

Effective and Privacy-Preserving Local Data Utilization with Distributed Machine Learning

Xue Jiang

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung einer

Doktorin der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr.-Ing. Pramod Bhatotia

Prüfende der Dissertation:

1. Prof. Dr. Jens Großklags
2. Prof. Dr. Stefanie Roos

Die Dissertation wurde am 20.12.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 14.07.2024 angenommen.

Abstract

Developing high-precision artificial intelligence (AI) products necessitates a diverse and large amount of real user data. However, user data are typically fragmented and stored in various local devices and data centers. In the conventional centralized setting, data analysts can directly gather these dispersed local data at the server side for conducting data mining and model training tasks. However, emerging privacy regulations have imposed strict restrictions on access and use of raw local data, which presents significant challenges for AI technology research and application. Therefore, how to enable data analysts to effectively utilize local data from different sources under comprehensive privacy protection is an urgent problem to be addressed in the current era of big data.

The problem of local data utilization can be addressed through two schemes: model-to-data transmission and data-to-model transmission. The model-to-data transmission approach uses distributed machine learning (ML) mechanisms, such as federated learning (FL), to conduct model training on the local side. During training, only the model parameters are exchanged, while the raw data remain on the local side. However, existing privacy-preserving federated learning (PPFL) algorithms cannot achieve a satisfactory balance among privacy, usability, and efficiency. In contrast, data-to-model transmission involves perturbing local data using state-of-the-art data anonymization techniques and sharing the privatized data with the server. The data can then be used for various downstream AI tasks. Nevertheless, existing data anonymization algorithms cannot comprehensively support the private sharing of high-dimensional data, unstructured data, and vertically partitioned data.

This thesis proposes a number of algorithms that employ the concept of distributed ML to address the challenges of both schemes. To start with, a novel local differential privacy (LDP)-based framework named SIGNDS-FL is introduced to improve the privacy-utility-efficiency trade-off in the model-to-data transmission scheme. The framework adopts a sign-based dimension selection strategy, which not only greatly alleviates the huge utility loss in previous LDP-FL frameworks but also significantly reduces the uplink communication overhead during FL training. The enhanced performance of the SIGNDS-FL framework further inspires the application of distributed ML to data-to-model transfer schemes. A number of algorithms applying deep learning-based synthetic data generation (DL-SDG) in a distributed manner are proposed to support private and high-utility data sharing in different scenarios. First, a framework called DP-FED-WAE is introduced, which combines a generative Wasserstein autoencoder (WAE) with the previously proposed SIGNDS-FL framework to effectively learn the statistical distributions of high-dimensional structure data without direct access to raw local data and generate high-fidelity synthetic data on the server side for downstream data mining and model training tasks. The concept of distributed DL-SDG is then extended to unstructured data, where a framework FEDSTDG is proposed for the private sharing of multivariate time series. The framework also incorporates an improved multi-dimension selection algorithm and an adaptive learning rate adjustment algorithm to the previous SIGNDS-FL framework to fur-

ther improve synthetic data utility. Finally, the idea is also applied to the scenarios of vertically partitioned data, where each client holds different attributes for the same group of data owners. A framework called VERTIGAN is introduced, which enables the server to gain insights into cross-attribute correlations among different parties and to generate high-utility synthetic data with the complete attribute set without the collection of raw local data.

Acknowledgments

I would like to take this opportunity to express my heartfelt appreciation to all those who have supported me during the successful completion of this thesis.

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Jens Grossklags, whose valuable guidance has profoundly influenced my growth in scientific research. His professional insights, constructive feedback, and constant encouragement have been instrumental in shaping the direction of my thesis.

Second, I am deeply grateful to have Professor Stefanie Roos and Professor Pramod Bhatotia serve on my thesis committee. Their insightful questions, helpful suggestions, and valuable feedback have inspired me to think more broadly about the problem and the impact of this work.

Next, I want to extend my profound thanks to my mentor, Dr. Xuebing Zhou from Huawei Munich Research Center, for leading me toward the research domain of privacy protection. Her patient guidance at work and warm care for me in life are important driving forces on my scientific path.

In addition, deep appreciation is felt towards my dear colleagues: Daniel Abril Castellano, Thomas Vannet, Derui Zhu, Yufei Zhang, Ricardo Mendes, Ahmed Frikha, and Krishna Kanth Nakka. I would like to thank them for the inspiring discussions and the pleasant working environment. Besides, I would also like to thank all my friends for their kind support and company.

Finally, I would like to express my earnest love to my father Sanyong Jiang, my mother Meiqun Ji, and my husband Chee Hung Koo for their unwavering encouragement and understanding. Also, profound thanks and remembrance are dedicated to my grandparents. Their affection and teaching have transcended the boundaries of time and space, and have consistently inspired my pursuits toward this achievement.

Table of Contents

Abstract	iii
Acknowledgments	v
Table of Contents	vi
List of Figures	xi
List of Tables	xv
List of Abbreviations	xvii
List of Symbols	xix
List of Publications	xxi
I Part A	1
1 Introduction	3
1.1 Motivation and Problem Statement	3
1.2 Challenges in Existing Solutions	5
1.2.1 Model-to-Data Transmission via Federated Learning	5
1.2.2 Data-to-Model Transmission via Privacy-Preserving Data Sharing	7
1.3 Research Goal and Guiding Research Questions	10
1.4 Road Map	11
1.4.1 Thesis Structure	12
1.4.2 Overview of Publications	12
2 Research Methods and Preliminaries	17
2.1 Differential Privacy and Its Variants	17
2.1.1 Differential Privacy	17
2.1.2 Rényi Differential Privacy	18
2.1.3 Local Differential Privacy	20
2.2 Federated Learning	21
2.2.1 Federated Learning for Horizontally-Partitioned Data	22
2.2.2 Federated Learning for Vertically-Partitioned Data	22
2.3 Synthetic Data Generation	24
2.3.1 Autoencoders	24
2.3.2 Generative Adversarial Network	25

II	Part B1	29
3	SIGNDS-FL: Horizontal Federated Learning with Local Differential Privacy	31
3.1	Abstract	32
3.2	Introduction	32
3.3	Related Work	35
3.4	SIGNDS-FL Framework	36
3.4.1	System Model	36
3.4.2	General Workflow of SIGNDS-FL	37
3.4.3	Local Differentially Private Dimension Selection	39
3.5	Experiments and Results	44
3.5.1	Experiment Setup	44
3.5.2	Performance in Simple Training Tasks	46
3.5.3	Performance in Complex Training Tasks	48
3.5.4	Performance Improvement with Multi-Dimension Selection	49
3.5.5	Analysis of Results	52
3.6	Discussion	53
3.7	Conclusion	53
4	DP-FED-WAE for Private Sharing of High-Dimensional Structured Data	55
4.1	Abstract	56
4.2	Introduction	56
4.3	Problem Statement	58
4.4	Proposed Solution	59
4.4.1	Data Pre-Processing and Design of the Generative Model	59
4.4.2	Training the Generative Model	61
4.4.3	Generating Synthetic Data and Data Post-Processing	64
4.5	Experiments and Results	64
4.5.1	Experiment Setup	65
4.5.2	Evaluation for Data Utility	69
4.5.3	Evaluation for Privacy Protection	75
4.6	Discussion	77
4.6.1	Extension to Other Data Types	77
4.6.2	Auxiliary Data for Pre-Training	78
4.7	Related Work	79
4.7.1	Data Collection Under Local Differential Privacy	79
4.7.2	Differentially Private Synthetic Data Generation	81
4.7.3	Efficiency and Privacy in Federated Learning	81
4.8	Conclusion	83
5	FEDSTDG for Private Sharing of Multivariate Time Series	85
5.1	Abstract	86
5.2	Introduction	86
5.3	Related Work	88
5.3.1	Differentially Private Synthetic Data Generation	88
5.3.2	Privacy-Preserving Federated Learning	89
5.4	Problem Statement	89

5.5	FEDSTDG Framework	91
5.5.1	Structure of the Recurrent Wasserstein Autoencoder	91
5.5.2	Training with Local Differentially Private Federated Learning	92
5.5.3	Overall Workflow of FEDSTDG	97
5.6	Experiments	97
5.6.1	Experiment Setup	97
5.6.2	Evaluation of Data Utility	100
5.6.3	Evaluation of Privacy Protection	106
5.6.4	Evaluation of Robustness	108
5.7	Conclusion	109
6	VERTIGAN for Private Sharing of Vertically-Partitioned Structured Data	111
6.1	Abstract	112
6.2	Introduction	112
6.3	Related Work	114
6.3.1	Data Analysis on Vertically-Partitioned Data	114
6.3.2	Differentially Private Data Synthesis	116
6.4	Problem Statement	116
6.5	Proposed Framework	118
6.5.1	Distributed GAN Against Data Inaccessibility	119
6.5.2	Multi-Output Generator Against Attribute Inconsistency	119
6.5.3	Collaborative Training with Differential Privacy	121
6.5.4	Overall Training Process	122
6.6	Experiments and Results	123
6.6.1	Experiment Setup	124
6.6.2	Utility: Statistical Similarity	127
6.6.3	Utility: AI Training Performance	129
6.6.4	Ablation Study	131
6.6.5	Empirical Privacy Analysis	134
6.7	Discussions and Future Work	135
6.7.1	Extension to Other Data Types	135
6.7.2	Reduction of Communication Cost	136
6.7.3	Protection for the Uploaded Gradients	136
6.8	Conclusion	137
III	Part C	139
7	Summary of Results	141
8	Limitations and Future Work	145
8.1	Inspection of Data Quality	145
8.2	Optimization of Hyperparameters	145
8.3	Performance Improvement Under High-Privacy Regimes	146
8.4	Reduction of Download Communication Cost in SIGNDS-FL	146
8.5	Comprehensive Privacy-Utility Assessments of Synthetic Data	147

Table of Contents

9	Conclusions	149
	Bibliography	151
	Appendix: Original Version of Publications	167
1	SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection	168
2	Privacy-Preserving High-dimensional Data Collection with Federated Generative Autoencoder	194
3	Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning	218
4	Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data	232

List of Figures

1.1	Privacy breach incidents and privacy regulations in recent decades.	4
1.2	Illustration of the centralized setting. The server directly collects private data from all clients to develop the AI model, which may cause privacy leakage and data compliance problems.	5
1.3	Illustration of the <i>model-to-data-transmission</i> solution. Instead of collecting private data for developing the AI model, the server broadcasts the model to the local side. Each client trains the model locally and only shares the model parameters with the server.	6
1.4	Illustration of two distinguish scenarios in <i>data-to-model-transmission</i> . In scenario (A), the raw data are initially collected by a group of trusted data collectors and shared with an untrusted third-party data analyst after data anonymization. In scenario (B), the data analyst also plays the role of data collector. Hence, the raw data are anonymized by the data owners themselves and then shared with the data analyst.	8
1.5	Structure of the thesis.	13
2.1	Example of model training on horizontally-partitioned data. Each local party holds the same set of features of a different set of samples. The server uses federated learning (FL) to increase the number of training samples to improve model performance. Here, a healthcare service seeks to use the data from different local devices to train a model for analyzing users' health conditions.	22
2.2	Example of model training on vertically-partitioned data, where each local party holds a different set of features of the same set of samples. The server uses FL to increase the number of features to improve model performance. Here, a credit card company seeks to use the data from an E-commerce company, a bank, and an online media company to train a credit score prediction model.	23
2.3	Example structure of an autoencoder (AE) model.	24
2.4	Example structure of an Wasserstein autoencoder (WAE) model.	25
2.5	Example structure of a generative adversarial network (GAN) model.	26

List of Figures

3.1	Overview of the SIGN-FL framework. At each global round, the server broadcasts the current global model to the local side. Each client trains the global model with local data and computes the local model update. Then, the client randomly samples a sign value and builds the top- k dimension set: if the sign value equals 1, the top- k set is built with the dimensions of the k largest update values; otherwise, it is built with the dimensions of the k smallest update values. An local differential privacy (LDP)-based dimension selection algorithm is then applied to select a set of “important” dimensions. The sampled sign value and the selected dimension set will be sent to the server. The server will construct sparse privatized local updates by assigning the sign value to the corresponding selected dimensions and finally use them to update the global model.	33
3.2	Example of using the inverse sampling trick to determine ν , namely the number of top- k dimensions in J	44
3.3	Utility of using PS and EM-MDS algorithms in multi-dimension selection regarding different output size h and privacy budget ϵ	44
3.4	Accuracy of logistic regression (LR) models trained on structured datasets with different privacy budgets ϵ	47
3.5	Accuracy of LR models trained on structured datasets with different group sizes.	48
3.6	Accuracy of neural network (NN)s and convolutional neural network (CNN)s trained on image datasets with different privacy budget ϵ	49
3.7	Accuracy of NNs and CNNs trained on image datasets with different group size.	50
3.8	Comparison of model accuracy regarding different dimension selection strategies.	51
3.9	Comparison of model convergence regarding different dimension selection strategies.	52
4.1	Overview of the DP-FED-WAE framework. The generative Wasserstein Autoencoder is first trained under the federated setting, which learns the distributions of real local data. An LDP algorithm SIGNDS is applied to the local updates to provide strict local privacy guarantees. After the model is trained, the <i>decoder</i> part is used to generate high-utility synthetic data. The generated data will be used for data mining and building AI services.	60
4.2	Relations among ϵ , p , and ξ . (a): given privacy budget ϵ and the expected top- k probability p , the minimum top- k ratio ξ required. (b): given the expected top- k ratio ξ and top- k probability p , the minimum privacy budget ϵ required.	64
4.3	Structure of the WAE model used for Adult dataset.	67
4.4	AVD of four-way joint distribution between the real and synthetic data with respect to different privacy levels.	70
4.5	AVD of m -way joint distributions between the real and synthetic data with respect to different dimensions of joint distribution.	71
4.6	CMD between the real and synthetic data with different privacy levels.	72
4.7	Correlation comparison between the real and synthetic data with $\epsilon = 8$ and $\xi = 0.1$. For each dataset, we present the correlations of the first 10 attributes. It can be seen that the synthetic data preserves similar correlations as real data.	73

4.8	Classification accuracy of NN models trained with real data (<i>Real Data</i>) and synthetic data generated by our framework (<i>WAE</i>) as well as by the baseline algorithms (<i>LoPub</i> , <i>LoCop</i>) under different privacy levels.	74
4.9	Classification accuracy of random forest (RF) models trained with real data (<i>Real Data</i>) and synthetic data generated by our framework (<i>WAE</i>) as well as by the baseline algorithms (<i>LoPub</i> , <i>LoCop</i>) under different privacy levels.	75
4.10	Classification accuracy of NN models with different numbers of records under the privacy level of $\epsilon = 8$	76
4.11	Classification accuracy of RF models with different numbers of records under the privacy level of $\epsilon = 8$	77
4.12	Classification accuracy of NN models with different numbers of users under different privacy levels.	78
4.13	Classification accuracy of RF models with different numbers of users under different privacy levels.	79
4.14	Results of data synthesis on image datasets.	80
5.1	Overview of the FEDSTDG framework. The recurrent Wasserstein autoencoder (RWAE) model is first trained under the federated setting, which learns the distributions of real local data. Two LDP algorithms, SUBMDS and MAGRR, are respectively applied for privatizing the local updates and the update magnitude. After the model is trained, the <i>decoder</i> part is used to generate high-utility synthetic data. The generated data will be used for data analysis and building AI services.	90
5.2	Structure of the RWAE model.	91
5.3	Comparison of the optimal output size under different privacy budgets and input top- k ratio. Given the <i>original</i> input size $d = 20000$ and top- k size $k = 100$, we vary the <i>actual</i> input size among $\tilde{d} = \{0.01d, 0.05d, 0.1d\}$, resulting in an input top- k ratio k/\tilde{d} among $\{0.5, 0.1, 0.05\}$. Then, we compare the corresponding optimal output size h^* regarding different expected output top- k ratio ζ^*	94
5.4	Comparison of maximum mean discrepancy (MMD) distance between real and synthetic data of all the datasets. For each dataset, we compare the results of non-private centralized and FL settings as well as different LDP-FL baselines under different privacy levels.	101
5.5	T-SNE visualization of the distribution of real and synthetic data on the four datasets. Each column represents the results of one dataset. Each row provides the distribution of synthetic data generated under non-private centralized and FL settings as well as using different LDP-FL algorithms with $\epsilon = 8$. SUBMDS and ADASUBMDS refer to the results of using the proposed methods. Red denotes the real data and blue denotes synthetic data. We also use the Synthcity library [127] to compute the coverage ratio between real and synthetic data.	102
5.6	Analysis of the impact of privacy split ratio. For each dataset, we train the RWAE model with ϵ_{mag} being 10%, 50%, 90% of the total privacy ϵ and compare the mean absolute error (MAE) of synthetic data under different privacy settings.	105

List of Figures

5.7 Analysis of the impact of per-round clients. For each dataset, we train the RWAE model with 50, 100, and 200 per-round clients and compare the MMD of synthetic data under different privacy settings. 106

5.8 Analysis of the impact of decay rate r and patience threshold T_{pat} . For each dataset, we respectively use decay rate $r \in \{0.5, 0.9\}$ and patience threshold $T_{pat} \in \{10, 50, 100\}$ to train the RWAE model for 500 rounds under $\epsilon = 6$. Results show the MMD of synthetic data during the training process. 107

5.9 The effectiveness in defending against model poison attacks evaluated on the four datasets. For each dataset, we present the predictive MAE of synthetic data generated under different poisoning attacks and privacy settings. 109

6.1 Overview of the system model. 117

6.2 General workflow of the VERTIGAN framework. 118

6.3 Workflow of the local training process. 120

6.4 AVD of 4-way joint distributions between the real and synthetic data with respect to different privacy levels. 128

6.5 AVD of m -way joint distributions between the real and synthetic data with respect to different dimensions of the joint distribution. 129

6.6 Correlation comparison between the real and synthetic data with $\epsilon = 8$. For each dataset, we present the correlations of 20 attributes, where each party contributes 10 attributes. It can be seen that the synthetic data preserves similar correlations as real data. 130

6.7 4-way AVD under the two-party settings with 10^4 , 10^5 , and 10^6 records under the privacy level $\epsilon = \{0.5, 2, 8\}$ 132

6.8 4-way AVD under the two-party settings with $\epsilon = 8$ and attribute split ratio from $\{0.1/0.9, 0.3/0.7, 0.5/0.5\}$. For each dataset, we compare the results of *random splitting* and *correlated splitting*, where the strongly correlated attributes are assigned to one of the parties. 132

6.9 4-way AVD under the two-party settings with $\epsilon = 8$ and the number of clients from $\{2, 4, 8, 16\}$. For each dataset, we compare the results of *random splitting* and *correlated splitting*, where the strongly correlated attributes are assigned to a subset of the parties. 133

6.10 Results of image data synthesis under a three-party setting. Each row represents the synthetic images generated by one local party under different privacy settings. 135

6.11 4-way average total variation distance (AVD) between the real and synthetic data with respect to different gradient sparsity ratios. 136

List of Tables

1.1	Limitations of existing work in privacy-preserving FL.	7
1.2	Comparison of existing techniques for privacy-preserving data sharing.	10
1.3	Mapping of challenges to the addressed research questions	12
1.4	Summary of embedded publications	15
3.1	Bibliographic details for P1	31
3.2	Details of datasets and models	45
4.1	Bibliographic details for P2	55
4.2	Datasets details	66
4.3	Structure of WAE models	66
4.4	Computation time of model training and synthetic data generation	69
4.5	Accuracy of membership inference attack	76
4.6	Classification accuracy of synthetic data generated by WAE models with (w) and without (w/o) pre-training	80
5.1	Bibliographic details for P3	85
5.2	Details of Datasets and Model Size	99
5.3	Comparison of synthetic data utility in artificial intelligence (AI) training tasks. For each dataset, we respectively use real and synthetic data to train one-layer long short-term memory (LSTM) models for <i>next-step prediction</i> and evaluate the models' predictive MAE on a <i>held-out</i> set of real data. Here we compare the MAE of synthetic data generated under different privacy settings.	104
5.4	Comparison of MMD of synthetic data generated by RWAEs with (w) and without (w/o) pretraining.	108
5.5	Averaged attack accuracy of the membership inference attack (MIA) attack under different privacy settings.	108
6.1	Bibliographic details for P4	111
6.2	Datasets details	125
6.3	One-hot dimensions and the number of model parameters under the two-party setting	125
6.4	Computation time (sec) of the proposed VERTIGAN framework and baseline DPLT+ algorithm regarding different datasets. For VERTIGAN, we perform 1500 global rounds and report the total training time.	127
6.5	Classification accuracy of MLP models evaluated on synthetic data generated under different privacy settings.	131
6.6	MIA accuracy under different privacy settings.	134

List of Tables

7.1 Summary of results 143

List of Abbreviations

AE	autoencoder
AI	artificial intelligence
AVD	average total variation distance
CCPA	California Consumer Privacy Act
CDP	central differential privacy
CE	categorical cross-entropy
CMD	correlation matrix distance
CNN	convolutional neural network
DL	deep learning
DL-SDG	deep learning-based synthetic data generation
DNN	deep neural network
DP	differential privacy
DPSGD	differentially private stochastic gradient descent
DS	dimension selection
EM	exponential mechanism
FL	federated learning
GAN	generative adversarial network
GDPR	General Data Protection Regulation
HE	homomorphic encryption
HPO	hyperparameter optimization
IoT	Internet-of-Things
LDP	local differential privacy
LR	logistic regression
LSTM	long short-term memory
MAE	mean absolute error
MIA	membership inference attack
ML	machine learning
MMD	maximum mean discrepancy
MSE	mean squared error
NN	neural network
PATE	private aggregation of teacher ensembles
PETs	privacy-enhancing technologies
PPFL	privacy-preserving federated learning
PRNG	pseudorandom number generator
RDP	Rényi differential privacy
RF	random forest
RWAE	recurrent Wasserstein autoencoder

List of Abbreviations

SDG	synthetic data generation
SMC	secure multi-party computation
VAE	variational autoencoder
VFL	vertical federated learning
VP	value perturbation
WAE	Wasserstein autoencoder
WGAN	Wasserstein GAN

List of Symbols

P_x	Distribution of real data
$P_{\hat{x}}$	Distribution of synthetic data
P_z	Distribution of latent space features
Q_ψ	Encoder of an autoencoder
G_θ	Decoder of an autoencoder
\mathcal{L}_{lat}	Latent penalty of an autoencoder
\mathcal{L}_{rec}	Reconstruction loss of an autoencoder
G	Generator of a generative adversarial network
D	Discriminator of a generative adversarial network
\mathcal{M}	A differential privacy mechanism
ϵ	Privacy cost of a differential privacy mechanism
δ	Failure probability of a differential privacy mechanism
$\Delta_2 f$	L_2 sensitivity of function f
ζ	Subsampling rate for privacy amplification
N	Number of per-round participants in federated learning
M	Global model in federated learning
E	Number of local epochs in federated learning
T	Number of global rounds in federated learning
X^i	Local dataset of client i in federated learning
M_t	Global model at round t in federated learning
M_t^i	Local model of client i at round t in federated learning
Δ_t^i	Local model update of client i at round t in federated learning
η	Local learning rate in federated learning
γ	Global learning rate in federated learning
Γ	Metric function for measuring the distance between two records
\mathcal{R}_ρ	Score of membership
d	Model dimensionality in SIGNDS-FL
S_{topk}	Set of the top- k indices in SIGNDS-FL
J_t^i	Set of indices selected by client i at global round t in SIGNDS-FL
\mathcal{J}	Domain of the output set in SIGNDS-FL
s_t^i	Sign value sampled by client i at global round t in SIGNDS-FL
k	Number of indices in the top- k set in SIGNDS-FL
h	Number of indices in the output set in SIGNDS-FL
u	Score function for the exponential mechanism in SIGNDS-FL
ϕ_u	Sensitivity of the score function in SIGNDS-FL
ν	Number of top- k indices in the output set in SIGNDS-FL
ν_{th}	Threshold of the number of top- k indices in the output set in SIGNDS-FL

List of Symbols

ω_τ	Number of output sets containing τ top- k indices in SIGNDS-FL
$ \mathcal{A} $	Number of attributes in attribute set \mathcal{A} in structured data
\mathcal{A}	Entire set of attributes in structured data
A	Subset of attributes in structured data
a_i	i -th attribute in structured data
Λ	Domain of attribute values in structured data
L	Number of timesteps in multivariate time series
W	Number of attributes in multivariate time series
$f_{l,w}$	Value of attribute w at timestep l
ρ	Subsampling ratio in SUBMDS
s_h	Output top- k ratio in SUBMDS
ζ^*	Desired output top- k ratio in SUBMDS
r	Decay rate in MAGRR
b	Decay flag in MAGRR
T_{pat}	Patience threshold in MAGRR
t_{pat}	Number of rounds requiring a learning rate decay in MAGRR
H	Number of data records held by each client in federated learning
Φ^i	Local pseudo-random number generator in VERTIGAN
ϑ	Random seed for pseudo-random number generator in VERTIGAN
C	L_2 clipping bound in VERTIGAN
B	Batch size for local training in VERTIGAN

List of Publications – Xue Jiang

Core Publications Included in This Thesis

- **Xue Jiang**, Xuebing Zhou, and Jens Grossklags (2022) SignDS-FL: Local Differentially Private Federated Learning with Sign-Based Dimension Selection. *ACM Transactions on Intelligent Systems and Technology*, 13(5): 1–22.
- **Xue Jiang**, Xuebing Zhou, and Jens Grossklags (2022) Privacy-Preserving High-Dimensional Data Collection with Federated Generative Autoencoder. *Proceedings on Privacy Enhancing Technologies*, 2022(1): 481–500.
- **Xue Jiang**, Yufei Zhang, Xuebing Zhou, and Jens Grossklags (2023) Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data. *Proceedings on Privacy Enhancing Technologies*, 2023(2): 236–250.

Working Papers Included in This Thesis (No Core Publication)

- **Xue Jiang**, Xuebing Zhou, and Jens Grossklags (2024) Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning. Working Paper, 2024.

Other Publications (Not Included in This Thesis)

- **Xue Jiang**, Xuebing Zhou, and Jens Grossklags (2022) Comprehensive Analysis of Privacy Leakage in Vertical Federated Learning During Prediction. *Proceedings on Privacy Enhancing Technologies*, 2022(2): 263–281.

Part I

Introduction to the Thesis

1 Introduction

1.1 Motivation and Problem Statement

In recent decades, there has been an exponential increase in the volume and diversity of data generated from various aspects of our lives, such as from social media, financial transactions, healthcare, and other online activities. It is estimated that the volume of data generated, consumed, copied, and stored will exceed 180 Zettabytes by 2025 [143]. Although these massive amounts of data contain valuable information about human populations, they are typically fragmented and stored in different local devices and organizational data warehouses. As such, being able to access and effectively utilize these local data is crucial for successful big data analytics. By analyzing the data obtained from different sources, researchers and businesses can gain broader insights into consumer behavior, preferences, and motivations and use the information to make informed decisions and develop advanced AI products that better meet the needs of consumers. For example, in an E-commerce scenario, retailers can analyze different customers' online shopping behaviors and personalize their product offerings to a specific group according to their purchasing preferences. Similarly, a healthcare company can aggregate the health records of patients in different hospitals to explore risk factors for certain diseases and develop targeted applications. Leveraging the wide range of data from different sources not only yields significant social and economic benefits, but also fosters innovation of technology, services, and products.

In the traditional AI product development, data analysts directly gather data from different local data sources into a central data warehouse for data mining and training effective machine learning (ML) models. However, as these data usually contain sensitive information about individuals, the direct collection of raw data may lead to serious privacy risks. More specifically, more than 30,000 cases of data breaches since 2004 have been recorded in [165], where the number of records exposed per incident ranges from millions to trillions. According to a recent report in 2023 [106], a staggering 2.6 billion personal records were leaked globally in 2021 and 2022, and the total number of data breaches in 2023 will further increase to be 20% higher than in 2022. A few of the particularly notorious data breach incidents are shown in Figure 1.1. It is estimated that the average total cost of a single data breach is nearly 4.5 million dollars. The frequent and severe repercussions of data breach incidents have garnered significant societal attention regarding the protection of data privacy and led many countries to implement a series of privacy protection regulations (Figure 1.1), such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) to set ground rules for the collection, usage, and sharing of personal data. Violations of these laws and regulations can lead to substantial business costs. It is predicted by Gartner that by 2024, "75% of the world's population will have personal data covered under modern privacy regulations, up from 10% in 2020" [50]. The increasing prominence of privacy concerns and the accelerated implementation of stringent privacy regulations have created a significant barrier

1 Introduction

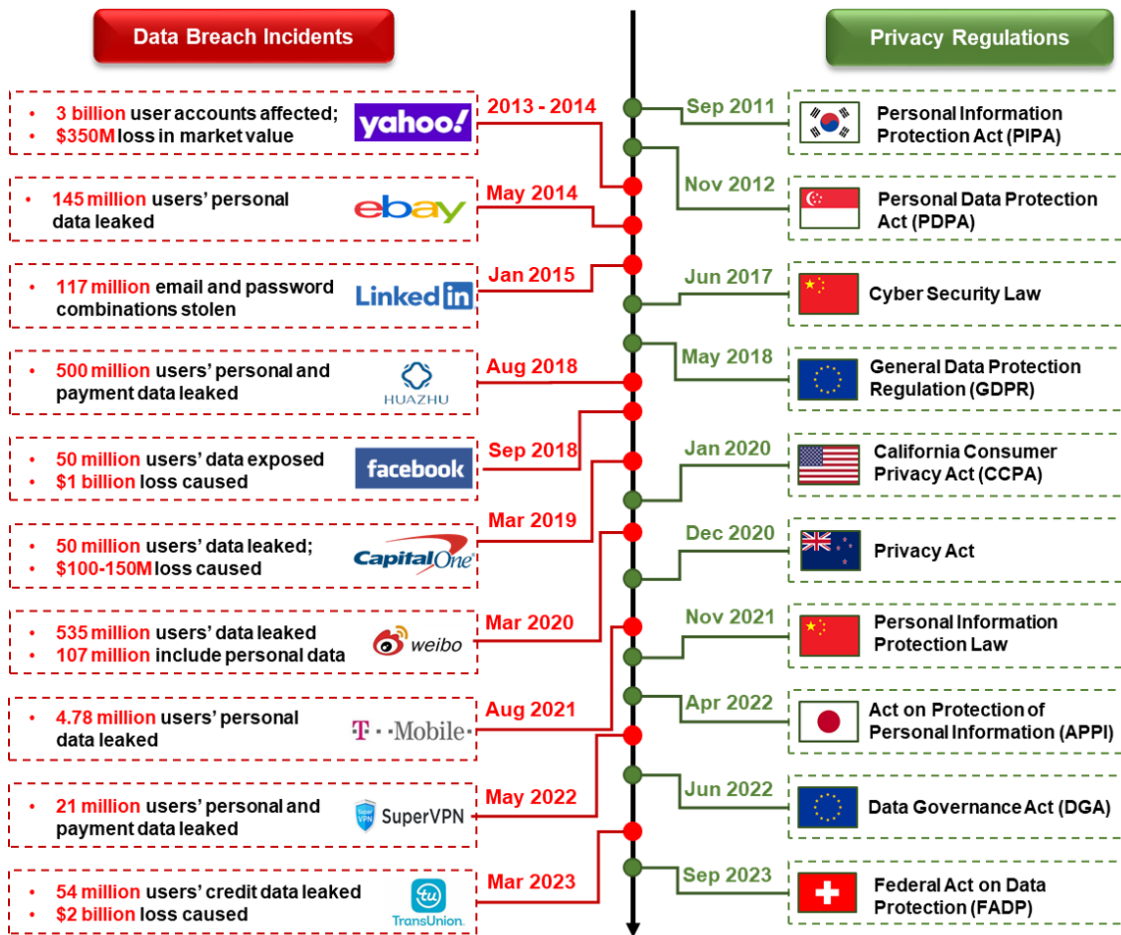


Figure 1.1 Privacy breach incidents and privacy regulations in recent decades.

for different service providers to freely gather and leverage raw personal data that are stored in local devices and data centers, as they would in a centralized setting (as illustrated in Figure 1.2). The lack of access to raw local data presents significant challenges in the research and application of AI technologies.

To enhance clarity and simplicity, this thesis will use the terms *central server* or simply *server* to refer to the party responsible for performing data mining and model training tasks, and *local client* or *client* to refer to the party that holds the raw user data. As introduced above, user data are usually dispersed on the local side and held by a number of clients, which cannot be directly accessed by the server due to privacy issues. **Therefore, there is a great need to explore potential solutions that enable the server to effectively utilize local data from different clients without violating privacy.**

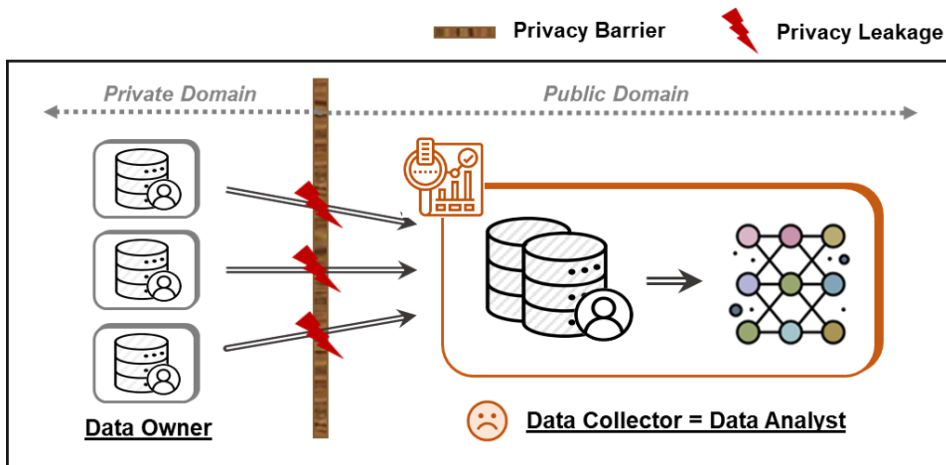


Figure 1.2 Illustration of the centralized setting. The server directly collects private data from all clients to develop the AI model, which may cause privacy leakage and data compliance problems.

1.2 Challenges in Existing Solutions

Current data-driven AI systems typically involve two main components: *model* and *data*. Hence, existing solutions that enable private utilization of local data can be categorized into two groups, namely *model-to-data transmission* and *data-to-model transmission*. In the model-to-data transmission solutions, the model is sent to the local side and the training process is performed in a distributed manner. This eliminates the need for the server to collect the raw data from the local client. In contrast, solutions based on data-to-model transmission employ advanced data anonymization techniques to privatize the raw local data. Then, only the anonymized data is shared with the server, which allows the model to be trained on the server side without compromising the privacy of the data owners. Next, we present the existing approaches and research challenges for each of these two solutions.

1.2.1 Model-to-Data Transmission via Federated Learning

The fundamental concept behind model-to-data transmission is to enable multiple clients to collaboratively train the ML model on the local side without sharing their raw data with the server. An illustration of the solution is shown in Figure 1.3. One of the representative technologies following this scheme is federated learning (FL) [108], a distributed ML mechanism that has attracted extensive research and attention in recent years in a wide range of scenarios such as healthcare [117, 131], communication networks [97, 120], Internet-of-Things (IoT) [84, 116], *etc.* During each training round of FL, the server broadcasts the current global model to the local clients. Each client uses the raw local data to train the model and sends the model updates back to the server. The server then aggregates the received information to update the global model and sends it to the local clients again in the next round. By only allowing the exchange of model parameters between the server and clients during the training process while keeping raw data on the local side, FL enables the server to effectively leverage

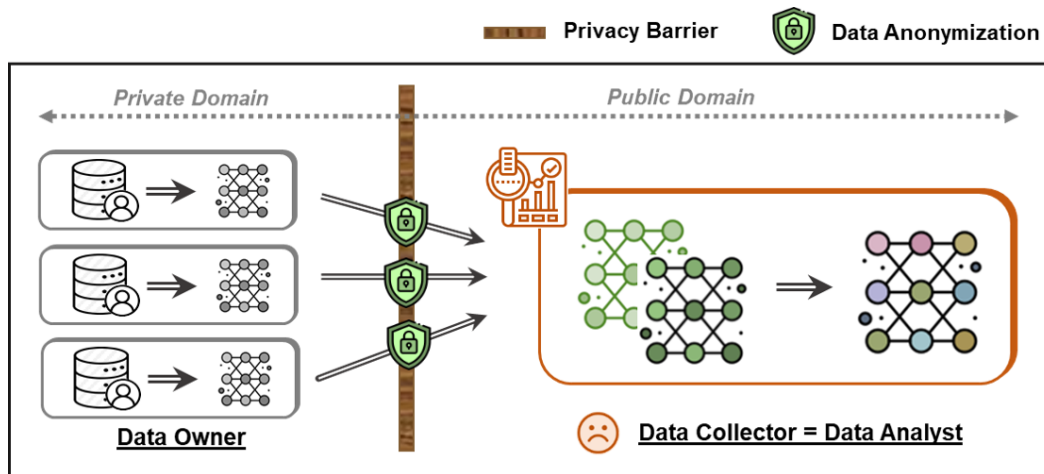


Figure 1.3 Illustration of the *model-to-data-transmission* solution. Instead of collecting private data for developing the AI model, the server broadcasts the model to the local side. Each client trains the model locally and only shares the model parameters with the server.

local data without directly accessing them. As a result, the mechanism provides significant privacy benefits in comparison to the centralized setting.

Nevertheless, since the model is iteratively trained on raw local data, it may still reveal a certain degree of sensitive information about these training data. Actually, a series of recent studies have pointed out that the original FL mechanism is vulnerable to various privacy attacks (e.g., [103, 71, 182]). For instance, since the model is trained on real local data, sensitive information from the training data might be encoded into the model updates. Therefore, an honest-but-curious server or any adversaries eavesdropping on a client's communications with the server may use the uploaded model updates to reconstruct the raw local data [186, 51, 182]. Moreover, as the local model updates are further used to update the global model, adversaries that have access to the global model may also apply membership inference [115, 112] or property inference attacks [112]: the former aims to infer whether a specific client has participated in the training process, while the latter aims to infer whether the local data contain certain properties that are independent of the main goal of the training task (e.g., given a facial expression classifier, infer whether the local data contains images of a particular gender).

In order to mitigate the potential privacy threats, different privacy-enhancing technologies (PETs) have been incorporated into the original FL, leading to the emergence of various privacy-preserving federated learning (PPFL) mechanisms. Some prior works [110, 7]) applied differential privacy (DP) [42], a rigorous mathematical notion of privacy, to protect the global model. More specifically, by clipping each local update and adding random noise to the global model, the algorithms limit the impact of each client's local data on the global model and hence achieve client-level central differential privacy (CDP) guarantees. However, these algorithms cannot prevent privacy leakages from local updates. Alternatively, crypto-based methods adopted secure multi-party computation (SMC) [14, 135] and homomorphic encryption (HE) [124, 104] protocols to encrypt the local model weights. The encrypted message will be aggregated on the server side and then decrypted, ensuring that the server cannot gain information about an individual client. Follow-up works [69, 78, 147, 57] have further com-

Table 1.1 Limitations of existing work in privacy-preserving FL.

Categories	Protection Methods	Actor	Target of Protection	Limitations
Crypto-based	SMC [14, 135]	Client	Local Model	<ul style="list-style-type: none"> • High communication cost • High computation cost • No protection for global models
	HE [124, 104]			
DP-based	CDP [110, 7]	Server	Global Model	<ul style="list-style-type: none"> • No protection for local models • Slight impact on model utility
	LDP [40, 184]	Client	Local Model Global Model	<ul style="list-style-type: none"> • Large impact on model utility
Hybrid	CDP+SMC [69, 78]	Client and Server	Local Model Global Model	<ul style="list-style-type: none"> • High communication cost • High computation cost
	CDP+HE [147, 57]			

bined the crypto protocols with DP [42], where random noise is added either on the local side or server side to ensure that the global model satisfies CDP guarantees. Nevertheless, these crypto-based solutions usually require extra communication and computation costs, hence may not be practical in large-scale scenarios. To mitigate both privacy and inefficiency problems, some other works [40, 184] have also proposed to perturb the local model updates, which provides strict LDP [82] guarantees for the local data. In comparison to the CDP-based solutions, LDP-based FL mechanisms provide strong privacy protection for both local and global models. Moreover, they facilitate communication and computation more efficiently compared to the crypto-based methods. Nevertheless, LDP-based algorithms usually suffer from significant utility loss, especially for high-dimensional models. In Table 1.1, we summarize the existing PPFL solutions and their limitations. **It is apparent that there is still a great need for PPFL algorithms that can achieve *privacy*, *utility*, and *efficiency* at the same time.**

1.2.2 Data-to-Model Transmission via Privacy-Preserving Data Sharing

Although the model-to-data transmission scheme provides a solution to perform model training without directly accessing the clients' local data, the server must repeat the training process each time the models are changed, resulting in additional communication and computational expenses. Instead of shifting the training process to the local side, an alternative solution is to let the clients first privatize the raw local data using data anonymization techniques. Then, the clients only need to share the processed data with the server, which will be used for multiple downstream data analysis and model training tasks. Moreover, the scheme can be further divided into the *cross-silo* scenario and the *cross-device* scenario, as shown in Figure 1.4. In the cross-silo scenario (scenario (A)), it is assumed that a group of trusted data collectors has gathered the raw data of individual data owners into several data centers and aims to share the anonymized datasets with a third-party data analyst. For instance, several hospitals having different patients' medical records want to share their patients' data with an AI company to

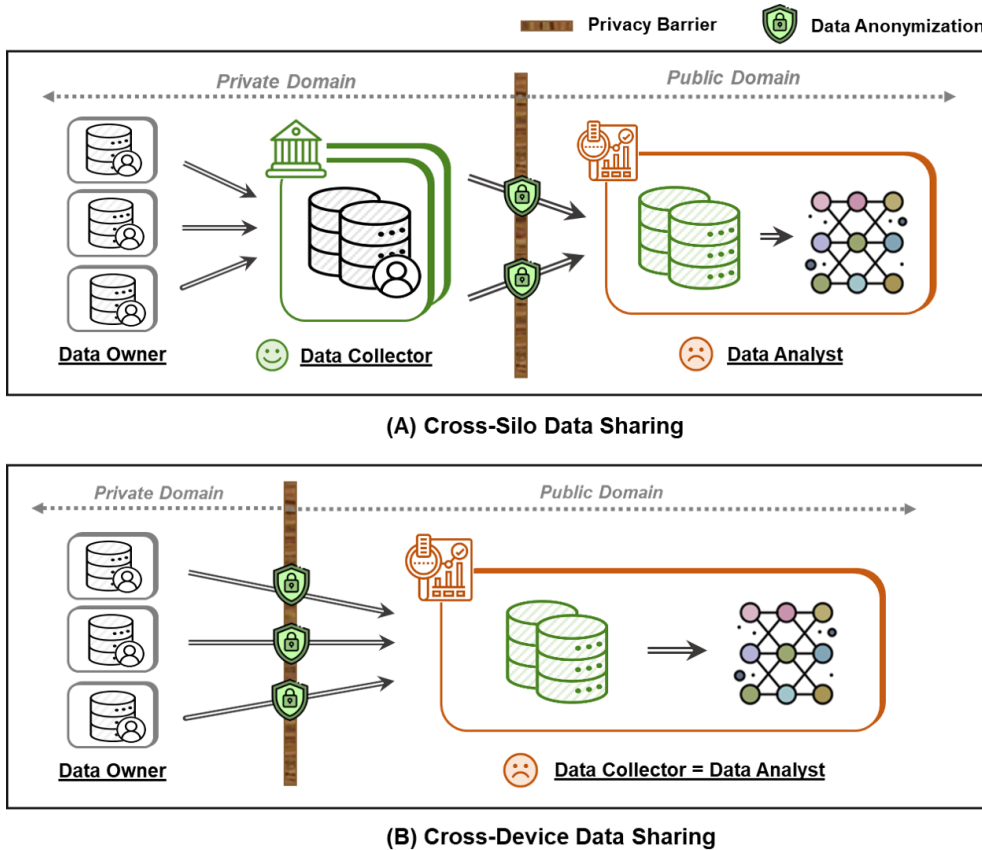


Figure 1.4 Illustration of two distinguish scenarios in *data-to-model-transmission*. In scenario (A), the raw data are initially collected by a group of trusted data collectors and shared with an untrusted third-party data analyst after data anonymization. In scenario (B), the data analyst also plays the role of data collector. Hence, the raw data are anonymized by the data owners themselves and then shared with the data analyst.

develop a cancer prediction model. Here, each data collector is considered a client, and the dataset stored in each data center contains private data belonging to *multiple* data owners. Conversely, in the cross-device scenario (scenario (B)), the data collector carries out the data analysis task directly. For instance, an online shopping application seeks to directly collect shopping preferences from users' end devices to train advanced recommendation algorithms. Here, there are no longer trusted data collectors. Each data owner is considered a client, and the dataset stored in each end device contains private data belonging to a *single* data owner.

According to the EU Article 29 Data Protection Working Party's 05/2014 opinion [123], the main anonymization techniques can be generally categorized into two groups, namely *generalization* and *randomization*. *Generalization* aims to reduce the granularity of a data representation by suppressing the attribute values with an asterisk '*' or replacing them with a broader category (for instance, replace *city* with *country*, or replace *date* with *month*). Some of the representative techniques include *k*-anonymity [141], *t*-closeness [105], and *l*-diversity [93]. Although such techniques ensure that an individual's data will be "hidden" in a larger group that contains multiple records with the same representation, the released datasets still contain

real personal data and hence may still reveal sensitive information of data owners. In contrast, *Randomization* aims to perturb the attribute values in order to break the strong link between the data record and the individual. Some common randomization techniques include *permutation*, *noise addition*, and *DP* [42]. *Permutation* swaps certain attribute values from one record with another to eliminate the link between attributes and individuals. However, the separate permutation of attributes may break the potential cross-attribute correlations and hence cause the data to become unusable for advanced data analysis tasks such as ML. *Noise addition* adds random noise to attribute values to reduce their accuracy, but it is difficult to measure the degree of privacy protection achieved by a certain amount of noise. In comparison, *DP-based* algorithms perturb the original dataset under formal privacy guarantees. Prior works first use probabilistic graph models [25, 179, 17] or copula functions [92, 49] to approximate low-dimensional marginal distributions of the original dataset. Then, the marginals are perturbed with random noise to achieve CDP guarantees for each individual record. Finally, the noisy marginals will be used to generate a synthetic dataset, which preserves similar statistical properties as the original dataset. As the generated synthetic data are fully synthetic and eliminate the linkage to the original data, the algorithms effectively reduce risks of re-identification attacks or attribute disclosure [122]. However, these algorithms are only feasible in the *cross-silo* scenarios, where each original dataset contains records belonging to *different* data owners and individuals' privacy can hence be protected via CDP. In contrast, in the cross-device scenario, each original dataset belongs to *one specific* data owner. CDP may not be sufficient in this case since the anonymized dataset preserves overall distributions of the original data and hence may still expose the individual's private information. To tackle this issue, a variant concept of differential privacy called LDP [82] has been introduced. LDP guarantees privacy for each individual's local data by ensuring that the randomization outcome of one client's data cannot be differentiated from the data of other clients. Nevertheless, prior research on LDP-based data sharing mainly focuses on one-dimensional statistics, such as frequency estimation [35, 44], heavy-hitter identification [10, 16], and itemset mining [128, 161], *etc.* Although a number of follow-up works [45, 130, 160] proposed solutions for multi-dimensional data, these algorithms still suffer from the *curse of dimensionality*: an increase of the number of attributes will not only increase the computation and communication cost but also lead to a significant degradation in data utility.

With the development of AI, deep learning-based synthetic data generation (DL-SDG) has attracted increasing attention in recent years. The main idea of such techniques is to train deep generative models such as AE [85, 144] and GAN [53] to learn the correlations and marginal distributions of the original data and then use the trained models to directly generate high-fidelity synthetic data. A number of existing works [122, 77, 23] have proposed to incorporate DP during the training of generative models to provide strict privacy guarantees for the original data. This can be considered an alternative solution for private data sharing under cross-silo scenarios, where each data collector can train a separate generative model using the data gathered from multiple data owners. It has been shown that under the same privacy guarantees, the DL-SDG-based algorithms can achieve much better utility for high-dimensional data in comparison to traditional marginal-based algorithms. Nevertheless, these algorithms are not directly applicable to cross-device scenarios, as the synthetic data released by each data owner may still reveal sensitive information about the individual. As such, **how to apply the concept of DL-SDG for cross-device data sharing is an important challenge to be solved in the data-to-model transmission scheme.** In addition, compared

Table 1.2 Comparison of existing techniques for privacy-preserving data sharing.

Anonymization Techniques	K-Anonymity [141, 105, 93]	DP		DL-SDG [122, 77, 23]
		CDP [25, 17, 49]	LDP [45, 130, 92]	
Data Sharing Scenarios	Cross-silo	Cross-silo	Cross-device	Cross-silo
Data Type	Structured	Structured	Structured	Structured, Unstructured
Data Partition Strategies	Horizontal, Vertical	Horizontal, Vertical	Horizontal	Horizontal
Pros	<ul style="list-style-type: none"> • Easy implementation 	<ul style="list-style-type: none"> • Strong privacy guarantees 		<ul style="list-style-type: none"> • Good data utility • Strong privacy guarantees when applying DP during model training • Support different data types
Cons	<ul style="list-style-type: none"> • Weak privacy guarantees • Not scalable • Only support structured data • Only support cross-silo scenarios 	<ul style="list-style-type: none"> • Only support structured data • Significant utility loss for high-dimensional data 		<ul style="list-style-type: none"> • Only support cross-silo scenario • Only support horizontally-partitioned data

with traditional anonymization algorithms that can only be used for sharing structured data, DL-SDG-based algorithms can flexibly adjust model architectures to support the release of other unstructured data types (such as images and time series) in the cross-silo scenarios. **Nevertheless, whether these solutions can be leveraged for sharing unstructured data under cross-device scenarios remains an unexplored issue.** Finally, although the existing DL-SDG algorithms can be applied in the cross-silo scenarios, they only focused on the horizontally-partitioned setting, where data collectors hold data of different individuals with the same attributes. However, in some other scenarios, the raw data are vertically-partitioned. The data collectors usually have different sets of attributes of the same group of individuals and the data analysts aim to combine all the attributes and create an integrated dataset to improve the data mining accuracy. Unfortunately, previous DL-SDG-based solutions are also not feasible in this case, as the synthetic data separately generated by each data collector cannot be linked to each other and combined into an integrated dataset. As such, **how to apply the concept of DL-SDG for sharing vertically-partitioned data is also an open question.**

1.3 Research Goal and Guiding Research Questions

This thesis aims to enable the efficient and private utilization of local data by solving a series of challenges in both model-to-data transmission and data-to-model transmission schemes. The subsequent part of this section outlines the guiding research questions (RQs) that will be investigated in this thesis to achieve this overarching goal.

RQ1: In the model-to-data transmission scheme, can the current PPFL frameworks be improved with a better balance in privacy, utility, and efficiency?

The objective of RQ1 is to address the challenge of the currently unfavorable privacy-utility-efficiency trade-off in *model-to-data transmission* solutions. As mentioned earlier, the existing PPFL frameworks either suffer from excessive communication and computation expenses, insufficient privacy protection, or a substantial reduction in usefulness. RQ1 seeks to investigate feasible enhancements to the existing PPFL frameworks to achieve a better balance between privacy, utility, and efficiency.

RQ2: In the data-to-model transmission, can the concept of DL-SDG-based privacy-preserving data sharing be applied in cross-device scenarios?

Considering the strong capabilities of DL-SDG solutions in generating high-fidelity synthetic data, RQ2 aims to investigate the viability of leveraging this technique for sharing horizontally-partitioned data in cross-device scenarios. The primary objective is to enable data analysts to obtain a set of synthetic data that preserves similar overall statistical properties as the original local data while ensuring that the private information of arbitrary data owners is not compromised.

RQ3: Can the DL-SDG-based cross-device data sharing solutions be extended to unstructured data types?

Building upon the findings of RQ2, RQ3 further seeks to investigate the feasibility of utilizing DL-SDG techniques for sharing unstructured data in cross-device scenarios. In comparison to structured tabular data, unstructured data such as images and time series are more intricate and have more complex semantic representations. Hence, the answer to this research question would further enhance the overall capability of the technique in sharing diverse types of data.

RQ4: Can the concept of DL-SDG-based privacy-preserving data sharing be applied to vertically-partitioned data?

Finally, in contrast to RQ2 and RQ3 which focus on horizontally-partitioned data, RQ4 is formulated to tackle the challenge of applying DL-SDG-based techniques to share vertically-partitioned data. In comparison to the horizontal setting, the generated synthetic data in the vertical setting are required to retain the correlations between attributes held by different clients, thereby increasing the complexity of the data-sharing process. The answer to this research question would further affirm the feasibility of DL-SDG-based techniques for private data sharing under different data partitioning strategies.

Each RQ and the corresponding publication is linked to and derived from the presented challenges. Table 1.1 provides an overview of which challenge is addressed by each RQ.

1.4 Road Map

This section provides an outline of the thesis structure and an overview of the publications included in this thesis.

Table 1.3 Mapping of challenges to the addressed research questions

Challenges in extant literature	RQ1	RQ2	RQ3	RQ4
C1: Lack of PPFL solutions for achieving satisfactory privacy-utility-efficiency balance	✓			
C2: Lack of DL-SDG-based solutions for private data sharing in cross-device scenarios		✓		
C3: Lack of DL-SDG-based solutions for sharing unstructured data in cross-device scenarios	✓	✓	✓	
C4: Lack of DL-SDG-based solutions for private sharing of vertically-partitioned data		✓		✓

Notes. C: Challenge; RQ: Research Question; ✓: Addresses challenge

1.4.1 Thesis Structure

This cumulative thesis is divided into three main parts: Part A – Introduction to the thesis, Part B – Publications, and Part C – Conclusions and Future Work. A road map of the thesis is illustrated in Figure 1.5. The contents of each part are briefly described as follows:

Part A: This part begins with the motivation and the problem statement followed by the objective and structure of the thesis (see Chapter 1). Then, the basic theoretical knowledge for understanding this thesis is provided in Chapter 2.

Part B: This part consists of four peer-reviewed contributions (Chapter 3 to Chapter 6), which include three peer-reviewed publications (Chapter 3, Chapter 4, and Chapter 6) and a peer-reviewed, revised working paper (Chapter 5). The objective of these publications is to tackle the challenges in Table 1.3 based on the idea of distributed ML. The first publication (Chapter 3) focuses on the model-to-data transmission scheme and introduces a novel LDP-based PPFL framework that achieves a better balance in privacy, utility, and efficiency. Based on the idea of distributed ML, the second publication (Chapter 4) deals with the current challenges of the data-to-model transmission scheme and presents a distributed DL-SDG-based framework for private sharing of horizontally-partitioned structured data in cross-device scenarios. Then, the third publication (Chapter 5) further extends the concept to unstructured data types and introduces a variant framework for sharing time-series data. Finally, the fourth publication in Chapter 6 focuses on the challenge of the private sharing of vertically-partitioned data, which further broadens the capabilities of distributed DL-SDG in real-life use cases.

Part C: The last part concludes the thesis. Chapter 7 presents a summary of the results of the publications. Then, the limitations of the thesis and potential directions for future research are discussed in Chapter 8. Finally, conclusions of the thesis are presented in Chapter 9.

1.4.2 Overview of Publications

The four peer-reviewed publications supporting the research goals of the thesis are embedded in Part B, as depicted in Figure 1.5. These publications are summarized in the following itemization, with each containing a brief overview of the research problem, the methodological approach adopted, and the primary contributions of the respective publication:

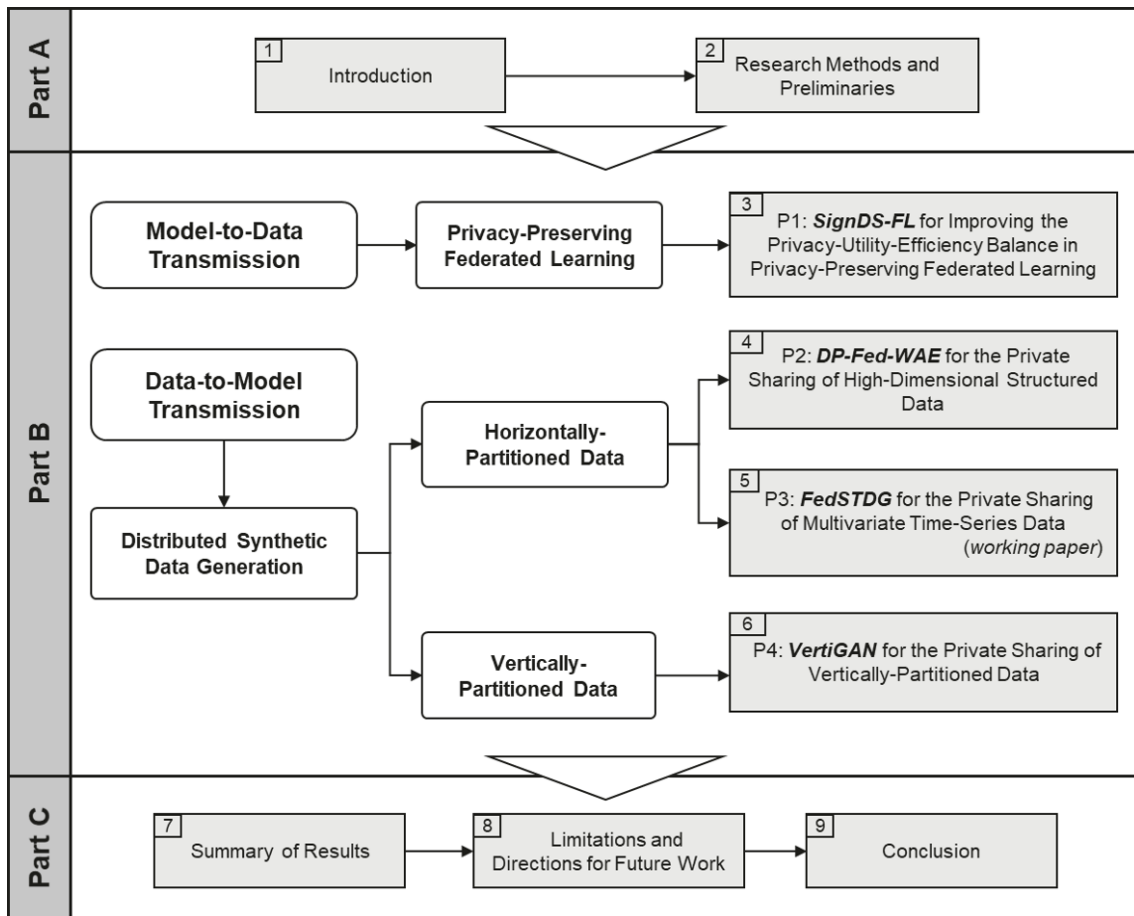


Figure 1.5 Structure of the thesis.

P1: SIGNDS-FL for improving the privacy-utility-efficiency balance in privacy-preserving federated learning. Although FL enables the efficient and private utilization of local data by moving the models to data, recent research highlights that the original FL framework is still vulnerable to privacy attacks. However, existing PPFL frameworks cannot achieve a satisfactory balance between privacy, utility, and efficiency: crypto-based solutions introduce extra communication and computation costs, CDP-based solutions do not provide local privacy protection, while LDP-based solutions suffer from significant degradation in model utility. This study aims to improve the privacy-utility-efficiency balance in LDP-FL. The main contributions of this work include (1) A novel framework called SIGNDS-FL based on private dimension selection, which achieves better model utility compared with existing LDP-FL frameworks; (2) An exponential mechanism (EM)-based multi-dimension selection (EM-MDS) algorithm that further improves model convergence and accuracy; (3) An extensive evaluation demonstrating the capability of SIGNDS-FL on a number of real-world datasets.

P2: DP-FED-WAE for the private sharing of high-dimensional structured data. In the data-to-model transmission scheme, each client must anonymize their raw local data before sharing them with the server. LDP is a widely used technique for private data sharing in cross-device scenarios (*i.e.*, without trusted data collectors). However, existing LDP-based

algorithms typically suffer from the *curse-of-dimensionality* problem: an increase in data features not only results in extra computation and communication costs but also poor data utility. On the other hand, although previous DL-SDG solutions can achieve a better utility on high-dimensional data, these algorithms cannot be directly applied to cross-device scenarios. This work mitigates the limitations of previous works following the idea of distributed ML. The main contributions include (1) A novel framework DP-FED-WAE that applies DL-SDG in a distributed fashion to support private data sharing in cross-device scenarios; (2) Adoption of a generative WAE which is privately trained under the aforementioned SIGNDS-FL framework in P1 to learn the overall distribution of all local data and used for generating high-fidelity synthetic data on the server side; (3) Extensive experimental results on a number of real-world datasets showing the advantages of the proposed framework in sharing high-dimensional data while striking a satisfactory utility-privacy balance.

P3: FEDSTDG for the private sharing of multivariate time-series data. Besides the widely explored structured (tabular) data, time-series data, with rich spatial-temporal information, have been pervasively used in finance [107], healthcare [178], and IoT [31] applications. Nevertheless, existing works (*e.g.*, [47]) are only feasible for the cross-silo scenarios, where a set of trusted data collectors already centralizes the individuals' time-series data and aims to share the data with a third-party service provider, while the cross-device scenarios *without* trusted data collectors are rarely considered. This work fills this research gap and provides a distributed DL-SDG-based solution for privately synthesizing the local time-series data *without* the requirement of the trusted data collector. The main contributions include (1) A novel framework FEDSTDG, which follows the idea in P2 but is capable of supporting private sharing of time-series data; (2) Enhancement of the SignDS-FL framework in P1 with an improved multi-dimension selection algorithm and adaptive global learning rate, which reduces the manual effort in hyperparameter tuning and achieves a better data utility; (3) An extensive evaluation on various examples of real-world time-series data for demonstrating the capability of our framework in preserving both utility and privacy.

P4: VERTIGAN the private sharing of vertically-partitioned data. Previous works on privacy-preserving data sharing usually focus on horizontally-partitioned data, where each client holds data with the same set of attributes. In contrast, the vertical setting, where clients hold different sets of attributes of the same group of data owners, is less studied. Existing DP-based solutions for sharing vertically-partitioned data only apply to low-dimensional structured data and suffer from large utility loss with an increase in dimensionality. In this study, we further extend the concept of distributed DL-SDG into private data sharing under vertical settings. The main contributions include (1) A novel framework VERTIGAN, which is comprised of one multi-output global generator and multiple local discriminators and is capable of learning the feature distribution of all data silos and generating high-utility synthetic data containing all the features; (2) A DP training process to ensure strict privacy guarantees for the local data; (3) An extensive evaluation using a number of real-world datasets to demonstrate the capability of our framework in achieving a satisfactory utility-privacy balance in the aggregation of vertically-partitioned local silo data.

Table 1.4 provides a summary of the publications embedded in Part B of this thesis, including information on the title, authors, outlet, type, and rank for each publication.

¹ACM Transactions on Intelligent Systems and Technology: <https://dl.acm.org/journal/tist>

²ACM Indexing: <https://dl.acm.org/journal/tist/indexing>

Table 1.4 Summary of embedded publications

No.	Title	Authors	Outlet	Type	Rank
P1	SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection	Jiang, Zhou, Grossklags	TIST 2022 ¹ (published)	Journal	Impact Factor 2021 ² : 10.489
P2	Privacy-Preserving High-Dimensional Data Collection with Federated Generative Autoencoder	Jiang, Zhou, Grossklags	PoPETS 2022 ³ (published)	Journal	CORE 2021 ⁴ : A
P3	Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning	Jiang, Zhou, Grossklags	PoPETS 2024 ⁵ (submitted)	Working Paper	CORE 2021 ⁴ : A
P4	Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data	Jiang, Zhang, Zhou, Grossklags	PoPETS 2023 ⁵ (published)	Journal	CORE 2021 ⁴ : A

³Proceedings on Privacy Enhancing Technologies, Volume 2022: <https://www.petsymposium.org/popets/2022/>

⁴CORE Ranking: <http://portal.core.edu.au/conf-ranks/1442/>

⁵Proceedings on Privacy Enhancing Technologies, Volume 2024: <https://www.petsymposium.org/popets/2024/>

2 Research Methods and Preliminaries

In this chapter, we aim to lay the foundation for the subsequent sections of this dissertation by presenting the necessary concepts and background information on differential privacy, federated learning, and synthetic data generation.

2.1 Differential Privacy and Its Variants

We start by providing an overview of differential privacy, covering its fundamental concepts. We will explain the original definition of differential privacy (DP), as well as the variations known as Rényi differential privacy (RDP) and local differential privacy (LDP).

2.1.1 Differential Privacy

DP [42] is a formal notion of privacy that is widely used in privacy-preserving data analysis applications. The original definition of DP is as follows:

Definition 1 ((ϵ, δ) -DP [42]). *A randomized mechanism $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ satisfies (ϵ, δ) -DP if for any two adjacent datasets $X, X' \in \mathcal{X}$ differing in one data sample and any measurable subset of outputs $Y \subseteq \mathcal{Y}$ the following statement holds:*

$$\Pr[\mathcal{M}(X) \in Y] \leq e^\epsilon \cdot \Pr[\mathcal{M}(X') \in Y] + \delta. \quad (2.1)$$

The definition ensures that the outputs of the randomization mechanism $\mathcal{M}(\cdot)$ on the neighboring datasets X and X' are very close to each other. Therefore, the adversary is unable to discern sensitive information about any particular record, hence providing stringent privacy protection for each individual in the dataset. The ϵ in Definition 1 refers to the privacy budget, namely the privacy guarantee in the worst case. The smaller ϵ , the stronger the privacy level is. The factor δ represents the probability of privacy leakage (*i.e.*, the probability of failure to hold the privacy guarantee). The smaller δ , the higher privacy. In practice, the value of δ should be negligible, particularly less than $1/|X|$, where $|X|$ is the number of records in the dataset. When $\delta = 0$, the mechanism satisfies pure ϵ -DP.

For a real-valued function f , a common approach to achieve DP guarantees is to introduce a certain amount of random noise to the outputs of f . The noise scale is carefully calibrated based on f 's sensitivity, which quantifies how much the output of a function can vary when a single individual's data is modified or removed from the dataset. The definition of L_p -sensitivity of a given function f goes as follows:

Definition 2 (L_p -sensitivity). For a real-valued function $f : X \rightarrow \mathbb{R}^d$, the L_p -sensitivity of f is defined as

$$\Delta_p f = \max_{X, X'} \|f(X) - f(X')\|_p \quad (2.2)$$

for all adjacent datasets X and X' , where $\|\cdot\|_p$ denotes the L_p -norm.

Following this paradigm, the widely used Gaussian mechanism independently adds zero-mean Gaussian noise to each dimension of the output, which is defined as follows:

Definition 3 (Gaussian mechanism [42]). For a real-valued function $f : X \rightarrow \mathbb{R}^d$ with an L_2 -sensitivity $\Delta_2 f$, the following Gaussian mechanism \mathcal{M}_σ satisfies (ϵ, δ) -DP:

$$\mathcal{M}_\sigma(x) = f(x) + \mathcal{N}(0, \sigma^2 I), \quad \text{where } \sigma = \frac{\Delta_2 f}{\epsilon} \sqrt{2 \log \frac{1.25}{\delta}}. \quad (2.3)$$

In addition, DP also holds two useful properties [42], namely robustness to *post-processing* and *sequential composition*. The *post-processing* property states that any deterministic or randomized function defined over an DP mechanism also satisfies DP. The *sequential composition* property states that interactively applying the DP mechanism on the same set of data yields an accumulated privacy cost.

Property 1 (Post-Processing). Let \mathcal{M} be an (ϵ, δ) -DP mechanism and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{M}$ is (ϵ, δ) -DP.

Property 2 (Sequential Composition). Suppose n mechanisms $\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$ respectively satisfy (ϵ, δ) -DP, and are sequentially computed on the same set of private data D , then a mechanism formed by $(\mathcal{M}_1, \dots, \mathcal{M}_n)$ satisfies $(\sum_{i=1}^n \epsilon_i, \sum_{i=1}^n \delta_i)$ -DP.

2.1.2 Rényi Differential Privacy

While the original DP defined an *upper* bound of the privacy loss, this bound can be sometimes loose, meaning that the true privacy loss is much less than the analytical result. In the case of iterative algorithms, where the overall privacy loss is the cumulative result of all the iterations, the difference between the calculated privacy bound and the actual value becomes even more pronounced. To overcome this challenge, recent works have delved into exploring novel variants of DP that offer tighter bounds on the privacy loss, particularly for iterative algorithms, while still upholding useful and meaningful privacy definitions. One of the widely used definitions is RDP [113], which uses the Rényi divergence [154] to measure the distance between two probabilities. The definition of Rényi divergence is as follows:

Definition 4 (Rényi divergence [154]). Given two probability distributions P and Q , the Rényi divergence of order $\alpha > 1$ is:

$$\mathcal{D}_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{X \sim Q} \left[\left(\frac{P(X)}{Q(X)} \right)^\alpha \right], \quad (2.4)$$

where $\mathbb{E}_{X \sim Q}$ is the expected value of X for the distribution of Q , $P(X)$ and $Q(X)$ denote the density of P and Q at X , respectively.

Based on Definition 4, the definition of RDP is stated as:

Definition 5 ($(\alpha, \epsilon(\alpha))$ -RDP [113]). *A randomized mechanism \mathcal{M} satisfies $(\alpha, \epsilon(\alpha))$ -RDP if for any two adjacent datasets X, X' differing in one data sample, the Rényi α -divergence between $\mathcal{M}(X)$ and $\mathcal{M}(X')$ satisfies*

$$\mathcal{D}_\alpha(\mathcal{M}(X) \parallel \mathcal{M}(X')) \leq \epsilon(\alpha). \quad (2.5)$$

Similar to DP, the Gaussian mechanism can also be used to achieve $(\alpha, \epsilon(\alpha))$ -RDP:

Definition 6 (Gaussian mechanism under RDP [113]). *For a real-valued function $f : X \rightarrow \mathbb{R}^d$ with an L_2 -sensitivity $\Delta_2 f$, the following Gaussian mechanism \mathcal{M}_σ satisfies $(\alpha, \epsilon(\alpha))$ -RDP:*

$$\mathcal{M}_\sigma(x) = f(x) + \mathcal{N}(0, \sigma^2 I), \quad \text{where } \sigma = \Delta_2 f \sqrt{\frac{\alpha}{2\epsilon(\alpha)}}. \quad (2.6)$$

In addition, RDP also preserves the properties of *post-processing* and *sequential composition*.

Property 3 (Post-processing under RDP). *Let \mathcal{M} be an $(\alpha, \epsilon(\alpha))$ -RDP mechanism, and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{M}$ also satisfies $(\alpha, \epsilon(\alpha))$ -RDP.*

Property 4 (Sequential Composition under RDP). *Suppose n mechanisms $\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$ respectively satisfy $(\alpha, \epsilon_i(\alpha))$ -RDP, and are sequentially computed on the same set of private data X , then a mechanism formed by $(\mathcal{M}_1, \dots, \mathcal{M}_n)$ satisfies $(\alpha, \sum_{i=1}^n \epsilon_i(\alpha))$ -RDP.*

Furthermore, the privacy guarantees under RDP can be converted to the original DP guarantees:

Lemma 1 (RDP to DP [113]). *If a mechanism \mathcal{M} satisfies $(\alpha, \epsilon(\alpha))$ -RDP, then \mathcal{M} satisfies $(\epsilon(\alpha) + \frac{\log 1/\delta}{\alpha-1}, \delta)$ -DP for any $\delta \in (0, 1)$.*

One of the major advantages of RDP is the tight composition in comparison to the original DP. For algorithms querying the same dataset multiple times, computing the accumulated privacy loss under RDP based on Property 4 and then converting to the original (ϵ, δ) -DP using Lemma 1 often yields a much lower privacy cost than directly applying the composition under the original definition. In addition, for iterative algorithms, the privacy cost of RDP can be further reduced by the subsampled mechanism:

Lemma 2 (RDP for Subsampled Mechanism [162]). *Given a dataset of a points drawn from a domain \mathcal{X} and a randomized mechanism \mathcal{M} that takes an input from \mathcal{X}^b for $b \leq a$, let the randomized algorithm $\mathcal{M} \circ \text{subsample}$ be defined as: (1) *subsample*: subsample without*

2 Research Methods and Preliminaries

replacement m data points of the dataset (sampling parameter $\zeta = b/a$), and (2) apply \mathcal{M} : a randomized algorithm taking the subsampled dataset as the input. For all integers $\alpha \geq 2$, if \mathcal{M} obeys $(\alpha, \epsilon(\alpha))$ -RDP, then the new randomized algorithm $\mathcal{M} \circ \text{subsamp1e}$ obeys $(\alpha, \epsilon'(\alpha))$ -RDP where

$$\begin{aligned} \epsilon'(\alpha) \leq & \frac{1}{\alpha - 1} \log \left(1 + \zeta^2 \binom{\alpha}{2} \min \{ 4(e^{\epsilon(2)} - 1), e^{\epsilon(2)} \min \{ 2, (e^{\epsilon(\infty)} - 1)^2 \} \} \right. \\ & \left. + \sum_{j=3}^{\alpha} \zeta^j \binom{\alpha}{j} e^{(j-1)\epsilon(j)} \min \{ 2, (e^{\epsilon(\infty)} - 1)^j \} \right). \end{aligned} \quad (2.7)$$

Intuitively, the subsampling further introduces a certain extent of uncertainty of whether a particular record is used for producing the final results, hence further amplifying the privacy protection of any individual record in the dataset.

Differentially Private Stochastic Gradient Descent

The privacy concept of privacy amplification by sampling is often used in differentially private stochastic gradient descent (DPSGD) algorithms. More specifically, in each iteration of DPSGD, a batch of b samples X_t is sampled from dataset X with a fixed subsampling probability $\zeta = b/|X|$, where b is the batch size. With the current model parameter θ_t and the loss function \mathcal{L} , the gradient of each sample $x_i \in X_t$ is computed as:

$$g_t(x_i) = \nabla_{\theta_t} \mathcal{L}(\theta_t, x_i). \quad (2.8)$$

Then, the DPSGD algorithm clips each per-example gradient based on a fixed L_2 -norm clipping bound C :

$$\bar{g}_t(x_i) = \text{CLIP}(g_t(x_i); C) = g_t(x_i) / \max \left(1, \frac{\|g_t(x_i)\|_2}{C} \right), \quad (2.9)$$

which ensures that for any two neighboring datasets, the L_2 -sensitivity of each query given by $\sum_{x_i \in X_t} \bar{g}_t(x_i)$ is bounded by C . Finally, the clipped gradients are summed and perturbed with Gaussian noise with a scale determined by C , namely:

$$\tilde{g}_g = \frac{1}{b} \left(\sum_{x_i \in X_t} \bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{1}) \right), \quad (2.10)$$

where σ is the noise multiplier determined by the privacy budget. Given the overall privacy budget, the total number of iterations, and the batch size b , the smallest σ can be computed using Definition 6, Property 4, and Lemma 2. Moreover, since the noise gradient \tilde{g}_t in each iteration satisfies DP guarantees, the final model also satisfies DP due to Property 3.

2.1.3 Local Differential Privacy

Finally, DP and RDP are usually applied in centralized settings where the data have already been collected by a trusted server. However, in local settings, we aim to ensure that each

client's local data will not be accessed by the server. Hence, the definition of LDP has been proposed [82], which provides strong local privacy guarantees for each user. The definition is as follows:

Definition 7 (LDP [82]). *A randomized mechanism $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ satisfies ϵ -LDP if and only if for any two inputs $x, x' \in \mathcal{X}$ and for any output $y \in \mathcal{Y}$:*

$$\Pr[\mathcal{M}(x) = y] \leq e^\epsilon \cdot \Pr[\mathcal{M}(x') = y]. \quad (2.11)$$

Similarly, LDP also holds the properties of robustness to *post-processing* and *sequential composition*.

Property 5 (Post-processing under LDP). *Let \mathcal{M} be an ϵ -LDP mechanism and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{M}$ is ϵ -LDP.*

Property 6 (Sequential Composition under LDP). *Suppose n mechanisms $\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$ respectively satisfy ϵ_i -LDP, and are sequentially computed on the same set of private data D , then a mechanism formed by $(\mathcal{M}_1, \dots, \mathcal{M}_n)$ satisfies $(\sum_{i=1}^n \epsilon_i)$ -LDP.*

2.2 Federated Learning

Over the past few decades, there has been a significant surge in the utilization of AI services in our everyday lives. Building effective AI services typically involves training ML and deep learning (DL) models with a substantial amount of user data. However, this data is often scattered and distributed across various local devices and entities. In the conventional approach of centralized learning, the service provider would gather data from all these local parties and train the ML model on a cloud server. Nevertheless, the advent of strict data regulations, such as GDPR, has limited the service provider's ability to collect raw user data. Consequently, this poses a considerable challenge to the traditional centralized paradigm. In this context, FL emerges as a promising solution that empowers service providers to effectively utilize local data without violating user privacy. By embracing a distributed learning paradigm, FL enables the training of ML models without the need to collect raw local data. The fundamental principle of FL is to distribute the computation tasks to the local side. In essence, AI models are collaboratively trained by multiple "local parties" (which can be different local devices or different organizations) under the coordination of a central server. Throughout the training process, only model updates or specific intermediate results are shared, while the individual local raw data remains on the local side.

According to the partitioning strategies of local data, existing FL solutions can be generally categorized into two main groups, respectively for horizontally-partitioned data and vertically-partitioned data.

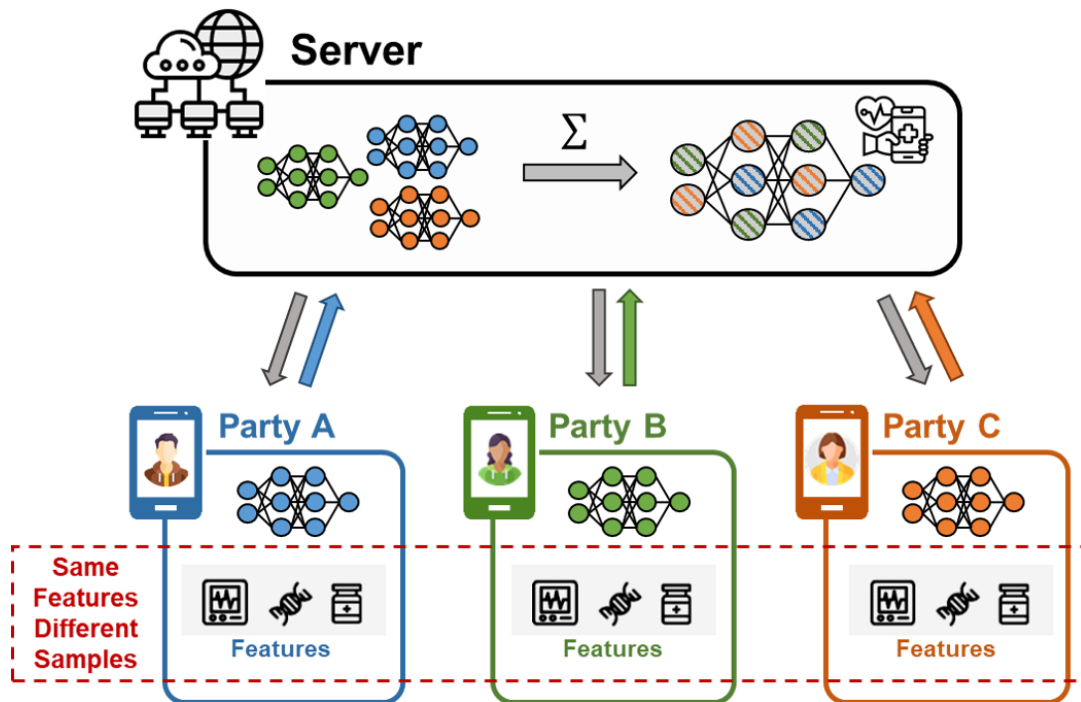


Figure 2.1 Example of model training on horizontally-partitioned data. Each local party holds the same set of features of a different set of samples. The server uses FL to increase the number of training samples to improve model performance. Here, a healthcare service seeks to use the data from different local devices to train a model for analyzing users' health conditions.

2.2.1 Federated Learning for Horizontally-Partitioned Data

Horizontally-partitioned data refers to scenarios where the data across multiple local parties are divided by samples. Each party possesses data from a distinct set of samples while sharing the same set of features. The server uses FL to increase the number of training samples to improve model performance. An example of FL in the horizontal setting is shown in Figure 2.1. In this particular case, a healthcare service, referred to as the *server*, seeks to train a model for analyzing users' health conditions. Each local party represents a user's smart device, such as smartphones, smart watches, etc., which collects and stores users' health data. To this end, the server first initializes a global model. Then, in each FL training round, the server distributes the current global model to all the local parties. Each party proceeds to train the model using its data and subsequently sends the model updates back to the server. The server aggregates all the local model updates to update the global model. This exchange process is repeated over several rounds until the global model achieves satisfactory performance.

2.2.2 Federated Learning for Vertically-Partitioned Data

On the other hand, vertically-partitioned data entails dividing the features or attributes of the data across multiple local parties (usually distinct organizations). Each party possesses a

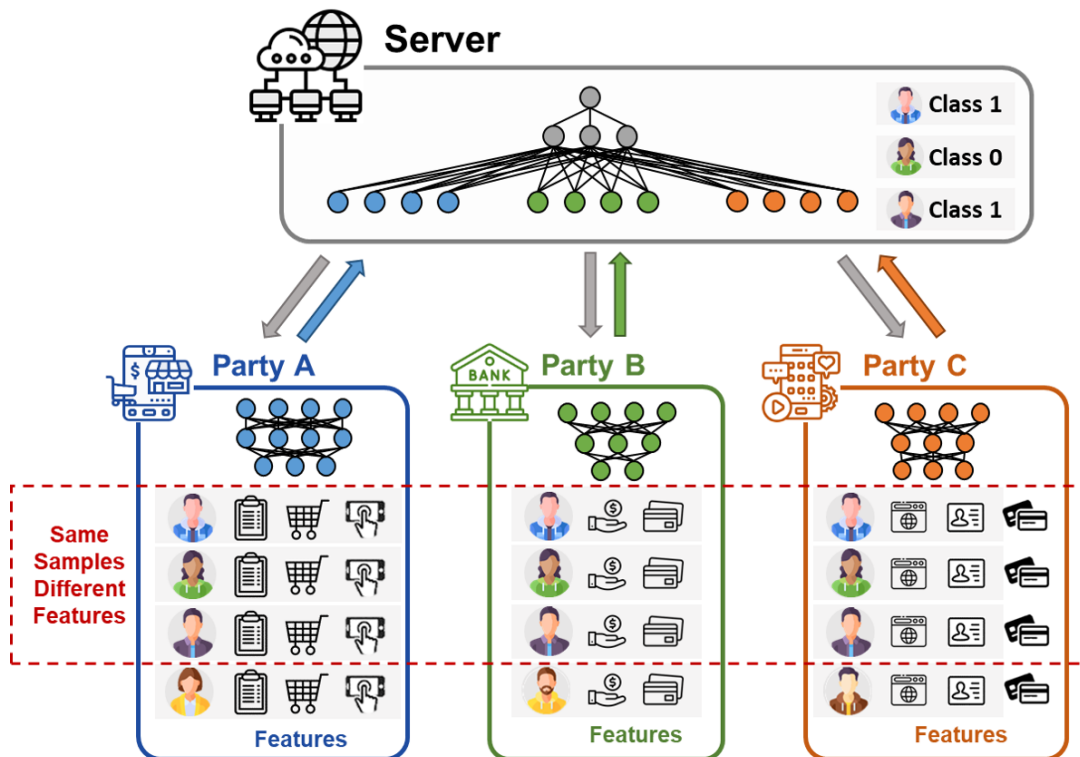


Figure 2.2 Example of model training on vertically-partitioned data, where each local party holds a different set of features of the same set of samples. The server uses FL to increase the number of features to improve model performance. Here, a credit card company seeks to use the data from an E-commerce company, a bank, and an online media company to train a credit score prediction model.

different set of features or attributes of the same group of users. The server wants to combine the features of all the parties to enhance the model's performance. To exemplify, consider the scenario depicted in Figure 2.2, where a credit card company collaborates with three parties: an E-commerce company, a bank, and an online media company. The goal of the server is to utilize features from diverse sources to create more precise user profiles and train a more accurate model for credit score prediction.

Nevertheless, as each party holds a different set of features, training a common global model on the local side, as done in the horizontal setting, is no longer feasible. Thus, in the vertical setting, each party holds a *distinct* local model (commonly known as the *bottom model*). In addition, a sample-wise data alignment should be applied before the training starts to ensure that the selected training records from each party belong to the same group of users. Then, during training, each party inputs their local data to the local model, and shares the model outputs, or so-called *embeddings*, with the server. On the server side, an additional model, known as the *top model*, is deployed. It takes the concatenation of all the local embeddings as input and finally outputs the predicted results. Based on the prediction loss, the gradient of the *top model* is calculated, which is then backpropagated to each local party to compute the gradient of the corresponding *bottom model*. The training process will be repeated for several iterations until the system achieves a satisfactory prediction accuracy.

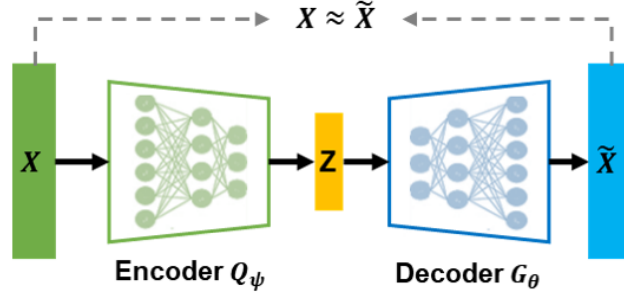


Figure 2.3 Example structure of an AE model.

2.3 Synthetic Data Generation

With the rapid advancement in machine learning and deep learning technologies, deep generative models are gaining immense popularity for their ability to capture real data distribution and generate high-fidelity synthetic data. The produced synthetic data can be used for various purposes, such as privacy-preserving data sharing and pre-training machine learning models. The objective of this dissertation is to delve into the exploration of two significant classes of deep generative models, namely AE and GAN, with a specific focus on their role in enabling the private utilization of local data.

2.3.1 Autoencoders

The AE [133] is a class of neural networks that are used to learn efficient and compressed feature representations in an unsupervised manner. An AE model, as shown in Figure 2.3, generally consists of two main parts: an *encoder* Q_ψ and a *decoder* G_θ . The encoder compresses the original high-dimensional input $x \sim P_x$ into the low-dimensional latent feature $z = Q_\psi(x)$ and the decoder maps z to the reconstructed output $\tilde{x} = G_\theta(z)$, which is of the same shape as x . The goal of training is to find an optimized pair of encoder and decoder, which minimizes the distance between x and $\tilde{x} = G_\theta(Q_\psi(x))$, namely

$$\mathcal{L}_{AE} = \mathbb{E}_{x \sim P_x} [\mathcal{L}_{rec}(x, G_\theta(Q_\psi(x)))], \quad (2.12)$$

where $\mathcal{L}_{rec}(\cdot, \cdot)$ is a metric for featuring the difference between two vectors. For instance, we can use the mean squared error (MSE) to measure the distance between continuous input vectors and the categorical cross-entropy (CE) for discrete input vectors. Building upon this, several variants of AE have been proposed for synthetic data generation, such as the variational autoencoder (VAE) [85] and WAE [144]. For example, an illustration of the WAE model is presented in Figure 2.4. The model adheres to the encoder-decoder architecture of AE. However, to augment its capabilities, an additional penalty term is introduced to enforce compliance of the latent space features with a predetermined prior distribution, commonly adopting the standard Gaussian distribution $\mathcal{N}(0, I)$. During the training process, the primary objective is to determine an optimal set of parameters that effectively minimizes the distance between the inputs and outputs while constraining the latent space features to follow the

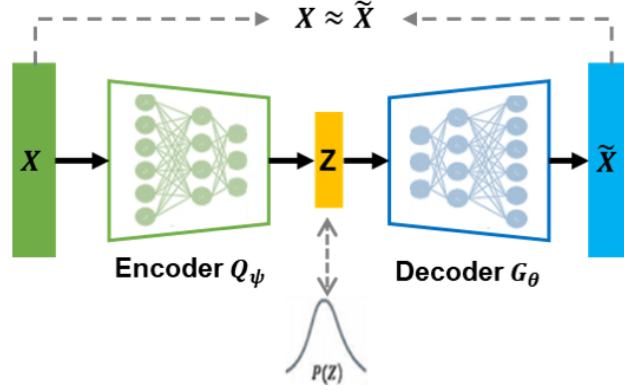


Figure 2.4 Example structure of an WAE model.

predefined prior distribution. Hence, the objective function of a WAE model can then be formulated as:

$$\mathcal{L}_{WAE} = \mathbb{E}_{x \sim P_x} [\mathcal{L}_{rec}(x, G_\theta(Q_\psi(x)))] + \lambda \cdot \mathcal{L}_{lat}(q_z, p_z), \quad (2.13)$$

where the latent space distance \mathcal{L}_{lat} is the MMD between the real latent distribution q_Z and the standard Gaussian distribution p_Z , and λ is a weight to balance both distances, which is usually set to 1 by default. Additionally, the MMD is computed as follows: given a batch of data sampled from the two distributions, i.e., $\{q^1, \dots, q^N\} \sim q_z$ and $\{p^1, \dots, p^N\} \sim p_z$, $\mathcal{L}_{lat}(q_z, p_z)$ can be empirically estimated as:

$$\mathcal{L}_{lat}(q_z, p_z) = \frac{1}{N(N-1)} \sum_{i \neq j} \kappa(p^i, p^j) - \frac{2}{N^2} \sum_{i,j} \kappa(p^i, q^j) + \frac{1}{N(N-1)} \sum_{i \neq j} \kappa(q^i, q^j), \quad (2.14)$$

where $\mathcal{K}(x, y) = \frac{\kappa}{\kappa + \|x - y\|_2^2}$. Given d_z as the dimension of latent layer and σ_z as the scale of the prior distribution, $\kappa = 2d_z\sigma_z^2$.

Furthermore, several previous works integrate advanced layers to the original AE model, to enhance their modeling capabilities for different types of data. For instance, [65, 28] apply convolutional layers into AE to extract intricate patterns within images, whereas [9, 56] adopt recurrent layers to capture spatiotemporal information in time-series data. The incorporation of advanced layers has significantly expanded their applicability across a wide range of data types and facilitated the modeling of complex structures within these datasets. In this dissertation, we will present an example of utilizing a modified AE model for the collection of time-series data.

2.3.2 Generative Adversarial Network

Besides AE models, GAN [53] is another class of unsupervised learning algorithms that have been extensively studied in the last decade due to its strong capability to generate high-fidelity synthetic data.

A GAN model typically consists of two components: a generator G and a discriminator D , as presented in Figure 2.5. The generator is responsible for producing synthetic data \tilde{x} by taking a random noise vector z from a certain latent distribution P_z and mapping it to the

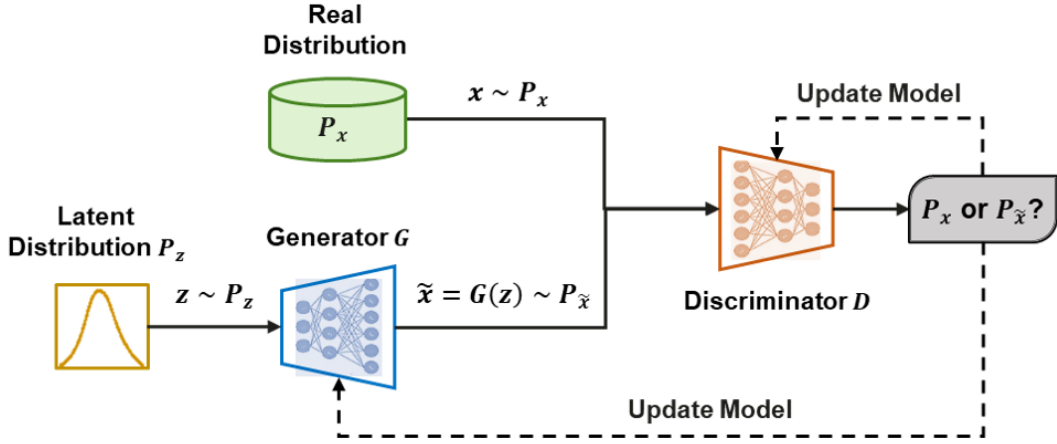


Figure 2.5 Example structure of a GAN model.

data space. On the other hand, the discriminator attempts to distinguish between real data samples x drawn from the real distribution P_x and synthetic data samples \tilde{x} generated by the generator using the latent distribution P_z , which is usually the standard Gaussian distribution $\mathcal{N}(0, I)$. This interaction between the generator and discriminator can be viewed as a two-player min-max game. The generator aims to improve the quality of the synthetic data to fool the discriminator, while the discriminator tries to discriminate between real and synthetic data with high accuracy. Through this adversarial training process, both models are trained simultaneously, constantly improving their abilities until a dynamic equilibrium is reached. The ultimate goal of a GAN is to approximate the real data distribution P_x with the synthetic data distribution $P_{\tilde{x}}$ in such a way that the discriminator cannot accurately distinguish between the two. The problem can be formulated with the following objective [53]:

$$\mathcal{L}_{GAN} = \mathbb{E}_{x \sim P_x} [\log D(x)] + \mathbb{E}_{\tilde{x} \sim P_{\tilde{x}}} [\log(1 - D(\tilde{x}))], \quad (2.15)$$

where P_x is the distribution of real data and $P_{\tilde{x}}$ is the distribution of synthetic data $\tilde{x} = G(z)$ with $z \sim P_z$.

By utilizing different generator and discriminator structures, GANs have been adjusted to generate various types of synthetic data such as tabular data [122, 171], images [80, 81], and time-series data [177, 70]. Nevertheless, the original GAN models usually suffer from problems such as training instability and failure to converge. Therefore, some other works proposed to modify the loss function to improve model convergence. The Wasserstein GAN (WGAN) [6, 164] is one of the well-known improved GANs. In comparison with the original loss function, WGAN proposed in [164] uses the Wasserstein-1 distance with an additional gradient norm penalty to achieve Lipschitz continuity. Given the real data x , the input noise $z \sim P_z$ and the synthetic data $\tilde{x} = G(z)$, the gradient penalty term can be written as

$$(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2, \quad \text{where } \hat{x} = \mu x + (1 - \mu)\tilde{x}. \quad (2.16)$$

Here \hat{x} is a weighted average between the real and synthetic data and $\mu \sim \mathbf{U}(0, 1)$ is a randomly sampled weight. Thus, the loss function for the generator and discriminator is formulated as follows:

$$\mathcal{L}_G = D(\tilde{x}) = D(G(z)), \quad (2.17)$$

$$\mathcal{L}_D = D(x) - D(\tilde{x}) + \lambda(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2, \quad (2.18)$$

where λ is the weight for the gradient penalty. By combining Equation (2.17) and Equation (2.18), the loss function of WGAN can be derived as:

$$\mathcal{L}_{WGAN} = \mathbb{E}_{\tilde{x} \sim P_{\tilde{x}}} [D(\tilde{x})] - \mathbb{E}_{x \sim P_x} [D(x)] + \lambda \mathbb{E}_{\hat{x} \sim P_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2]. \quad (2.19)$$

Part II

Peer-Reviewed Publications¹

¹In this part, the original publications are slightly modified, including the unification of format, the organization of sections, and the correction of spelling errors and grammar issues. Furthermore, the references included in each research study have been integrated and are presented at the end of this dissertation. The published papers in the original format can be found in the Appendix.

3 SIGNDS-FL: Horizontal Federated Learning with Local Differential Privacy

Table 3.1 Bibliographic details for P1

Title	SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection
Authors	Xue Jiang ^{1,2} (xue.jiang@tum.de) Xuebing Zhou ² (xuebing.zhou@huawei.com) Jens Grossklags ¹ (jens.grossklags@in.tum.de) ¹ Technical University of Munich (TUM), Boltzmannstraße 3, 85748 Garching, Germany ² Huawei Munich Research Center, Riesstraße 25, 80992 Munich, Germany
Type	Journal
Outlet	ACM Transactions on Intelligent Systems and Technology ³
Ranking	Impact Factor in 2022 ⁴ : 10.489 Rank in the field of Information Systems ⁴ : 36/163 Rank in the field of Artificial Intelligence ⁴ : 13/145
Status	Published
Citation	Jiang, X., Zhou, X., & Grossklags, J. (2022). SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection. ACM Transactions on Intelligent Systems and Technology (TIST), 13(5), 1-22.
Copyright	This work is published under a Creative Commons Attribution-NonCommercial International 4.0 (CC BY-NC 4.0) License ⁵ .
Author Contributions	Xue Jiang developed the core idea for the paper and took primary responsibility for designing the methodology, acquiring data, implementing experiments, evaluating results, and drafting the manuscript. Xuebing Zhou and Jens Grossklags provided valuable feedback and suggestions on the methodology and experiments, and assisted in reviewing and enhancing the manuscript.

³<https://dl.acm.org/journal/tist>

⁴<https://dl.acm.org/journal/tist/indexing>

⁵<https://creativecommons.org/licenses/by-nc/4.0/>

3.1 Abstract

In this chapter, we first focus on addressing the data silo problem using model transmission. As discussed in Section 1.2.1, FL, as one of the representative solutions, has attracted increasing attention due to achievements in computational efficiency and privacy preservation. However, recent research highlights that the original FL framework may still reveal sensitive information of clients' local data from the exchanged local updates and the global model parameters. Local Differential Privacy (LDP), as a rigorous definition of privacy, has been applied to FL to provide formal privacy guarantees and prevent potential privacy leakage. However, previous LDP-FL solutions suffer from considerable utility loss with an increase of model dimensionality. Recent work [99] proposed a two-stage framework that mitigates the dimension-dependency problem by first selecting one "important" dimension for each local update and then perturbing the dimension value to construct the sparse privatized update. However, the framework may still suffer from utility loss because of the insufficient per-stage privacy budget and slow model convergence.

In this work, we propose an improved framework, SIGNDS-FL, which shares the concept of dimension selection with [99], but saves the privacy cost for the value perturbation stage by assigning random sign values to the selected dimensions. Besides using the single-dimension selection algorithms in [99], we propose an Exponential Mechanism-based Multi-Dimension Selection (EM-MDS) algorithm that further improves model convergence and accuracy. We evaluate the framework on a number of real-world datasets with both logistic regression models and deep neural networks and show that our framework significantly outperforms the previous LDP-FL solutions and enjoys an advanced utility-privacy balance.

3.2 Introduction

machine learning (ML) has been widely applied in solving societal challenges in recent years. Traditional centralized learning mechanisms gather all the client data for model training and therefore suffer from high computational complexity and privacy issues. Due to these problems, federated learning (FL) [108] has been proposed, where the ML models are jointly trained by multiple local devices (also referred to as *clients* or *users*) under the coordination of a central server. As training tasks are distributed to local devices and clients' private data are never uploaded to the server, the framework enjoys distinctive advantages in both computational efficiency and privacy protection. FL is increasingly used in real-life scenarios such as health care [96], recommendation systems [174], and other mobile edge computing applications [97].

Although FL enjoys significant privacy benefits in comparison to centralized learning, recent works have demonstrated that FL is still vulnerable to various privacy attacks such as reconstruction attacks [186] and membership inference attacks [115], as the exchanged local updates and the global model parameters may reveal sensitive information of the private local data. Motivated by this, an increasing number of privacy-preserving federated learning (PPFL) frameworks with privacy-enhancing techniques such as homomorphic encryption (HE) [32], secure multi-party computation (SMC) [176] and differential privacy (DP) [42] have been proposed, aiming to prevent potential privacy leakage in FL. However, the cryptography-based

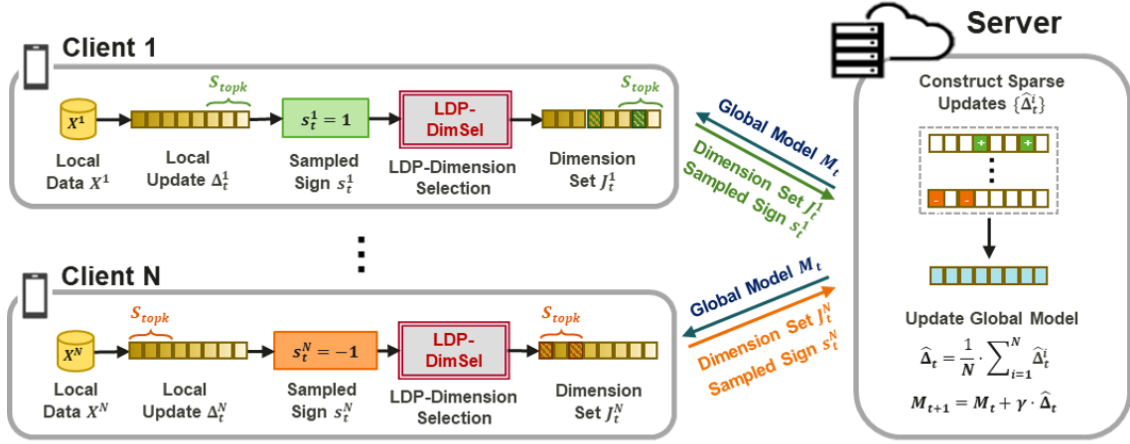


Figure 3.1 Overview of the SIGN-FL framework. At each global round, the server broadcasts the current global model to the local side. Each client trains the global model with local data and computes the local model update. Then, the client randomly samples a sign value and builds the top- k dimension set: if the sign value equals 1, the top- k set is built with the dimensions of the k largest update values; otherwise, it is built with the dimensions of the k smallest update values. An LDP-based dimension selection algorithm is then applied to select a set of “important” dimensions. The sampled sign value and the selected dimension set will be sent to the server. The server will construct sparse privatized local updates by assigning the sign value to the corresponding selected dimensions and finally use them to update the global model.

solutions may not be practical to large-scale FL scenarios due to the massive additional communication and computation costs, as discussed in [95].

In comparison, DP-FL frameworks use randomization algorithms (*e.g.*, injecting random noise) to perturb the model updates or the model parameters and do not impose significant additional communication and computation costs. Moreover, the randomization algorithms follow a strict DP definition and can effectively prevent attackers from inferring information from local data. The DP-FL frameworks in [110, 7] add Gaussian noise on the server side in order to protect the privacy of the global model. However, they assume the presence of a trusted server. Thus, a more practical solution is to apply local differential privacy (LDP) to FL, which perturbs the local updates before sending them to the server. Nevertheless, as discussed in [79], it is very challenging for LDP-FL frameworks to achieve a satisfactory privacy-utility balance, especially for high-dimensional models. For instance, [136] proposes the first LDP-FL framework using the sparse vector technique. However, its DP guarantee is per dimension and is less effective for large models [121]. Later works, *i.e.*, [40, 158, 184] adopted LDP mean estimation algorithms to perturb the local updates. However, the injected noise in these solutions is in essence proportional to the model dimensionality, making them only applicable to simple ML models. A recent work proposed FEDSEL [99], a *two-stage* LDP-FL framework that includes a *dimension selection (DS)* stage and a *value perturbation (VP)* stage. Given an original local update vector, the DS stage first builds a top- k dimension set containing the dimensions of the k largest *absolute* update values and privately selects one “important” dimension from the top- k set. Then, in the VP stage, the value of the selected dimension is perturbed via the LDP algorithms in [40, 158] and used to construct a sparse privatized local update.

Although FEDSEL [99] mitigates the dimension-dependency problem in previous works, the framework may still suffer from utility loss due to the following reasons. First, since the privacy budget is consumed by both stages, for high-privacy regimes (when the privacy budget is small), each stage may obtain only an insufficient privacy budget and cause large randomness. Moreover, the framework only selects one dimension for each local update, which may lead to slow model convergence, especially for high-dimensional models. In this work, we address the above challenges from two perspectives. First, we propose a novel LDP-FL framework, SIGNDS-FL (as shown in Figure 3.1), which aims to mitigate the problem of an insufficient per-stage privacy budget in the high-privacy regime. The main idea is to save the privacy budget for the VP stage in [99] by building the top- k set according to the *real* local update values and assigning sign values instead of the perturbed dimension values to the selected dimensions. The sign values are randomly sampled by each client on the local side, which is used for determining the top- k set and constructing the privatized local update. Moreover, besides adopting the single-dimension selection algorithms as in [99], we give the first attempt to explore multi-dimension selection algorithms. To select h dimensions under the ϵ -LDP guarantee, a naive approach is to independently perform the single-dimension selection h times. However, the privacy budget for selecting each dimension should then be less than ϵ/h , which will lead to a significant degradation in model utility. Inspired by the exponential mechanism (EM) [111] and its extension in frequency estimation on set-valued data [159], we propose an Exponential Mechanism-based Multi-Dimension Selection (EM-MDS) algorithm. In comparison to the naive approach, EM-MDS utilizes the privacy budget more efficiently and helps improve model convergence and accuracy.¹

Our major contributions can be summarized as follows:

- We propose SIGNDS-FL, a novel LDP-FL framework. Different from [99], we propose to build the top- k dimension set according to *real* update values and construct the privatized updates based on sign values randomly sampled by the clients. Our solution saves the privacy budget for the VP stage in [99] and achieves better model accuracy under the same privacy level.
- We further extend the single-dimension selection strategy in [99] to multi-dimension selection and propose EM-MDS, a novel algorithm based on the idea of the *exponential mechanism* [111]. The algorithm can more effectively utilize the privacy budget in comparison to the naive approach and achieves better model utility.
- We evaluate the performance of our framework on a number of real-world datasets and ML models and compare the results with previous works. For the simple tasks of training logistic regression models on structured datasets, our framework achieves an accuracy loss of only 1% \sim 2% under a privacy level $\epsilon \geq 4$ in comparison to a 5% \sim 15% decrease of accuracy for the baseline methods. For the complex tasks that train deep neural networks on image datasets, the accuracy loss of our framework under a privacy level $\epsilon \geq 8$ is also less than 8% and at best only 2%. Extensive experimental results demonstrate that our framework significantly outperforms the previous LDP-FL solutions and enjoys an advanced utility-privacy balance.

¹The implementation code can be found at: https://gitee.com/mindspore/mindspore/blob/r1.6/mindspore/lite/java/java/fl_client/src/main/java/com/mindspore/flclient/SecureProtocol.java

The remainder of the chapter is organized as follows. In Section 3.4, we introduce the system model and the proposed methodology in detail. The evaluation experiments and results are then presented in Section 3.5. In Section 3.6, we discuss the potential applications of the proposed method. Finally, conclusions and avenues for future work are presented in Section 3.7.

3.3 Related Work

In recent years, DP [42] has been widely used as a strict criterion for privacy protection in data analysis [41], data publishing [179], and machine learning [22, 1]. Moreover, an increasing number of studies also incorporate DP into FL in order to reduce potential privacy leakage and offer privacy guarantees for the framework. Previous works by [110] and [7] proposed to add Gaussian noise on the server side to protect the privacy of the global model. However, such solutions cannot prevent privacy leakage from clients' local updates. Follow-up works further adopted crypto-based algorithms to strengthen local privacy. For instance, [69] presented a framework that incorporates DP with SMC while [147] introduced a hybrid solution with DP and HE. Yet, the solutions require extra communication and computation costs during the key-distribution phase and cannot be applied to large-scale scenarios.

Due to the privacy and efficiency issues in the abovementioned solutions, a more practical approach is to use LDP [82] to privatize the original local updates before sending them to the server. Considering that the local updates are numerical vectors and the global aggregation on the server side computes the average of all the local updates, a natural way is to use the LDP mean estimation algorithms. Given a local update vector $\Delta \in [-1, 1]^d$, a naive solution is to independently perturb each dimension using the Laplace mechanism [42], *i.e.*, $\hat{\Delta} = \Delta + \text{Lap}(\frac{2d}{\epsilon})$. However, the noise scale is essentially linear to the dimension d and will result in a significant utility loss for high-dimensional models. In order to reduce the noise scale, [40] further proposed a method based on randomized response (RR) [163] (referred to as DM) that maps each original value to two possible constants $\{-B, B\}$, which are determined by d and ϵ . Although the algorithm achieves a lower noise scale, it is relatively sophisticated and does not achieve ϵ -LDP when d is even [119]. Based on DM, [158] further proposed a piecewise mechanism (PM), which returns a sparse privatized vector with at most h dimensional values, where $h = \max\{1, \min\{d, \lfloor \frac{\epsilon}{2.5} \rfloor\}\}$. More specifically, for each selected dimension $\Delta[j]$, the algorithm first computes the noised results x_j and let the $\hat{\Delta}[j] = \frac{d}{h}x_j$. In this case, the communication cost is reduced to $O(h)$ in comparison with $O(d)$ in DM. The authors also proposed a hybrid mechanism (HM), which combines DM and PM to achieve an optimized worst-case variance. Additionally, a follow-up work by [184] proposed an improved PM-SUB algorithm, which further reduced the variance when ϵ is large. Although [158] and [184] increase the per-dimension privacy budget to ϵ/h by only reporting the value of h random dimensions, the perturbed values are finally enlarged by d/h for an unbiased estimation, which increases the injected noise at the same time. Based on the above LDP mean estimation-based solutions, a recent work by [99] further proposed FEDSEL, a *two-stage* LDP-FL framework that includes a DS stage and a VP stage. In the DS stage, LDP-based dimension selection algorithms are applied to select one "important" dimension from the top- k dimension set (*i.e.*, the set of dimensions with the k largest *absolute* update values); in the VP stage, the value of the selected dimension is perturbed via the LDP algorithms in

[40, 158]. Finally, a sparse privatized local update is constructed and returned to the server. Although [99] mitigates the *dimension-dependency* problem apparent in previous works, the privacy budget is still consumed by two stages. When the privacy budget is small, each stage may obtain only an insufficient privacy budget and cause large randomness. Also, only selecting one dimension for each local update may lead to slow model convergence, especially for high-dimensional models.

In addition to solutions following a strict LDP definition, alternative notions of LDP have also been investigated. For instance, [13] introduced minimax differential privacy, which relaxes *local privacy* by limiting the prior knowledge of the attackers, thus allowing the algorithm to be performed under a much larger privacy budget (e.g., $\epsilon > 500$). Similarly, [148] proposed the ϵ -LDPFed framework based on Condensed LDP (ϵ -CLDP) [54]. The algorithm defines ϵ as the privacy cost under CLDP and requires $\epsilon \ll \epsilon$ to ensure meaningful privacy protection. However, they adopted $\epsilon = 1$ in the evaluation experiments, which is equivalent to using a very large ϵ under the original LDP definition and results in weak privacy protection. As we mainly focus on LDP-FL under the original LDP definition, the above two solutions are out of scope for this paper.

3.4 SIGNDS-FL Framework

Although the LDP-FL framework in [99] shows considerable performance improvements in comparison to previous works, it may still suffer from utility loss because of the insufficient per-stage privacy budget and slow model convergence. In this section, we will present our solution to address these limitations. To start with, we will describe our system model. Then, we will introduce our enhanced LDP-FL framework SIGNDS-FL, which aims to further reduce the privacy cost in [99] and mitigate the utility loss in high-privacy regimes. Finally, we will present EM-MDS, a new privacy-preserving multi-dimension selection algorithm that helps improve model convergence.

3.4.1 System Model

A common FL scenario is considered in this work, where a machine learning model M is jointly trained by a number of local clients under the coordination of a central server. Each client i holds a local dataset X^i that contains the client's private personal data. At each global round t , the server selects a group of N clients and distributes the current global model M_t . Each client i in the group trains the global model for several gradient descent steps with his local data X^i and obtains the local model M_t^i . Then, the client computes the local update $\Delta_t^i = M_t^i - M_t$ and sends Δ_t^i back to the server. On the server side, all the local updates are aggregated and averaged, which is then used to update the global model as below:

$$\Delta_t = \frac{1}{N} \sum_{i=1}^N \Delta_t^i, \quad M_{t+1} = M_t + \Delta_t. \quad (3.1)$$

The updated global model M_{t+1} is distributed again to local clients to start the next round.

Since the raw local updates Δ_t^i are derived based on clients' personal data, directly uploading Δ_t^i to the server may reveal private information of raw local data. Here, the server is assumed to be *honest-but-curious*, which follows the system protocols but tries to infer sensitive information of local users from Δ_t^i . Thus, to prevent such privacy leakage, the clients use a randomization algorithm \mathcal{M} to perturb the local updates and send the privatized updates to the server. The randomization algorithm \mathcal{M} follows the strict LDP definition, ensuring that the server lacks access to the original local updates and providing formal privacy guarantees for clients' local data.

3.4.2 General Workflow of SIGNDS-FL

Motivated by the limitations of [99], we propose SIGNDS-FL, an improved LDP-FL framework. The main idea is to substitute the VP stage in [99] by assigning a constant value to the selected dimensions in order to save privacy costs. In this way, with the same privacy level, the proposed method can achieve less randomness and thus higher accuracy in the dimension selection stage. However, since the parameter values may have different signs, assigning the same constant value to all the selected dimensions may result in a significant change in the gradient descent direction. To address the problem, the SIGNDS-FL framework adopts an extra sign variable $s \in \{-1, 1\}$, which is used for dimension selection and for the construction of the sparse privatized local updates. The idea is inspired by the SIGNSGD algorithm proposed by [12]. The authors quantized each local update value to its sign value to reduce the communication cost and proved that the algorithm can still enjoy a satisfying convergence rate. Thus, in our algorithm, each client randomly samples a sign value s on the local side. If $s = 1$, the dimension indices of the k *largest* values in the local update are used to build the top- k dimension set and perform the private dimension selection process. The selected dimensions will then be assigned with $s = 1$ to construct the sparse privatized local update. Intuitively, the k *largest* values are highly likely to be larger than zero. Therefore, assigning the positive sign value to the selected dimension will not cause much difference in the model update direction, which mitigates the impact on model accuracy. Similarly, when $s = -1$, the dimension indices of the k *smallest* values are used for building the top- k set. The selected dimensions are assigned with the negative sign value for constructing the sparse privatized local update.

The general training workflow of SIGNDS-FL is presented in Algorithm 1. At each global round t , the server selects a group of N clients and broadcasts the current global model M_t to the clients (Lines 2-4). On the local side, the client i trains the global model for several gradient descent epochs with his local data X^i and computes the local update Δ_t^i (Lines 13-17). Then, the client sorts Δ_t^i by *real* update values, randomly samples a sign value $s_t^i \in \{-1, 1\}$ with equal probability, and builds the top- k dimension set S_{topk} following the idea described above (Lines 18-23). Given the privacy budget ϵ , different LDP dimension selection algorithms are applied to privately select a set of dimensions J_t^i (Line 24), which, together with the sampled sign s_t^i , will be returned to the server. The server then constructs the sparse privatized local update $\hat{\Delta}_t^i$ by assigning the sign value s_t^i to all the dimensions contained in J_t^i (Lines 6-7). Finally, all the sparse local updates are aggregated and used to update the global model with a global learning rate γ (Lines 9-10). The updated global model M_{t+1} is distributed again to local clients to start the next round.

Algorithm 1: SIGNDS-FL

Input: $M_1 \in \mathbb{R}^d$: initial global model; T : global aggregation rounds; N : number of per-round clients; E : number of local epochs; η : local learning rate; d : local update size; k : top- k set size; h : output dimension set size; γ : global learning rate; ϵ : privacy budget

Output: Trained model M

Server executes:

```

1: for global round  $t = 1, \dots, T$  do
2:   Randomly select a group of  $N$  clients
3:   for client  $i = 1, \dots, N$  in parallel do
4:     Broadcast current global model  $M_t$ 
5:     Receive the dimension set and sign  $J_t^i, s_t^i = \mathbf{LocalUpdate}(M_t, E, \eta, \epsilon, k, h)$ 
6:     Initialize the sparse privatized local update  $\hat{\Delta}_t^i = [0, \dots, 0]^d$ 
7:     For  $j \in \{1, \dots, d\}$ , if  $j \in J_t^i$ , set  $\hat{\Delta}_t^i[j] = s_t^i$ 
8:   end for
9:   Aggregate all the sparse local updates:  $\hat{\Delta}_t = \frac{1}{N} \sum_{i=1}^N \hat{\Delta}_t^i$ 
10:  Update global model  $M_{t+1} = M_t + \gamma \cdot \hat{\Delta}_t$ 
11: end for
12: Return Global model  $M = M_{T+1}$ 

```

LocalUpdate($M_t, E, \eta, \epsilon, k, h$):

// Run on the client side

```

13: Initialize local model  $M_t^i \leftarrow M_t$ 
14: for epoch  $e = 1, \dots, E$  do
15:    $M_t^i = M_t^i - \eta \cdot \nabla \mathcal{L}(M_t^i, X^i)$ 
16: end for
17: Calculate local update:
     $\Delta_t^i = M_t^i - M_t$ 
18: Randomly sample a sign  $s_t^i \in \{1, -1\}$  with probability  $\Pr[s_t^i = 1] = 0.5$ 
19: if  $s_t^i = 1$  then
20:   Select dimensions indices of  $k$  largest values in  $\Delta_t^i$  to build  $S_{topk}$ 
21: else
22:   Select dimensions indices of  $k$  smallest values in  $\Delta_t^i$  to build  $S_{topk}$ 
23: end if
24: Obtain the private dimension set
     $J_t^i = \mathbf{LDP-DimSel}(S_{topk}, d, k, h, \epsilon)$ 
25: Return  $J_t^i, s_t^i$ 

```

In the later sections, we will introduce two private dimension selection algorithms that provide strict ϵ -LDP guarantees to clients' local data. In addition, since the sign values are randomly sampled by the clients and are unrelated to the local data, the sparse privatized local updates constructed on the server side also satisfy ϵ -LDP.

Theorem 1. *For the original local update Δ of any client, if the dimension selection algorithm used in Algorithm 1 satisfies ϵ -LDP, the sparse privatized local update $\hat{\Delta}$ also satisfies ϵ -LDP.*

Proof. According to Section 2.1.3, for any client with two possible local updates Δ, Δ' , the privatized local update $\hat{\Delta}$ satisfies ϵ -LDP if and only if $\frac{\Pr[\hat{\Delta}|\Delta]}{\Pr[\hat{\Delta}|\Delta']} \leq \exp(\epsilon)$.

In Algorithm 1, the construction of the privatized local update can be decomposed into two steps: privately selecting a dimension set J according to the sampled sign value s and assigning each dimension $j \in J$ with s . This can be formulated as

$$\frac{\Pr[\hat{\Delta}|\Delta]}{\Pr[\hat{\Delta}|\Delta']} = \frac{\Pr[J|\Delta] \cdot \prod_{j \in J} \Pr[\hat{\Delta}[j]|\Delta[j]]}{\Pr[J|\Delta'] \cdot \prod_{j \in J} \Pr[\hat{\Delta}[j]|\Delta'[j]]}, \quad (3.2)$$

and since the dimension selection step satisfies the ϵ -LDP guarantee, it holds that:

$$\frac{\Pr[J|\Delta]}{\Pr[J|\Delta']} \leq \exp(\epsilon). \quad (3.3)$$

Additionally, each selected dimension is assigned with the sign value s , which is a constant and is independent of the real dimension value. Thus, we have:

$$\Pr[\hat{\Delta}[j]|\Delta[j]] = \Pr[\hat{\Delta}[j]|\Delta'[j]] = 1, \quad (3.4)$$

where $j \in J$. To sum up the above considerations, it holds that

$$\frac{\Pr[\hat{\Delta}|\Delta]}{\Pr[\hat{\Delta}|\Delta']} = \frac{\Pr[J|\Delta] \cdot 1}{\Pr[J|\Delta'] \cdot 1} \leq \exp(\epsilon), \quad (3.5)$$

which completes the proof. \square

Note that the privacy guarantees achieved by Theorem 1 hold for any ϵ -LDP dimension selection algorithms and are agnostic to the model structures.

3.4.3 Local Differentially Private Dimension Selection

Next, we will introduce two private dimension selection algorithms used in the framework that provide strict ϵ -LDP guarantees to the local updates and local data.

Algorithm 2: PS for single dimension selection [99]

Input: S_{topk} : top- k dimension set; d : local update size; k : top- k set size; ϵ : privacy budget.

Output: J : selected dimension

1: Sample a Bernoulli variable x such that

$$\Pr[x = 1] = p_{sin} = \frac{\exp(\epsilon) \cdot k}{d - k + \exp(\epsilon) \cdot k}$$

2: **if** $x = 1$ **then**

3: Randomly sample a dimension $J \in \{a \in \{1, \dots, d\} | a \in S_{topk}\}$

4: **else**

5: Randomly sample a dimension $J \in \{a \in \{1, \dots, d\} | a \notin S_{topk}\}$

6: **end if**

7: Return J

Single-dimension Selection

We start with the single-dimension selection algorithms, which only select one dimension for each local update. This has been investigated in [99]. Here, we briefly introduce one of the proposed algorithms called *perturbed sampling* (PS), which is presented in Algorithm 2. Given the top- k dimension set S_{topk} , a dimension J is randomly sampled as

$$J \in \begin{cases} \{a \in \{1, \dots, d\} | a \in S_{topk}\} & \text{with probability } p_{sin} \\ \{a \in \{1, \dots, d\} | a \notin S_{topk}\} & \text{with probability } 1 - p_{sin} \end{cases}. \quad (3.6)$$

Namely, the dimension J is sampled from the top- k dimension set with a probability p_{sin} and otherwise from the non-top- k dimension set. Let p_{sin} be the top- k probability. Thus, the privacy guarantee of Algorithm 2 is as follows:

Lemma 3. *Algorithm 2 satisfies ϵ -LDP when $p_{sin} \leq \frac{\exp(\epsilon) \cdot k}{d - k + \exp(\epsilon) \cdot k}$.*

Proof. For each client, given any two possible local updates Δ and Δ' and the sampled sign value s , let S_{topk}, S'_{topk} be the corresponding top- k dimension sets and $J \in \{1, \dots, d\}$ be any output dimension. Given the top- k probability p_{sin} , the probabilities of sampling the dimension J from the top- k set and from the non-top- k set are respectively $p_{sin} \cdot \frac{1}{k}$ and $(1 - p_{sin}) \cdot \frac{1}{d-k}$. Thus, when $p_{sin} \leq \frac{\exp(\epsilon) \cdot k}{d - k + \exp(\epsilon) \cdot k}$ it holds that

$$\frac{\Pr[J|\Delta]}{\Pr[J|\Delta']} = \frac{\Pr[J|S_{topk}]}{\Pr[J|S'_{topk}]} \leq \frac{\Pr[J|J \in S_{topk}]}{\Pr[J|J \notin S'_{topk}]} = \frac{p_{sin} \cdot \frac{1}{k}}{(1 - p_{sin}) \cdot \frac{1}{d-k}} \leq \exp(\epsilon), \quad (3.7)$$

which completes the proof. \square

Multi-Dimension Selection

Although Algorithm 2 can efficiently select top- k dimensions under LDP guarantees, it only returns one dimension for each local model update, which may result in the loss of valuable parameter information and a slow convergence for high-dimensional models. This motivates us to consider whether it is possible to generalize the algorithm by returning multiple dimensions. In order to select h dimensions under ϵ -LDP, a naive method is to repeatedly apply Algorithm 2 for h times. However, according to the sequential composition property (Property 6), the privacy budget for selecting *each* dimension should be less than ϵ/h , which will lead to a significant decrease in the probability p_{sin} and a degradation of model accuracy. Therefore, better dimension selection algorithms are needed. Since the dimension indices here are non-numerical values, traditional LDP algorithms such as Laplace and Gaussian mechanisms [42] cannot be directly applied. In contrast, the EM [111] is a widely used method for handling such non-numerical queries. Inspired by this idea, we further propose an enhanced EM-MDS algorithm to improve model convergence and utility.

Consider a top- k dimension set S_{topk} and an output set J with h elements. Let $\nu = |S_{topk} \cap J|$ be the number of intersections between the two sets, which is equivalent to the number of top- k dimensions contained in J . A score function $u(S_{topk}, J)$ is then defined as an indicator function to highlight whether the top- k dimensions contained in J are larger than a certain threshold ν_{th} , namely

$$u(S_{topk}, J) = \mathbb{1}(|S_{topk} \cap J| \geq \nu_{th}) = \mathbb{1}(\nu \geq \nu_{th}), \quad (3.8)$$

where $0 \leq \nu \leq h$ and $1 \leq \nu_{th} \leq h$. Furthermore, let ϕ_u be the sensitivity of u , it can be derived that

$$\phi_u = \max_{J \in \mathcal{J}} ||u(S_{topk}, J) - u(S'_{topk}, J)|| = 1 - 0 = 1, \quad (3.9)$$

where \mathcal{J} is the domain of the output set, and S_{topk} and S'_{topk} are two random top- k dimension sets. With the above definitions, the multi-dimension selection process can be defined as follows:

Definition 8 (EM-MDS). *Given the top- k dimension set S_{topk} of a local update, one can randomly sample an output dimension set $J \in \mathcal{J}$ with the following probability:*

$$\begin{aligned} p_{mul} &= \frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J))}{\sum_{J' \in \mathcal{J}} \exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J'))} = \frac{\exp(\epsilon \cdot \mathbb{1}(\nu \geq \nu_{th}))}{\sum_{\tau=0}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon \cdot \mathbb{1}(\tau \geq \nu_{th}))}, \\ &= \frac{\exp(\epsilon \cdot \mathbb{1}(\nu \geq \nu_{th}))}{\sum_{\tau=0}^{\tau=\nu_{th}-1} \omega_{\tau} + \sum_{\tau=\nu_{th}}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon)} \end{aligned} \quad (3.10)$$

where ν is the number of top- k dimensions contained in J , ν_{th} is the threshold of the score function and $\omega_{\tau} = \binom{k}{\tau} \binom{d-k}{h-\tau}$ is the number of possible combinations in \mathcal{J} that contains τ top- k dimensions.

The next question is: how to determine an appropriate threshold ν_{th} to achieve a satisfactory model utility? From Equation (3.10), the probability that the sampled J contains τ top- k dimensions given a threshold ν_{th} can further be derived as follows:

$$p_{mul}(\nu = \tau | \nu_{th}) = \begin{cases} \omega_{\tau} / \Omega & \text{if } 0 \leq \tau < \nu_{th} \\ \omega_{\tau} \cdot \exp(\epsilon) / \Omega & \text{if } \nu_{th} \leq \tau \leq h \end{cases}, \quad (3.11)$$

3 SIGNDS-FL: Horizontal Federated Learning with Local Differential Privacy

where $\Omega = \sum_{\tau=0}^{\nu_{th}-1} \omega_{\tau} + \sum_{\tau=\nu_{th}}^h \omega_{\tau} \cdot \exp(\epsilon)$ is the denominator part of Equation (3.10). Moreover, the expectation of ν given the threshold ν_{th} can be calculated as

$$\mathbb{E}_{mul}[\nu|\nu_{th}] = \sum_{\tau=0}^{\tau=h} \tau \cdot p_{mul}(\nu = \tau|\nu_{th}). \quad (3.12)$$

Intuitively, the higher $\mathbb{E}_{mul}[\nu|\nu_{th}]$, the higher the probability that the sampled J contains more top- k dimensions and the better the model utility. Therefore, the optimum threshold ν_{th}^* can be determined as the threshold that achieves the highest $\mathbb{E}_{mul}[\nu|\nu_{th}]$, namely

$$\nu_{th}^* = \operatorname{argmax}_{\nu_{th} \in \{1, \dots, h\}} \mathbb{E}_{mul}[\nu|\nu_{th}]. \quad (3.13)$$

By summarizing all the above design considerations, the workflow of our EM-MDS algorithm is presented in Algorithm 3. Given all the input settings (d, k, h, ϵ) , the optimal threshold ν_{th}^* is firstly determined based on Equation (3.12) and Equation (3.13) (Line 1). Then, the dimension selection process is conducted as in Definition 8. Nevertheless, as the output domain \mathcal{J} contains $\binom{d}{k}$ possible combinations, directly sampling a set J from \mathcal{J} with a certain probability is computationally expensive, especially when d and k are large. To further improve the efficiency, the trick of inverse sampling is applied, which firstly samples a random variable β from the uniform distribution $\mathbf{U}(0, 1)$ and determines the target number of top- k dimensions ν according to the cumulative distribution of $p_{mul}(\nu|\nu_{th}^*)$ (Lines 3-13). An example of the inverse sampling trick is illustrated in Figure 3.2. In this case, the computational cost of Algorithm 3 is efficiently reduced from $O(\binom{d}{k})$ to $O(d)$. Finally, the output dimension set J is constructed by randomly sampling ν dimensions from the top- k dimension set and $h - \nu$ dimensions from the non-top- k set.

Now, we present the privacy and utility analysis of the EM-MDS algorithm.

Lemma 4. *Algorithm 3 satisfies ϵ -LDP.*

Proof. For each client, given any two possible local updates Δ and Δ' and the sampled sign value s , let S_{topk}, S'_{topk} be the corresponding top- k dimension sets. For any output dimension set $J \in \mathcal{J}$, let $\nu = |S_{topk} \cap J|$, $\nu' = |S'_{topk} \cap J|$ be the number of intersections between J and both top- k sets. With the sampling probability defined in Equation (3.10) it holds that

$$\begin{aligned} \frac{\Pr[J|\Delta]}{\Pr[J|\Delta']} &= \frac{\Pr[J|S_{topk}]}{\Pr[J|S'_{topk}]} = \frac{\sum_{J' \in \mathcal{J}} \frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J'))}{\exp(\frac{\epsilon}{\phi_u} \cdot u(S'_{topk}, J'))}}{\sum_{J' \in \mathcal{J}} \frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J'))}{\exp(\frac{\epsilon}{\phi_u} \cdot u(S'_{topk}, J'))}} = \frac{\sum_{\tau=0}^{\tau=\nu_{th}-1} \omega_{\tau} + \sum_{\tau=\nu_{th}}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon)}{\sum_{\tau=0}^{\tau=\nu_{th}-1} \omega_{\tau} + \sum_{\tau=\nu_{th}}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon)} \\ &= \frac{\exp(\epsilon \cdot \mathbb{1}(\nu \geq \nu_{th}))}{\exp(\epsilon \cdot \mathbb{1}(\nu' \geq \nu_{th}))} \leq \frac{\exp(\epsilon \cdot 1)}{\exp(\epsilon \cdot 0)} = \exp(\epsilon), \end{aligned} \quad (3.14)$$

which completes the proof. \square

As for the utility analysis, we compare the performance of our EM-MDS algorithm with the naive method of applying Algorithm 2 for multi-dimension selection. More specifically, with the output size h , the naive way repeats the PS algorithm h times with the privacy budget ϵ/h for

Algorithm 3: EM-MDS for multi-dimension selection

Input: S_{topk} : top- k dimension set; d : local update size; k : top- k set size; h : output size; ϵ : privacy budget.

Output: J : dimension set with h elements

- 1: Determine the optimum threshold $\nu_{th}^* = \underset{\nu_{th} \in \{1, \dots, h\}}{\operatorname{argmax}} \mathbb{E}_{mul}[\nu | \nu_{th}]$
- 2: Compute denominator $\Omega = \sum_{\tau=0}^{\tau=\nu_{th}^*-1} \omega_{\tau} + \sum_{\tau=\nu_{th}^*}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon)$, where $\omega_{\tau} = \binom{k}{\tau} \binom{d-k}{h-\tau}$
- 3: Randomly sample β from uniform distribution $\mathbf{U}(0, 1)$
- 4: Initialize $\tau = 0$, $\mathcal{F}(\tau) = \omega_0 / \Omega$
- 5: **while** $\mathcal{F}(\tau) < \beta$ **do**
- 6: $\tau = \tau + 1$
- 7: **if** $\tau < \nu_{th}^*$ **then**
- 8: $\mathcal{F}(\tau) = \mathcal{F}(\tau) + \omega_{\tau} / \Omega$
- 9: **else**
- 10: $\mathcal{F}(\tau) = \mathcal{F}(\tau) + \omega_{\tau} \cdot \exp(\epsilon) / \Omega$
- 11: **end if**
- 12: **end while**
- 13: Let $\nu = \tau$ be the number of top- k dimensions
- 14: Sample ν dimensions from $\{a \in \{1, \dots, d\} | a \in S_{topk}\}$ and append to J
- 15: Sample $h - \nu$ dimensions from $\{a \in \{1, \dots, d\} | a \notin S_{topk}\}$ and append to J
- 16: Return J

each iteration. Thus, the probability of sampling from the top- k set is $p_{sin} = \frac{\exp(\epsilon/h) \cdot k}{d-k + \exp(\epsilon/h) \cdot k}$. Given a sampled dimension set J , the probability distribution and the expectation of ν can be derived as follows:

$$p_{sin}(\nu = \tau) = \binom{h}{\tau} \cdot p_{sin}^{\tau} \cdot (1 - p_{sin})^{h-\tau}, \quad \mathbb{E}_{sin}[\nu] = \sum_{\tau=0}^{\tau=h} \tau \cdot p_{sin}(\nu = \tau). \quad (3.15)$$

For the EM-MDS algorithm, we firstly determine the optimum ν_{th}^* based on Equation (3.13) and then calculate the expectation $\mathbb{E}_{mul}[\nu] = \mathbb{E}[\nu | \nu_{th}^*]$ as in Equation (3.12).

We select different settings of h and ϵ to empirically compare the privacy-utility trade-off of the naive method and the proposed method. More specifically, we use $\mathbb{E}[\nu]/h$, the expected ratio of top- k dimensions contained in the output set, to represent the model utility. Intuitively, a higher expected ratio means that there is a higher probability of selecting dimensions from the top- k set, which contributes to the model utility. The comparison results are presented in Figure 3.3. It can be seen that both algorithms achieve the same expected ratio when $h = 1$. This can be derived from Equation (3.10), where the EM-MDS algorithm is equivalent to the PS algorithm when $h = 1$. Moreover, with the same privacy budget ϵ , increasing the output dimension h will lead to a lower top- k ratio and a lower model utility. This is because a large h causes each dimension allocated with less privacy budget, thus a lower probability to be sampled from the top- k set. Nonetheless, the EM-MDS algorithm achieves a distinctively higher top- k ratio in the output set compared with the naive method. In particular, the higher the privacy budget ϵ , the larger the difference. This is due to the fact that EM-MDS considers all the combinations of the output set and assigns higher probabilities to those with more

3 SIGNDS-FL: Horizontal Federated Learning with Local Differential Privacy

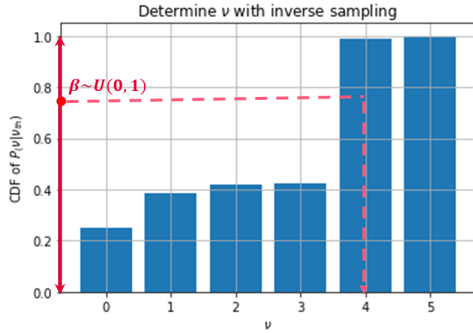


Figure 3.2 Example of using the inverse sampling trick to determine ν , namely the number of top- k dimensions in J .

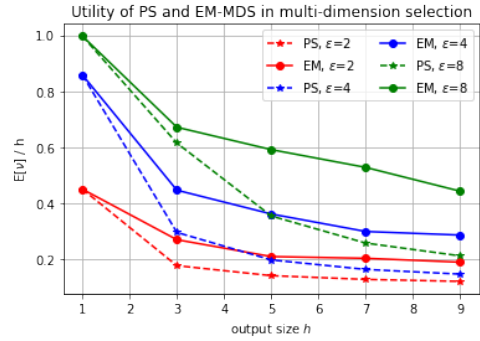


Figure 3.3 Utility of using PS and EM-MDS algorithms in multi-dimension selection regarding different output size h and privacy budget ϵ .

top- k dimensions. As a result, the algorithm can utilize the privacy budget more efficiently in comparison to the naive method and achieve a better model utility.

3.5 Experiments and Results

After implementing the proposed framework, we now report the results of comprehensive experiments with a number of open-source datasets to evaluate its performance. In this section, we will first introduce the experimental settings and then discuss the evaluation results.

3.5.1 Experiment Setup

We first introduce the experimental settings including the datasets and models used in the experiments, baseline algorithms, and parameter configurations.

Datasets and Models

Six open-source datasets are used for evaluating the performance of our framework. Each dataset contains multi-dimensional data records used for classification tasks:

- The **Census** dataset [38] contains records drawn from the 1990 United States census data, which include 68 personal attributes such as gender, income, and marriage status. The dataset is used for a classification task to determine the duration of people’s active duty service.
- The **Adult** dataset [86] originally contains records with 15 personal attributes such as age, occupation, education, and gender. The goal is to train a binary classifier that determines whether a person earns more than 50 thousand dollars a year. We use a processed version from [126] that converts the original attributes into 123 binary features.

Table 3.2 Details of datasets and models

Data Type	Dataset	#Records	#Features	#Classes	Model	#Parameters
Structured Data	Census	2,458,285	68	3	LR	207
	Adult	32,561	123	2	LR	248
	USPS	9,298	256	10	LR	2570
	HAR	10,299	561	6	LR	3372
Image Data	FMNIST	70,000	784	10	NN	76,330
					CNN	44,426
	EMNIST	131,600	784	47	NN	79,919
					CNN	47,571

- The **USPS** dataset [67] is a digit dataset provided by the U.S. Postal Service. The dataset contains 9298 samples with 256 features, which are categorized into 10 classes.
- The **Human Activity Recognition (HAR)** [5] dataset contains 10299 sensor data records of 30 volunteers which can be categorized into 6 daily activities. Each data record has 561 features representing different time and frequency domain variables.
- The **Fashion-MNIST (FMNIST)** [168] dataset contains 70,000 article images (from Zalando), which are categorized into 10 classes. Each record is a grey-scale image of size 28×28 .
- The **Extended-MNIST (EMNIST)** [30] dataset consists of 131,600 handwritten letters, each is a grey-scale image of size 28×28 . Here, we use the EMNIST-Balanced dataset, where the samples are evenly categorized into 47 classes.

We conduct experiments to evaluate the performance of our framework in both simple and complex training tasks. For simple training tasks, we train logistic regression (LR) models on the structured datasets (*i.e.*, **Census**, **Adult**, **USPS**, **HAR**). For complex tasks, we conduct experiments on the image datasets (**FMNIST**, **EMNIST**) using neural network (NN) as well as convolutional neural network (CNN) models. Here, we use a 2-layer NN with 96 neurons on the hidden layer and LeNet [88], a widely used CNN that consists of two convolution layers (each followed by a max-pooling layer) and three fully connected layers to perform the experiments.

Details about the datasets and models are presented in Table 3.2, which include the number of records, features, and classes for each dataset as well as the number of parameters of the used models.

Baselines

Various previous works mentioned in Section 3.3 are used as our baselines, namely: DM [40], PM [158], HM [158], PM-SUB [184] and FEDSEL [99]. For FEDSEL, we choose the ratio of the privacy budget for the dimension selection stage to be 0.1 and 0.5, which is referred to in the experiment results as FEDSEL0.1 and FEDSEL0.5.

Parameter Configurations

In the experiments, we set the global round $T = 500$, where 250 clients participate in each round. Each client possesses 10 data records. The global learning rate is set to $\gamma = 0.05$. For local training, each client updates the model for $E = 10$ epochs with a learning rate $\eta = 0.001$. For the LDP dimension selection algorithms, we set the size of the top- k dimension set $k = 0.1d$, where d is the size of local updates (*i.e.*, the number of model parameters). Moreover, we vary the privacy budget ϵ to explore the influence of privacy on the performance of the framework. In our experiments, we choose $\epsilon \in \{0.5, 1, 2, 4, 8, 12\}$. Additionally, we repeat each experiment 10 times and report the average accuracy.

3.5.2 Performance in Simple Training Tasks

We first evaluate the performance of our SIGNDS-FL framework in simple training tasks. Here, we adopt the single-dimension selection algorithm (Algorithm 2) for a fair comparison, which is referred to as SIGNDSBIT. We train LR models on the four structured datasets and compare the model accuracy with the baselines under different settings.

Model Accuracy Regarding the Privacy Level

To start with, we analyze how the framework performs under different privacy budgets. We conduct the training process under different LDP-FL frameworks with $\epsilon \in \{0.5, 1, 2, 4, 8\}$ and compare the model accuracy, as shown in Figure 3.4. We use the error bars here and also in later results to represent the 95% confidence level. It can be observed that the model accuracy of all methods improves with an increase of ϵ . This aligns with the privacy-utility tradeoff in LDP-FL frameworks: the larger ϵ , the less randomness added for privacy protection, and the better the model utility. Moreover, although the LDP mean estimation-based baselines (DM, PM, HM, and PM-SUB) can achieve relatively satisfactory accuracy for low-dimensional datasets (**Adult**, **Census**), they suffer from an obvious accuracy loss for datasets with more features (**USPS**, **HAR**). This is because the noise scale in these algorithms is proportional to model dimensionality. Thus, for the **USPS** and **HAR** datasets, the increase in model size results in a larger noise scale and a higher accuracy loss. In comparison, FEDSEL and SIGNDS-FL can effectively mitigate the *dimension-dependency* problem. However, SIGNDS-FL can achieve even better performance than FEDSEL. This is likely due to the fact that FEDSEL splits the privacy budget for selecting the top- k dimensions and perturbing the dimension values, whereas our framework applies all the privacy budget for better dimension selection results. In addition, FEDSEL adds random noise to the dimension value, while SIGNDS-FL replaces the dimension value with the sign values. This can better preserve the real model update direction and further improve model convergence. It can be seen that SIGNDS-FL consistently outperforms the baselines for all the datasets. With $\epsilon \geq 4$, the baseline methods suffer from at minimum 5% ~ 15% decrease of accuracy for the **USPS** and **HAR** datasets, while our framework only causes around 1% ~ 2% accuracy loss. The results illustrate that the SIGNDS-FL framework can effectively address the *dimension-dependency* problem in baseline algorithms and achieve better model utility under the same privacy levels.

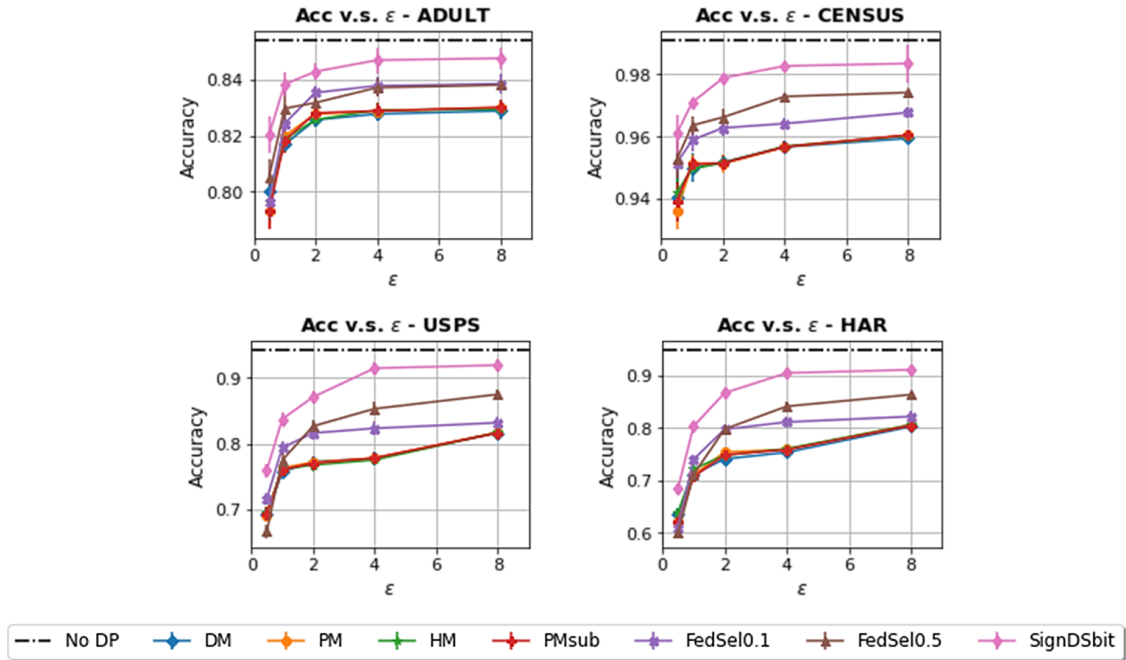


Figure 3.4 Accuracy of LR models trained on structured datasets with different privacy budgets ϵ .

Model Accuracy Regarding the Group Size

We further investigate how the choice of group size N (*i.e.*, the number of participating clients in each round) influences the training performance. We conduct experiments varying the group size $N \in \{100, 250, 500, 750\}$ under a privacy level of $\epsilon = 4$ and compare the change of model accuracy for the different LDP-FL solutions. The results in Figure 3.5 show that training with a larger group size can in general help improve model accuracy. In addition, our framework consistently outperforms the baselines under the same group size settings. Moreover, it can be observed that the baseline algorithms are more sensitive to the group size in comparison to SIGNDS-FL, especially with larger models. More specifically, for **USPS** and **HAR**, the accuracy change of our framework is less than 8%, but around 10% \sim 15% for the baseline methods. This is due to the baseline algorithms adding random noise to the real dimension values, which alters the direction of the selected dimensions. Thus, the algorithms require a sufficiently large group size to reduce the impact of noise on the direction of the averaged local updates as well as on the model utility. In comparison, our SIGNDS-FL framework replaces the real dimension values with their corresponding sign values, which preserves the direction of the selected dimensions. Therefore, our framework shows higher robustness to different choices of group sizes in comparison to the baselines and can still achieve satisfactory performance even with small group sizes.

3 SIGNDS-FL: Horizontal Federated Learning with Local Differential Privacy

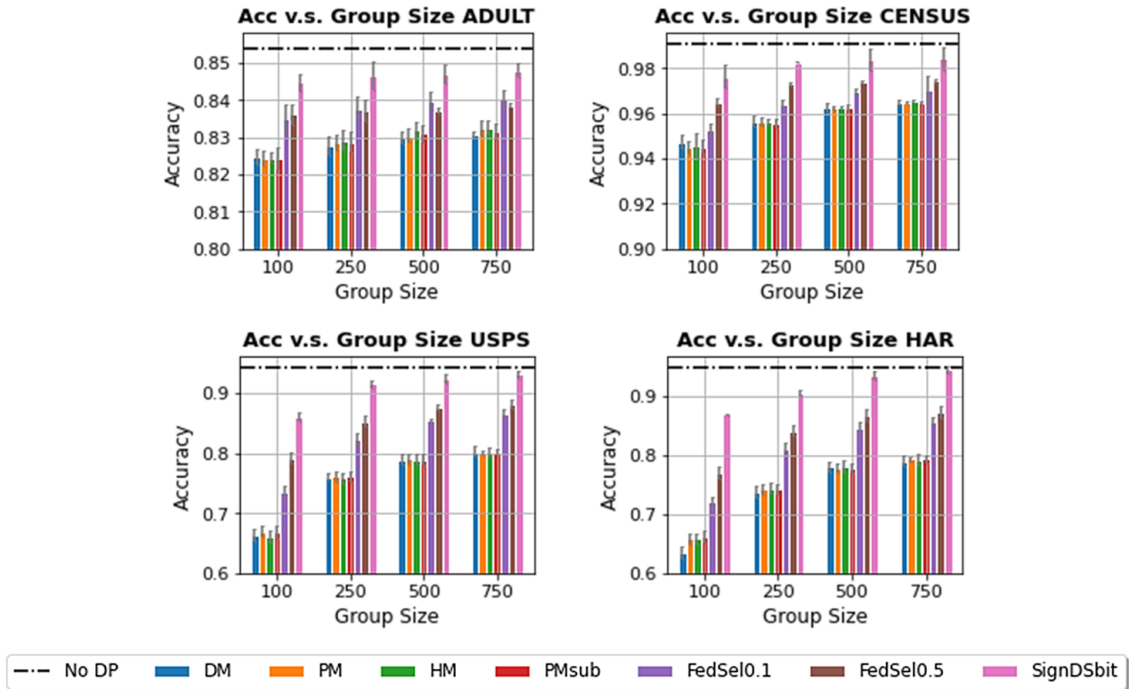


Figure 3.5 Accuracy of LR models trained on structured datasets with different group sizes.

3.5.3 Performance in Complex Training Tasks

Next, we evaluate the performance of our framework on complex tasks. We respectively train the 2-layer NNs and CNNs on both **FMNIST** and **EMNIST** datasets and compare the results with the baseline methods.

Model Accuracy Regarding the Privacy Level

Firstly, we analyze the impact of the privacy budget ϵ on model accuracy. The results are presented in Figure 3.6, where we can see that the baseline algorithms based on LDP mean estimation cannot obtain an acceptable model accuracy for complex training tasks even with $\epsilon = 8$. This occurs since the injected noise in these algorithms gets proportionally larger for high-dimensional deep neural network (DNN), which may require an extremely large ϵ to mitigate the randomness. Moreover, although FEDSEL shows a better performance, there is still a distinctive decrease in accuracy. In comparison, our framework achieves a notable accuracy improvement. For the **FMNIST** dataset, the accuracy loss is around 4% for 2-layer NNs and CNNs. For **EMNIST**, the accuracy loss is respectively around 10% and 15% for both models. It follows that saving the privacy budget for better dimension selection results and using the sign values to preserve the model update direction can contribute to improving the model utility. Moreover, it can be observed that the models trained on **FMNIST** always have a higher accuracy than those trained on **EMNIST**. This is because **EMNIST** has 47 classes in comparison to **FMNIST** which only has 10 classes. Thus, the increase in data variants leads to more difficulty in model training.

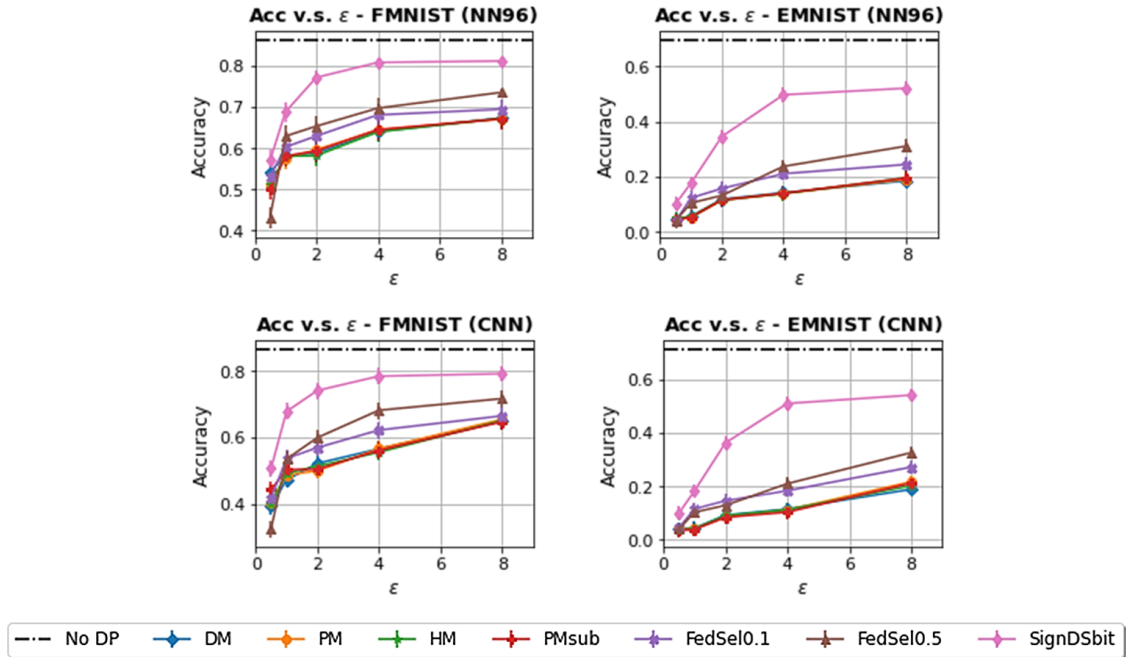


Figure 3.6 Accuracy of NNs and CNNs trained on image datasets with different privacy budget ϵ .

Model Accuracy Regarding the Group Size

We also analyze the impact of group size N on the accuracy of complex models. As in Section 3.5.2, we conduct experiments varying the group size $N \in \{100, 250, 500, 750\}$ under a privacy level of $\epsilon = 4$ and compare the model accuracy. The results are shown in Figure 3.7. It can be seen that the baseline algorithms cannot yield negligible accuracy loss for the complex models even with an increase in group size. On the other hand, our proposed method consistently outperforms the baseline methods under different group sizes. The model accuracy can be further improved by 1% ~ 5% when increasing the group size from 250 to larger than 500. The results demonstrate that the proposed SIGNDS-FL framework can effectively support complex training tasks with high-dimension deep neural network (DNN)s.

3.5.4 Performance Improvement with Multi-Dimension Selection

Although our framework shows significant accuracy improvements for training DNNs in comparison to the baseline methods, there is still an obvious gap in model accuracy between the private training setting and the non-private setting. This may be due to the fact that we only select one dimension for each local update, which slows down the model convergence and thus results in a decrease in model accuracy. In the following, we aim to explore whether our multi-dimension selection algorithm EM-MDS can further help improve model accuracy. We respectively train NNs and CNNs under SIGNDS-FL using the EM-MDS algorithm with h , namely the number of selected dimensions, to be 3 and 5. We choose different privacy budgets $\epsilon \in \{1, 4, 8, 12\}$ in the experiments. In addition, we also analyze the model accuracy when using the naive multi-dimension selection approach, which simply repeats the

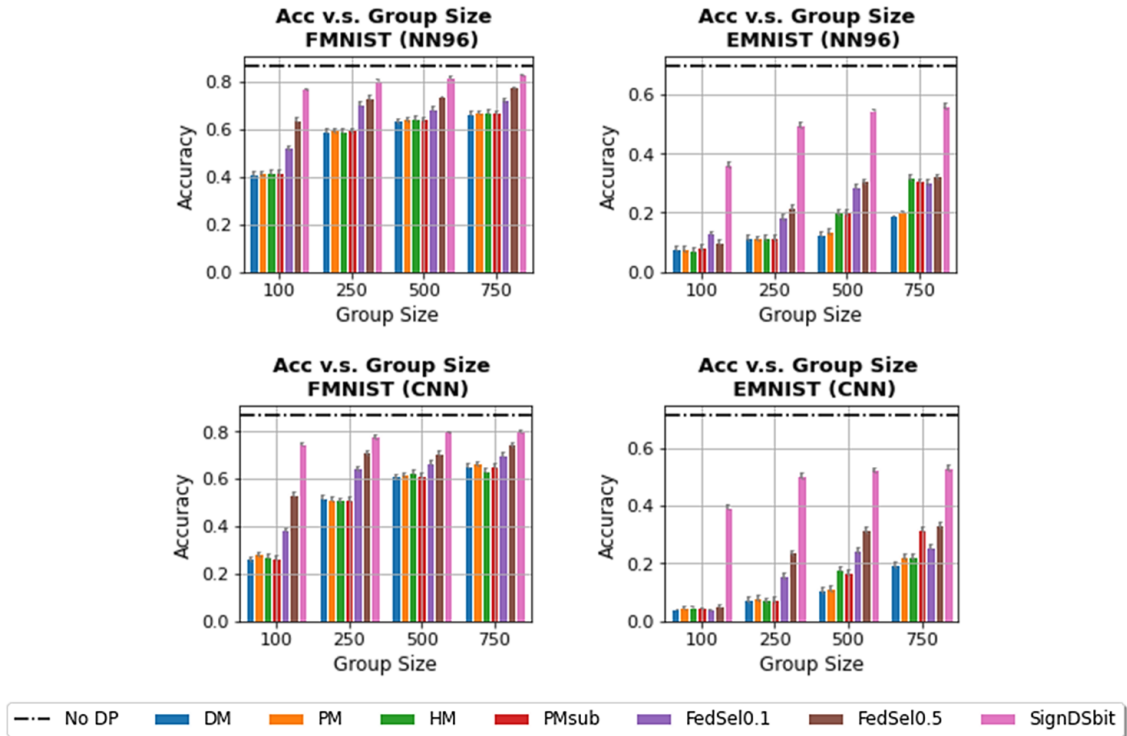


Figure 3.7 Accuracy of NNs and CNNs trained on image datasets with different group size.

single-dimension selection algorithm h times with a privacy budget of ϵ/h for each time (as mentioned in Section 3.4.3).

Comparison of Model Accuracy Regarding the Privacy Level

It can be seen that the naive multi-dimension selection method may not contribute to the model utility under high privacy regimes. In particular, when $\epsilon \leq 4$, selecting multiple dimensions via the naive method causes a distinctive decrease of accuracy in comparison to the single-dimension selection results. In contrast, the proposed EM-MDS algorithm can achieve similar or even slightly better accuracy under the same ϵ . This aligns with the utility analysis in Figure 3.3. More specifically, the naive method evenly splits the privacy budget for each output dimension. When the total ϵ is small, each dimension may be allocated with an insufficient privacy budget and the output set may only obtain a low top- k ratio. In comparison, EM-MDS considers all the combinations of the output set and assigns higher probabilities to the combinations with more top- k dimensions. Thus, the algorithm can achieve a higher top- k ratio in the output set and a better model utility. Moreover, there is an obvious accuracy improvement with EM-MDS as ϵ increases. For example, with $\epsilon \geq 8$, EM-MDS gets better accuracy with $h = 5$ in comparison to the single-dimension selection algorithm while the naive approach still shows a decrease in accuracy. As a result, with the same privacy guarantee, our EM-MDS can more efficiently utilize the privacy budget for selecting more dimensions in comparison to the naive approach and can further improve model utility. In comparison to the

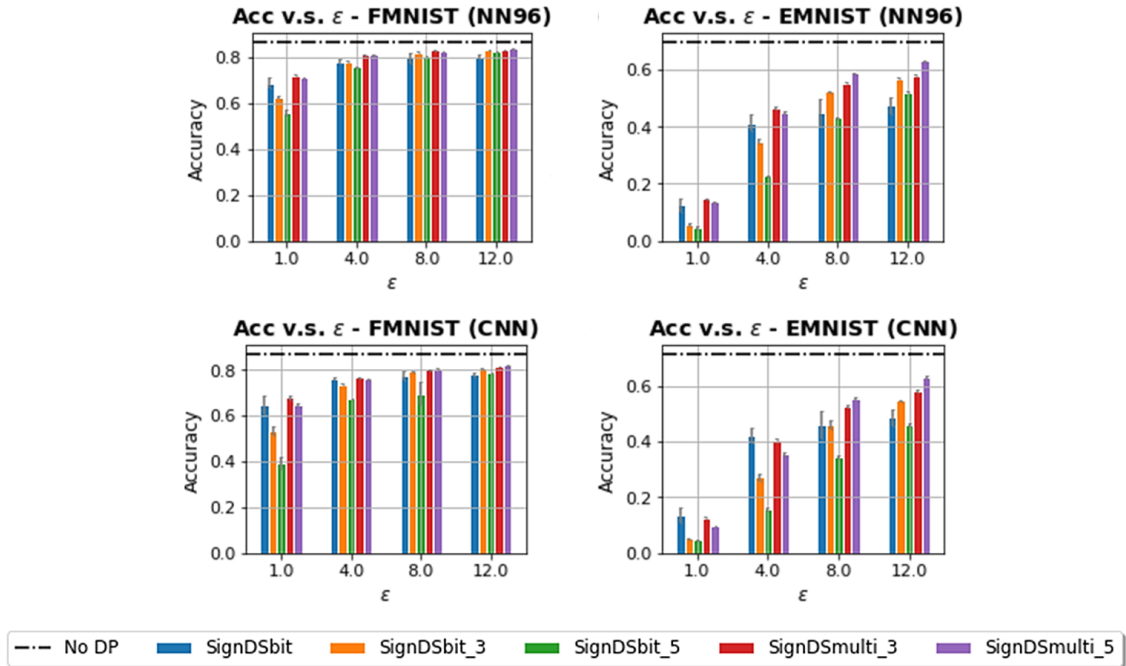


Figure 3.8 Comparison of model accuracy regarding different dimension selection strategies.

results in Section 3.5.3, with $\epsilon \geq 8$, EM-MDS achieves around a 2% increase of accuracy for the **FMNIST** dataset and more than an 8% accuracy improvement for the **EMNIST** dataset.

Comparison of Model Convergence

We further visualize the model accuracy regarding the global rounds in order to investigate whether the EM-MDS algorithm can help speed up model convergence. We compare the results of both models and datasets under the privacy level $\epsilon = 8$ and present the results of different dimension selection strategies in Figure 3.9. Intuitively, model convergence faces a trade-off between the output size h and the randomness during the dimension selection process. As discussed above, a larger output size may not always contribute to model convergence since it may result in an increase in randomness in dimension selection when the privacy budget is insufficient. On the other hand, when ϵ is large enough, an increase in h will speed up model convergence since more information of the local updates is used during training. Similar to the results in Figure 3.8, using the naive approach with $h = 5$ triggers an even slower convergence speed in comparison to the single-dimension selection strategy, which is because of the insufficient privacy budget for dimension selection. In contrast, EM-MDS with $h = 5$ shows the fastest convergence over all the selection strategies. This indicates that EM-MDS can help improve model convergence and thus requires fewer communication rounds for model training.

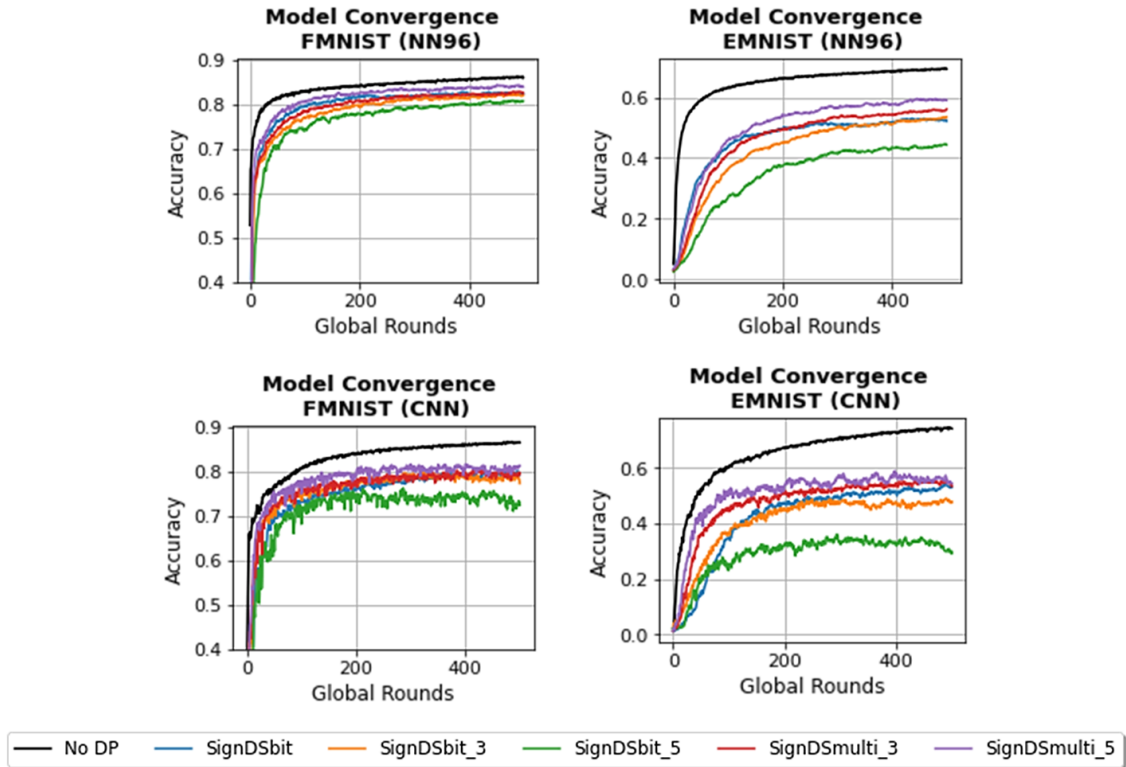


Figure 3.9 Comparison of model convergence regarding different dimension selection strategies.

3.5.5 Analysis of Results

From the results in Section 3.5.2 and Section 3.5.3, it can be seen that the LDP mean estimation-based baselines are effective for low-dimensional models but suffer from large accuracy loss for high-dimensional models. This is because the injected noise in these algorithms grows proportionally with the model size, which causes significant utility loss. Moreover, although FEDSEL can mitigate this *dimension-dependency* problem for LR models, it still poses an obvious accuracy loss for DNNs. On the contrary, the SIGNDS-FL framework achieves better accuracy for both simple LR models and complex DNNs. In addition, we observe that the baseline algorithms are more sensitive to the group size and require a sufficiently large group size to mitigate the impact of noise on model convergence. In comparison, SIGNDS-FL can still obtain acceptable model accuracy even with small group sizes. The results demonstrate the viability of SIGNDS-FL in real-life applications. On the one hand, the framework can effectively support not only simple LR models but also complex DNNs. On the other hand, the framework can also be extended to scenarios with only a small number of local clients.

The results in Section 3.5.4 illustrate how the proposed EM-MDS algorithm can further improve the accuracy of complex models. The results show that the naive multi-dimension selection algorithm may have a negative impact on model accuracy, especially when ϵ is small. This is because the naive approach evenly splits the privacy budget across each output dimension. This causes each dimension to be allocated with an insufficient privacy budget and

thus a low probability of being selected from the top- k dimensions. In contrast, with the same privacy budget, our EM-MDS algorithm follows the idea of exponential mechanism [111] and assigns higher probabilities to the output combinations with higher top- k ratios. Therefore, the top- k dimensions will be more likely to be selected, which contributes to a better model utility. With the increase of ϵ , the model accuracy of using EM-MDS is better than single-dimension SIGNDS-FL. The results demonstrate that the EM-MDS algorithm can be applied to improve the performance of SIGNDS-FL in training high-dimensional models, which further strengthens the framework’s practicality in real-world applications.

3.6 Discussion

In this work, we introduced an efficient and privacy-preserving LDP-FL framework with a satisfactory privacy-utility balance. The framework enjoys significantly lower communication and computation costs than crypto-based solutions and thus can be applied in a variety of large-scale privacy-sensitive applications. For example, in mobile edge computing applications, one can use the framework for privately training user profile models for providing keyboard suggestions [175] or ranking browser history suggestions [59]. In computer vision applications [100], the framework can be applied to train object detection models while ensuring the privacy of the local images. Besides the existing applications of supervised learning tasks, it is also possible to extend the framework to support other ML tasks, such as unsupervised learning, reinforcement learning, and semi-supervised learning [79]. For instance, one can apply the framework to privacy-preserving distributed synthetic data generation scenarios [146, 74], where the generative models are trained privately using the proposed framework to learn the distribution of local data and then to generate synthetic data on the server side. This can be considered as an alternative approach for privacy-preserving data collection. In addition, the framework can also be integrated into private semi-supervised learning applications, where both labeled and unlabeled local data are utilized for updating the global model [172]. It should be noted that despite the large variety of possible applications and ML model scenarios, the main idea of the LDP-FL protocol remains unchanged.

3.7 Conclusion

FL has recently attracted increased attention due to its computational efficiency and privacy benefits. However, the naive FL framework is still vulnerable to a number of privacy attacks as the exchanged local updates and the final model can still reveal sensitive information of clients’ local data. LDP, as a strong notion of privacy, has been recently applied to the local side of federated learning to provide privacy guarantees for clients’ local data. However, previous LDP-FL solutions cannot provide satisfactory outcomes for high-dimensional models.

In this work, we propose SIGNDS-FL, an efficient and privacy-preserving federated learning framework based on LDP dimension selection. The main idea is to privately select a set of “important” dimensions of the local updates under strict LDP guarantees and to construct sparse privatized local updates using sign values, which are randomly determined by the clients. Moreover, we propose EM-MDS, an efficient multi-dimension selection algorithm

3 SIGNDS-FL: Horizontal Federated Learning with Local Differential Privacy

that can better utilize the privacy budget and contribute to improving model convergence and accuracy. We evaluate the framework on many real-world structured and image datasets using simple LR models as well as DNNs. Extensive experimental results demonstrate that our framework significantly outperforms the previous LDP-FL solutions and enjoys a favorable utility-privacy balance.

4 DP-FED-WAE for Private Sharing of High-Dimensional Structured Data

Table 4.1 Bibliographic details for P2

Title	Privacy-Preserving High-dimensional Data Collection with Federated Generative Autoencoder
Authors	Xue Jiang ^{1,2} (xue.jiang@tum.de) Xuebing Zhou ² (xuebing.zhou@huawei.com) Jens Grossklags ¹ (jens.grossklags@in.tum.de)
	¹ Technical University of Munich (TUM), Boltzmannstraße 3, 85748 Garching, Germany ² Huawei Munich Research Center, Riesstraße 25, 80992 Munich, Germany
Type	Journal
Outlet	Proceedings on Privacy Enhancing Technologies, Volume 2022 ³
Ranking	CORE 2021 ⁴ : A
Status	Published
Citation	Jiang, X., Zhou, X., & Grossklags, J. (2022). Privacy-Preserving High-dimensional Data Collection with Federated Generative Autoencoder. Proceedings on Privacy Enhancing Technologies, 2022(1), 481-500.
Copyright	This work is published under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 ⁵ (CC BY-NC-ND 3.0) License. Copyright held by the owner/author(s).
Author Contributions	Xue Jiang developed the core idea for the paper and took primary responsibility for designing the methodology, acquiring data, implementing experiments, evaluating results, and drafting the manuscript. Xuebing Zhou and Jens Grossklags provided valuable feedback and suggestions on the methodology and experiments, and assisted in reviewing and enhancing the manuscript.

³<https://www.petsymposium.org/popets/2022/>

⁴<http://portal.core.edu.au/conf-ranks/1442/>

⁵<https://creativecommons.org/licenses/by-nc-nd/3.0/>

4.1 Abstract

Although FL addresses the data silo problem by distributing the model training task to the local side, the solution is task-specific. Namely, the collaborative training procedure needs to be repeated for each different training task or model structure. In contrast to model transmission, an alternative solution is data transmission. The main idea is to anonymize the local data on the local side and only send the privatized data to the server, which will be used to support various downstream data analysis and model training tasks. One of the state-of-the-art privatization technologies is local differential privacy (LDP). However, existing LDP algorithms are not applicable to high-dimensional data; not only because of the increase in computation and communication costs, but also poor data utility.

In this work, we introduce a novel data synthesis-based solution for addressing the *curse of dimensionality* problem in LDP-based high-dimensional data collection. Different from existing works on synthetic data generation, we focus on the scenario where the data are distributed on the local side and are inaccessible to the server. With the combination of a generative autoencoder, federated learning, and differential privacy, our framework is capable of privately learning the statistical distributions of local data and generating high-utility synthetic data on the server side without revealing users' private information. We evaluated the framework in terms of data utility and privacy protection on a number of real-world datasets and showed that our framework outperforms the LDP-based baseline algorithms in capturing joint distributions and correlations of attributes and generating high-utility synthetic data. Extensive experimental results demonstrate the capability and efficiency of our framework in synthesizing high-dimensional data while striking a satisfactory utility-privacy balance.

4.2 Introduction

With the rapid development of network and computer technologies, large and diverse quantities of multi-dimensional person-specific data are frequently generated on local devices such as smartphones and IoT sensors. These data usually contain rich information of univariate and multivariate (joint) distributions describing user profiles, which is valuable for data analysts to explore the hidden correlations and patterns of data from different perspectives and to obtain a better understanding of the characteristics of user groups. For instance, a digital healthcare application may utilize users' physical information (*i.e.*, temperature, blood pressure, activity signals, *etc.*) for health monitoring and disease predictions, while an online shopping website may take users' age, gender, and purchase history for providing suitable product recommendations. In principle, the more dimensions the data consist of, the more information can be used for describing an individual user; thus, the more accurate the decision-making system can be. Therefore, the collection of multi-dimensional data can be of significant help to companies and organizations in designing and building effective business intelligence & AI services.

However, since the data are generated based on individuals' ongoing behaviors, the direct collection can reveal sensitive information about them and lead to severe privacy problems (see, for example, [18, 11]). local differential privacy (LDP) [82], as a state-of-the-art data anonymization mechanism, has been recently deployed by major technology organizations

such as Apple [35], Google [44], and Microsoft [36] for privacy-preserving data collection. By locally randomizing the user data before sending it to the server, the LDP algorithms ensure that the server cannot access the original user data, but is able to learn the population’s overall statistics. However, prior research on LDP-based data collection mainly focuses on one-dimensional statistics, such as frequency estimation [35, 44], heavy-hitter identification [10, 16], and itemset mining [128, 161], *etc.* However, since the attributes in multi-dimensional data are usually correlated, the server is particularly interested in learning the correlations and joint distributions among attributes.

Directly applying the above-mentioned LDP algorithms for estimating the joint distributions of multi-dimensional data faces a foundational problem: the *curse-of-dimensionality*. The domain size increases exponentially with data dimensionality, which will lead to extremely large communication costs and storage complexity, as well as a significant degradation in data utility. To reduce the large communication overhead, Fanti et al. [45] proposed to separately collect data of each dimension under LDP and to estimate the joint distributions using expectation maximization (EM). However, the algorithm only supports estimates of the joint distribution of two attributes. Further, Ren et al. [130] introduced LOPUB, which splits the w -dimensional data into m -dimensional clusters ($m < w$) using dependence graphs and estimates m -way joint distributions via an EM-based and Lasso regression-based approach. However, the algorithm still suffers from high computational complexity and low data utility when m is large. Based on these facts, alternative solutions for privacy-preserving high-dimensional data collection are still greatly needed.

Recently, data synthesis has been considered a promising approach for addressing data privacy issues in business intelligence & AI services. With the strong capabilities of characterizing the joint distributions and correlations of high-dimensional data, deep generative models are increasingly used for generating high-utility and low-sensitivity synthetic data. In this work, we follow the idea of data synthesis and propose DP-FED-WAE, a privacy-preserving framework for high-dimensional categorical data collection. Different from prior work on differentially private synthetic data generation algorithms [179, 122, 145], which mainly focuses on the centralized setting where the real data are already collected by the server, our framework conducts the data synthesis *without* collecting real local data. The main idea is to train a (generative) Wasserstein autoencoder (WAE) [144] under the federated learning (FL) [108] setting to learn the distributions of the high-dimensional local data and then to generate high-quality synthetic data on the cloud server. Moreover, we propose a novel local randomization algorithm SIGNDS, which is applied to a client’s local updates to prevent potential privacy leakages in FL. The algorithm provides a strict ϵ -LDP privacy guarantee for any client’s local dataset.

In comparison with previous data collection approaches, our framework shows significant advantages in both *data utility* and *privacy protection*. As for data utility, the WAE model has a strong capability in capturing correlations and joint distributions of high-dimensional data and generating high-utility synthetic data. The generated synthetic data can be easily scaled up to replace the real data for data analysis and AI training tasks. As for privacy, the generated data are fully synthetic, which effectively reduce risks of re-identification attacks or attribute disclosure [122]. Moreover, training the WAE model under the LDP-FL setting not only avoids the collection of raw user data but also provides comprehensive privacy guarantees to the framework. Our contributions can be summarized as follows:

4 DP-FED-WAE for Private Sharing of High-Dimensional Structured Data

- We propose DP-FED-WAE, an efficient and privacy-preserving framework that effectively combines a generative autoencoder, FL, and DP for collecting high-dimensional categorical data. Based on the idea of data synthesis, the framework effectively solves the *curse-of-dimensionality* problem in LDP-based data collection solutions. The synthetic data preserves high utility and can replace real data for data mining and AI training tasks.
- We further propose a novel local randomization algorithm SIGNDS, which perturbs clients' local updates and prevents potential privacy leakages in FL. We prove that the algorithm follows a strict ϵ -LDP definition and provides a strong local privacy guarantee to any client's local data.
- We have implemented our framework and evaluated the performance in terms of data utility and privacy protection using real-world datasets containing 68–124 classification attributes. Through comparison with the LDP-based algorithms, we show that the synthetic data generated by our framework always preserve much closer joint distributions and correlations to real data. Also, the accuracy loss of the model trained with synthetic data generated by our framework is significantly reduced in comparison to the baseline method. With a local privacy guarantee $\epsilon = 8$, we reduced the accuracy loss from 10% ~ 30% to less than 3% and at best even less than 1%. Extensive evaluation experiments show that our framework has outperforming capability and efficiency in collecting high-dimensional data while striking a satisfactory utility-privacy balance.

4.3 Problem Statement

In this work, we consider a scenario where a large number of local users hold high-dimensional personal data. A central server aims to estimate joint distributions of these high-dimensional data and to generate similar synthetic data for data analysis or designing new AI services. Here, we assume the server to be *honest-but-curious*, who follows the system protocols but tries to infer sensitive information of local users. Thus, to protect local privacy, we require the server not to have access to raw local data but only anonymized versions of data or their feature representations.

Assume that there are N local users, each holding one or more data records, which have $|\mathcal{A}|$ attributes $\mathcal{A} = \{a_1, \dots, a_{|\mathcal{A}|}\}$. Each attribute a_i has a domain Λ_i of possible values. The full domain for the $|\mathcal{A}|$ -dimensional data record is denoted as $\Lambda = \Lambda_1 \times \dots \times \Lambda_{|\mathcal{A}|}$, where \times is the Cartesian product. The total domain size is $|\Lambda| = \prod_{i=1}^{|\mathcal{A}|} |\Lambda_i|$, which increases exponentially with data dimensionality $|\mathcal{A}|$. Based on the notation above, the problem can then be formulated as follows: given a $|\mathcal{A}|$ -dimensional private dataset X distributed among N local users, a central server aims to generate a synthetic dataset \tilde{X} without access to raw data X . The synthetic dataset \tilde{X} has the same attributes \mathcal{A} , and preserves similar joint distributions of X , namely

$$P_{\tilde{X}}(a_1 \cdots a_{|\mathcal{A}|}) \approx P_X(a_1 \cdots a_{|\mathcal{A}|}). \quad (4.1)$$

It can be further derived that the synthetic data \tilde{X} also preserves m -way joint distributions as X . Namely, given an m -way attribute combination $A \subseteq \mathcal{A}$, we have

$$P_{\tilde{X}}(A) \approx P_X(A). \quad (4.2)$$

4.4 Proposed Solution

As discussed previously, existing LDP algorithms are impractical for collecting high-dimensional data due to both high computation and communication costs and poor data utility (e.g., [44, 45, 130]). In order to solve the curse-of-dimensionality problem, we propose DP-FED-WAE, an efficient privacy-preserving framework for collecting high-dimensional categorical data. The framework contains three main components: a generative autoencoder, FL, and DP. Following the idea of recent data synthesis techniques, the framework utilizes generative autoencoders to learn the statistical distributions and correlations of high-dimensional user data and then to generate high-utility synthetic data on the server side. Different from existing works of synthetic data generation where the real data are already available to the server (e.g., [179, 122, 145]), our framework focuses on the scenario where the real data are distributed on local devices. Therefore, we propose to train the generative autoencoder under the FL setting, which only exchanges model parameters during the training process and keeps the raw user data inaccessible to the server. Furthermore, we incorporate DP during the training process in order to prevent potential privacy leakages in FL. In comparison to the previous DP-FL frameworks that add DP noise on the server side [110, 7], we propose a novel local randomization algorithm that perturbs the local updates before uploading them to the server. This ensures that the server cannot gain access to real local updates and efficiently prevents local privacy leakages. We prove that the randomization algorithm follows a strict LDP definition and provides a strong *local privacy guarantee* to each client's local dataset.

The overall workflow is presented in Figure 4.1, which is processed in the following sequence:

1. The local clients first process the original categorical data into a numerical form, which can be used for training the generative autoencoder. At the same time, the server defines the structure of the generative autoencoder based on the dimensionality of local data and initializes the model.
2. The generative autoencoder is then collaboratively trained under the FL mechanism that is incorporated with LDP to achieve strict privacy guarantees.
3. After the model gets trained, the decoder is extracted for generating synthetic data. The generated data will be finally converted back to categorical form and used for data mining and building machine learning models.

4.4.1 Data Pre-Processing and Design of the Generative Model

Since the original data are categorical and cannot be directly processed by machine learning models, we first convert the data into numerical form. Here, we use a *one-hot encoding* to encode each categorical attribute into a binary vector. Each entry in the binary vector stands for a unique attribute value and the entry of the given value is set to 1 while all the others are

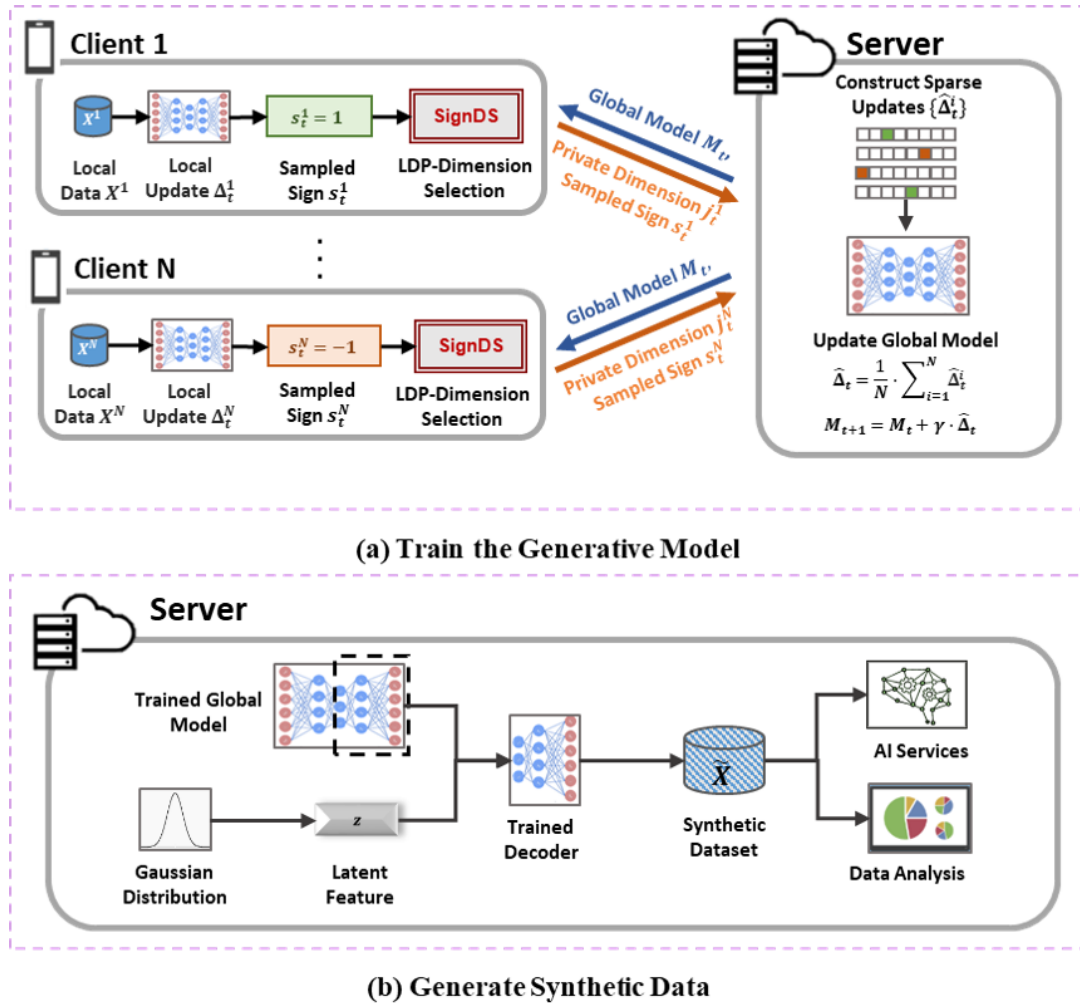


Figure 4.1 Overview of the DP-FED-WAE framework. The generative Wasserstein Autoencoder is first trained under the federated setting, which learns the distributions of real local data. An LDP algorithm SIGNDS is applied to the local updates to provide strict local privacy guarantees. After the model is trained, the *decoder* part is used to generate high-utility synthetic data. The generated data will be used for data mining and building AI services.

set to 0. Finally, we concatenate all the binary vectors into one vector as the input data for the generative model.

In this framework, we choose the Wasserstein autoencoder (WAE) as the generative model in our framework, which provides better data synthesis capability in comparison to the variational autoencoder (VAE) [85] and less training difficulty than the generative adversarial network (GAN) [53]. As a variant from the family of autoencoders, WAE preserves the encoder-decoder architecture. The encoder Q_ψ compresses the original high-dimensional input $x \sim P_x$ into the low-dimensional latent feature $z = Q_\psi(x)$ and the decoder G_θ maps z to the reconstructed output $\tilde{x} = G_\theta(z)$, which is the same shape as x . The distance between the original input and the reconstructed output can be presented as $\mathcal{L}_{rec}(x, G_\theta(Q_\psi(x)))$. In addition, a regularizer term $\mathcal{L}_{lat}(q_z, p_z)$ is applied to measure the distance between the latent space distribution q_z and certain prior distribution p_z . The final objective function of the WAE model can thus be formulated as follows:

$$\mathcal{L}_{WAE} = \mathbb{E}_{x \sim P_x} [\mathcal{L}_{rec}(x, G_\theta(Q_\psi(x)))] + \lambda \cdot \mathcal{L}_{lat}(q_z, p_z), \quad (4.3)$$

where λ is a hyperparameter for balancing the two terms. The goal of training is to find an optimal set of parameters, which minimizes the distance between the inputs and outputs while restricting the latent space to follow the prior distribution.

We design the WAE models with fully-connected hidden layers. We apply the *relu* activation on the output of each hidden layer for better training performance. Moreover, since the inputs are binary vectors, we use the *sigmoid* activation on the output layer, which restricts the output value within $[0, 1]$. Then, we calculate the binary cross-entropy of each input/output dimension and compute the average as the reconstruction distance $\mathcal{L}_{rec}(x, G_\theta(Q_\psi(x)))$. For the latent space distance $\mathcal{L}_{lat}(q_z, p_z)$, we use the standard Gaussian distribution as the prior distribution p_z and use the maximum mean discrepancy (MMD) to measure the distance between the latent space distribution q_z and p_z , as in [144]. Given a batch of data sampled from the two distributions, *i.e.*, $\{q^1, \dots, q^N\} \sim q_z$ and $\{p^1, \dots, p^N\} \sim p_z$, $\mathcal{L}_{lat}(q_z, p_z)$ can be empirically estimated as

$$\mathcal{L}_{lat}(q_z, p_z) = \frac{1}{N(N-1)} \sum_{i \neq j} \mathcal{K}(p^i, p^j) - \frac{2}{N^2} \sum_{i, j} \mathcal{K}(p^i, q^j) + \frac{1}{N(N-1)} \sum_{i \neq j} \mathcal{K}(q^i, q^j), \quad (4.4)$$

where $\mathcal{K}(x, y) = \frac{\kappa}{\kappa + \|x - y\|_2^2}$. Given d_z as the dimension of latent layer and σ_z as the scale of the prior distribution, $\kappa = 2d_z\sigma_z^2$. We choose λ equal to 1.

4.4.2 Training the Generative Model

Previous LDP-FL frameworks (*e.g.*, [42, 40, 158]) evenly split the privacy budget across dimensions and apply the perturbation independently. However, the per-dimension privacy budget becomes extremely small for high-dimensional models, which results in a significant increase in noise. A recent work [99] proposed a *two-stage* LDP-FL framework, which splits the privacy budget into a *dimension selection (DS)* stage and a *value perturbation (VP)* stage. In the DS stage, the local update is sorted by *absolute* value, and one "important" dimension is privately selected from the top- k dimensions; in the VP stage, the value of the selected dimension is perturbed. Finally, a sparse local update is constructed and returned to the server.

Algorithm 4: SIGNDS

Input: $\Delta \in \mathbb{R}^d$: local update; k : size of the top- k set; ϵ : privacy budget; s : sampled sign
Output: j : selected dimension index

- 1: **if** $s = 1$ **then**
- 2: Select dimensions of k *largest* values in Δ to build the top- k dimension set S_{topk}
- 3: **else**
- 4: Select dimensions of k *smallest* values in Δ to build the top- k dimension set S_{topk}
- 5: **end if**
- 6: Sample a Bernoulli variable x such that $\Pr[x = 1] = \frac{e^\epsilon \cdot k}{d - k + e^\epsilon \cdot k}$
- 7: **if** $x = 1$ **then**
- 8: Randomly sample a dimension $j \in \{a \in \{1, \dots, d\} | a \in S_{topk}\}$
- 9: **else**
- 10: Randomly sample a dimension $j \in \{a \in \{1, \dots, d\} | a \notin S_{topk}\}$
- 11: **end if**
- 12: **Return** j

Although [99] mitigated the dimension-dependency problem by only selecting one "important" dimension, the privacy budget is still consumed by the two stages. In high-privacy scenarios (where the privacy budget is small), each stage may therefore obtain only an insufficient privacy budget and cause large randomness.

Motivated by the limitations of [99], we propose a *sign-based dimension selection* algorithm SIGNDS, as presented in Algorithm 4. The main idea is to substitute the VP stage by assigning a constant value to the selected dimension. Since the parameter values may have different signs, we introduce an extra variable $s \in \{-1, 1\}$, which is randomly sampled by the client with equal probability. Then, given each local update Δ , we build the top- k dimension set S_{topk} according to Δ 's *real values* and the sampled sign s : if $s = 1$, S_{topk} is built with the dimensions of the k *largest* values; otherwise, it is built with the dimensions of the k *smallest* values. We refer to the dimensions included in S_{topk} as *top- k dimensions* and the rest as *non-top- k dimensions*. Then, a dimension index j is randomly sampled as follows:

$$j \in \begin{cases} \{a \in \{1, \dots, d\} | a \in S_{topk}\} & w.p. \quad p \\ \{a \in \{1, \dots, d\} | a \notin S_{topk}\} & w.p. \quad 1 - p \end{cases}, \quad (4.5)$$

Namely, the index j is sampled from the *top- k dimensions* with a probability of p and otherwise from the *non-top- k dimensions* with a probability of $1 - p$. We refer to p as the *top- k probability*. Finally, the dimension index j and the sampled sign value s are returned to the server. Since our algorithm does not return the dimension value to the server, we save the privacy budget for the value perturbation stage in [99]. With the same privacy level, we can now achieve less randomness and thus higher accuracy in dimension selection.

In the following, we provide the privacy guarantee and utility analysis of Algorithm 4.

Lemma 5. *Algorithm 4 satisfies ϵ -LDP when the top- k probability $p \leq \frac{e^\epsilon \cdot k}{d - k + e^\epsilon \cdot k}$.*

Proof. For each client, given the sampled sign s and any output dimension $j \in \{1, \dots, d\}$, let S_{topk}, S'_{topk} be the top- k dimension set of any two possible local update vectors Δ and Δ' .

Given the top- k probability p , the probability of sampling a dimension j from the top- k set and the non-top- k set are respectively $p \cdot \frac{1}{k}$ and $(1-p) \cdot \frac{1}{d-k}$. Thus, when $p \leq \frac{e^\epsilon \cdot k}{d-k+e^\epsilon \cdot k}$ we have

$$\frac{\Pr[j|\Delta]}{\Pr[j|\Delta']} = \frac{\Pr[j|S_{topk}]}{\Pr[j|S'_{topk}]} \leq \frac{\Pr[j|j \in S_{topk}]}{\Pr[j|j \notin S'_{topk}]} = \frac{p \cdot \frac{1}{k}}{(1-p) \cdot \frac{1}{d-k}} \leq e^\epsilon \quad (4.6)$$

which completes the proof. \square

In addition to the privacy guarantee, we are also interested in how to choose proper k and ϵ in order to achieve a certain top- k probability p . Let $\xi = k/d$ be the ratio of the top- k parameters regarding the total number of parameters. A $\xi = 1$ means to randomly select one dimension from the entire dimension group. Intuitively, the smaller ξ , the closer the parameter values of top- k dimensions to the real largest (or smallest) value and the better the model utility. We derive relations among ϵ , p , and ξ as follows:

Corollary 1. *With a fixed privacy budget ϵ , in order to achieve a probability p , ξ should satisfy $\xi \geq \frac{p}{e^\epsilon \cdot (1-p) + p}$.*

Corollary 2. *With a fixed top- k ratio ξ , in order to achieve a probability p , ϵ should satisfy $\epsilon \geq \log \frac{p \cdot (1-\xi)}{(1-p) \cdot \xi}$.*

Proof. From Lemma 5, we have $p \leq \frac{e^\epsilon \cdot k}{d-k+e^\epsilon \cdot k} = \frac{e^\epsilon \cdot \xi}{1-\xi+e^\epsilon \cdot \xi}$. Thus, with a fixed ϵ , we have $\xi \geq \frac{p}{e^\epsilon \cdot (1-p) + p}$; with a fixed ξ , we have $\epsilon \geq \log \frac{p \cdot (1-\xi)}{(1-p) \cdot \xi}$. \square

Corollary 1 states that with a fixed privacy budget ϵ , a smaller top- k ratio ξ leads to a decrease of top- k probability p . Moreover, given an expected top- k probability p and a predefined top- k ratio ξ , the minimum required privacy budget ϵ can be calculated using Corollary 2. We further visualize the relations of ϵ , p , and ξ in Figure 4.2. As shown in Figure 4.2a, in high-privacy scenarios (e.g., $\epsilon \leq 2$), the required top- k ratio ξ differs distinctively with the choices of p . Namely, we have to choose a large ξ in order to ensure that the index is more likely to be sampled from top- k dimensions. As ϵ increases (e.g., $\epsilon \geq 6$), ξ does not differ much regarding p . In other words, we can always achieve a high top- k probability even with a small top- k ratio. In Figure 4.2b, we further present the minimum ϵ under various ξ and p .

We now describe the overall training process presented in Algorithm 5. At each global round t , the server selects a group of n clients and broadcasts the current global model M_t . On the local side, each client i in the group trains the global model for several epochs with his local data X^i and computes the local update Δ_t^i . Then, the client randomly samples a sign $s_t^i \in \{-1, 1\}$ with equal probability and uses it along with the predefined privacy budget ϵ_r to privately select a dimension index j_t^i of the local update. Finally, s_t^i and j_t^i are returned to the server. After receiving the dimension j_t^i and the sampled sign s_t^i , the server builds a sparse local update $\hat{\Delta}_t^i$ and assigns s_t^i to the selected dimension. Since the selected dimension j_t^i satisfies the ϵ -LDP guarantee and the assigned sign value s_t^i is unrelated to the local data, according to LDP's robustness to post-processing (Property 5), the sparse local updates also satisfy ϵ -LDP. Finally, the server aggregates all the sparse local updates and updates the global model with a global learning rate γ . The updated global model M_{t+1} is distributed to local clients to start the next round.

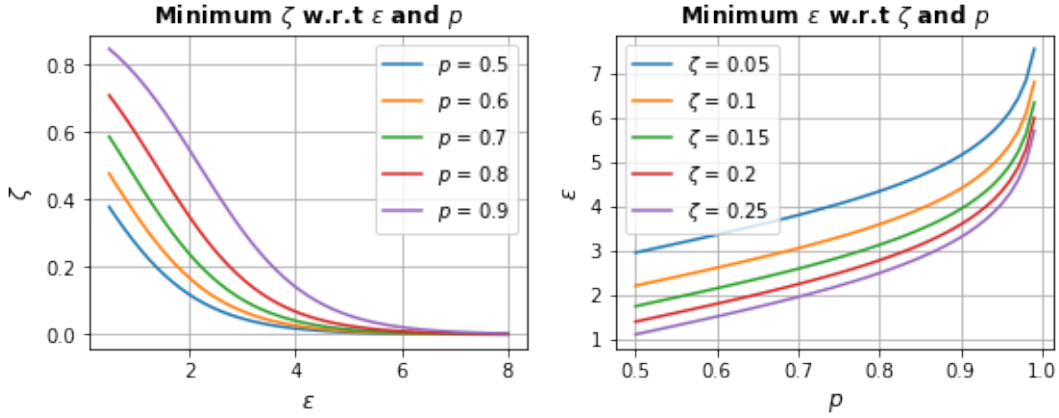


Figure 4.2 Relations among ϵ , p , and ξ . (a): given privacy budget ϵ and the expected top- k probability p , the minimum top- k ratio ξ required. (b): given the expected top- k ratio ξ and top- k probability p , the minimum privacy budget ϵ required.

Note that according to the sequential composition property (Property 6), if the same client repeatedly participates in the training and submits the local update for multiple global rounds, the overall privacy guarantee for his local data will be accumulated. Assume each client is allowed to participate in at most t_r global rounds. In order to ensure an overall privacy guarantee of ϵ -LDP for each client's local data after the whole training process, the per-round privacy guarantee should satisfy $\epsilon_r \leq \epsilon/t_r$. Moreover, during the training process, we monitor the number of rounds each client participates in. If a client has reached the maximum participating rounds (which is t_r here), he is not allowed to participate in the later training process.

4.4.3 Generating Synthetic Data and Data Post-Processing

Once the model has been trained, the server can use the decoder part to generate synthetic data. Recall that the latent space features are enforced to follow the standard Gaussian distribution p_z . Therefore, we can simply generate random latent features from p_z and feed them into the decoder. The decoder output has the same length as the encoded input described in Section 4.4.1, where each dimension is a numerical value between 0 and 1.

Finally, we need to convert the synthetic data back to categorical form. Given an output vector, we first split it into pieces of short vectors, each representing one categorical attribute. Then, for each short vector, we choose the entry with the maximum value as the attribute value. In the end, we concatenate all the categorical labels into one vector as the final synthetic data. The synthetic data will be used for data analysis and training of machine learning models.

4.5 Experiments and Results

We implemented the proposed framework and performed comprehensive experiments with a number of open-source datasets to evaluate its performance.

Algorithm 5: Training the WAE Model

Input: $M_1 \in \mathbb{R}^d$: initial global model; N : number of per-round clients; E : number of local epochs; η : local learning rate; k : number of parameters in the top- k set of each local update; T : number of global aggregation rounds; γ : global learning rate; ϵ_r : per-round privacy budget

Output: Trained WAE model M

Server executes:

```

1: for global round  $t = 1, \dots, T$  do
2:   Randomly select a group of  $N$  clients
3:   for client  $i = 1, \dots, N$  in parallel do
4:     Broadcast current global model  $M_t$ 
5:     Receive sampled sign and dimension  $s_t^i, j_t^i = \mathbf{LocalUpdate}(M_t, E, \eta, \epsilon_r, k)$ 
6:     Build sparse local update  $\hat{\Delta}_t^i = [0, \dots, 0]^d$  and set  $\hat{\Delta}_t^i[j_t^i] = s_t^i$ 
7:   end for
8:   Aggregate local updates:  $\hat{\Delta}_t = \frac{1}{N} \sum_{i=1}^N \hat{\Delta}_t^i$ 
9:   Update global model  $M_{t+1} = M_t + \gamma \cdot \hat{\Delta}_t$ 
10: end for
11: Return Global model  $M = M_{T+1}$ 

```

LocalUpdate($M_t, E, \eta, \epsilon_r, k, h$):

// Run on the client side

```

13: Initialize local model  $M_t^i \leftarrow M_t$ 
14: for epoch  $e = 1, \dots, E$  do
15:    $M_t^i = M_t^i - \eta \cdot \nabla \mathcal{L}(M_t^i, X^i)$ 
16: end for
17: Calculate local update:  $\Delta_t^i = M_t^i - M_t$ 
18: Randomly sample a sign  $s_t^i \in \{1, -1\}$  with probability  $\Pr[s_t^i = 1] = 0.5$ 
19: Dimension selection  $j_t^i = \mathbf{SignDS}(\Delta_t^i, k, \epsilon_r, s_t^i)$ 
20: Return  $s_t^i, j_t^i$ 

```

4.5.1 Experiment Setup

Datasets and WAE Models

We used four open-source datasets for evaluating the performance of our framework. Each dataset contains multi-dimensional data records, which were used for classification tasks:

- The **Census** dataset [38] contains records drawn from the 1990 United States census data, which include 68 personal attributes such as gender, income, and marriage status. We used the dataset for a classification task to determine the duration of people’s active duty service.
- The **Twitter** dataset [83] contains records with 77 attributes such as the number of discussions, average discussion length, and the number of authors, which are used to predict the number of active discussions, namely the popularity magnitude of each

Table 4.2 Datasets details

Dataset	Type	Num. Records	Num. Attributes	Domain Size
Census	Integer	2458285	68	2^{150}
Twitter	Integer	140707	78	2^{181}
Vehicle	Binary	98528	101	2^{101}
Adult	Binary	32561	124	2^{124}

Table 4.3 Structure of WAE models

Dataset	Num.Params	Model Structure
Census	76524	Input-Dense(96, relu)-Dense(24) -Dense(96, relu)-Output(sigmoid)
Twitter	94961	Input-Dense(128, relu)-Dense(36) -Dense(128, relu)-Output(sigmoid)
Vehicle	13093	Input-Dense(64, relu)-Dense(16)
Adult	16060	-Dense(64, relu)-Output(sigmoid)

instance. In our experiment, we quantified the values of each attribute into five bins. The goal was to classify the level of popularity of each instance.

- The **Vehicle** dataset [39] contains data collected in wireless distributed sensor networks. Each record has 100 attributes representing data collected from different acoustic and seismic sensors. The goal was to train a classifier for vehicle type classification.
- The original **Adult** dataset [86] contains records with 15 personal attributes such as age, occupation, education, and gender. The goal was to train a binary classifier that determines whether a person earns more than 50K a year. We used the processed version from [126], which converted the original attributes into binary features.

We present details of each dataset in Table 4.2, which include the number of records and attributes, the length of the *one-hot* encoded input, and the total domain size. Since the number of user data should be large in order to preserve data utility (which will be discussed in Section 4.7.1), in the following experiments we simulated the large-scale distributed scenario by assuming there were 5×10^4 clients, each holding two data records. Hence, we randomly sampled 10^5 records for each dataset. For datasets with more than 10^5 records (*i.e.*, **Census** and **Twitter**), we did the sampling *without* replacement.

We varied the structure of the WAE models to fit the input size of different datasets. Details of the WAE models can be found in Table 4.3. For binary datasets (*i.e.*, **Vehicle** and **Adult**), small WAE models with a latent-layer size of 16 were already sufficient to achieve satisfactory data synthesis performance. On the other hand, for the other two complex datasets that had distinctively higher domain sizes, we used larger models with a higher latent-layer size to better capture the hidden distribution and cross-attribute correlations. Moreover, it is also possible to use auxiliary data to further optimize the model structure, which we will discuss in Section 4.6.2. We also provide an example structure of the WAE model used for the **Adult** dataset (Figure 4.3), where FC represents fully-connected layers, BCE represents the binary cross-entropy and MMD represents the MMD penalty.

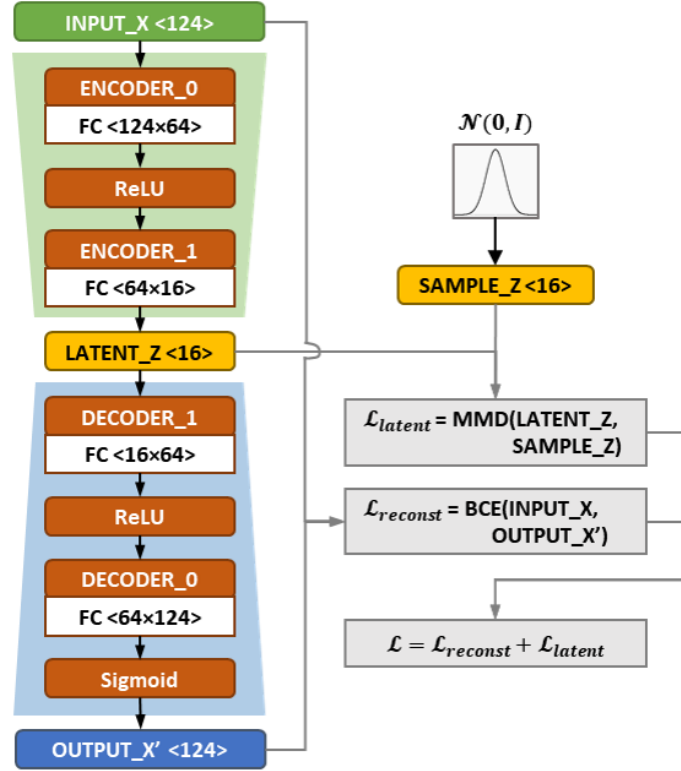


Figure 4.3 Structure of the WAE model used for **Adult** dataset.

Baseline Methods

In the following experiments, we have used LOPUB [130] and LOCOP [160] as our baseline algorithms. Both algorithms apply LDP directly on the *local data* and send the randomized data to the server. The local randomization follows the RAPPOR algorithm [44]. As derived in [130], given w as the number of attributes in local data, H as the number of hash functions and p_f as the flip probability, the overall privacy for each individual client is

$$\epsilon = 2 \cdot w \cdot H \cdot \ln((2 - p_f)/p_f). \quad (4.7)$$

Then, the randomized data will be aggregated on the server side for estimating the joint distributions and attribute dependencies. Such information will then be finally used for constructing the synthetic dataset: LOPUB generates the dependency graph based on a dependence threshold ϕ and estimates m -way joint distributions to generate the synthetic data; LOCOP leverages multivariate Gaussian copula to determine attribute dependencies and generates synthetic data by only using one- and two-way joint distributions. For both algorithms, we used the Lasso-based regression for estimating the joint distributions. In addition, we followed [130] to choose the number of the hash function $H = 4$ and the dependence threshold $\phi = 0.4$.

Evaluation Metrics

We evaluated the performance of our framework from two perspectives, namely the *data utility evaluation* and the *privacy evaluation*:

- For the data utility evaluation, we first compared the statistical distributions of synthetic data and real data. Then, we used different machine learning models to investigate the utility of synthetic data in AI training tasks. Intuitively, synthetic data with high utility should show similar statistical properties and model accuracy as real data.
- For the privacy evaluation, we investigated the capability of our framework against membership inference attacks, where an attacker aimed to use the synthetic dataset to determine whether a target data was used for training the WAE model.

Parameter Configurations

In the experiments, we assumed there were 5×10^4 clients. We set the global round $T = 5000$, and $N = 10$ clients were sampled to train the WAE model in each global round; namely, each client was sampled once during the whole training process. We set the global learning rate $\gamma = 1$ due to the good empirical performance. For local training, each client updated the model for $E = 10$ epochs. We used the Adam optimizer with a default local learning rate $\eta = 0.001$ for all the WAE models. For the local randomization, we chose the top- k ratio ξ from $\{0.05, 0.1, 0.25\}$ and the privacy budget $\epsilon \in \{0.5, 1, 2, 4, 6, 8\}$ to explore the influence of privacy on the framework performance.

It should be noted that ϵ here was the *overall* local privacy budget for each client. As mentioned in Section 4.4.2, if each client participated in t_r global training rounds, the per-round privacy budget should satisfy $\epsilon_r \leq \epsilon/t_r$. Since we assumed that each client only participated once during the whole training process, we have $t_r = 1$ and the per-round privacy budget is equal to the overall privacy budget. Moreover, we would like to emphasize that the selected ϵ values are *reasonable* local privacy guarantees for collecting w -dimensional data. Consider the privacy guarantee of the baseline algorithms (Equation (4.7)), with the number of the hash function $H = 1$ and a flipping probability $p_f = 0.5$, we already have $\epsilon = 150$ for the **Census** dataset with $w = 68$. For the **Adult** dataset with $w = 124$, the overall ϵ is even 272, which is significantly larger than our setting.

Computation Environments

We performed all the experiments on a server with an Intel E5-2470 2.40GHz CPU. In Table 4.4, we report the computational time of 1) 10 epochs of local training on each client; 2) one round of local updates aggregation and global model update on the server side; 3) generation of 10^5 synthetic data records on the server.

Table 4.4 Computation time of model training and synthetic data generation

Dataset		Adult	Vehicle	Census	Twitter
Training	Client	1.06 s	0.93 s	1.28 s	1.25 s
	Server	3.51 ms	3.05 ms	4.95 ms	4.92 ms
Data Generation		7.38 s	7.37 s	9.44 s	10.47 s

4.5.2 Evaluation for Data Utility

In this section, we evaluate the utility of the synthetic data generated using our framework in comparison to the baseline. The evaluations can be generally divided into *statistical comparison* and *AI training performance*.

Statistical Comparison

For the statistical comparison, the goal is to investigate whether the synthetic data generated by our framework can preserve the joint distributions and correlations of real data. Intuitively, synthetic data with high utility should show similar statistical properties as real data. We have respectively compared m -way joint distributions and the cross-attribute correlations to analyze the utility of the synthetic data.

Comparison of Joint Distributions For the analysis of joint distributions, we used the average total variation distance (AVD) to quantify the distribution difference between the real data and synthetic data, as suggested in [130], which is defined as

$$AVD = \frac{1}{2} \sum_{A \subseteq \mathcal{A}} |P_{real}(A) - P_{syn}(A)|, \quad (4.8)$$

where $P_{real}(A)$ and $P_{syn}(A)$ are m -way joint distributions of real data and synthetic data. More specifically, given an m -way attribute combination A with a domain size of $|\Lambda_A|$, P_{real} and P_{syn} are $|\Lambda_A|$ -dimensional vectors, where each entry is the probability of a specific value combination (namely the ratio of occurrence in the entire real or synthetic dataset). For each dataset, we randomly chose 100 combinations of m attributes and calculated the average distribution difference.

We first analyzed the AVD of all three algorithms with respect to the privacy level ϵ . For each dataset, we respectively compared the AVD of the synthetic data generated by the baseline algorithms and by our framework. In Figure 4.4, we present the results for the four-way joint distribution with different privacy budgets. The error bars represent the 95% confidence interval (also for the remaining experimental results). It can be seen that the AVD of all the algorithms decreases with the increase of ϵ . For all datasets, the synthetic data generated by our framework (referred to as *WAE*) have smaller AVD in comparison to the baseline methods (referred to as *LoPub* and *LoCop*), indicating that the synthetic data generated by our framework preserves better multivariate distributions than the baseline methods. Also, we notice that the non-binary datasets **Census** and **Twitter** usually show larger AVD in comparison to the other two binary datasets. This is due to the fact that the non-binary datasets have a larger

4 DP-FED-WAE for Private Sharing of High-Dimensional Structured Data

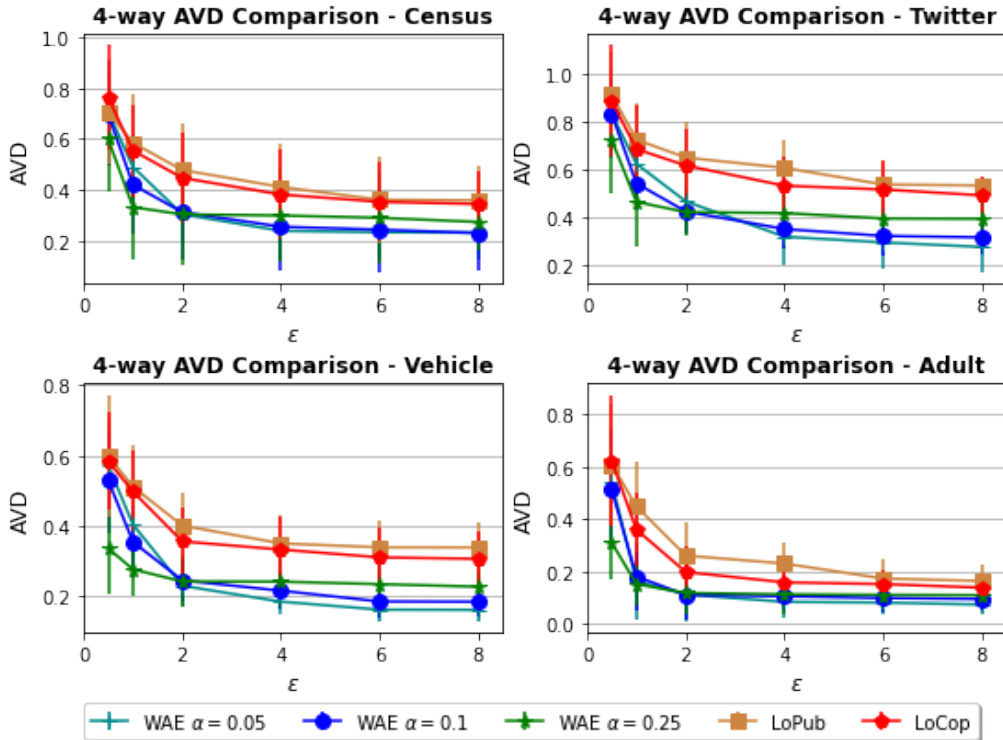


Figure 4.4 AVD of four-way joint distribution between the real and synthetic data with respect to different privacy levels.

domain size, which leads to lower frequencies of the potential attribute combinations. Therefore, it is more difficult for the generative models to find meaningful mappings between the original input space and the compact latent space, which results in a comparatively larger difference between the synthetic data and real data. Moreover, we observe that for our solution, when the privacy budget ϵ is small, the synthetic data with a larger top- k ratio have smaller AVD; while for larger ϵ , the synthetic data with smaller ξ show better utility. This complies with the discussion in Section 4.4.2. By intuition, the smaller ξ is, the better model performance is. However, when ϵ is small, the decrease of ξ leads to a significant decrease in top- k probability p , which increases the randomness of dimension selection and affects the model convergence. As ϵ increases, p is always relatively high and does not differ much regarding to ξ . In this case, a smaller ξ enhances the model performance and thus improves the utility of the synthetic data.

We further analyzed the AVD of all the algorithms with regard to the dimension of joint distributions m , in order to get a deeper insight into our framework's capability on complex statistics. For each dataset, we tested the m -way AVD where $m \in \{2, 3, 4, 5, 6\}$ and present the results under $\epsilon = 4$ in Figure 4.5. It can be seen that for all the datasets, the AVD increases with a larger m . In addition, our proposed solution consistently outperforms the baseline algorithms. More specifically, the AVD of the baseline algorithms is close to our framework when m is small, yet gets distinctively larger with an increase of m . This indicates that our framework can effectively capture the information of high-dimensional joint distributions of real data.

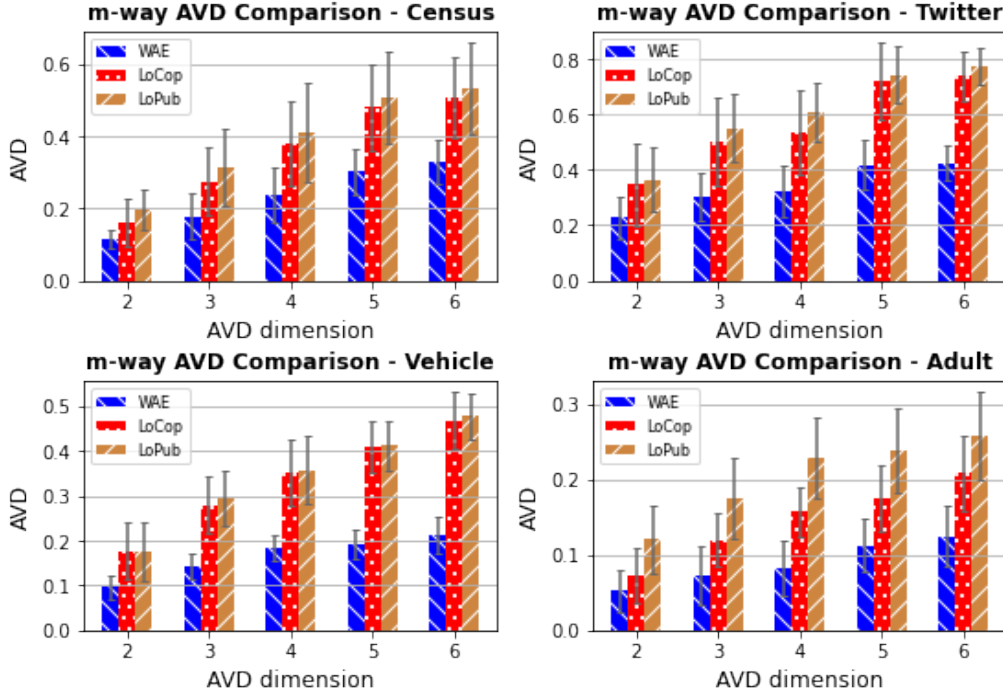


Figure 4.5 AVD of m -way joint distributions between the real and synthetic data with respect to different dimensions of joint distribution.

Comparison of Correlation For the comparison of correlation, we have respectively computed the Pearson correlation coefficient of the real and synthetic dataset and used the correlation matrix distance (CMD) [61] to measure the distance between the two correlations, which is defined as follows:

$$CMD = 1 - \frac{\text{tr}\{R_{real}R_{syn}\}}{\|R_{real}\|_2\|R_{syn}\|_2}, \quad (4.9)$$

where R_{real} and R_{syn} are correlation coefficient matrices of real and synthetic data, $\text{tr}(\cdot)$ is the matrix trace, $\|\cdot\|_2$ is the Frobenius norm. The correlation matrix distance (CMD) is bounded by $[0, 1]$, where zero means the two correlation matrices are identical.

For each dataset, we calculated the CMD of the synthetic data generated by both the baseline algorithms and our framework under different privacy levels and compared the results in Figure 4.6. It can be seen that with the same ϵ , the baseline algorithms always show a much larger CMD in comparison to the results of our framework. Although increasing the ϵ helps to reduce the CMD, it is still insufficient for preserving the multivariate correlations of real data. On the other hand, the synthetic data generated by our framework shows a distinctive decrease with the increase of ϵ . In particular, the CMD is close to zero when $\epsilon \geq 4$, indicating that the synthetic data have similar cross-attribute correlations as real data.

We further visualized the correlation coefficient matrix of real data and synthetic data with heat maps in order to better understand the capability of our method in capturing and preserving the cross-attribute correlations. Figure 4.7 shows the comparison result of the different datasets with $\epsilon = 8$ and $\xi = 0.1$. For each dataset, we present the correlations of the first 10 attributes. From the visualization results, it can be seen that the correlation of synthetic data is similar

4 DP-FED-WAE for Private Sharing of High-Dimensional Structured Data

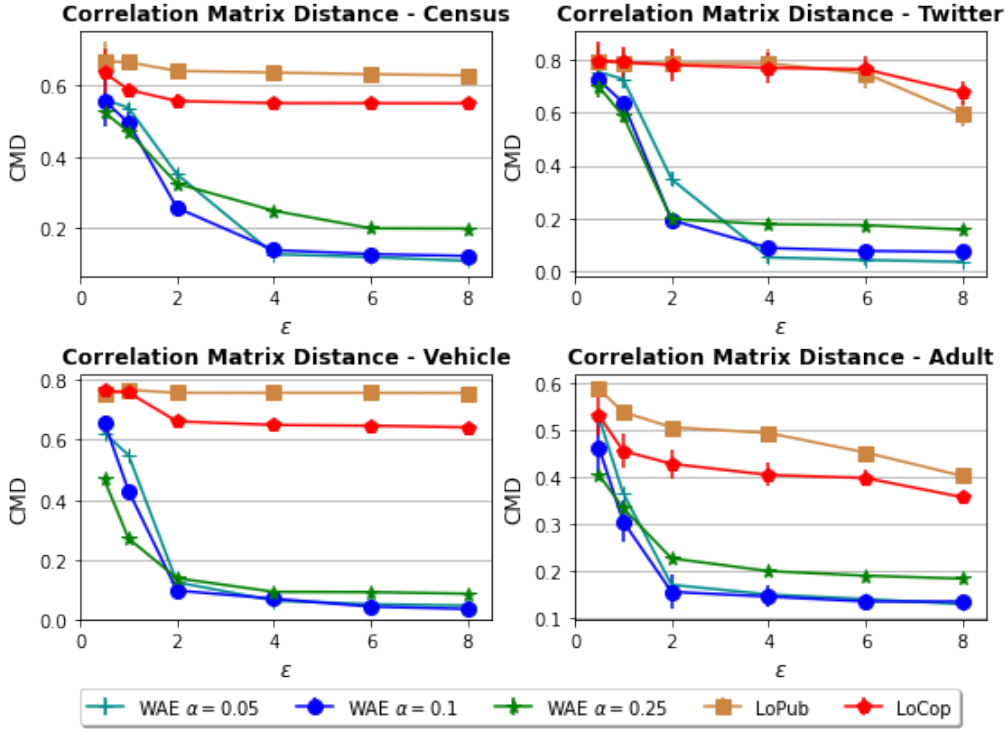


Figure 4.6 CMD between the real and synthetic data with different privacy levels.

to the correlation of real data, indicating that the synthetic data successfully preserves the attribute correlations of real data.

AI Training Performance

Next, we used different machine learning models to evaluate the utility of synthetic data in different AI training tasks. More specifically, we trained two classification models M_{real} , M_{syn} , respectively, with real data and synthetic data, and tested both models with an amount of held-out real data. Then, we compared the test accuracy Acc_{real} and Acc_{syn} , which represent the test accuracy of M_{real} , M_{syn} . If Acc_{syn} was close to Acc_{real} , we considered that the synthetic data are of high utility.

For each dataset, we used a two-layer neural network (NN) and random forest (RF) as the classification models. We trained each classification model 10 times and calculated the averaged Acc_{syn} . In Figure 4.8 and Figure 4.9, we present the results of Acc_{real} as well as Acc_{syn} evaluated on the synthetic data generated by all three methods under different privacy levels. It can be seen that the Acc_{syn} of the baselines shows, in general, a distinctive distance from Acc_{real} on both evaluation models and only has slight improvement with larger privacy budget ϵ . In comparison, the Acc_{syn} of our method consistently outperforms the baselines for both classification algorithms. With an increase of ϵ , the Acc_{syn} gradually gets close to Acc_{real} . Moreover, we observe higher Acc_{syn} with the decrease of top- k ratio ξ . In particular, with $\epsilon = 8$ and $\xi = 0.05$, the reduction of Acc_{syn} is less than 1% for the **Census** and **Adult**

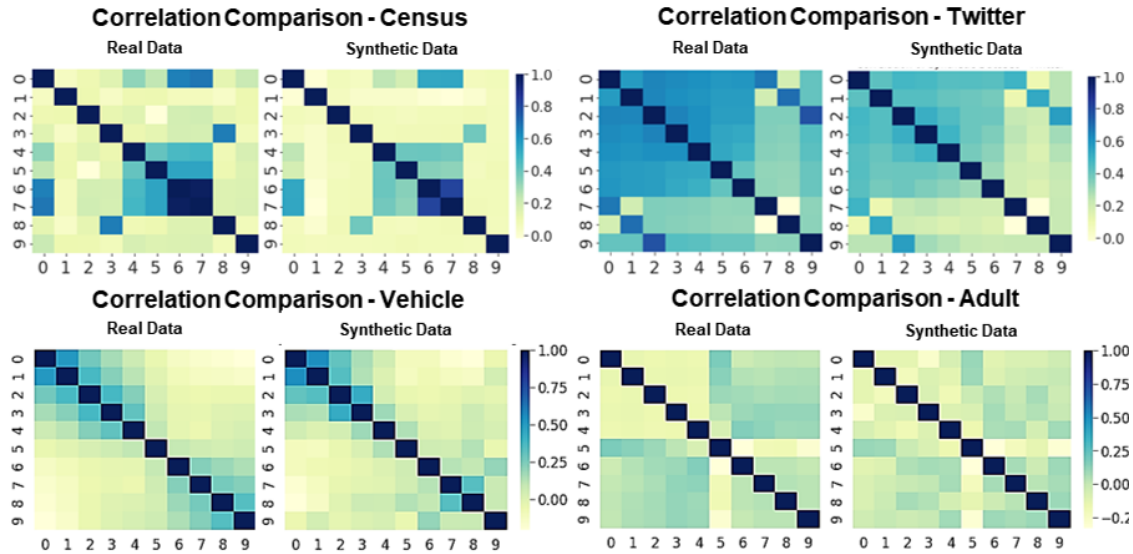


Figure 4.7 Correlation comparison between the real and synthetic data with $\epsilon = 8$ and $\xi = 0.1$. For each dataset, we present the correlations of the first 10 attributes. It can be seen that the synthetic data preserves similar correlations as real data.

dataset and less than 3% for the other two datasets. The results above further indicate that the synthetic data generated by our framework largely preserves the joint distributions and hidden correlations of real data, and can replace real data for AI training tasks.

Impact of the Number of Records

In the above experiments, we assumed a group size of 5×10^4 clients and in total 10^5 records. We further investigated how the number of records impacts the utility of synthetic data. We varied the number of records among $\{10^4, 10^5, 10^6\}$ (thus the total number of clients is respectively $\{5 \times 10^3, 5 \times 10^4, 5 \times 10^5\}$). Similar to previous experiments, we assumed that each client held two data records and only participated once during the whole training process. For the experiments with 10^4 , we set the total global rounds $T = 500$ with $n = 10$ clients for each round. For the experiments with 10^6 records, we set the total global rounds $T = 5000$ with $n = 100$ clients for each round.

We have evaluated the accuracy of both classification models (*i.e.*, two-layer NN and RF) with respect to the number of records and present the results in Figure 4.10 and Figure 4.11. Here we compare the results under a privacy of $\epsilon = 8$ (and $\xi = 0.1$ for our method). Although both algorithms show higher classification accuracy with a larger number of records, the baseline algorithms still cannot achieve significant improvements even with the largest number of records. In comparison, the classification accuracy of our method constantly outperforms the baseline algorithms. In addition, we notice that the classification accuracy in the experiments with 10^4 records is distinctively lower than others. This is because the generative model is underfitted when trained on a limited number of records and thus cannot generate high-utility synthetic data. On the other hand, although a larger number of local data (*e.g.*, 10^6) ensures the generative model is fully-trained, the model performance does not improve much after

4 DP-FED-WAE for Private Sharing of High-Dimensional Structured Data

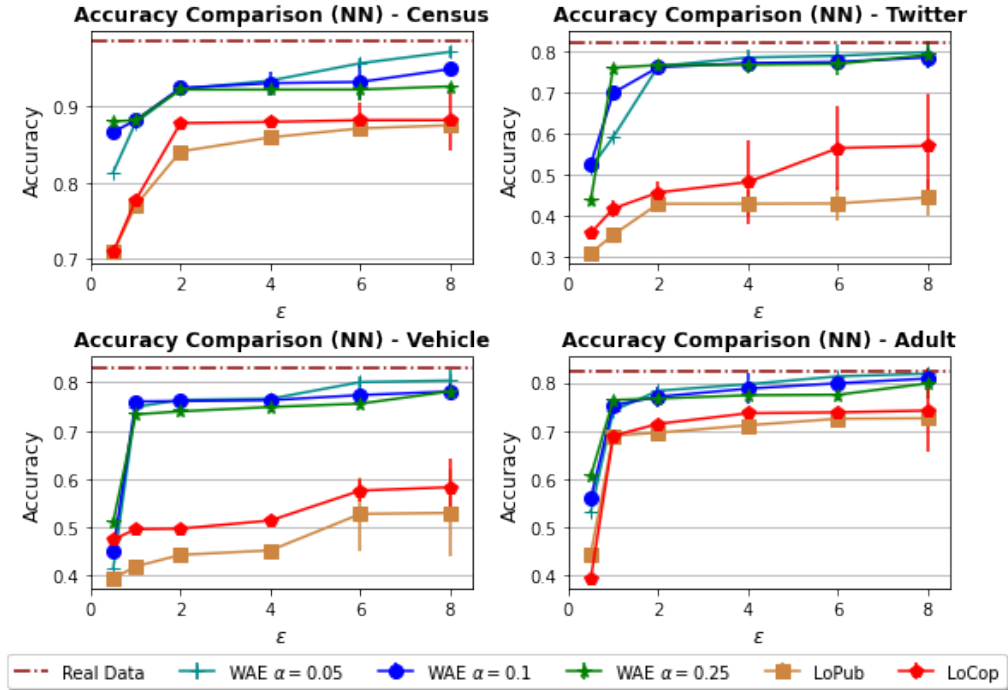


Figure 4.8 Classification accuracy of NN models trained with real data (*Real Data*) and synthetic data generated by our framework (*WAE*) as well as by the baseline algorithms (*LoPub*, *LoCop*) under different privacy levels.

achieving convergence and thus cannot reach much improvement regarding classification accuracy.

Impact of the Number of Users

In previous experiments, we assumed that there were a large number of users (*e.g.*, 5×10^4) who each only participated once during the entire training process. We further extended the scenario by assuming there were fewer clients and each client was selected multiple times for model training. To this end, we respectively assumed there were 5×10^4 , 5×10^3 , 5×10^2 clients, each with 2, 20, and 200 data records. Each client participated in 1, 10, and 100 global training rounds and the corresponding per-round privacy budget ϵ_r equals ϵ , $\epsilon/10$ and $\epsilon/100$ (ϵ is the total privacy cost).

For each dataset, we conducted experiments with the total privacy budget $\epsilon \in \{2, 4, 8\}$ and evaluated the accuracy of both classification models regarding the number of users. The results are shown in Figure 4.12 and Figure 4.13. It can be generally seen that participating in multiple training rounds can cause a distinctive impact on the framework performance and data utility, especially for datasets with larger generative models (**Census** and **Twitter**). This is because with a fixed total privacy budget the per-round privacy budget is inversely proportional to the participating rounds. Therefore, having each client participate in multiple training rounds will significantly increase the randomness injected during model training and affect the model

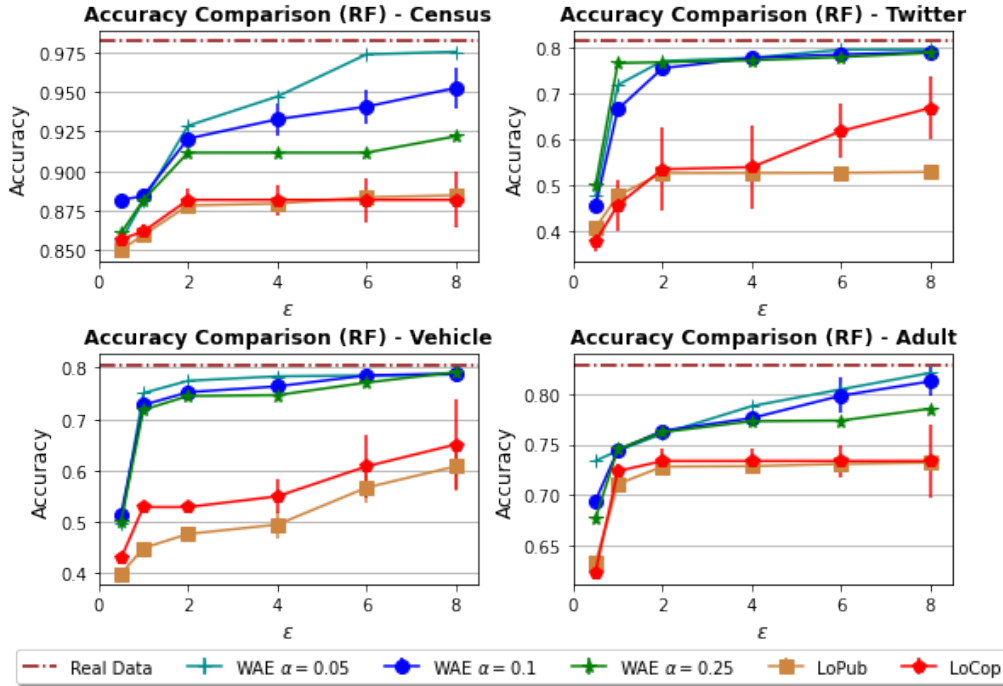


Figure 4.9 Classification accuracy of RF models trained with real data (*Real Data*) and synthetic data generated by our framework (*WAE*) as well as by the baseline algorithms (*LoPub*, *LoCop*) under different privacy levels.

convergence. Thus, we need to further increase the total privacy budget ϵ to achieve satisfying data utility.

4.5.3 Evaluation for Privacy Protection

Although a larger privacy budget ϵ has a distinctive contribution to data utility, this may be at the expense of privacy. Currently, there is no concrete understanding of how to choose an appropriate ϵ in practice for a satisfactory utility-privacy trade-off. In this section, we have empirically analyzed the privacy protection capabilities of our framework against the membership inference attack (MIA). We followed the MIA protocol of [138]. The protocol assumed that the attacker holds a reference dataset that shares similar distributions as the real training data. The attacker respectively trains a pair of generative models G_{in} and G_{out} using the reference data *with* and *without* the target record. Then, an attack model is trained to distinguish the synthetic data generated by G_{in} and G_{out} , which can be considered as a binary classification task. Finally, given the published synthetic dataset, the attacker can use the attack model to test whether the synthetic data is generated by a model trained with the target record, namely whether the target record is included in the generative model's training dataset.

We randomly picked 30 records as the target record. For each target record, we trained generative models under different privacy settings and repeated the attack 10 times. In each attack trial, we used a list of machine learning models such as support vector machines, logistic regression models, KNN models, RFs, and NNs as the attackers and picked the highest attack

4 DP-FED-WAE for Private Sharing of High-Dimensional Structured Data

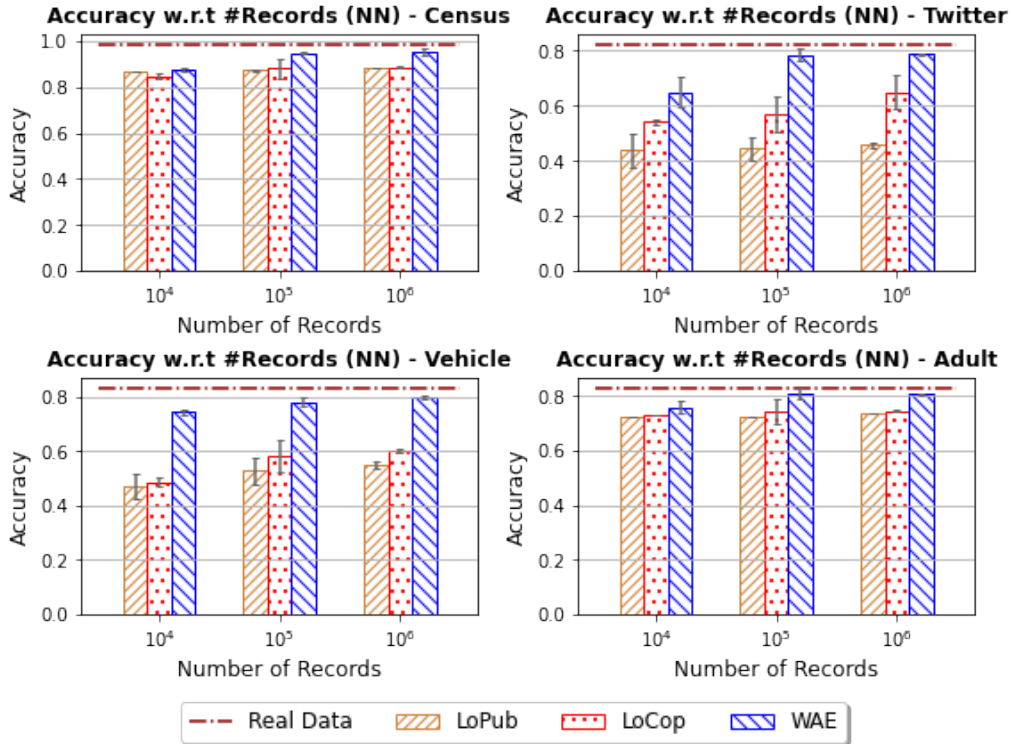


Figure 4.10 Classification accuracy of NN models with different numbers of records under the privacy level of $\epsilon = 8$.

Table 4.5 Accuracy of membership inference attack

Dataset	Census	Twitter	Vehicle	Adult
No Privacy	0.735	0.698	0.637	0.642
$\epsilon = 8$	0.574	0.547	0.546	0.535
$\epsilon = 2$	0.548	0.529	0.513	0.519
$\epsilon = 0.5$	0.529	0.524	0.507	0.506

accuracy over all the attackers. Finally, we compute the averaged attack accuracy against all the target records under different privacy settings and present the results in Table 4.5. It can be seen that synthetic data generated by the non-private generative model is more likely to reveal the membership information of the target record. The attack accuracy of all the datasets is more than 60% and even up to 73.5% for the **Census** dataset. On the other hand, applying DP can effectively reduce the attack accuracy. With $\epsilon = 0.5$, the attack accuracy is reduced by 13% ~ 20%. Even with $\epsilon = 8$, the attack accuracy can still be reduced by 8% ~ 16% and is close to 50%, namely the accuracy of a random guess. The results demonstrate that our framework is able to reduce the risk of membership inference attacks and provide privacy protection to the local data.

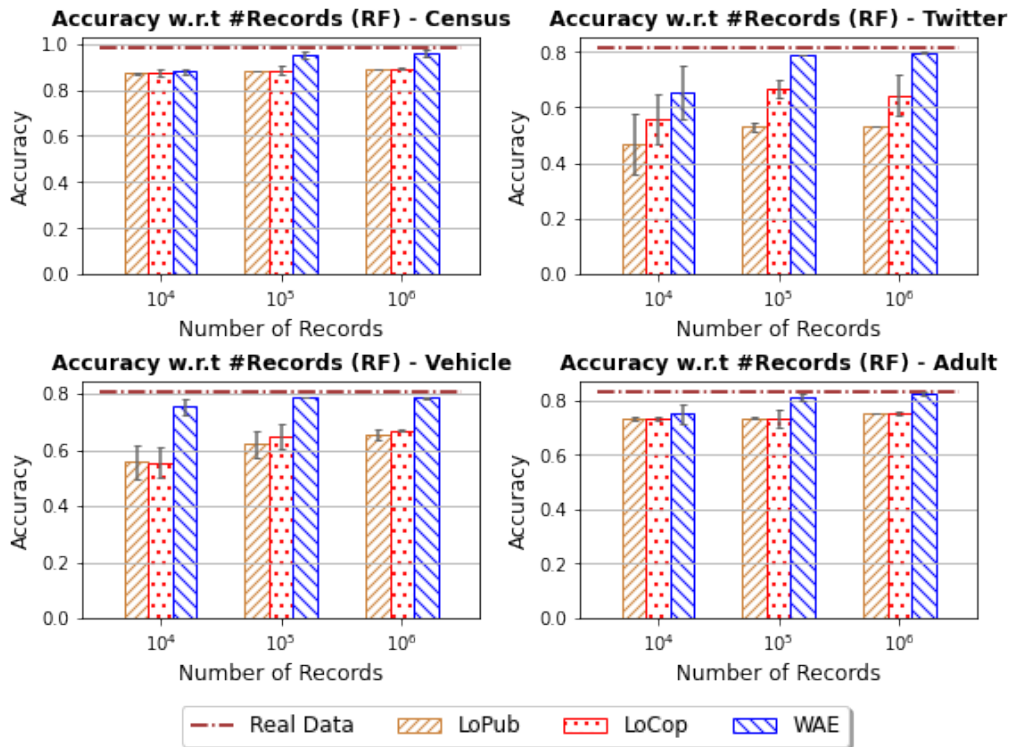


Figure 4.11 Classification accuracy of RF models with different numbers of records under the privacy level of $\epsilon = 8$.

4.6 Discussion

4.6.1 Extension to Other Data Types

In this paper, we demonstrated that our framework performs well on high-dimensional categorical data. In order to do so, we converted the categorical data into numerical form for training the WAE model and then reversed the model's numerical outputs back to categorical form. In future studies, it is also possible to modify the current model structure and the loss function in order to extend the framework for supporting other data types, image data, and text data. For instance, for image data, we can further apply convolution layers to enhance the feature extraction capability and use the mean squared error instead of the cross-entropy to measure the reconstruction distance. Despite the variation of the generative models, the main idea of training the model under privacy-preserving federated learning and generating synthetic data remains unchanged. In Figure 4.14, we give the synthesis results evaluated on the **MNIST** [89] and **Fashion-MNIST** [168] dataset. For each dataset, we show the synthetic data produced by generated models trained under different privacy settings, namely, non-private, with privacy of $\epsilon = 8$ and $\epsilon = 4$. Note that the synthetic data are randomly generated and may look different from real data. However, it can be observed that our framework is also capable of synthesizing image datasets, and generated images have better quality with an increase of the privacy budget.

4 DP-FED-WAE for Private Sharing of High-Dimensional Structured Data

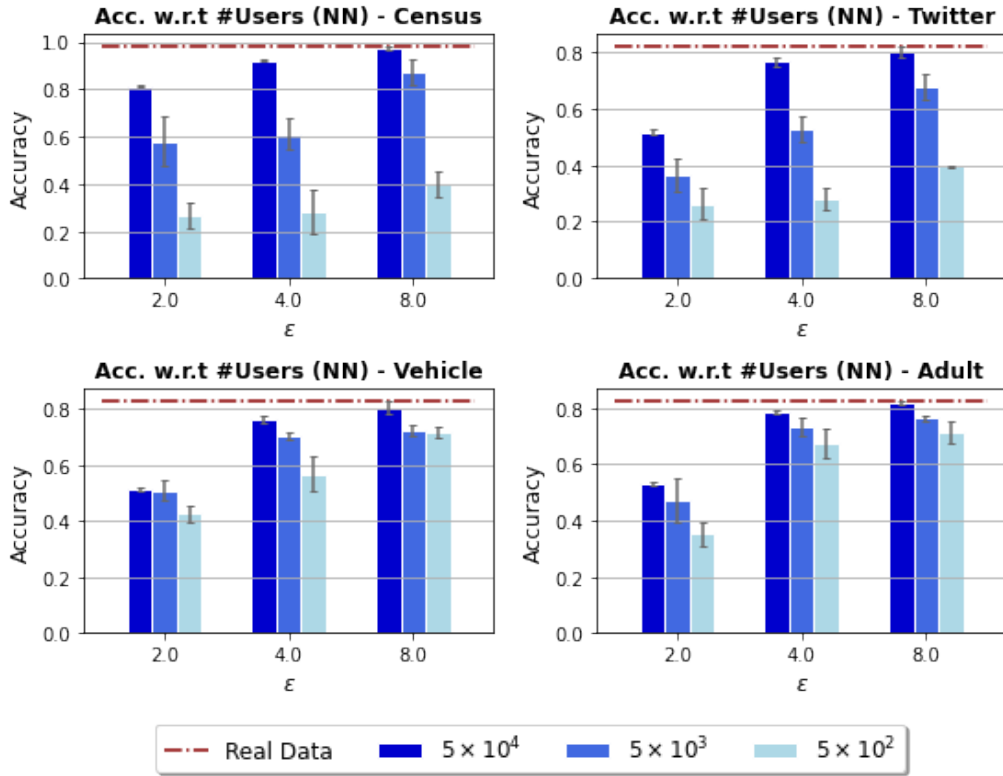


Figure 4.12 Classification accuracy of NN models with different numbers of users under different privacy levels.

4.6.2 Auxiliary Data for Pre-Training

Before applying the WAE model for collecting local user data, the server needs to design the model structure. An appropriate model structure helps to enhance the capability of capturing the local data distributions and thus the utility of synthetic data. In our scenario, the server only knows the *basic properties* of the data to be collected, such as the number of attributes and the domain of each attribute. The server can thus use some auxiliary data to optimize the model structure. The auxiliary data here refer to certain public datasets or the random data generated by uniform sampling from the domain of the local data. The server can use such data to simulate the data collection process and tune the model structure by evaluating the utility of the synthetic data. Moreover, the auxiliary data can also be used for pre-training the WAE model before applying the model in the data collection process, so as to improve the model convergence and the utility of synthetic data.

In Table 4.6, we compare the utility of synthetic data generated by WAE models with (w) and without (w/o) pre-training under the setting of $\xi = 0.1$ and $\epsilon \in \{4, 8\}$. For each dataset, we have randomly generated an auxiliary dataset only using the *basic properties* of real data, as mentioned before. We used the auxiliary dataset to pre-train the WAE model and applied the pre-trained model to the data collection process. We respectively evaluated the utility of synthetic data generated in both scenarios based on the classification accuracy of NNs and RFs. For both types of models, we observe that the synthetic data generated by pre-trained

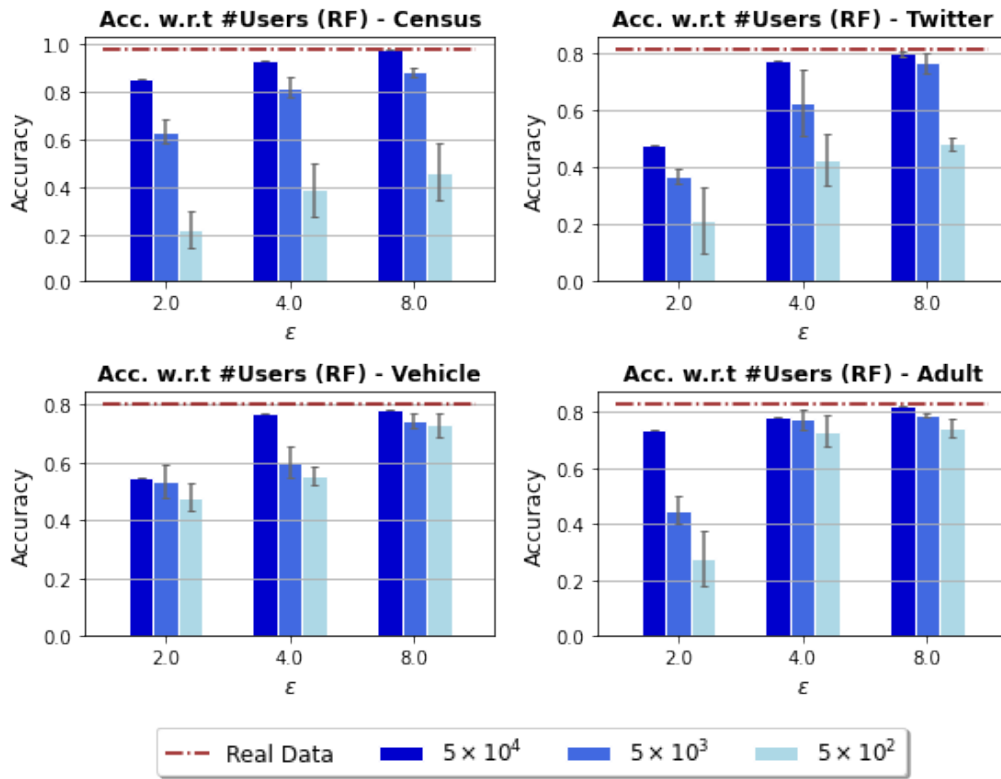


Figure 4.13 Classification accuracy of RF models with different numbers of users under different privacy levels.

WAE models achieve 1 ~ 2% increase in classification accuracy. The results demonstrate that using auxiliary data to pre-train the WAE model is feasible to enhance the model convergence in the data collection process and further improve the synthetic data utility.

4.7 Related Work

4.7.1 Data Collection Under Local Differential Privacy

Differential privacy (DP) [42], as a strong mathematical formalization of privacy, has been used as a criterion for privacy protection in data publishing, data mining, and machine learning ([41, 179, 1]). However, traditional DP assumes a trusted server (data curator), who first collects the original user data, and then performs data analysis under differential privacy. In order to eliminate the assumptions of trustworthy servers, LDP [82] has been proposed, which provides strong privacy guarantees to local data. By utilizing local randomization algorithms, the server cannot infer any individual's original data but can learn the overall statistics of the whole population. However, prior research on LDP mainly focuses on collecting one-dimensional statistics, such as frequency estimation ([44, 35]), heavy-hitter identification ([10, 16]), and itemset mining ([128, 161]). Regarding scenarios with multi-dimensional data, Alaggar et al. [3, 2] proposed using Bloom filters to encode local data and analyze aggregate

4 DP-FED-WAE for Private Sharing of High-Dimensional Structured Data

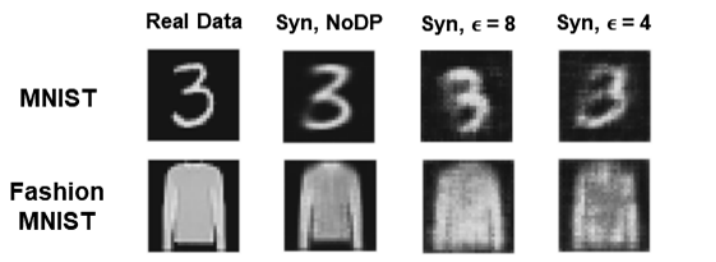


Figure 4.14 Results of data synthesis on image datasets.

Table 4.6 Classification accuracy of synthetic data generated by WAE models with (w) and without (w/o) pre-training

Dataset		Accuracy - NN		Accuracy - RF	
		w/o	w	w/o	w
Census	$\epsilon = 8$	0.948	0.960	0.952	0.965
	$\epsilon = 4$	0.929	0.935	0.932	0.940
Twitter	$\epsilon = 8$	0.786	0.798	0.788	0.796
	$\epsilon = 4$	0.771	0.782	0.778	0.785
Vehicle	$\epsilon = 8$	0.781	0.795	0.788	0.794
	$\epsilon = 4$	0.762	0.789	0.763	0.782
Adult	$\epsilon = 8$	0.808	0.815	0.812	0.820
	$\epsilon = 4$	0.787	0.798	0.775	0.789

gated statistics. However, their works do not involve the estimation of joint distributions and cross-attribute correlations, which differs from our objectives.

Directly applying the above LDP-based algorithms to estimate complex statistics of high-dimensional data will cause extremely large communication overhead as well as a degradation in data utility. Consider, for example, RAPPOR [44], a state-of-the-art LDP-based data collection algorithm. For w -dimensional binary data with $w = 32$, we have domain size $|\Lambda| = 2^{32} \approx 4.3 \times 10^{10}$. Directly applying RAPPOR consumes a communication cost and a storage space of $O(|\Lambda|)$ [130]. Also, for high-dimensional input domains, it is common for each user to have a unique feature combination. Therefore, it is essential to collect a large number of data in order to cover all the possible combinations in the feature domain. Given a domain size $|\Lambda|$, as a general rule of thumb [44], the number of user data N should follow $\sqrt{N}/10 \geq |\Lambda|$. In the above example, $N \geq 100 \cdot 2^{64} \approx 1.8 \times 10^{21}$. All of these requirements are impractical for real-world applications. In subsequent research, Fanti et al. [45] proposed to separately collect data of each dimension under RAPPOR and estimate the joint distributions using expectation maximization (EM). Although the algorithm significantly reduces the communication overhead between clients and the server, it only supports to estimate the joint distribution of two attributes. Based on [45], Ren et al. proposed LOPUB [130], which reduces w -dimensional data to m -dimensional clusters ($m < w$) using dependence graphs and estimates m -way joint distributions with an EM-based and Lasso regression-based approach. However, the algorithm still suffers from high computational complexity and low data utility when m is large. An improved scheme, LOCOP [160] was further proposed, which lever-

ages multivariate Gaussian copula to estimate cross-attribute dependencies and to construct synthetic data.

Instead of directly randomizing the local data, our framework uses deep generative models to learn the data distributions and to generate synthetic data without accessing real data, which effectively enhances data utility. In our experiments, we used LOPUB and LOCOP as the baselines to compare the frameworks' performance in terms of data utility.

4.7.2 Differentially Private Synthetic Data Generation

Differentially private synthetic data generation has been extensively studied over recent years as an alternative solution to privacy-preserving data publishing. Previous works ([25, 179]) analyzed statistical distributions of original data under differential privacy and used them to generate synthetic data. Later works have proposed using differentially private generative models ([122, 145]) to directly generate high-utility synthetic data. However, these works only focus on the centralized setting, where the server has already collected the real data and uses them to generate private synthetic data. In contrast, our approach is practical for a distributed setting, where the server cannot access the real data, but is interested in learning statistical information about the data.

Recent works by Augenstein et al. [7] and Triastcyn et al. [146] also investigate the synthetic data generation under the distributed setting. [7] aims to use generative models to detect errors and bugs in local data. However, as they claimed, such applications do not require high-fidelity generation. On the other hand, although [146] focused on generating and publishing synthetic data, their method is only limited to image data. In addition, they adopted a weaker measure of privacy to preserve the model performance. In comparison to both works, our framework is able to generate synthetic data with high utility and fidelity, which can replace real data in data mining and AI training tasks. Moreover, we apply strict LDP randomization on the client side, which provides strong privacy guarantees for clients' local privacy.

4.7.3 Efficiency and Privacy in Federated Learning

Communication Efficiency in Federated Learning

It is widely acknowledged that communication cost can be a bottleneck for FL, especially when training high-dimensional models. Recently, a number of *upload link* and *download link* compression methods have been proposed to alleviate the communication cost in FL.

The *upload link* compression applies quantization or sparsification on the model updates to reduce the communication cost from clients to the server. The main idea of quantization is to reduce the number of bits of update values. For instance, Seide *et al.* [134] proposed the 1-BIT SGD, which quantizes the update values larger than a pre-defined threshold to 1 and the rest to 0. Similarly, Bernstein *et al.* [12] proposed SIGNSGD and SIGNUM, where the update values are quantized to their sign values. The study theoretically proved that SIGNSGD can effectively reduce the communication cost while enjoying a satisfying convergence rate. In contrast, sparsification aims to transmit only a subset of update values. For instance, the top- k mechanisms (*e.g.*, [37, 4]) only keep the top- k largest magnitude values of each model

update and set the others to 0. Stich *et al.* [139] proposed a similar scheme with memory. Ivkin *et al.* [68] further proposed to use the count-sketch algorithm to approximately select the top- k updates.

On the other hand, reducing the communication cost of the *download link* has recently also gained increasing attention. Caldas *et al.* [19] gave the first attempt at the *download link* compression and proposed FEDERATED DROPOUT, which extracts small sub-models from the original high-dimensional and sends to the client side for local training. Based on this idea, Bouacida *et al.* further proposed [15] an adaptive federated dropout algorithm, which builds the sub-models using an adaptive activation score map. In addition, Jiang *et al.* [76] proposed to gradually prune the global model during the training process to achieve better communication and computational efficiency without loss of accuracy.

Privacy Protection in Federated Learning

Although FL enjoys significant privacy benefits in comparison to centralized learning, recent works showed that FL is still vulnerable to various privacy attacks [51, 115] against the exchanged local updates and the global model. Thus, an increasing number of studies propose to incorporate DP into the FL framework. Some works (*e.g.*, [110, 7]) add Gaussian noise on the server side to protect the privacy of the global model. However, such solutions cannot prevent privacy leakages from the local updates. Thus, other works propose hybrid frameworks (*e.g.*, [69, 147, 43]), which use crypto-based solutions such as HE and SMC to further achieve local privacy protection. However, these solutions require extra communication and computation costs during the key-distribution phase and thus, may not be practical in large-scale scenarios.

Considering the privacy and efficiency issues in the above-mentioned solutions, a more practical solution is to apply LDP to FL, which perturbs the local updates before sending them to the server. Previous LDP-FL frameworks (*e.g.*, [40, 158, 184]) perturb the local updates using private mean estimation algorithms. However, these algorithms evenly split the privacy budget across dimensions and the injected noise is proportional to the model dimension, making them only applicable to simple ML models. A recent work proposed FEDSEL [99], a *two-stage* LDP-FL framework that includes a DS stage and a VP stage. The DS stage first sorts the local update by *absolute value* and then privately selects one "important" dimension from the top- k dimension set (namely, the set of k dimensions with the largest absolute values). Then, in the VP stage, the value of the selected dimension is perturbed via the LDP algorithms in [158] and used to construct a sparse privatized local update. Although [99] mitigates the dimension-dependency problem by only selecting one "important" dimension, the privacy budget is still consumed by the two stages. In high-privacy scenarios, each stage may therefore obtain only an insufficient privacy budget and cause large randomness.

Inspired by the effectiveness of SIGNSGD [12] and top- k sparsification ([99, 37, 4]), we propose a novel local randomization algorithm called SIGNDS. The main idea is to save the privacy budget for the VP stage in [99] by assigning sign values instead of the perturbed dimension values to the selected dimensions. With the same privacy budget, we can achieve less randomness and thus higher model utility.

4.8 Conclusion

Building effective business and AI services requires the collection of personal data, which often introduces challenges related to an insufficient amount of data on the one hand, and privacy violations on the other hand. Recently, deep generative model-based data synthesis techniques have created opportunities for addressing these challenges: the generative models have strong capabilities in capturing the cross-attribute correlations of real data and can easily generate large-scale high-utility data; in addition, since the generated data are fully synthetic and cannot be linked to any particular individual, re-identification attacks or attribute disclosure becomes almost impossible.

In this work, we have followed the idea of data synthesis and proposed DP-FED-WAE, a privacy-preserving framework for high-dimensional data collection. The framework utilizes a (generative) Wasserstein autoencoder to learn the joint distributions and correlations of high-dimensional user data and generate high-utility synthetic data on the server side. Moreover, we applied a novel LDP-FL framework for training the autoencoder, which not only avoids the collection of real local data but also provides strong local privacy guarantees. Experimental evaluation with real-world datasets shows that our framework significantly outperforms the LDP-based baseline algorithms for high-dimensional data collection and synthesis. The synthetic data generated by our framework preserves very similar statistical properties as real data and can replace real data for data mining and model training tasks.

5 FEDSTDG for Private Sharing of Multivariate Time Series

Table 5.1 Bibliographic details for P3

Title	Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning
Authors	Xue Jiang ^{1,2} (xue.jiang@tum.de) Xuebing Zhou ² (xuebing.zhou@huawei.com) Jens Grossklags ¹ (jens.grossklags@in.tum.de) ¹ Technical University of Munich (TUM), Boltzmannstraße 3, 85748 Garching, Germany ² Huawei Munich Research Center, Riesstraße 25, 80992 Munich, Germany
Type	Working Paper
Status	Under review at time of thesis submission.
Copyright	Authors retain copyright of this working paper.
Author Contributions	Xue Jiang developed the core idea for the paper and took primary responsibility for designing the methodology, acquiring data, implementing experiments, evaluating results, and drafting the manuscript. Xuebing Zhou and Jens Grossklags provided valuable feedback and suggestions on the methodology and experiments, and assisted in reviewing and enhancing the manuscript.

Note: Since submission of the thesis, this working paper (in slightly modified form) has been accepted and published as: Xue Jiang, Xuebing Zhou, & Jens Grossklags (2024) Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning. *IEEE Access*, 12:157067–157082. DOI: <https://doi.org/10.1109/ACCESS.2024.3481652>. The published paper is licensed under a Creative Commons Attribution - Non-Commercial - NoDerivatives 4.0 License.

5.1 Abstract

Developing effective data analysis tools and AI models for time-series data usually faces data insufficiency and privacy issues. While synthetic data generation (SDG) has been considered a promising technique for addressing these challenges, prior works assume that the central server can directly collect the real client data. However, this may not always be achievable in real-life scenarios, as the clients may be unwilling to share their private data.

In this paper, we consider a more realistic setting where the clients' local data are *inaccessible* to the server. We propose FEDSTDG, the first privacy-preserving framework for *distributed* multivariate time-series synthesis. With the combination of federated learning (FL), local differential privacy (LDP), and a recurrent generative autoencoder, our framework achieves learning the temporal correlations and distributions of local multivariate time series *without* collecting the real data. The trained autoencoder will be used to generate high-utility synthetic data on the server side. We further propose a novel SUBMDS algorithm for privatizing the local model updates during FL training, which achieves a remarkable utility improvement compared to the previous LDP-FL algorithms. Additionally, we also propose MAGRR, a privacy-preserving algorithm that adaptively adjusts the FL learning rate for better model convergence. We evaluate the performance of our framework on a number of real-world datasets. Experimental results on four open-source datasets demonstrate the capability and efficiency of our framework in generating high-quality time-series data. Under the same privacy level, the synthetic data generated using the SUBMDS algorithm yields around 20% ~ 85% reduction in the downstream prediction error compared to using previous LDP-FL algorithms. Additionally, empirical analysis of privacy and adversarial attacks shows that our framework can effectively improve privacy protection and robustness in the FL process.

5.2 Introduction

With the rapid development of network and computer technologies, large and diverse quantities of person-specific data are frequently generated on local devices such as smartphones and IoT sensors. Besides the widely-explored tabular data and image data, there has also been an increasing number of studies in modeling time-series data in finance [107], healthcare [178], and IoT [31] applications. These data usually consist of a series of measurements gathered periodically over time. Data analysts can use the rich information contained in these time-series data to explore the hidden temporal distributions and correlations among attributes from different perspectives and develop algorithms for classification and forecasting tasks. For instance, a digital healthcare application may utilize users' physical information for health monitoring and disease predictions, while a location-based service may use the drivers' historical trajectory to forecast potential future locations. However, the development of such AI services usually faces two major challenges: data insufficiency and privacy issues. On the one hand, building reliable machine learning (ML) models usually requires adequate training data to prevent overfitting and achieve satisfactory performance. On the other hand, directly using real personal data for training the ML models may cause severe privacy problems and may even violate legal requirements (e.g., in the context of the GDPR).

In recent years, synthetic data generation (SDG) has been considered one of the potential techniques for addressing the above challenges in business intelligence & AI services. Extensive studies have proposed to use generative models to generate different types of synthetic data, such as tabular data [122], images [129], and time series [177] for a variety of real-world applications. Such techniques enjoy advantages in both data utility and privacy protection. The well-designed generative models have strong capabilities for capturing the joint distributions and hidden correlations of real data and can flexibly produce high-fidelity synthetic data for data analysis and building accurate ML models at a large scale. Also, the generated data are fully synthetic, which effectively reduces the risks of re-identification attacks or attribute disclosure [122]. Some later works [170, 77] also incorporate differential privacy (DP) into the training process to provide formal privacy guarantees to the algorithms. Nevertheless, these previous works only consider the synthetic data generation (SDG) under the centralized setting, where the real data are already collected by a data curator (referred to as the *cloud server*) and will be used to train the generative models for privacy-preserving data publishing. This may not always be realistic since the data owners (referred to as *local clients*) may be reluctant to share their personal data with untrusted servers. To address this problem, some recent works [7, 146, 74] propose solutions for *distributed SDG*. The generative models are trained using the federated learning (FL) mechanism [108], which only exchanges model parameters and keeps the real data on the local side. However, existing *distributed SDG* solutions only focus on structured data and images (see [98], Table 2), which cannot be directly applied to time-series data.

In this paper, we offer the first attempt at *distributed SDG* of time-series data. Inspired by prior work conducted by Jiang et al. [74], we propose FEDSTDG, an efficient and privacy-preserving framework that achieves synthesizing the clients' local time-series data *without* the collection of real data. Our framework shows several advantages compared to [74]. To start with, we extend the Wasserstein autoencoder (WAE) architecture used in [74] into a recurrent Wasserstein autoencoder (RWAE), which can better capture the temporal distributions and correlations of multivariate time-series data. The model is then trained under the federated learning setting to learn the distribution of local data without collecting the raw data. Finally, the trained model can be used to generate high-fidelity synthetic data on the server side. In addition, [74] proposed a local differential privacy (LDP) algorithm, SIGNDS, to prevent privacy leakage for the local model updates. However, for each local update vector, the algorithm only selects one dimension index of an "important" parameter to send to the server, which suffers from a particularly slow convergence speed with large models. A follow-up study [75] introduced EM-MDS, a privacy-preserving algorithm that employs the exponential mechanism to select multiple dimensions under LDP guarantees. In this paper, we propose enhancements to EM-MDS from two perspectives. Firstly, we introduce an improved multi-dimensional selection algorithm called SUBMDS, which applies parameter subsampling to enable the selection of more dimension indices under the same privacy guarantee. Additionally, we present MAGRR, which adaptively adjusts the FL learning rate in a private manner and facilitates a rapid and stable model convergence. By incorporating both algorithms, our framework demonstrates superior model convergence and synthetic data utility in comparison to prior methods.

Our major contributions can be summarized as follows:

- We propose FEDSTDG, a privacy-preserving framework for the distributed synthesis of multivariate time-series data. With the combination of LDP, FL, and a RWAE, our framework enables the learning of local time-series data distributions and the generation of high-fidelity synthetic data *without* the need for centralized data collection. To the best of our knowledge, this is the first work of synthetic time-series data generation under a *distributed* setting.
- We introduce two innovative approaches, SUBMDS and MAGRR, as improvements to existing LDP-FL algorithms. The former applies parameter subsampling to enable the selection of more dimension indices, while the latter adaptively adjusts the learning rate to achieve a rapid and stable model convergence. The integration of both algorithms leads to superior synthetic data utility compared to existing methods.
- We use a number of real-world time-series datasets to evaluate the performance of our framework. Extensive evaluation results demonstrate that our framework has superior capability and efficiency in synthesizing time-series data while achieving satisfactory privacy protection and robustness.

The remainder of the paper is organized as follows. In Section 5.3, we discuss prior work on synthetic data generation and privacy-preserving FL. The problem statement is described in Section 5.4 and our proposed framework is introduced in detail in Section 5.5. The evaluation experiments and results are then presented in Section 5.6. Finally, conclusions are presented in Section 5.7.

5.3 Related Work

5.3.1 Differentially Private Synthetic Data Generation

Differentially private synthetic data generation has been extensively studied over recent years as an alternative solution to privacy-preserving data publishing. Previous works [92, 179] analyzed statistical distributions of original structured data under DP and used them to generate synthetic data. Later works proposed to use DP generative models such as generative adversarial networks [53] and autoencoders [85, 144] to directly generate high-utility synthetic data, which can be flexibly applied to tabular data [77], images [23], and time series [47]. However, prior SDG solutions mainly focus on the centralized setting, where the server has already collected the real clients' data for generating private synthetic data. This may not always be realistic since the clients may refuse to share their personal local data with untrusted servers. In order to address the problem, some recent works [7, 146] introduced solutions for distributed SDG. The generative models are trained using the FL mechanism, which only exchanges model parameters and keeps the real data on the local side. However, existing distributed SDG solutions only focus on structured data and image data. In this paper, we fill this research gap and propose the first framework for synthetic *time series* data generation. The framework learns the spatial-temporal distributions of raw local data and generates synthetic time series on the server side to support downstream data analysis tasks.

5.3.2 Privacy-Preserving Federated Learning

Although FL enjoys significant privacy benefits in comparison to centralized learning, recent works showed that FL is still vulnerable to various privacy attacks against the local model updates [51] and the final global model [115]. A number of existing works propose to incorporate crypto-based solutions such as homomorphic encryption (HE) [147] and secure multi-party computation (SMC) [69] into FL for protecting the local model updates. However, these solutions cannot be well-scaled due to the extra communication and computation costs during the encryption and key-distribution phase. Some other works proposed to apply LDP to FL, where the local updates are perturbed before being sent to the server. However, previous private mean estimation-based LDP-FL frameworks (*e.g.*, [40, 158, 184]) easily suffer from significant utility loss, as the injected noise in these algorithms is, in essence, proportional to the number of model parameters. To address the issue, some recent works [99, 74] proposed dimension selection-based solutions, which only select one "important" dimension index for each local update vector under LDP guarantees and use the shared indices on the server side to construct corresponding sparse updates to update the global model. Nevertheless, both algorithms only select one dimension for each local update, which may lead to slow model convergence, especially for high-dimensional models. A follow-up work [75] introduced EM-MDS, an advanced LDP multi-dimension selection algorithm. Rather than repetitively conducting single-dimension selection with evenly divided privacy budgets, the algorithm treats each group of indices as a complete entity and adopts the exponential mechanism to assign higher probabilities to subsets containing more "important" indices. In this paper, we further enhance the existing algorithm with a reduced input size and plan an adaptive learning rate, which achieves better model utility under the same privacy guarantees.

5.4 Problem Statement

In this paper, we consider a scenario where a number of clients hold multivariate time-series data on the local side. Such data can be, for instance, clients' daily activities collected by wearable devices, or vehicle trajectories collected during driving. A central server aims to investigate the distribution and correlations of these time-series data and generate similar synthetic data for data analysis and designing AI services. Here, we assume the server to be *honest-but-curious*, who follows the system protocols but tries to infer sensitive information of local users. Hence, to protect local privacy, we require the server *not to* have direct access to raw local data.

The problem can be formulated as follows: given a private multivariate time-series dataset $\mathbf{X}_{1:L}$ with W attributes and L time steps, each sample $\mathbf{x}_{1:L}^i \in \mathbf{X}_{1:L}$ is denoted as

$$\mathbf{x}_{1:L}^i = \begin{bmatrix} F_1^i \\ \vdots \\ F_L^i \end{bmatrix} = \begin{bmatrix} f_{1,1}^i & \cdots & f_{1,W}^i \\ \vdots & \ddots & \vdots \\ f_{L,1}^i & \cdots & f_{L,W}^i \end{bmatrix}, \quad (5.1)$$

where F_l^i represents the vector of multivariate features of the l^{th} timestamp and $f_{l,w}$ is the feature of the w^{th} attribute. Note that each record has the same number of W attributes. Assume the private dataset is distributed among N local users. A central server aims to

5 FEDSTDG for Private Sharing of Multivariate Time Series

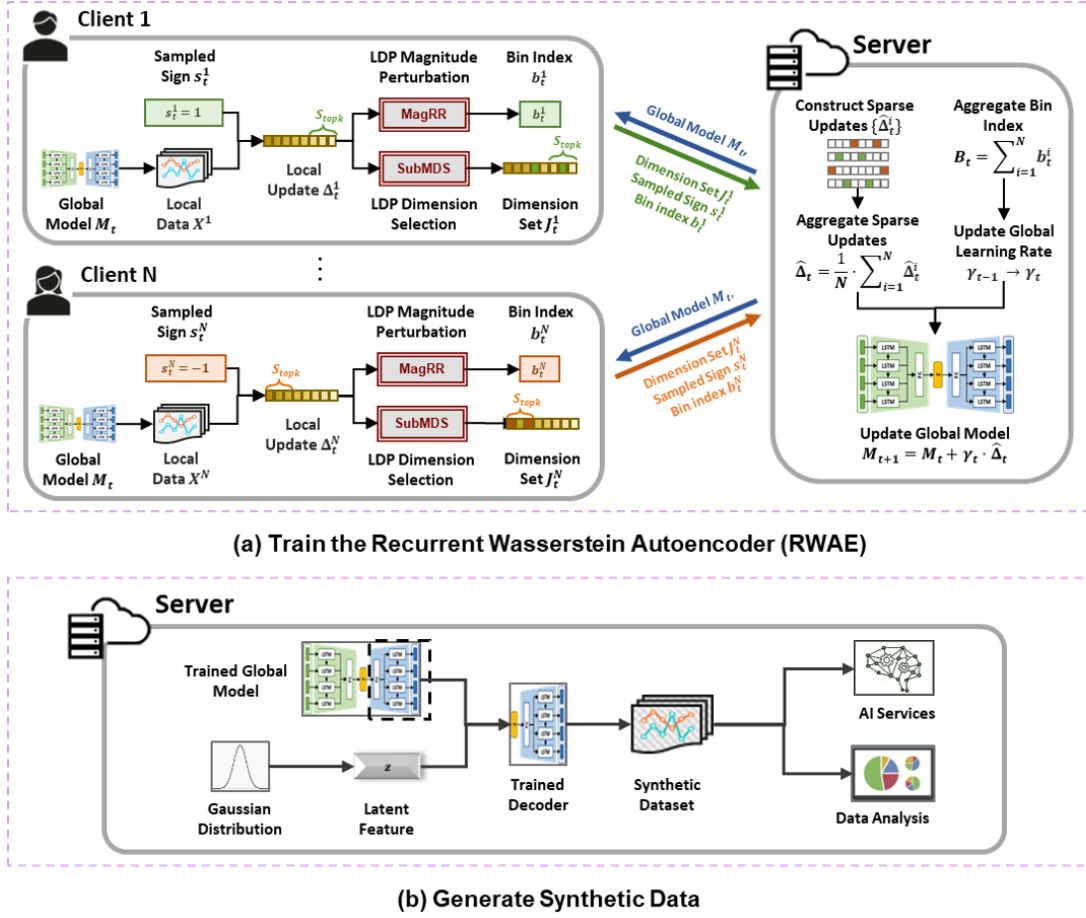


Figure 5.1 Overview of the FEDSTDG framework. The RWAE model is first trained under the federated setting, which learns the distributions of real local data. Two LDP algorithms, SUBMDS and MAGRR, are respectively applied for privatizing the local updates and the update magnitude. After the model is trained, the *decoder* part is used to generate high-utility synthetic data. The generated data will be used for data analysis and building AI services.

generate a synthetic dataset $\tilde{X}_{1:L}$ *without* accessing the real local data. The synthetic dataset $\tilde{X}_{1:L}$ should have the same number of attributes as $X_{1:L}$. Moreover, $\tilde{X}_{1:L}$ can preserve the temporal distributions as in $X_{1:L}$, *i.e.*,

$$P(X_{1:L}) \approx P(\tilde{X}_{1:L}). \quad (5.2)$$

The above objective can be further simplified with a conditional distribution among time steps as follows:

$$P(X_l | X_{1:l-1}) \approx P(\tilde{X}_l | \tilde{X}_{1:l-1}), \quad (5.3)$$

which states that the synthetic data can well approximate the real data at any time l .

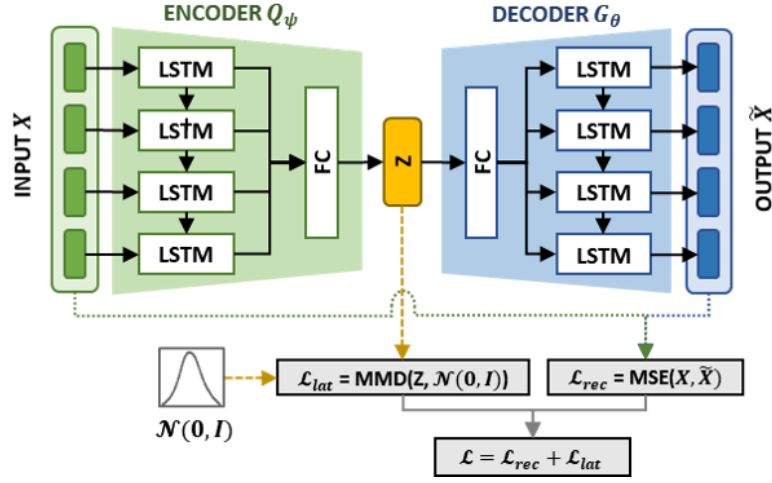


Figure 5.2 Structure of the RWAE model.

5.5 FEDSTDG Framework

In this paper, we propose FEDSTDG, a privacy-preserving framework for distributedly synthesizing the multivariate local time series *without* collecting raw data. The overall workflow of our FEDSTDG is presented in Figure 5.1. More specifically, our framework first trains a RWAE model under the FL setting to learn the spatial-temporal distributions of local data and then generates high-utility synthetic data on the server side, which can be used for downstream data mining and model training tasks. Moreover, on the basis of the sign-based dimension selection concept in [74, 75], we propose an improved LDP-FL algorithm that achieves a better privacy-utility balance. We first introduce an enhanced algorithm SUBMDS to support private multi-dimension selection. The algorithm effectively mitigates the privacy leakage from upload updates while enjoying a better model convergence compared to existing solutions. Additionally, we introduce the MAGRR algorithm, which enables adaptive learning rate adjustments during the training process. We prove that both SUBMDS and MAGRR satisfy a strict LDP definition. By integrating these key components, our framework eliminates the need for a trusted server and ensures strict privacy protection for local data. In the following, we will first introduce the RWAE model and LDP-FL algorithms used in our framework and then describe the overall workflow in detail.

5.5.1 Structure of the Recurrent Wasserstein Autoencoder

Although [74] proposed to use WAE [144] for the data synthesis, the previous model only applies to structured data. In this paper, we extend the model by incorporating long short-term memory (LSTM) [63] layers (a type of recurrent layer) in both encoder and decoder to better capture the spatial-temporal information of the original time-series data. The structure of our RWAE model is presented in Figure 5.2. For the encoder, the time-series data are sequentially input into the LSTM layer. Each cell in the LSTM layer takes in input at a given time step and combines it with information from previous time steps to compute the output of the current step, which is used as input for future time steps. The information passed from

the previous cells enables the layer to learn the temporal correlations of input time series. Moreover, the layer adopts a forget gate, an input gate, and an output gate to compute the cell output, which helps to mitigate the gradient vanishing problem compared to the traditional recurrent layers. The outputs of all the cells are then concatenated and input into a fully connected layer to compute the latent features. The decoder is constructed with a symmetric structure as the encoder, where the latent features are mapped back to the input space using another fully connected layer and an LSTM layer. Each cell in the LSTM layer of the decoder outputs features of a specific time step, and the combination of all these outputs forms the final synthetic data. Our RWAE model follows the same objective function as WAE models, which consists of a reconstruction distance measured using MSE and a latent space distance measured using MMD. Moreover, we set the balancing parameter $\lambda = 1$.

5.5.2 Training with Local Differentially Private Federated Learning

Next, we will introduce the improved LDP-FL algorithm adopted in the FEDSTDG framework, which consists of a subset multi-dimension selection algorithm SUBMDS and a private learning rate adjustment algorithm MAGRR.

Subset Multi-Dimension Selection (SubMDS)

As discussed previously, LDP-FL is one of the privacy-enhancing solutions to prevent privacy leakage in FL. However, existing private mean estimation-based LDP-FL algorithms suffer from a significant utility loss as the noise scale is proportional to the number of model parameters. To improve upon this, we introduced the SIGNDS-FL algorithm in Chapter 3, which adopts a dimension selection protocol to address the *curse-of-dimensionality* problem in existing LDP-FL algorithms. For each model update vector, the algorithm first constructs a top- k set S_{topk} with the dimension indices of k distinctively updated parameters. A set of dimension indices is then selected from S_{topk} under LDP guarantees, which are sent to the server to update the global model. We further introduced two algorithms, namely PS and EM-MDS, for single-dimension selection and multi-dimension selection, respectively.

Upon a closer examination of the EM-MDS algorithm [75] (Section 3.4.3), we note that the optimal threshold $\nu_{th}^*(h^*)$ and the corresponding optimal output size h^* can be automatically determined. First, for each fixed output size h , we search the optimal threshold $\nu_{th}^*(h)$. More specifically, the expectation of ν in the sampled output set J given a certain threshold $\nu_{th}(h)$ can be derived as follows:

$$\mathbb{E}[\nu|\nu_{th}(h)] = \sum_{\tau=0}^h \tau \cdot p(\nu = \tau|\nu_{th}(h)) = \sum_{\tau=0}^{\nu_{th}-1} \frac{\tau \cdot \omega_{\tau}}{\Omega} + \sum_{\tau=\nu_{th}}^h \frac{\tau \cdot \omega_{\tau} \cdot \exp(\epsilon)}{\Omega}, \quad (5.4)$$

where $\Omega = \sum_{\tau=0}^{\nu_{th}(h)-1} \omega_{\tau} + \sum_{\tau=\nu_{th}(h)}^h \omega_{\tau} \cdot \exp(\epsilon)$ as the denominator of Equation (3.10). Intuitively, the larger $\mathbb{E}[\nu|\nu_{th}(h)]$, the more likely that the sampled J contains more top- k dimension indices and the better the model utility. Therefore, the optimal threshold $\nu_{th}^*(h)$ can be determined as the threshold that achieves the maximum $\mathbb{E}[\nu|\nu_{th}(h)]$, namely

$$\nu_{th}^*(h) = \operatorname{argmax}_{\nu_{th} \in [1, h]} \mathbb{E}[\nu|\nu_{th}(h)]. \quad (5.5)$$

Next, we calculate *output top- k ratio*, namely the proportion of top- k dimension indices within the output set, as $\varsigma_h = \mathbb{E}[\nu | \nu_{th}^*(h)]/h$. Intuitively, a larger output size h means more indices are selected, while a higher ratio ς_h suggests selected indices are more likely to be top- k indices. To achieve a good balance between model convergence and utility, we set a minimum acceptable *output top- k ratio* ς^* . Then, we respectively compute the output top- k ratio ς_h for each output size h and choose the largest size that meets the requirement of ς^* to be the optimal size h^* . Namely,

$$h^* = \underset{h \in [1, \text{inf}]}{\text{argmin}} \varsigma_h = \underset{h \in [1, \text{inf}]}{\text{argmin}} \frac{\mathbb{E}[\nu | \nu_{th}^*(h)]}{h} \quad \text{s.t.} \quad \varsigma_h > \varsigma^*. \quad (5.6)$$

Then, we use Equation (5.5) to compute the corresponding optimal threshold $\nu_{th}^*(h^*)$.

Through empirical tests, we discovered that the optimal output size h^* depends not only on the privacy budget ϵ but also on the *input top- k ratio* k/d , which is the ratio of the top- k set S_{topk} to the total number of local update parameters. To illustrate this, let's assume the original local update has $d = 20000$ parameters and the top- k set contains the indices of $k = 100$ parameters. Keeping k fixed, we select the actual input size \tilde{d} from $\{0.01d, 0.05d, 0.1d\}$, resulting in *input top- k ratio* k/\tilde{d} of $\{0.5, 0.1, 0.05\}$. Next, we calculate the optimal output size h^* for different *output top- k ratio* ς^* and present the results in Figure 5.3. We use different colors to represent various *input top- k ratio* and distinct line patterns to differentiate the privacy budget levels. The results show that, under the same privacy level, a larger input top- k ratio k/d leads to a larger h^* . This suggests that reducing the input size \tilde{d} may accelerate model convergence and enhance model utility.

The insightful result motivates us to introduce a new algorithm, SUBMDS. The algorithm aims to enhance model performance by using a reduced input size \tilde{d} . The process of SUBMDS is illustrated in Algorithm 6. Given the original local update $\Delta \in \mathbb{R}^d$, we first randomly select $\tilde{d} = \rho \cdot d$ model parameters and construct the input set S_{in} using the corresponding dimension indices. Here, ρ is referred to as the *subsampling ratio*. Then, a sign value s is sampled and used to determine the construction of the top- k set S_{topk} . Next, with a predefined privacy budget ϵ and an expected output top- k ratio ς^* , we compute the optimal output size h^* and threshold ν_{th}^* based on Equation (5.6) and Equation (5.5). These values are then used to calculate the probability defined in Equation (3.10). It should be noted that in our algorithm, the input set S_{in} only contains \tilde{d} dimension indices. Hence, the output domain is shrunk to $\mathcal{J} = \{i | i \in S_{in}\}^h$ and the number of output sets containing ν top- k dimensions is reduced to $\omega_\nu = \binom{k}{\nu} \binom{\tilde{d}-k}{h^*-\nu}$. Following [75], we use the inverse sampling technique [155] to build the output set. Specifically, the number of top- k dimensions included in J is determined by a random variable β drawn from the uniform distribution $\mathbf{U}(0, 1)$ and the cumulative distribution function $\mathcal{F}(\nu | \nu_{th}^*)$. Finally, J is constructed by randomly selecting ν indices from S_{topk} and $h - \nu$ indices from the remaining indices set $S_{in} \setminus S_{topk}$.

As the subset of parameters is randomly sampled and is irrelevant to the original local update, we can prove that our SUBMDS algorithm satisfies ϵ -LDP guarantees.

Lemma 6. *The SUBMDS algorithm satisfies ϵ -LDP.*

Proof. For each client, given any two arbitrary local updates Δ, Δ' and an input set S_{in} containing \tilde{d} randomly sampled dimension indices, let S_{topk} and S'_{topk} be the top- k sets con-

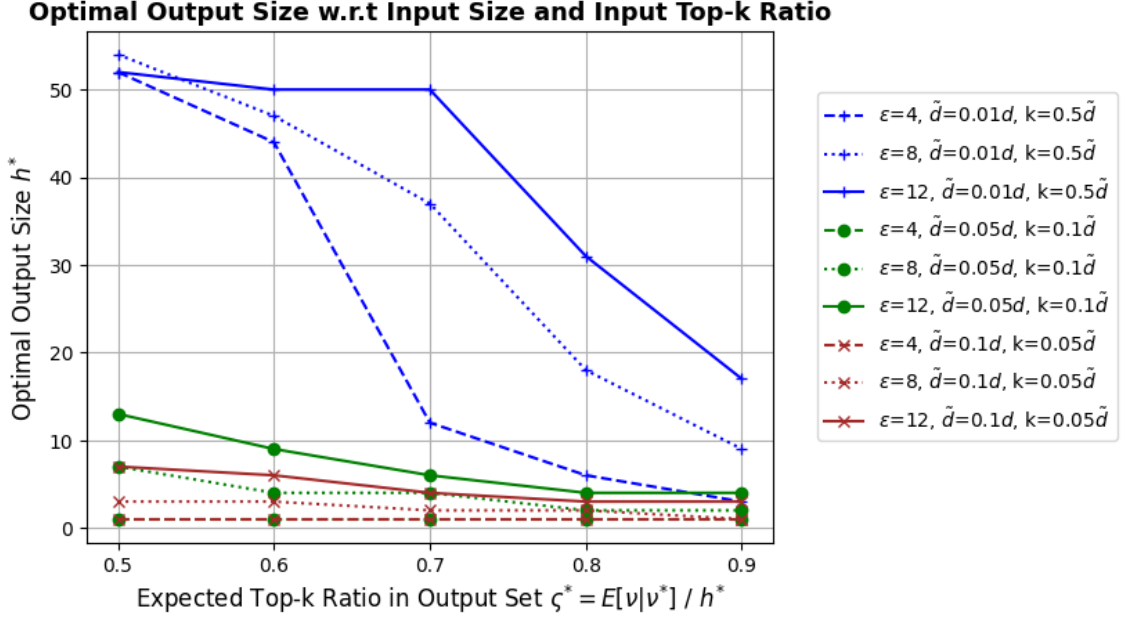


Figure 5.3 Comparison of the optimal output size under different privacy budgets and input top- k ratio. Given the *original* input size $d = 20000$ and top- k size $k = 100$, we vary the *actual* input size among $\tilde{d} = \{0.01d, 0.05d, 0.1d\}$, resulting in an input top- k ratio k/\tilde{d} among $\{0.5, 0.1, 0.05\}$. Then, we compare the corresponding optimal output size h^* regarding different expected output top- k ratio ζ^* .

structured from S_{in} . For any output set $J \in \mathcal{J}$, where $\mathcal{J} = \{i | i \in S_{in}\}^h$ being the output domain, let $\nu = |S_{topk} \cap J|$, $\nu' = |S'_{topk} \cap J|$ be the number of intersected indices between J and both top- k sets. With the sampling probability defined in Equation (3.10) it holds that

$$\begin{aligned} \frac{\Pr[J|\Delta]}{\Pr[J|\Delta']} &= \frac{\Pr[J|S_{topk}, S_{in}]}{\Pr[J|S'_{topk}, S_{in}]} = \frac{\sum_{J' \in \mathcal{J}} \frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J))}{\exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J'))}}{\sum_{J' \in \mathcal{J}} \frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S'_{topk}, J))}{\exp(\frac{\epsilon}{\phi_u} \cdot u(S'_{topk}, J'))}} = \frac{\sum_{\tau=0}^{\nu_{th}-1} \omega_{\tau} + \sum_{\tau=\nu_{th}}^h \omega_{\tau} \cdot \exp(\epsilon)}{\sum_{\tau=0}^{\nu'_{th}-1} \omega_{\tau} + \sum_{\tau=\nu'_{th}}^h \omega_{\tau} \cdot \exp(\epsilon)} \quad (5.7) \\ &= \frac{\exp(\epsilon \cdot \mathbf{1}(\nu \geq \nu_{th}))}{\exp(\epsilon \cdot \mathbf{1}(\nu' \geq \nu_{th}))} \leq \frac{\exp(\epsilon \cdot 1)}{\exp(\epsilon \cdot 0)} = \exp(\epsilon), \end{aligned}$$

which completes the proof. \square

MAGR for Adaptive Global Learning Rate

After receiving clients' output index sets and sampled sign values, the server constructs corresponding sparse updates and uses their average to update the global model parameters. However, both previous works [74, 75] used a constant global learning rate γ throughout training, which impacts the model's utility. A large γ results in fast convergence but may cause a significant discrepancy between the real and the sparse local updates as training keeps on. Conversely, a small γ requires more rounds for convergence. To address this, an ideal approach is to apply an adaptive γ that starts large and progressively decreases during training. Therefore, we introduce a novel algorithm MAGRR to adapt the global learning rate in

Algorithm 6: SUBMDS

Input: $\Delta \in \mathbb{R}^d$: local update with d parameters; k : size of top- k set; ϱ : subsampling ratio; ϵ : privacy budget;

- 1: Randomly sample $\tilde{d} = \varrho \cdot d$ parameters from Δ to build Δ_{in}
- 2: Use the dimension indices of parameters in Δ_{in} to build an input set S_{in}
- 3: Randomly sample a sign value s from $\{1, -1\}$
- 4: **if** $s = 1$ **then**
- 5: Build a top- k set S_{topk} with the indices of k *largest* parameters in Δ_{in}
- 6: **else**
- 7: Build a top- k set S_{topk} with the indices of k *smallest* parameters in Δ_{in}
- 8: **end if**
- 9: Given \tilde{d} and k , compute the optimal output size h^* and threshold ν_{th}^* according to Equation (5.5) and Equation (5.6)
- 10: Compute denominator $\Omega = \sum_{\tau=0}^{\nu_{th}^*-1} \omega_{\tau} + \sum_{\tau=\nu_{th}^*}^{h^*} \omega_{\tau} \cdot \exp(\epsilon)$, where $\omega_{\tau} = \binom{k}{\tau} \binom{\tilde{d}-k}{h^*-\tau}$
- 11: Randomly sample $\beta \sim \mathbf{U}(0, 1)$
- 12: Initialize $\nu = 0$ and $\mathcal{F}(\nu|\nu_{th}^*) = \omega_0/\Omega$
- 13: **while** $\mathcal{F}(\tau) < \beta$ **do**
- 14: $\nu = \nu + 1$
- 15: **if** $\nu < \nu_{th}^*$ **then**
- 16: Let $\mathcal{F}(\nu|\nu_{th}^*) = \mathcal{F}(\nu|\nu_{th}^*) + \omega_{\nu}/\Omega$
- 17: **else**
- 18: Let $\mathcal{F}(\nu|\nu_{th}^*) = \mathcal{F}(\nu|\nu_{th}^*) + \exp(\epsilon) \cdot \omega_{\nu}/\Omega$
- 19: **end if**
- 20: **end while**
- 21: Construct J by sampling ν indices from S_{topk} and $h^*-\nu$ indices from the remaining set $S_{in} \setminus S_{topk}$
- 22: **Return** J

a private manner. The main idea is to collect the largest magnitude of each client's local update and use the aggregated result to determine the global learning rate for the next training round. In addition, as the real update magnitude cannot be directly published due to privacy issues, the algorithm randomizes the real magnitude under LDP guarantees and only shares the privatized value. More specifically, with the current round's global learning rate γ , each client computes the largest magnitude m in the local update (where $m \geq 0$). Subsequently, we quantize m into two bins, namely, $[0, r \cdot \gamma)$ and $[r \cdot \gamma, \infty)$, where r is the decay rate. We use the bin index b as a flag to indicate whether the global learning rate should be decayed. Then, we apply a binary randomized response (RR) [163] to flip the index. Given a privacy budget ϵ , we have:

$$\hat{b} = \begin{cases} b & w.p. \frac{\exp(\epsilon)}{\exp(\epsilon)+1} \\ 1-b & w.p. \frac{1}{\exp(\epsilon)+1} \end{cases}, \quad (5.9)$$

The randomized bin index will be sent to the server. Here, we present the privacy analysis of the MAGRR algorithm.

Lemma 7. *The MAGRR algorithm satisfies ϵ -LDP.*

Algorithm 7: MAGRR

Input: $\Delta \in \mathbb{R}^d$: local update with d parameters; γ : global learning rate ; \hat{b} : aggregated decay flag; r : decay rate; ϵ : privacy budget; T_{pat} : patience threshold; t_{pat} : number of rounds requiring a decay in learning rate.

LocalPert(Δ, γ, ϵ):

// Run on the client side

- 1: Get the largest magnitude value m from Δ
- 2: Compute the decay flag b :
 if $m \in [0, r \cdot \gamma)$, **set** $b = 1$; **elif** $m \in [r \cdot \gamma, \infty)$, **set** $b = 0$
- 3: Perturb the decay flag b :

$$\hat{b} = \begin{cases} b & w.p. \frac{\exp(\epsilon)}{\exp(\epsilon)+1} \\ 1-b & w.p. \frac{1}{\exp(\epsilon)+1} \end{cases} \quad (5.8)$$

- 4: **Return** \hat{b}

ServerAggr($\hat{b}, \gamma, r, T_{pat}, t_{pat}$):

// Run on the server side

- 1: **if** $\hat{b} \geq 0.5$ **then**
- 2: $t_{pat} = t_{pat} + 1$
- 3: **if** $t_{pat} = T_{pat}$ **then** $\gamma' = r \cdot \gamma$, $t_{pat} = 0$; **else** $\gamma' = \gamma$
- 4: **else**
- 5: $\gamma' = \gamma$, $t_{pat} = 0$
- 6: **end if**
- 7: **Return** γ', t_{pat}

Proof. For each client, given two arbitrary real bin indices b, b' and the perturbed index as \hat{b} , with the flip probability defined in Equation (5.9) it holds that

$$\frac{\Pr[\hat{b}|b]}{\Pr[\hat{b}|b']} \leq \frac{\Pr[\hat{b} = b|b]}{\Pr[\hat{b} = 1-b'|b']} = \frac{\frac{\exp(\epsilon)}{\exp(\epsilon)+1}}{\frac{1}{\exp(\epsilon)+1}} = \exp(\epsilon), \quad (5.10)$$

which completes the proof. \square

After receiving the perturbed flags, the server will update the global learning rate accordingly. If more than half of the flags are 1, meaning the majority of local clients' update magnitude is less than $r \cdot \gamma$, we adjust the learning rate to $r \cdot \gamma$; otherwise, we keep γ unchanged. However, adjusting the learning rate in every round may lead to unstable training due to variations in update magnitudes across clients and the randomness in flag reporting. To address this, we introduce a patience threshold T_{pat} and only modify the global learning rate when the condition of majority flags being 1 persists for more than T_{pat} rounds. In the following experiments in Section 5.6, we use default values of $r = 0.5$ and $T_{pat} = 50$.

5.5.3 Overall Workflow of FEDSTDG

We now describe the overall workflow of our FEDSTDG framework presented in Figure 5.1.

First, we train the RWAE within a federated setting, as depicted in Algorithm 8. At each global round t , the server randomly selects a subset of N clients and broadcasts the current global model M_t and the global learning rate γ_t . Each local client i trains the global model using its private data X^i and calculates the local update Δ_t^i . Then, given a privacy budget ϵ_{idx} , the client utilizes the SUBMDS algorithm to sample the random sign value s_t^i and compute the private index set J_t^i . Additionally, with a privacy budget ϵ_{mag} , the client computes the decay flag based on the local update magnitudes. Then, the client sends the respective b_t^i , s_t^i , and J_t^i to the server. Note that for each local update, we sequentially apply the SUBMDS and MAGRR algorithms to obtain the private index set and the decay flag of the learning rate. According to the DP’s sequential composition theorem (Property 6), the total privacy guarantee to *each local update* is $\epsilon = \epsilon_{idx} + \epsilon_{mag}$.

After receiving the information from local clients, the server constructs the sparse update using each pair of s_t^i and J_t^i and computes the aggregated model update $\hat{\Delta}_t$. Moreover, the server also calculates an aggregated decay flag \hat{b}_t based on the local decay flags and modifies the global learning rate γ_t accordingly. Finally, the server employs $\hat{\Delta}_t$ and \hat{b}_t to update the global model parameters, and the new global model M_{t+1} is distributed to local clients for the next training round.

Once the model has been trained, the server can use the *decoder* G_θ to produce synthetic time-series data. Recall that the latent space features are enforced to follow the standard Gaussian distribution p_z . Therefore, we can simply sample random latent features z from p_z and feed them into G_θ to generate high-utility synthetic time-series data, which can be further used for data analysis and building AI services.

5.6 Experiments

We implemented the proposed framework using the TensorFlow platform and conducted comprehensive experiments with real-world datasets to evaluate its performance. We conduct the proposed experiments on an Intel 1.8 GHz Core i7 CPU. In this section, we will introduce the experimental settings and discuss the evaluation results.

5.6.1 Experiment Setup

Datasets and Model Architectures

We use four real-world multivariate time-series datasets for evaluating the performance of our framework.

Algorithm 8: Training the RWAE

Input: $M \in \mathbb{R}^d$: global model with d parameters; T : number of aggregation rounds; N : number of clients in each round; E : number of local epochs; η : local learning rate; k : size of top- k set; ϱ : subsampling ratio; γ : global learning rate; t_{pat} : accumulated number of decay rounds; T_{pat} : threshold for decay rounds; r : decay rate; $\epsilon_{idx}, \epsilon_{mag}$: privacy budgets.

Output: Trained WAE model M

Server executes:

- 1: Initialize the global model M_1 and the global learning rate γ_1
- 2: **for** global round $t = 1, \dots, T$ **do**
- 3: Randomly select a group of N clients
- 4: **for** client $i = 1, \dots, N$ in parallel **do**
- 5: Broadcast the global model M_t and the global learning rate γ_t to the local side
- 6: $b_t^i, s_t^i, J_t^i = \mathbf{LocalUpdate}(M_t, E, \eta, \epsilon_{idx}, \epsilon_{mag}, \gamma_t, k, \varrho, r)$
- 7: Construct $\hat{\Delta}_t^i = [0, \dots, 0]^d$; for $j \in J_t^i$, set $\hat{\Delta}_t^i[j] = s_t^i$
- 8: **end for**
- 9: Compute the aggregated model update $\hat{\Delta}_t = \frac{1}{N} \sum_{i=1}^N \hat{\Delta}_t^i$
- 10: Compute the aggregated decay flag $\hat{b}_t = \frac{1}{N} \sum_{i=1}^N b_t^i$
- 11: Update global learning rate $\gamma_{t+1}, t_{pat} = \mathbf{MagRR.ServerAggr}(\hat{b}_t, \gamma_t, r, T_{pat}, t_{pat})$
- 12: Update global model $M_{t+1} = M_t + \gamma_{t+1} \cdot \hat{\Delta}_t$
- 13: **end for**
- 14: **Return** Global model $M = M_{T+1}$

LocalUpdate($M_t, E, \eta, \epsilon_{idx}, \epsilon_{mag}, \gamma_t, k, \varrho, r$):

// Run on the client side

- 15: Initialize local model $M_t^i \leftarrow M_t$
- 16: **for** epoch $e = 1, \dots, E$ **do**
- 17: $M_t^i = M_t^i - \eta \cdot \nabla \mathcal{L}(M_t^i, X^i)$
- 18: **end for**
- 19: Compute local update: $\Delta_t^i = M_t^i - M_t$
- 20: Get the output set $J_t^i, s_t^i = \mathbf{SubMDS}(\Delta_t^i, k, \varrho, \epsilon_{idx})$
- 21: Get the decay flag $b_t^i = \mathbf{MagRR.LocalPert}(\Delta_t^i, r, \gamma_t, \epsilon_{mag})$
- 22: **Return** b_t^i, s_t^i, J_t^i

Table 5.2 Details of Datasets and Model Size

	#Records	#Features	#Steps	#Parameters
Sine	50,000	5	30	29,269
Stock	4,400	6	24	26,358
Energy	19,735	27	24	29,729
Air	9,333	13	24	27,485

Sine [177]: The dataset contains 50,000 multivariate sinusoidal sequences simulated using the open-source code provided in the original paper ¹. Each record contains five attributes, representing the sequence with different frequencies and phase.

Stock [46]: The dataset contains daily historical Google stocks data originating from August 2004, which sums to approximately 4400 records. Each record contains six attributes such as volume, opening prices, and closing prices. The goal is to predict future stock prices and volume.

Energy [20]: The dataset contains 19,735 records, each with 27 attributes presenting the energy usage as well as temperature and humidity in different areas of each period of time. The goal is to predict energy usage in the future.

Air [33]: The dataset contains 9358 hourly records of an Air Quality Multisensor Device, including the hourly concentrations and sensor responses of different types of gas. The goal is to predict the future trend of gas concentration. We remove the date and time attributes in the original data and replace the missing values with zeros.

For all the datasets, we normalize the record values to $[0,1]$. For the encoder part of the RWAE models, we use single-layer LSTM with 32 hidden units followed by a fully connected layer with 16 units. The layers in the decoder are of the same hidden units as the encoder but in a reversed order. Furthermore, we use the *sigmoid* activation for the output layer to restrict the reconstructed values to be within $[0,1]$. Details of the datasets and the corresponding model size are presented in Table 5.2.

Evaluation Metrics

We evaluate the performance of our framework considering three aspects, namely *data utility*, *privacy protection*, and *robustness*. For the evaluation of data utility, we follow the evaluation approaches in [177] and assess the quality of synthetic data from three perspectives:

- *Fidelity*: The synthetic data should preserve similar correlations and distributions of real data.
- *Diversity*: The synthetic data should be diversely distributed and cover most of the variety of real data.
- *Usefulness*: The synthetic data should have similar performance as real data in AI training tasks.

¹https://bitbucket.org/mvdschaar/mlforhealthlabpub/src/master/alg/timegan/data_loading.py

In this study, we utilize the MMD distance Equation (2.14), t-SNE analysis [153], and mean absolute error (MAE) for next step prediction to assess the *fidelity*, *diversity*, and *usefulness*, respectively. To evaluate privacy protection, we examine the effectiveness of our framework against membership inference attack (MIA) by comparing the attack accuracy under various privacy levels. Finally, we evaluate the robustness of our framework against untargeted poisoning attacks. .

Baselines

In the following experiments, we investigate the performance of our framework under different privacy settings. We also compare the results *with* and *without* adaptive global learning rate adjustment, which are respectively referred to as SUBMDS and ADASUBMDS. We also compare the performance of the RWAE model trained with baseline LDP-FL algorithms, including SIGNDS [75] with single-dimension selection (referred to as SignDS-S) and with multi-dimension selection (referred to as SIGNDS-M). We also include a baseline that evenly splits the privacy budget for multi-dimension selection (referred to as SIGNDS-E). Moreover, we also compare the model trained under the non-private centralized setting and federated setting and the DP centralized setting using the DPSGD algorithm.

Hyperparameter Configurations

In each experiment, we conduct the FL training for 1000 global rounds, where $N = 50$ clients are sampled in each round. For the local training, we assume each local client has 10 time-series records and trains the RWAE model for $E = 10$ epochs using the *Adam* optimizer with a default learning rate $\eta = 0.001$. We further choose various privacy budgets $\epsilon \in \{0.5, 2, 4, 6, 8\}$ to explore the influence of privacy on the framework performance. For all the algorithms, we keep $k = 0.05d$, where d is the total number of model parameters and k is the number of indices in S_{topk} . For SUBMDS, we set the subsampling rate ρ to 0.1. Hence, the input top- k ratio for SIGNDS-S, SIGNDS-M, SIGNDS-E is 0.05, while for SUBMDS is $0.05/0.1 = 0.5$. Moreover, for all three algorithms, we set the desired output top- k ratio ζ^* to 0.8. In experiments *without* adaptive global learning rate, we set ϵ_{idx} to ϵ and γ to 5. In experiments *with* adaptive global learning rate (ADASUBMDS), we set ϵ_{idx} to 0.9ϵ and ϵ_{mag} to 0.1ϵ . The initial value of γ is set to 25, the decay rate r is 0.5, and the patience threshold T_{pat} is 50.

5.6.2 Evaluation of Data Utility

We first evaluate the utility of the synthetic data under different privacy levels regarding the *fidelity*, *diversity*, and *usefulness*.

Analysis of Fidelity

To start with, we investigate the fidelity of synthetic data in comparison to real data. To this end, we use the MMD distance in Equation (2.14) to measure the distribution difference between

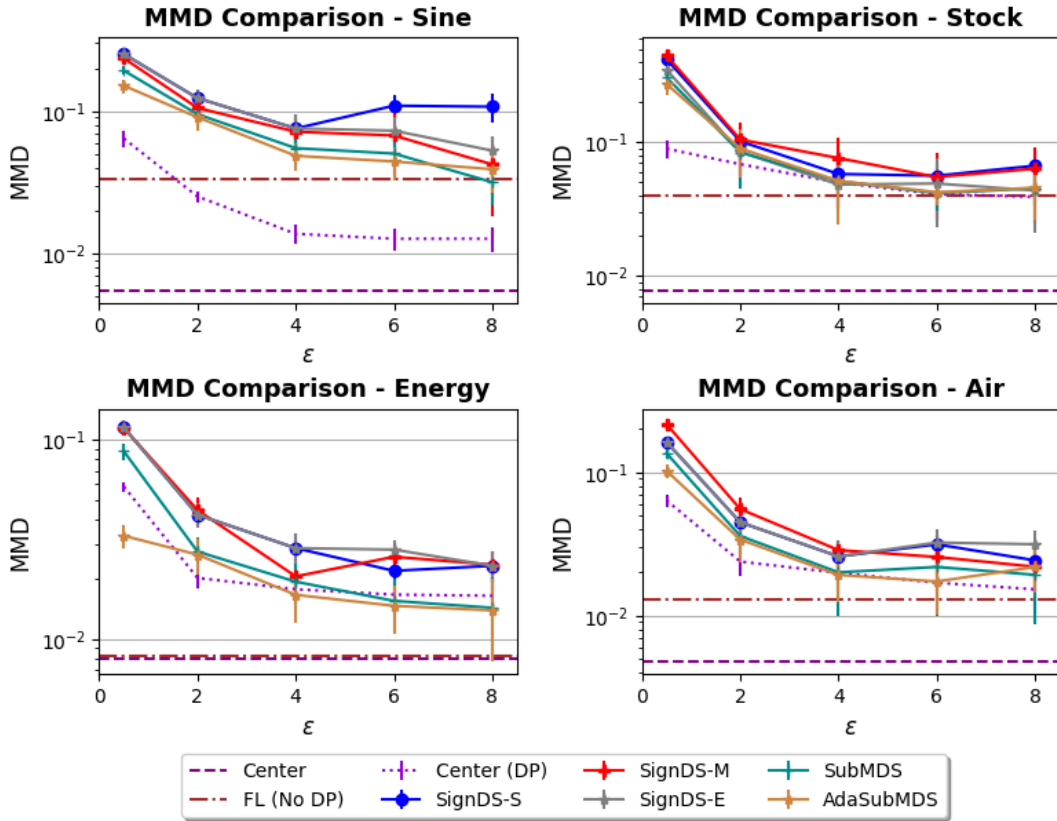


Figure 5.4 Comparison of MMD distance between real and synthetic data of all the datasets. For each dataset, we compare the results of non-private centralized and FL settings as well as different LDP-FL baselines under different privacy levels.

real data and synthetic data. Intuitively, the smaller the MMD, the more similar the synthetic data is to real data. For each dataset, we respectively compare the MMD of the RWAE models trained under private and non-private centralized settings and FL settings and present the results in Figure 5.4. It can be seen that, for the RWAE models trained with LDP-FL algorithms, the increase of the privacy budget ϵ generally leads to a decrease in MMD, as the model is perturbed by less noise during the training process. Moreover, our SUBMDS and ADASUBMDS algorithms achieve a distinctive improvement compared to the baseline algorithms, particularly with smaller privacy budgets. Specifically, under the same privacy level, both algorithms can achieve an approximate 30% – 40% reduction in MMD compared to the single-dimension SIGNDS (SIGNDS-S), and around 20% – 30% compared to the multi-dimension SIGNDS (SIGNDS-M). When ϵ increases, the MMD is approaching the results of the non-private FL setting. The results suggest that reducing the input dimensions and adopting a larger top- k ratio can effectively accelerate model convergence and help improve synthetic data utility.

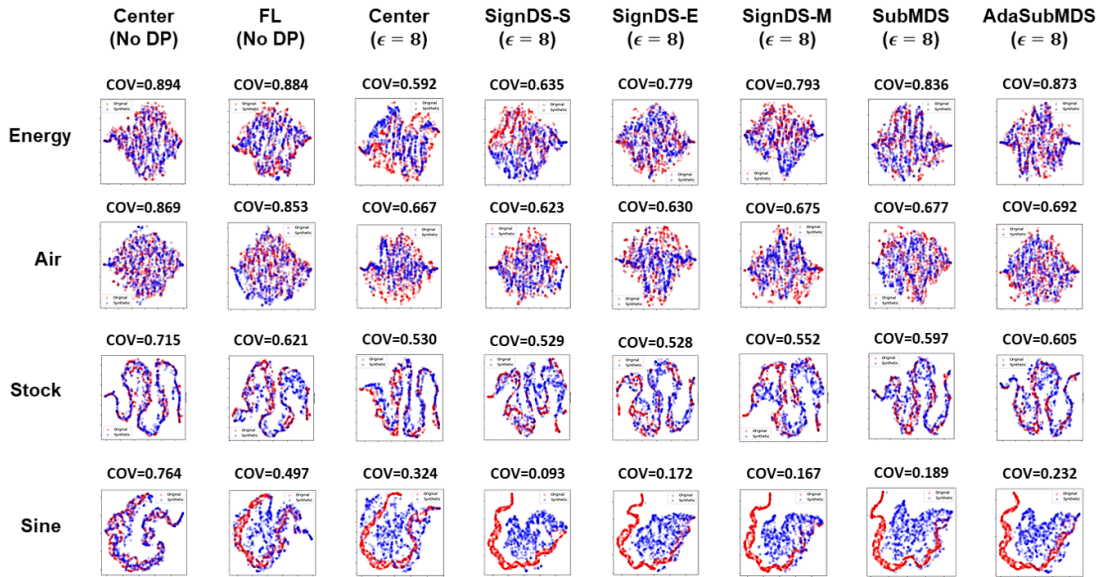


Figure 5.5 T-SNE visualization of the distribution of real and synthetic data on the four datasets. Each column represents the results of one dataset. Each row provides the distribution of synthetic data generated under non-private centralized and FL settings as well as using different LDP-FL algorithms with $\epsilon = 8$. SUBMDS and ADASUBMDS refer to the results of using the proposed methods. Red denotes the real data and blue denotes synthetic data. We also use the Synthcity library [127] to compute the coverage ratio between real and synthetic data.

Analysis of Diversity

Next, we qualitatively assess the diversity of the synthetic data. Here, we conduct a t-SNE analysis [153] to visually depict the distribution of real and synthetic data from different datasets in a two-dimensional space. Intuitively, synthetic data with good utility should be characterized by a distribution with high diversity and cover the distribution of real data. In Figure 5.5, we present the visualization results of the four datasets generated by the RWAE model under the non-private centralized and federated settings, as well as using different LDP-FL algorithms with $\epsilon = 8$. For each dataset, we respectively compare the distribution between the real data (red) and the synthetic data (blue). Moreover, we also use the Synthcity library [127]² to compute the coverage ratio between real and synthetic data. It can be seen that the synthetic data generated by the baseline LDP-FL algorithms fail to cover the entirety of the real distribution. In contrast, the synthetic data generated using our SUBMDS and ADASUBMDS algorithms can better overlap with real data and achieve a higher coverage ratio. The results illustrate that our framework can better capture the distributions of different variants of real data and generate diversified synthetic data.

Analysis of Usefulness

We further analyze the usefulness of synthetic data in AI training tasks. We use the real and synthetic data to train a one-layer LSTM model for *next-step prediction*, respectively. Then,

²<https://github.com/vanderschaarlab/synthcity>

we evaluate the model performance using a *held-out* set of real data. Intuitively, if the LSTM models trained with synthetic data demonstrate similar performance as those trained with real data, we say that the synthetic data is of high usefulness and can replace real data for downstream AI training tasks. Here, we use the MAE to measure the performance of the LSTM models.

The predictive MAE of all datasets under various privacy settings is presented in Table 5.3. The MAE of models trained with real data is also included as a reference. It is evident that both SUBMDS and ADASUBMDS exhibit significantly lower predictive errors compared to the other two baselines, particularly for small privacy budgets. Specifically, with $\epsilon = 0.5$, the ADASUBMDS algorithm reduces the MAE by at least 20% compared to the baseline algorithms, and achieves a maximum reduction of 85% on the **Stock** dataset. This further demonstrates the effectiveness of our multi-dimensional selection algorithms in enhancing the performance of RWAEs and the quality of synthetic data. Furthermore, as ϵ increases, the MAE of the SUBMDS and ADASUBMDS algorithms gradually approaches that of the real data. With $\epsilon = 8$, the increase in MAE for both algorithms is less than 0.09 and only 0.002 at best when compared to models trained with real data. These results demonstrate that the synthetic data generated by our framework has higher usefulness compared to the baselines and can serve as a replacement for real data in AI training tasks.

Ablation Study

We further conduct a series of ablation studies under the ADASUBMDS setting to investigate how the split ratio of privacy budgets and the number of per-round clients will impact the framework’s performance.

Impact of the Split of Privacy Budgets In previous experiments, we found that allocating 90% of the total privacy budget to dimension selection enhances the performance of ADASUBMDS compared to baselines under the same privacy budget. In this section, we further investigate how the different split ratios of privacy budgets impact the model performance. To this end, we conduct experiments under $\epsilon = \{2, 4, 8\}$ with ϵ_{idx} being 10%, 50%, and 90% of the total budget, respectively, and compare the predictive MAE of synthetic data generated under different privacy settings. The results of the four datasets are shown in Figure 5.6. It can be observed that for the same total privacy value ϵ , choosing a larger split ratio for dimension selection typically results in a lower MAE, namely better data utility. This suggests the model’s update direction is more crucial for achieving a satisfactory convergence in comparison to the update magnitude. Given that the dimension selection process significantly impacts the update direction, it should receive a higher proportion of privacy to reduce randomness during training. Additionally, for the magnitude perturbation, since we implement a quantization on the real value and only modify the global learning rate following T_{pat} rounds, it can withstand greater randomness and therefore be assigned less privacy.

Impact of the Per-round Clients In addition to the split ratio of privacy budgets, we also analyze the impact of the number of participating clients in each training round on the framework’s performance. We choose 50, 100, and 200 per-round clients respectively, and train

Table 5.3 Comparison of synthetic data utility in AI training tasks. For each dataset, we respectively use real and synthetic data to train one-layer LSTM models for *next-step prediction* and evaluate the models’ predictive MAE on a *held-out* set of real data. Here we compare the MAE of synthetic data generated under different privacy settings.

Datasets	Sine			Stock		
	Real	Center	FL	Real	Center	FL
Non-Private Setting	0.020	0.027	0.038	0.012	0.013	0.014
Central-DP Setting	$\epsilon = 0.5$	$\epsilon = 2$	$\epsilon = 8$	$\epsilon = 0.5$	$\epsilon = 2$	$\epsilon = 8$
	0.124	0.103	0.051	0.018	0.015	0.014
FL Setting	$\epsilon = 0.5$	$\epsilon = 2$	$\epsilon = 8$	$\epsilon = 0.5$	$\epsilon = 2$	$\epsilon = 8$
SignDS-S [74]	0.211	0.141	0.130	0.131	0.022	0.017
SignDS-E	0.210	0.145	0.127	0.130	0.021	0.017
SignDS-M [75]	0.196	0.141	0.111	0.118	0.021	0.015
SubMDS	0.159	0.120	0.107	0.028	0.016	0.014
AdaSubMDS	0.134	0.108	0.110	0.020	0.015	0.014
Datasets	Energy			Air		
	Real	Center	FL	Real	Center	FL
Non-Private Setting	0.031	0.043	0.043	0.038	0.043	0.047
Central-DP Setting	$\epsilon = 0.5$	$\epsilon = 2$	$\epsilon = 8$	$\epsilon = 0.5$	$\epsilon = 2$	$\epsilon = 8$
	0.060	0.056	0.050	0.083	0.081	0.074
FL Setting	$\epsilon = 0.5$	$\epsilon = 2$	$\epsilon = 8$	$\epsilon = 0.5$	$\epsilon = 2$	$\epsilon = 8$
SignDS-S [74]	0.118	0.066	0.054	0.117	0.087	0.079
SignDS-E	0.116	0.065	0.054	0.116	0.087	0.078
SignDS-M [75]	0.108	0.062	0.053	0.113	0.085	0.071
SubMDS	0.084	0.054	0.051	0.097	0.082	0.070
AdaSubMDS	0.058	0.052	0.050	0.087	0.083	0.070

the model with $\epsilon \in \{2, 4, 8\}$. The MMD of the four datasets under different privacy settings is shown in Figure 5.7. For both datasets, it is evident that increasing the number of per-round clients further reduces the MAE. With $\epsilon \leq 4$, increasing the per-round clients from 50 to 200 results in around a 50% ~ 75% decrease in MMD. This outcome is due to the aggregation of privatized local updates on the server side. Consequently, when ϵ is small, using more per-round clients helps mitigate the impact of local randomization on the aggregated global update, thereby improving the convergence of the global model. Conversely, when ϵ is large, the randomness caused by the local protection process is already significantly reduced. Thus, the number of per-round clients does not significantly enhance the utility of synthetic data.

Impact of the Decay Rate and Patience Threshold Finally, we study the impact of r and T_{pat} on the model convergence. We train the RWAE with $r \in \{0.5, 0.9\}$ and $T_{pat} \in \{10, 50, 100\}$, and use the MMD of synthetic data after each round of aggregation for analysis. Intuitively, a smaller r leads to a faster decay of the learning rate, while a smaller T_{pat} implies a more frequent decay of the learning rate. Figure 5.8 illustrates the performance of the model with privacy budget $\epsilon = 6$. We observe that $r = 50$ and $T_{pat} = 10$ result in slower convergence and poorer model performance, as the learning rate reduces quickly at the beginning

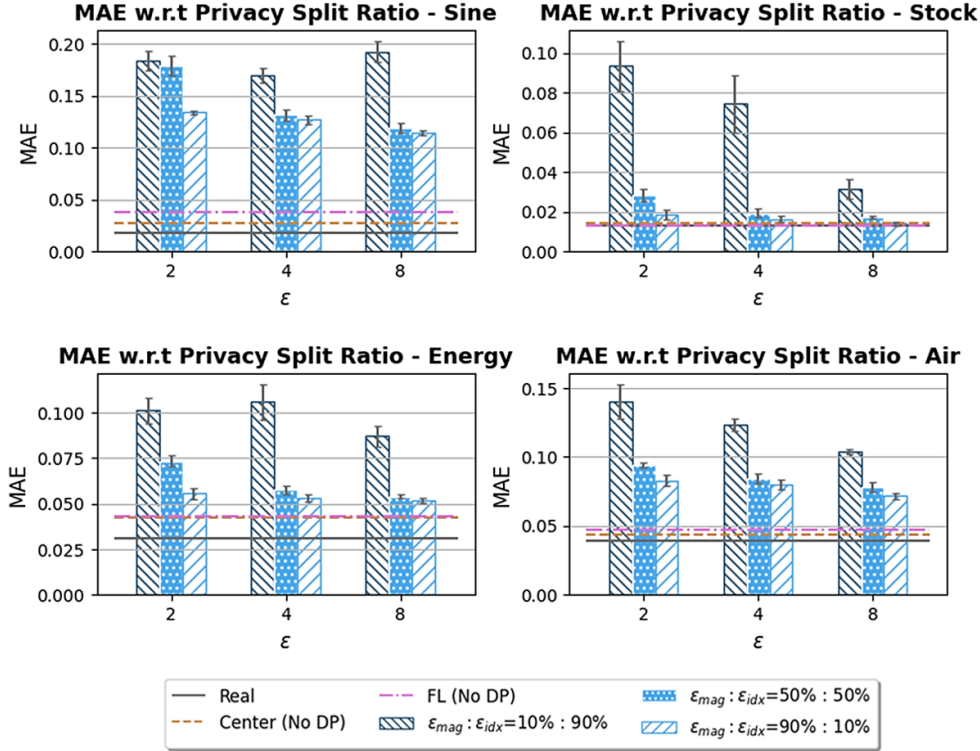


Figure 5.6 Analysis of the impact of privacy split ratio. For each dataset, we train the RWAE model with ϵ_{mag} being 10%, 50%, 90% of the total privacy ϵ and compare the MAE of synthetic data under different privacy settings.

of training. Conversely, $r = 0.9$ and $T_{pat} = 100$ exhibit unstable convergence, as the learning rate deviates significantly from the true magnitude, leading to a greater discrepancy in sparse updates. Furthermore, we notice that the ideal combination of r and T_{pat} may vary across datasets, suggesting potential improvement by tuning hyperparameters.

Pre-training with Auxiliary Data Finally, we investigate whether the performance of SUB-MDS can be further improved with pre-training. We assume that the server has a small number of auxiliary data that are from a similar distribution as real local data. The server uses these data to pre-train the RWAE model and then applies the warm-up model for FL training. In Table 5.4, we respectively compare the MMD between the real data and the synthetic data generated by the model with and without pre-training. For each dataset, we use 100 auxiliary data to pre-train the model for 50 epochs. We report the results of training, especially with $\epsilon = 2$ and $\epsilon = 8$. It can be seen that for all the datasets, synthetic data generated by pre-trained RWAE models can achieve a lower MMD. In particular, when $\epsilon = 2$, the MMD has been reduced by around 30% after using pre-training. This is because a larger amount of randomness is injected during the FL training when ϵ is small, which hinders the model convergence. Since the pre-trained model has already learned some information about the data distribution, it may suffer from less difficulty during FL training in comparison to training the model from scratch and can therefore obtain a better synthetic data utility. On the other hand,

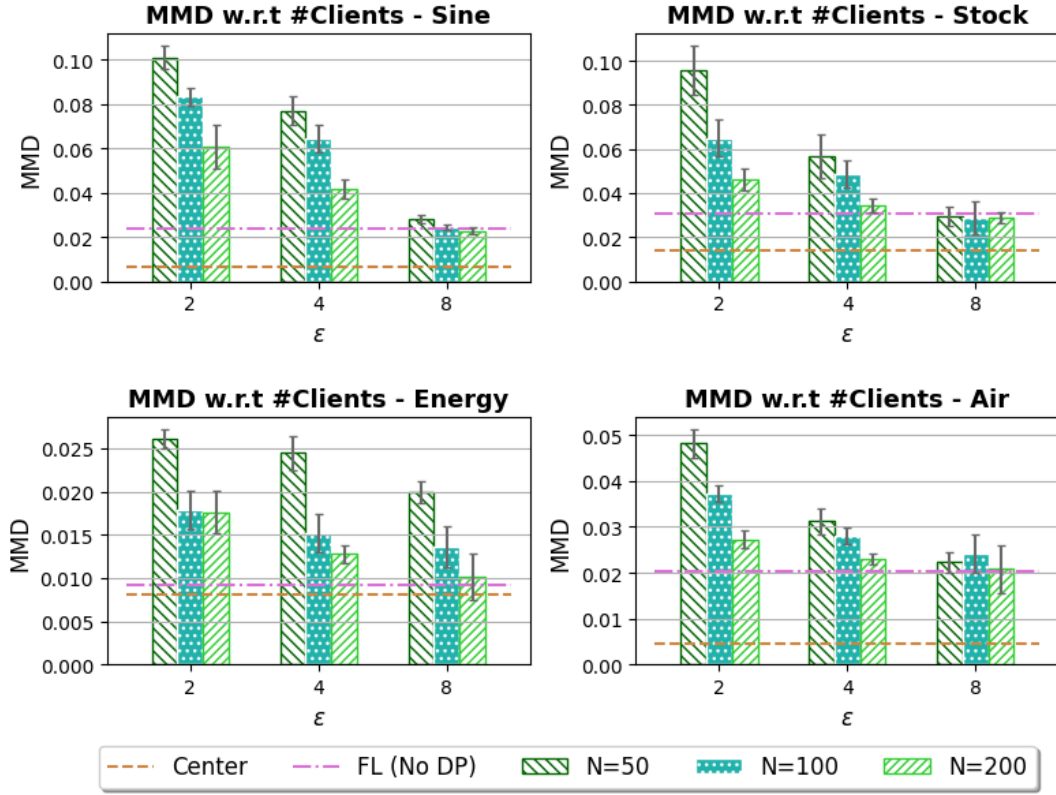


Figure 5.7 Analysis of the impact of per-round clients. For each dataset, we train the RWAE model with 50, 100, and 200 per-round clients and compare the MMD of synthetic data under different privacy settings.

when ϵ is large, the FL training is less affected by the local randomization process. Thus, the improvement of MMD by using pre-training is less significant.

5.6.3 Evaluation of Privacy Protection

Although DP provides a formal definition for the level of privacy protection, there is currently no concrete interpretation of how to choose an appropriate ϵ in practice for a satisfactory utility-privacy trade-off. Therefore, we empirically analyze the performance of our framework in defending the membership inference attack (MIA). We follow the black-box MIA proposed in [62], where the attacker uses the distance between a target record and a published synthetic dataset to infer whether the record is used to train the RWAE model. Intuitively, the generative models tend to produce synthetic data that are more similar to the training data. Therefore, the more synthetic records close to the target record, the higher the probability the target record is included in the training data. Let x be a target record and \mathcal{X}_{syn} be the published synthetic dataset, we denote $\mathcal{R}_\rho(x) = \{x' | \Gamma(x, x') \leq \rho\}$ as the ρ -neighborhood of x under a certain

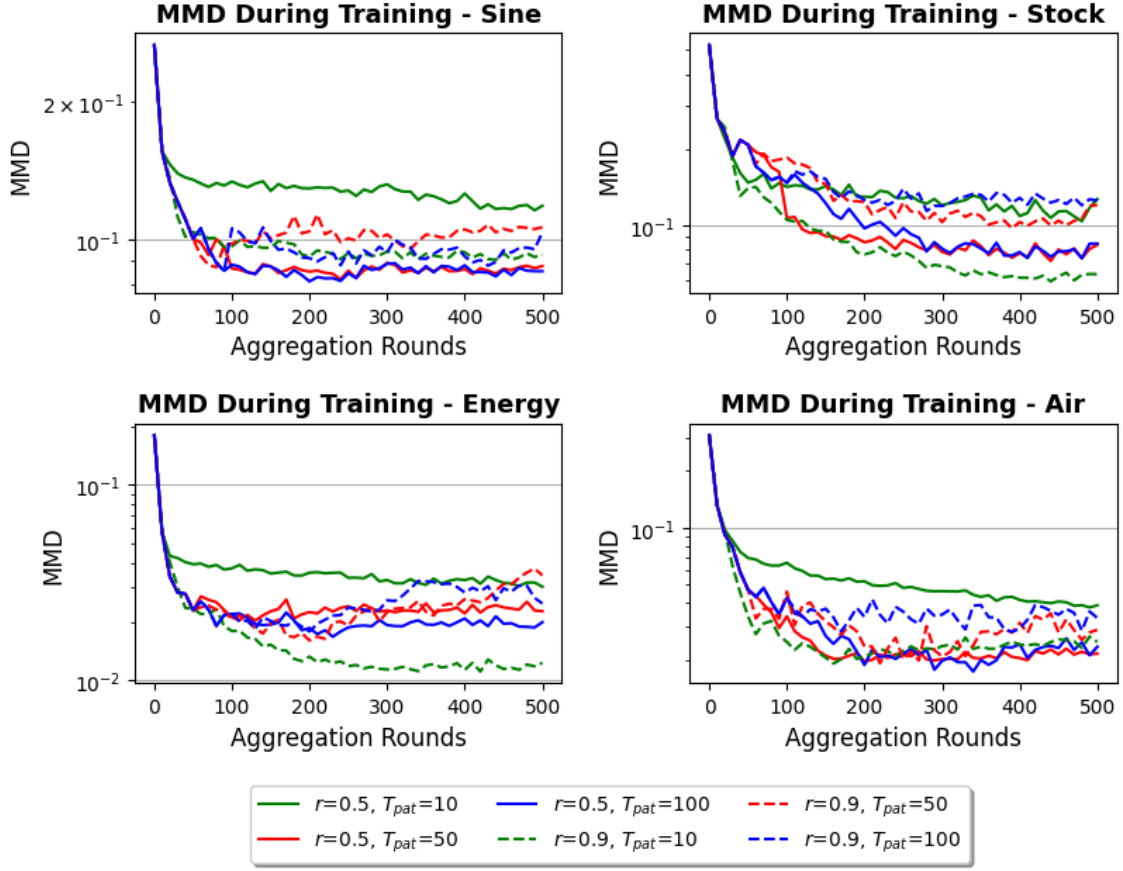


Figure 5.8 Analysis of the impact of decay rate r and patience threshold T_{pat} . For each dataset, we respectively use decay rate $r \in \{0.5, 0.9\}$ and patience threshold $T_{pat} \in \{10, 50, 100\}$ to train the RWAE model for 500 rounds under $\epsilon = 6$. Results show the MMD of synthetic data during the training process.

distance metric Γ . The attacker then computes the ratio of synthetic records that fall into the neighborhood of x , namely

$$\mathcal{R}_\rho(x) = \frac{1}{|\mathcal{X}_{syn}|} \sum_{x_{syn} \in \mathcal{X}_{syn}} \mathbb{1}[x_{syn} \in \mathcal{U}_\rho(x)], \quad (5.11)$$

where x_{syn} is an individual synthetic record in \mathcal{X}_{syn} and $|\mathcal{X}_{syn}|$ is the size of the synthetic dataset. Clearly, the higher $\mathcal{R}_\rho(x)$, the more likely that x is in the training data.

In our experiments, we construct the target dataset with 100 training records and 100 testing records (which are not used for training the RWAE model). The ground truth labels of the training and testing records are 1 and 0, respectively. For each experiment, we generate a synthetic dataset \mathcal{X}_{syn} containing 10^5 records. Then, we compute the Euclidean distance between the target and synthetic records and further derive the ratio $\mathcal{R}_\rho(x)$ of each target record. Here, we follow [62] and set ρ as the median of the minimum distance of each target record. After sorting the $\mathcal{R}_\rho(x)$ of all the target records, we assign a prediction label 1 to the 100 records with the largest $\mathcal{R}_\rho(x)$ and a prediction label 0 to the rest. Finally, we compute

Table 5.4 Comparison of MMD of synthetic data generated by RWAEs with (w) and without (w/o) pretraining.

	$\epsilon = 2$			$\epsilon = 8$		
	w/o	w	improve	w/o	w	improve
Sine	0.101	0.064	0.037	0.024	0.020	0.004
Stock	0.095	0.070	0.025	0.028	0.023	0.005
Energy	0.026	0.019	0.007	0.020	0.017	0.003
Air	0.048	0.034	0.014	0.028	0.025	0.003

Table 5.5 Averaged attack accuracy of the MIA attack under different privacy settings.

Dataset	Center (No DP)	FL (No DP)	SubMDS		
			$\epsilon = 8$	$\epsilon = 2$	$\epsilon = 0.5$
Sine	0.706	0.634	0.586	0.548	0.514
Stock	0.717	0.650	0.613	0.559	0.501
Energy	0.613	0.594	0.579	0.531	0.509
Air	0.643	0.621	0.615	0.587	0.539

the attack accuracy regarding the ground truth labels and the predicted results. We repeat each experiment 5 times and present the averaged attack accuracy under different privacy settings in Table 5.5. It can be seen that synthetic data generated by the non-private RWAEs are still likely to reveal the membership information of the target record. For the four datasets, the attack accuracy on synthetic data trained under the non-private FL setting can be more than 60%, and the accuracy under the non-private centralized setting can even exceed 70%. On the other hand, training the RWAEs with DP can help mitigate information leakage. More specifically, with $\epsilon = 0.5$, the attack accuracy is reduced by 11% \sim 21% and is close to 50%, namely the performance of random guesses. Even with $\epsilon = 8$, the attack accuracy can still be reduced by 3% \sim 12%. The results demonstrate that our framework is able to reduce the risk of MIA and provide privacy protection to the local data.

5.6.4 Evaluation of Robustness

Finally, we investigate whether the proposed SUBMDS algorithm can also help prevent poison attacks and improve the robustness of FL training. Here we focus on the scenarios of model poisoning attacks, where a number of malicious local clients modify the real local updates in order to disturb the model convergence. We consider two common attack strategies, namely *random updates* and *sign flip*. The former replaces the real update with a random update, while the latter flips the signs of update values. In the experiments, we presume a presence of 20% malicious clients. We then train the RWAE models using the non-private setting as well as the SUBMDS algorithm with $\epsilon \in \{2, 8\}$ and evaluate the predictive MAE of synthetic data under both model poisoning attacks. In Figure 5.9 we report the predictive MAE of the four datasets. Noticeably, for RWAEs trained in the non-private setting, a significant increase in predictive MAE is observed on applying both poisoning attacks, suggesting that the training process is easily influenced by the attacks. In contrast, training RWAEs with SUBMDS effectively reduces the MAE. In particular, with $\epsilon = 8$, the MAE under the *random updates* attack

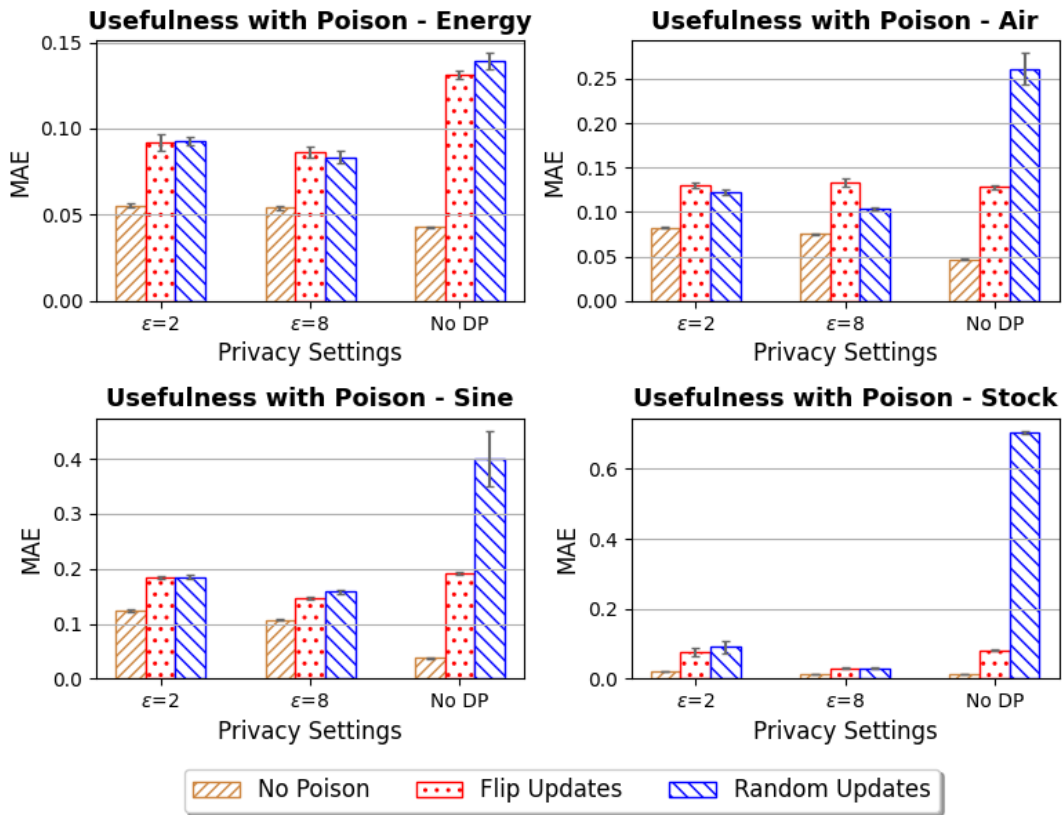


Figure 5.9 The effectiveness in defending against model poison attacks evaluated on the four datasets. For each dataset, we present the predictive MAE of synthetic data generated under different poisoning attacks and privacy settings.

is reduced by 0.25 for **Sine** and 0.7 for **Stock** in comparison to the non-private setting. This is because SUBMDS only selects a subset of dimensions for each local update, which limits the impact of random updates on the model convergence. Moreover, although decreasing ϵ increases the MAE for both poisoned and unpoisoned models because of the larger randomness introduced during training, it can still lessen the effect of poisoning attacks compared to the non-private setting. The results demonstrate that applying SUBMDS algorithm on the local side can help mitigate different poison attacks and help improve the robustness of FL.

5.7 Conclusion

synthetic data generation (SDG) has recently attracted increasing attention because of its potential to address data insufficiency and privacy issues in data mining and building AI services. However, most prior SDG works are conducted under the centralized setting, which assumes the real client data have been collected by the server and will be used for private data publishing. On the other hand, although recent works propose solutions for generating synthetic data under the distributed setting, they are currently only limited to structured and image data.

5 FEDSTDG for Private Sharing of Multivariate Time Series

In this paper, we offer the first attempt to generate synthetic time-series data under the distributed setting. We propose FEDSTDG, an effective framework for synthesizing local multivariate time-series data with comprehensive privacy guarantees. The framework utilizes a time-series autoencoder to learn the distributions and correlations of real local data and generate high-fidelity synthetic data on the server side. Moreover, we propose SUBMDS, a novel local randomization algorithm for preventing potential privacy leakages in our framework. We show that the algorithm not only provides strict LDP guarantees but also contributes to improving the utility of synthetic data compared to existing LDP-FL algorithms. Finally, we introduce MAGRR, a privacy-preserving algorithm for adaptively adjusting the global learning rate during FL training. Extensive evaluation with real-world datasets demonstrates the capability and efficiency of our framework in synthesizing time-series data under strong privacy protection. The synthetic data preserves similar statistical properties as real data and can be easily scaled up for future data mining and AI training tasks.

6 VERTIGAN for Private Sharing of Vertically-Partitioned Structured Data

Table 6.1 Bibliographic details for P4

Title	Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data
Authors	Xue Jiang ^{1,2} (xue.jiang@tum.de) Yufei Zhang ¹ (yufei.zhang@tum.de) Xuebing Zhou ² (xuebing.zhou@huawei.com) Jens Grossklags ¹ (jens.grossklags@in.tum.de)
	¹ Technical University of Munich (TUM), Boltzmannstraße 3, 85748 Garching, Germany ² Huawei Munich Research Center, Riesstraße 25, 80992 Munich, Germany
Type	Journal
Outlet	Proceedings on Privacy Enhancing Technologies, Volume 2023 ³
Ranking	CORE 2021 ⁴ : A
Status	Published
Citation	Jiang, X., Zhang, Y, Zhou, X., & Grossklags, J. (2023). Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data. Proceedings on Privacy Enhancing Technologies, 2023(2), 236-250.
Copyright	This work is published under a Creative Commons Attribution 4.0 License ⁵ .
Author Contributions	Xue Jiang developed the core idea for the paper and took primary responsibility for designing the methodology. Yufei Zhang was responsible for acquiring data, implementing experiments and evaluating results. The manuscript was drafted by Xue Jiang, and Xuebing Zhou and Jens Grossklags provided valuable feedback and suggestions on the methodology and experiments, and assisted in reviewing and enhancing the manuscript.

³<https://www.petsymposium.org/popets/2023/>

⁴<http://portal.core.edu.au/conf-ranks/1442/>

⁵<https://creativecommons.org/licenses/by/4.0/>

6.1 Abstract

In the previous chapter, we focused on the scenarios where data are horizontally-partitioned, where all the local datasets share the same set of features. In this chapter, we further explore privacy-preserving data transmission under a vertical setting. Namely, the local parties hold different features of the same set of users. This setting is also common in real life, as the user data collected by different AI service providers are usually different. Since these data describe the user profiles from various perspectives, gathering and analyzing the data from all the local parties would enable the developers to gain a better understanding of the user population. Nevertheless, the publication of vertically-partitioned data faces a dilemma: on the one hand, the original data cannot be directly shared by local parties due to privacy concerns; on the other hand, independently privatizing the local datasets before publishing may break the potential correlation between the cross-party attributes and lead to a significant utility loss. Prior solutions compute the privatized multivariate distributions of different attribute sets for constructing a synthetic integrated dataset. However, these algorithms are only applicable to low-dimensional structured data and may suffer from large utility loss with the increase in data dimensionality.

Following the idea of synthetic data generation, we further propose VERTIGAN, the first framework based on a generative adversarial network (GAN) for publishing vertically-partitioned data with privacy protection. The framework adopts a GAN model comprised of one multi-output global generator and multiple local discriminators. The generator is collaboratively trained by the server and local parties to learn the distribution of all parties' local data and is used to generate a high-utility synthetic integrated dataset on the server side. Additionally, we apply differential privacy (DP) during the training process to ensure strict privacy guarantees for the local data. We evaluate the framework's performance on a number of real-world datasets. Extensive experimental results demonstrate the capability and efficiency of our framework in synthesizing vertically-partitioned data while striking a satisfactory utility-privacy balance.

6.2 Introduction

With the rapid development of network and computer technologies, large and diverse quantities of user data have been extensively collected and stored by different companies and institutes (referred to as local parties). These data usually contain rich information characterizing user profiles, which is valuable for data mining and building AI services. Due to the variety of service scenarios, the user data are often vertically partitioned and distributed among these local parties. That is, the local dataset held by each party usually contains different attributes of the same group of users. Considering that the more attributes the data consist of, the more information can be used for describing an individual user, it is practical for local parties to collaborate with each other and publish an integrated dataset with all the attributes for better decision making or building high-accuracy services. For instance, in a healthcare scenario, a group of specialist hospitals could publish a joint dataset to study potential correlations between different types of illnesses such as cancer, and heart and lung diseases. Similarly, in a smart finance scenario, a loan company could use a dataset jointly published by a bank and

an e-commerce company to more deeply explore the key attributes that may result in higher default risk. More generally, integrating and analyzing these vertically-partitioned datasets enables data analysts to explore the hidden correlations of attributes from different perspectives and thus obtain a better understanding of the characteristics of user groups. This can be of significant help in designing optimized data mining algorithms and machine learning models.

However, publishing vertically-partitioned datasets has to be cognizant of the restrictions of data protection regulations such as the GDPR and users' privacy concerns. On the one hand, since the local data are generated based on users' ongoing behaviors and may contain sensitive information of individual users, directly sharing the original local datasets with an untrusted third party may lead to serious privacy leakage (see, for example, [18, 11]). On the other hand, the local parties can use state-of-the-art privacy-enhancing techniques, such as differential privacy (DP) [42], to process the real data and only share the privatized datasets. Nevertheless, each party individually privatizing the local data may break the correlations and joint distributions among attributes held by different parties and lead to distinctive utility loss in the published dataset. Therefore, solutions for publishing vertically-partitioned data under a satisfactory privacy-utility balance are greatly needed.

In comparison to the substantial attention given to privacy-preserving data mining and machine learning under a vertical setting, algorithms for publishing the vertically-partitioned data are still barely studied. Prior works [72, 87] proposed two-party publication protocols under k -anonymity guarantees [141]. Unfortunately, later studies [166, 140] pointed out that k -anonymity models are vulnerable to various privacy attacks and cannot provide sufficient privacy protection. Follow-up work [114] proposed the first algorithm for publishing vertically-partitioned data under DP guarantees. However, the algorithm is limited to two-party scenarios and requires pre-defined taxonomy trees for all categorical attributes. Recent work by Tang *et al.* [142] proposed to use a latent tree model [181] to represent the cross-attribute distributions in the original dataset and privatizes the latent tree parameters via a distributed Laplace protocol to achieve ϵ -DP for each local dataset. Although the work by Tang *et al.* [142] effectively improves data utility and efficiency compared to [114], the algorithm evenly splits the privacy budget to all the attribute pairs. Therefore, the noise scale may increase exponentially with the data dimensionality and cause significant utility loss. Moreover, the algorithm is limited to discrete structured datasets and cannot support other data types.

In recent years, data synthesis has increasingly been considered a useful approach for addressing data insufficiency problems in developing AI applications. With the strong capabilities of characterizing the correlations and distributions of high-dimensional data, deep generative models such as generative adversarial network (GAN) are increasingly used for generating high-utility and low-sensitivity synthetic data. Although some recent works (*e.g.*, [146, 74]) also proposed training the generative models under the federated learning (FL) framework to avoid the direct collection of real local data, the solutions all focus on the horizontal setting, which cannot be directly applied to vertically-partitioned data.

In this work, we address this research gap and propose VERTIGAN, the first GAN-based framework for privacy-preserving publication of vertically-partitioned data. The framework adopts a distributed GAN architecture, comprised of a global generator and multiple local discriminators. By using a collaborative training strategy, the global generator is trained without accessing the real local data. Moreover, we adopt a multi-output structure for the generator, which enables the model to directly learn the correlations and distributions of the attributes

held by different local parties and generate synthetic integrated data. Finally, we inject DP perturbation during the training process, which ensures that the generator and the synthetic data satisfy strict DP guarantees for each local party. The main contributions of our approach are as follows:

- We propose VERTIGAN, an efficient and privacy-preserving framework for publishing vertically-partitioned data. The framework trains a multi-output global generator to directly learn the distribution of all parties' local data and to generate high-utility synthetic integrated data on the server side. To the best of our knowledge, this is the first framework based on a deep generative model for private data publication under the vertical setting.
- We introduce a distributed training strategy, where the global generator is updated based on the gradients calculated by the local discriminators. The strategy eliminates the need to access real local data when training the global generator. Moreover, we apply DP perturbation during the training process to provide a strict privacy guarantee for each local dataset.
- We implement our framework and evaluate the performance on a number of real-world datasets containing 68–1501 classification attributes. Through comparison with the previous statistics-based algorithms, we show that the synthetic data generated by our framework always preserve much closer joint distributions and correlations to real data. Moreover, with a local privacy guarantee $\epsilon = 8$, we achieve around 2% ~ 15% improvement in classification accuracy compared to the baseline algorithms. Extensive evaluation experiments show that our framework has outperforming capability and efficiency in collecting high-dimensional data while offering a favorable utility-privacy balance.

6.3 Related Work

6.3.1 Data Analysis on Vertically-Partitioned Data

In recent decades, data analysis on vertically-partitioned data has attracted increasing attention. Different from the horizontal setting, vertical partitioning refers to the scenario where local parties collect different attributes of the same set of users. Existing applications on vertically-partitioned data include, for instance, jointly training ML models using attributes of all the local parties, or publishing an integrated dataset for future data mining.

Machine Learning Under Vertical Setting

In the context of machine learning (ML), prior studies by Vaidya *et al.* proposed a series of secure multi-party computation (SMC) protocols [176] for training different models on vertically-partitioned data, including Bayes classifier [150], and decision trees [151], etc. Hardy *et al.* [58] proposed a vertical federated learning (VFL) framework that trained LR models using homomorphic encryption (HE) [32]. Yang [173] further applied the quasi-Newton method in VFL to reduce the number of communication rounds. Some other works [27, 167] also proposed solutions for tree-based models and neural networks [132]. Besides using crypto-based tech-

nologies such as HE and SMC to ensure security in VFL, recent works [26, 157] further proposed to incorporate DP into the training process to provide strict privacy guarantees for local data.

On the other hand, some recent works also investigate potential privacy attacks against VFL, which include label inference attacks and feature reconstruction attacks. In the label inference attacks, the parties without ground-truth labels aim to use the back-propagated gradients to infer the sample labels. Several existing attacks proposed to explore the difference of the gradient norms [94] or the sign of the last-layer gradients [101, 187]. Other research [48] also proposed a semi-supervised learning approach that first estimated the bottom-layer parameters and then used the “completed” model to “generate” the label of arbitrary samples. Apart from the label leakage, some other works [102, 73] also studied the feature leakage in VFL, where the party obtaining the model predictions tries to reconstruct the input features of other parties. Nevertheless, existing attacks against VFL only focused on classification models, where the attackers either try to infer the ground-truth labels or need to use the model predictions to reconstruct local features. In contrast, in this paper, we use the GAN model for data synthesis, which does not involve such label (or prediction) information. Hence, the above-mentioned attacks in VFL are no longer applicable.

Data Publication Under Vertical Setting

Compared to the extensive set of studies on machine learning under the vertical setting, there are still only limited prior works on publishing vertically-partitioned data. Prior works in [72, 87] proposed SMC-based protocols for two-party data publication under k -anonymity guarantees [141]. Nevertheless, later studies [166, 140] pointed out that k -anonymity models are vulnerable to various privacy attacks and cannot provide sufficient privacy protection. In contrast, DP [42] is considered as a more principled approach for private data publication. Mohammed *et al.* proposed DistDiffGen [114], the first algorithm for publishing vertically-partitioned data under DP guarantees. DistDiffGen first generalizes the raw data using a distributed exponential mechanism and then adds noise to the distributions to ensure ϵ -DP. However, the algorithm is limited to two-party scenarios and requires pre-defined taxonomy trees for all categorical attributes, which may not always be available in practice. Later work by Tang *et al.* [142] proposed an improved differentially private latent tree (DPLT) algorithm, which first uses a latent tree model [181] to represent the cross-attribute distributions in the original dataset and then privatizes the latent tree parameters via a distributed Laplace protocol to achieve ϵ -DP for each local dataset. The latent tree model will then be used for generating a synthetic dataset. Although [142] significantly improves the data utility and efficiency in comparison to [114], it is still limited to discrete attributes. Moreover, since the privacy budget is evenly split over all the attribute pairs, the noise scale may increase exponentially with the increased data dimensionality and cause a large utility loss.

In this paper, we propose a distributed GAN-based protocol for publishing vertically partitioned data in a private manner. Compared to previous works, our solution can support the publication of high-dimensional datasets with strict DP guarantees. Moreover, the framework can be further extended to support other types of data such as numerical data and images.

6.3.2 Differentially Private Data Synthesis

DP data synthesis has been extensively studied over recent years as one of the solutions for privacy-preserving data publishing. Previous statistics-based works [25, 179] computed joint distributions of original structured data under DP guarantees and used them to generate synthetic datasets. However, these methods can only be applied to structured data and may suffer from a significant utility loss with the increase in data dimensionality.

Inspired by the rapid evolution of deep learning, later works proposed to directly train generative models such as autoencoders [85, 6] and generative adversarial networks (GANs) [53] and to generate high-utility synthetic data. Nevertheless, simply training these generative models without protection may still lead to privacy leakage. For instance, prior work [152, 8] showed that GANs may unintentionally memorize the training data. Moreover, Hayes *et al.* [60] proposed different membership inference attacks against the trained generator and discriminators. Later works also demonstrated that the membership information can be revealed from the generated synthetic data [62, 24, 138]. In addition, Zhou *et al.* [185] performed a property inference attack, which uses the synthetic data to infer the macro-level information of training data (*e.g.*, the ratio of samples regarding a certain property).

DP has been considered one of the countermeasures against such privacy attacks. Existing DP data synthesis algorithms are generally divided into two categories, namely by using differentially private stochastic gradient descent (DPSGD) [1] or private aggregation of teacher ensembles (PATE) [121]. The DPSGD-based algorithms [170, 183] perturb the model gradients in each iteration by clipping and adding Gaussian noise to ensure DP guarantees. The PATE-based algorithms [77, 156] first train a group of teacher models (*e.g.*, the discriminator in GAN) on non-overlapping subsets of original data and then use the noisy predictions from the teacher group to train the student model (*e.g.*, the generator). Nevertheless, previous data synthesis algorithms mainly focus on the centralized setting, where the server has already collected the clients' real data. This may not always be realistic since the clients may refuse to share their personal local data with untrusted servers. Therefore, some recent works also proposed to train the generative autoencoders [74] and GANs [146, 180] under the FL framework to avoid the collection of original data. However, existing solutions only focus on the horizontal setting, where the local data shares the same set of attributes. In contrast, in this paper, we conduct the first attempt at the GAN-based DP data synthesis for vertically-partitioned data.

6.4 Problem Statement

In this work, we focus on the scenario where the user data are vertically partitioned and distributed over multiple local parties. Each party possesses a different set of attributes of the same group of samples. A central server aims to integrate these local datasets in a private manner and publish a joint dataset containing all the attributes. The joint dataset will be further used by external data analysts for downstream data mining and model training tasks.

An illustration of the system setting is shown in Figure 6.1. We assume there are N local parties $\mathcal{P}_1, \dots, \mathcal{P}_N$. Each party \mathcal{P}_i has a local dataset containing a *different* set of attributes $A^i = \{a_1^i, \dots, a_{|A^i|}^i\}$. Here, the attribute sets can be either partially overlapping or non-overlapping. Moreover, each party may hold samples not covered by other parties. Therefore,

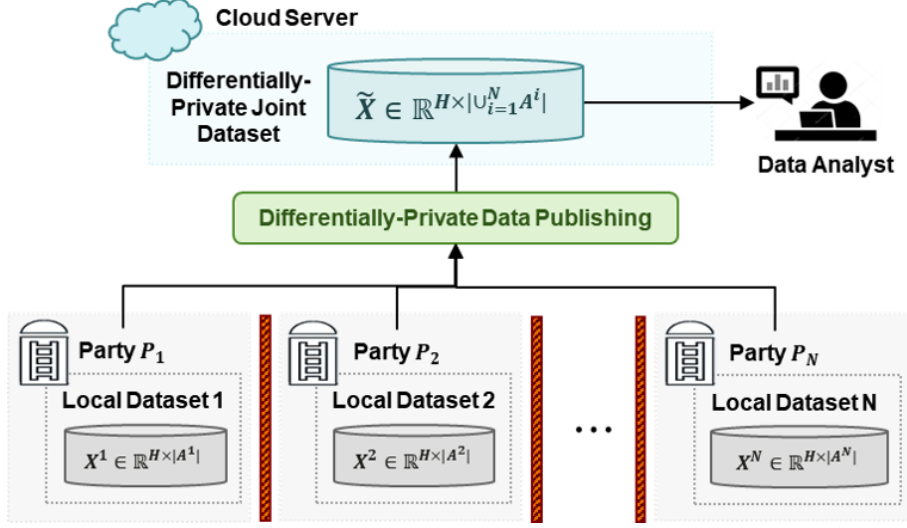


Figure 6.1 Overview of the system model.

we assume that the local data have certain alignable sample IDs (e.g., ID number, cellphone number, etc.). The local parties can use private set intersection (PSI) protocols (e.g., [34, 66, 29]) to determine the intersecting sample IDs without exposing the non-intersect samples. Then, each party sorts the common samples according to their IDs and obtains the final training dataset $X^i \in \mathbb{R}^{H \times |A^i|}$, where H is the number of samples and $|A^i|$ is the number of attributes.

The goal of the task is to design a privacy-preserving framework, where a central server can collaborate with all the local parties and publish a private joint dataset $\tilde{X} \in \mathbb{R}^{H \times |\cup_{i=1}^H A^i|}$ that contains the full set of attributes. The joint dataset \tilde{X} preserves both single-party and cross-party attribute correlations. More specifically, consider local parties \mathcal{P}_i and \mathcal{P}_j respectively holding local datasets $X^i \in \mathbb{R}^{H \times |A^i|}$ and $X^j \in \mathbb{R}^{H \times |A^j|}$, then the distribution of \tilde{X} should satisfy

$$P_{\tilde{X}}(A^i) \approx P_{X^i}(A^i), \quad P_{\tilde{X}}(A^i, A^j) \approx P_{X^i, X^j}(A^i, A^j). \quad (6.1)$$

Following previous works, we assume that the local parties and the central server are *honest-but curious*, who correctly follow the protocols but try to infer sensitive information of other local datasets. Moreover, we also consider the threat posed by external data analysts, who aim to use the published joint dataset to re-identify sensitive information of specific users. Based on the considerations above, it is required that there is no information exchange among local parties and each party does not know the attribute set of other parties. Moreover, we assume that the server cannot directly access the raw local data but is aware of the full attribute set and the size of the training dataset. Finally, the published dataset should satisfy strict DP guarantees and not reveal the privacy of individual users in the local datasets.

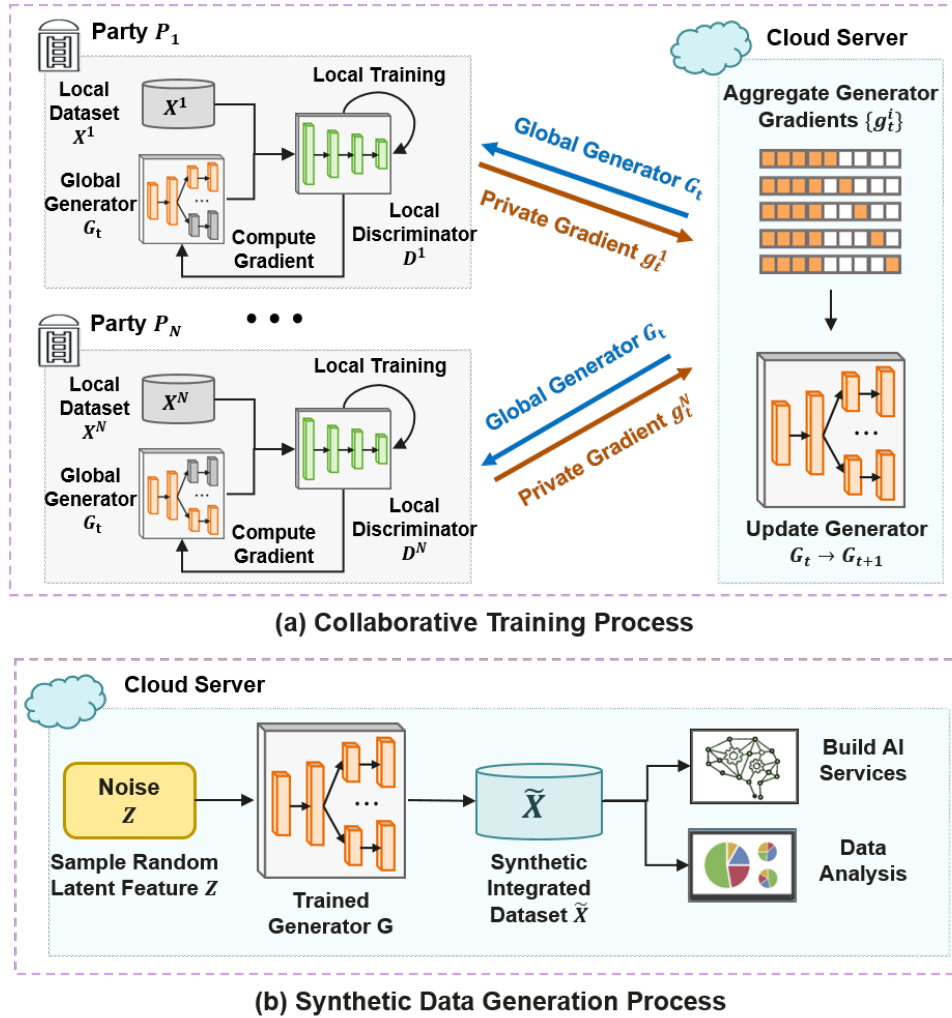


Figure 6.2 General workflow of the VERTIGAN framework.

6.5 Proposed Framework

Although previous works proposed statistics-based algorithms for publishing vertically-partitioned data under DP guarantees, the solutions are only limited to low-dimensional structured data and may suffer from large utility loss with the increase in domain size. Following the idea of data synthesis, we propose VERTIGAN, the first GAN-based framework for differentially-private publication of vertically-partitioned data. The overall workflow of the framework is presented in Figure 6.2, which consists of two phases, namely the *collaborative training process* and the *synthetic data generation process*. In the first process, a GAN model is collaboratively trained by the server and all the local parties to learn the correlations and distributions of all the local datasets in a private manner. In the second phase, the generator part is used to directly generate synthetic integrated data that contains attributes held by all the local parties. The synthetic data preserves similar statistical properties to real data and can be alternatively used for downstream data analysis and AI training tasks.

Nevertheless, training the GAN model on distributed vertically-partitioned data faces several challenges. To start with, in this work, we focus on the scenario where the real data are distributed on the local side and cannot be directly shared with the server. Hence, the model cannot be simply trained as in the centralized setting due to *data inaccessibility*. Moreover, in the vertical setting, the attribute sets held by the local parties are usually different from each other, which is referred to as *attribute inconsistency* in this work. This causes existing solutions that train GANs in the horizontal FL framework to be inapplicable. Finally, recent contributions (e.g., [137, 21]) point out that the ML models may memorize information in training data and suffer from different privacy attacks. Therefore, *privacy protection* techniques should be applied during model training to prevent potential privacy leakage. We apply corresponding solutions in the VERTIGAN framework to address the above-mentioned challenges. In the following sections, we will respectively introduce each solution in detail.

6.5.1 Distributed GAN Against Data Inaccessibility

Different from other generative models, GANs are usually built with two independent networks, namely a generator and a discriminator. The two networks are trained in an adversarial manner to improve their own performance. By taking advantage of GANs' separate generator-discriminator architecture, the VERTIGAN framework applies a distributed training strategy to address *data inaccessibility* problems. More specifically, the framework deploys a global generator G on the server side and multiple discriminators $\{D^1, \dots, D^N\}$ on the local side. The global generator takes in random latent features and outputs synthetic data for each local party, while the local discriminators are trained on the local side to distinguish between real data and synthetic data. The ultimate goal of the framework is to obtain a well-trained global generator on the server side that is capable of producing high-utility synthetic data without violating the privacy of real local data.

The training process is conducted in cooperation with the server and all the local parties, as shown in Figure 6.2. During each global training round, the server broadcasts the current global generator to all the local parties for generating synthetic data. Each party first uses its real local data and the corresponding part of synthetic data to train its local discriminator and then uses the trained discriminator to compute the generator's gradient. Finally, the gradients from all the local parties will be aggregated on the server side and used to update the global generator. In Figure 6.3, we also present a detailed illustration of the local training process. It can be seen that the local data are only used for training the local discriminator D^i , and the global generator G is only updated based on the gradient computed by the trained discriminators. Moreover, only the information (weights and gradients) of the generator is exchanged between the local and server side, while the discriminators and the real data are always kept on the local side. In this way, the framework can facilitate the training of the global generator without direct access to the real local data.

6.5.2 Multi-Output Generator Against Attribute Inconsistency

Moreover, in this work, we consider the scenario where the user data are vertically-partitioned and distributed among M local parties. Since the local parties under this setting may hold

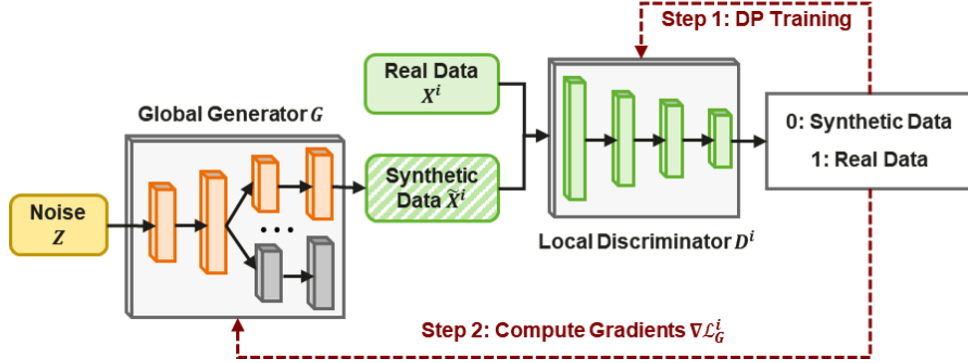


Figure 6.3 Workflow of the local training process.

different sets of attributes, the conventional single-output generators are not applicable to the framework. In order to address the *attribute inconsistency* problem, we propose a multi-output structure for the global generator. The generator consists of several common layers (denoted as G^0) and M separate follow-up branches (denoted as $\{G^1, \dots, G^N\}$). Each branch G^i produces synthetic data with attributes of one local party \mathcal{P}_i . Given a batch of input feature Z , the global generator is capable of concurrently producing synthetic data $\{\tilde{X}^1, \dots, \tilde{X}^N\}$ for all the local parties. Here, $\tilde{X}^i = G^i(G^0(Z))$ corresponds to the data generated from the i -th branch.

We follow the optimization approach of WGAN introduced in Section 2.3.2 to iteratively train the global generator and local discriminators in the proposed framework. On the one hand, the training of the discriminators on the local side is conducted as under the centralized setting. Here, the loss function for the i -th local discriminator D^i is:

$$\mathcal{L}_D^i = D^i(x^i) - D^i(\tilde{x}^i) + \lambda(\|\nabla_{\hat{x}^i} D(\hat{x}^i)\| - 1)^2, \quad (6.2)$$

where x^i is the real data of the i -th local party, $\tilde{x}^i = G^i(G^0(z))$ is the synthetic data generated by the i -th branch of G , \hat{x}^i is the gradient penalty as defined in Equation (2.16), and λ is the weight for the gradient penalty. Once the discriminators have been trained for several iterations, they will be used to compute the gradient of the global generator. The loss function for the global generator G can be computed as the sum of the loss regarding all the local discriminators, each of which is derived following Equation (2.17):

$$\mathcal{L}_G = \sum_{i=1}^N \mathcal{L}_G^i = \sum_{i=1}^N D^i(G^i(G^0(z))). \quad (6.3)$$

The generator's gradient $\nabla \mathcal{L}_G$ can be further derived as

$$\begin{aligned} \nabla \mathcal{L}_G &= \frac{\partial \sum_{i=1}^N \mathcal{L}_G^i}{\partial G} = \sum_{i=1}^N \frac{\partial \mathcal{L}_G^i}{\partial [G^0, G^1, \dots, G^N]} \\ &= \left[\frac{\partial \mathcal{L}_G^1}{\partial G^0}, \frac{\partial \mathcal{L}_G^1}{\partial G^1}, 0, \dots, 0 \right] + \dots + \left[\frac{\partial \mathcal{L}_G^N}{\partial G^0}, 0, 0, \dots, \frac{\partial \mathcal{L}_G^N}{\partial G^N} \right] \\ &= \left[\sum_{i=1}^N \frac{\partial \mathcal{L}_G^i}{\partial G^0}, \frac{\partial \mathcal{L}_G^1}{\partial G^1}, \dots, \frac{\partial \mathcal{L}_G^N}{\partial G^N} \right] \end{aligned} \quad (6.4)$$

which is the sum of the generator gradients from all the local parties. Hence, by aggregating all the returned generator gradients, the server achieves to use the sum of the gradients to update the global generator. It can be seen from Equation (6.4) that the parameters of each branch G^i are updated based on the gradients from party \mathcal{P}_i , while the parameters of the common layers G^0 are updated by the gradients from all the local parties. Therefore, the multi-output structure enables the global generator to automatically capture the correlations and distributions of attributes across local parties during the training process and directly generate synthetic integrated data with the entire attribute set.

6.5.3 Collaborative Training with Differential Privacy

In the previous sections, we illustrate how the VERTIGAN framework enables a global generator to learn the hidden correlations of attributes across all the local parties without actually accessing the real local data. Nevertheless, recent studies (e.g., [137, 21]) showed that the trained generator may reveal sensitive information of real local data under various privacy attacks. In order to mitigate potential privacy risks, we further apply DP during the training process, which provides strict privacy guarantees to the local datasets.

Considering the global generator does not directly access real local data, we follow previous DP-GAN algorithms [170, 183] and only perturb the gradients of local discriminators to achieve privacy protection. Specifically, in each update step of the discriminator, we first sample a batch of real local data and synthetic data, and then compute the corresponding gradients $\{g_D^{i,b}\}_{b \in B}$. Each gradient $g_D^{i,b}$ is then clipped by a pre-defined L_2 -norm bound C , namely

$$\bar{g}_D^{i,b} = \text{clip}(g_D^{i,b}, C) = g_D^{i,b} / \max(1, \|g_D^{i,b}\|_2 / C). \quad (6.5)$$

Next, we sum up all the clipped gradients, add random Gaussian noise $\mathcal{N}(0, \sigma^2 C^2 I)$, and divide the perturbed gradient by the batch size B as shown below:

$$\tilde{g}_D^i = \frac{1}{B} \left(\sum_{b=1}^B \bar{g}_D^{i,b} + \mathcal{N}(0, \sigma^2 C^2 I) \right). \quad (6.6)$$

The gradient \tilde{g}_D^i is used to update the discriminator parameters.

Since the local discriminator is repeatedly updated during the training process, according to the composition property, the total privacy cost should be accumulated. Considering that RDP achieves a much tighter privacy estimation in comparison to the traditional DP (as mentioned in Section 2.1.2), we first compute the overall privacy cost under the RDP definition and then convert it back to the traditional DP definition. To start with, the privacy cost of each gradient perturbation under RDP is derived as follows:

Corollary 3. *With a noise scale $\mathcal{N}(0, \sigma^2 C^2)$, the perturbed gradient \tilde{g}_D^i satisfies $(\alpha, \alpha/2\sigma^2)$ -RDP.*

Proof. Let $f = \sum_{b=1}^B \bar{g}_D^b = \sum_{b=1}^B \text{clip}(g_D^b, C)$ be the sum of all the gradients clipped by an L_2 -norm bound of C . The sensitivity of f can be derived as:

$$\Delta_2 f = \max_{X, X'} \|f(X) - f(X')\|_2 \leq C. \quad (6.7)$$

Algorithm 9: VERTIGAN - Workflow of Server

Input: G : global generator; N : number of local parties; T : global training rounds; η : learning rate; OPT : optimizer for the GAN model.

Output: Trained global generator G

Server executes:

```

1: Initialize global generator  $G$ 
2: for each local party  $i = 1, \dots, N$  do
3:   LocalInitialization() // Run on the local side
4: end for
5: for each global round  $t = 1, \dots, T$  do
6:   Sample random seed  $\vartheta$ 
7:   for local party  $i = 1, \dots, M$  do
8:     Distribute  $\vartheta$  and  $G$  to the local party  $i$ 
9:     Get local gradient  $g_G^i = \mathbf{LocalUpdate}(\vartheta, G)$ 
10:  end for
11:  Aggregate local gradients  $g_G = \sum_{i=1}^N g_G^i$ 
12:  Update generator  $G \leftarrow OPT.update(G, g_G, \eta)$ 
13: end for
14: return  $G$ 

```

Furthermore, the gradient perturbation process can be denoted as $\mathcal{M}_f = f + \mathcal{N}(0, \sigma^2 C^2 I)$. Based on Definition 6, the privacy cost of \mathcal{M}_f under the order α is

$$\epsilon(\alpha) = \frac{(\Delta_2 f)^2 \cdot \alpha}{2 \cdot \sigma^2 C^2} = \frac{C^2 \cdot \alpha}{2 \cdot \sigma^2 C^2} = \frac{\alpha}{2\sigma^2}. \quad (6.8)$$

As shown in Equation (6.6), the perturbed gradient will be divided by a batch size B , and the result \tilde{g}_D^i will be actually used to update the discriminator. Since B is unrelated to the real data, according to the post-processing property (Property 3), the final discriminator update \tilde{g}_D^i also satisfies $(\alpha, \epsilon(\alpha))$ -RDP. \square

According to Lemma 2, the privacy guarantee can be further amplified by subsampling. Given M as the total number of training data and B as the batch size, we compute the sampling rate as $\zeta = H/B$ and derive the amplified privacy cost $\epsilon'(\alpha)$ following Lemma 2. Next, assume the discriminator has updated for T_{total} steps during the entire training process, then the overall privacy cost is $(\alpha, T_{total} \cdot \epsilon'(\alpha))$ -RDP. We further convert privacy cost back to the traditional (ϵ, δ) -DP definition according to Lemma 1. Finally, since the global generator is trained on the local discriminators, according to Property 3, the global generator also satisfies (ϵ, δ) -DP for the corresponding local dataset.

6.5.4 Overall Training Process

With the above design considerations, we now describe the overall training process presented in Algorithm 9 and Algorithm 10.

Before the training starts, the server initializes the global generator G . On the local side, each party also initializes its local discriminator D^i . Moreover, considering that the local parties may have personalized privacy requirements, we let each local party individually compute the noise scale σ^i . With the universally configured batch size B , global rounds T , and local steps E , the discriminator’s total update step is derived as $T_{total} = T \cdot E$. Following the privacy accounting process described in Section 6.5.3, the required σ^i under the target privacy budget (ϵ^i, δ^i) can be determined accordingly. Finally, since each local party holds different attributes of the same group of samples, the local training data should be sample-wise aligned during each global round. A naive solution is to let the server randomly sample multiple batches of data indices for selecting the real data as well as input features for generating the synthetic data, and then broadcast all the information to the local side. However, this may cause extra communication costs, especially for large training batches. To address the issue, our framework applies a pseudorandom number generator (PRNG) Φ^i at *each* local party to realize the data alignment. Following prior works [14, 109], we use secure PRNGs to achieve comprehensive security guarantees. Moreover, we require that all the local PRNGs use the same algorithm and are deployed with the same configuration. Therefore, according to the reproducibility of PRNG, given the same random seed ϑ , each Φ^i is able to produce the same sequence of indices of real data or input features sampled from the standard Gaussian distribution. By using the PRNG, the server only needs to randomly sample a random seed and broadcast it to all the local parties in each global round, which significantly improves communication efficiency. Also, considering that existing secure PRNGs based on standard cryptographic primitives can have an output rate of gigabytes per second on modern CPUs [79], their computation cost is negligible compared to the local training time.

In each global training round, the server broadcasts the current global generator G as well as the random seed ϑ to all the local parties. Each party \mathcal{P}^i first sets Φ^i with the random seed ϑ and then updates the local discriminator D^i for E steps using the real data X^i and the synthetic data $\tilde{X}^i = G^i(G^0(Z))$ sampled by Φ^i . We apply the DP perturbation in each update step, where the batch of gradients is clipped by L_2 bound C and perturbed with random Gaussian noise $\mathcal{N}(0, \sigma^{i2} C^2 I)$. The noise scale σ^i is determined in the initialization process. Then, the local discriminator is used to compute the gradient g_G^i of the current global generator, which will be returned to the server for updating the parameters of the global generator parameters. The global training process is conducted for T rounds. Once the training completes, the server can use the global generator G to directly generate the synthetic dataset with attributes of all the local parties.

6.6 Experiments and Results

We implemented the proposed framework using the Tensorflow library and performed comprehensive experiments with a number of open-source datasets to evaluate its performance. In this section, we first introduce the experimental settings and then discuss the evaluation results.

Algorithm 10: VERTIGAN - Workflow of Local Party i

Input: G : global generator; D^i : party i 's local discriminator; X^i : party i 's local data; Φ^i : party i 's local PRNG; T : global training rounds; E : local update steps; B : batch size; η : learning rate; C : L_2 clipping bound; (ϵ^i, δ^i) : party i 's privacy budget; OPT : optimizer for the GAN model.

LocalInitialization():

- 1: Initialize local discriminator D^i , local PRNG Φ^i
- 2: Given the target (ϵ^i, δ^i) and the pre-defined (B, T_{total}, E) , compute the required noise scale σ^i

LocalUpdate(ϑ, G):

- 3: Get local data X^i , set $\Phi^i.set_seed(\vartheta)$
// Train local discriminator
 - 4: **for** $t = 1, \dots, E$ **do**
 - 5: Sample indices $J = \Phi^i.random_choice(size=B)$
 - 6: Sample input noise $Z = \Phi^i.random_normal(size=B)$
 - 7: **for** $b = 1, \dots, B$ **do**
 - 8: Let $x = X^i[J[b]]$, $\tilde{x} = G^i(G^0(Z[b]))$
 - 9: Compute $\mathcal{L}_D^{i,b}(x, \tilde{x})$ and $g_D^{i,b} = \nabla \mathcal{L}_D^{i,b}(x, \tilde{x})$
 - 10: Clip gradient $\tilde{g}_D^{i,b} = g_D^{i,b} / \max(1, \|g_D^{i,b}\|_2/C)$
 - 11: **end for**
 - 12: Aggregate gradients and add noise $\tilde{g}_D^i = \frac{1}{B}(\sum_{b=1}^B \tilde{g}_D^{i,b} + \mathcal{N}(0, \sigma^i{}^2 C^2 I))$
 - 13: Update discriminator $D^i \leftarrow OPT.update(D^i, \tilde{g}_D^i, \eta)$
 - 14: **end for**
// Compute generator gradient
 - 15: Sample input noise $Z = \Phi^i.random_normal(size=B)$
 - 16: Compute $g_G^i = \frac{1}{B} \sum_{b=1}^B \nabla \mathcal{L}_G(G^i(G^0(Z[b])))$
 - 17: **return** g_G^i
-

6.6.1 Experiment Setup

Datasets and Models

We used six multi-dimensional classification datasets for evaluating the performance of the VERTIGAN framework:

- **Web** [125] contains records with 124 binary attributes extracted from each web page. The goal was to train a classifier to determine whether the web page belongs to a category.
- **Vehicle** [39] contains data collected in wireless distributed sensor networks. Each record has 100 binary attributes representing data collected from different acoustic and seismic sensors. The goal was to train a classifier for vehicle type classification.
- **Census** [38] contains records drawn from the 1990 United States census data, including 68 personal attributes such as gender, income, and marital status. We used the dataset to classify the duration of people's active duty service.

Table 6.2 Datasets details

Dataset	Type	Num. Records	Num. Attributes	Domain Size
Census	Integer	2,458,285	68	2^{150}
Twitter	Integer	140,707	78	2^{181}
Web	Binary	36,974	124	2^{124}
Vehicle	Binary	98,528	101	2^{101}
HAR	Binary	10,299	561	2^{561}
Dilbert	Binary	10,000	1501	2^{1501}

Table 6.3 One-hot dimensions and the number of model parameters under the two-party setting

Dataset	Party 1		Party 2		Server
	One-hot Dim.	#Param. D^1	One-hot Dim.	#Param. D^2	#Param. G
Census	137	9,592	145	10,732	53,760
Twitter	195	19,307	174	15,313	94,768
Web	124	7,813	123	7,751	39,416
Vehicle	100	5,101	102	5,305	26,857
HAR	562	158,485	566	160,745	812,949
Dilbert	1,500	788,551	1,505	794,190	2,681,446

- **Twitter** [83] contains records with 77 attributes such as the number of discussions and average discussion length, which are used to predict the popularity magnitude of each instance. In our experiment, we quantified the values of each attribute into five bins. The goal was to classify the level of popularity of each instance.
- **Activity** [5] contains sensor records describing six daily activities. Each data record has 561 attributes representing different time and frequency domain variables. We normalize each attribute and convert the data to binary form.
- **Dilbert** was originally provided in [90] for object recognition. We use the processed version in [55], where the records are categorized into five classes. We take the first 1500 attributes from the processed data to exclude the irrelevant variables mentioned in [55]. Then, we normalize each attribute and convert the data to binary form.

Details of each dataset are presented in Table 6.2, including the data type, the number of records and attributes, and the domain size. In the experiments, we assume that each party holds 10^5 data records. To this end, we randomly sample 10^5 records from each original dataset and partition the datasets by feature. If the original dataset contains fewer records, the data are sampled with replacements. We further use one-hot encoding to convert the original categorical attributes to the numerical form for model training.

We design the global generator and local discriminators as multi-layer neural networks (NNs) and determine their layer size according to the one-hot dimension of the local datasets. The local discriminators are two-layer NNs whose output is a scalar between 0 and 1. The global generator is a multi-output model, which has two common layers followed by a number of sep-

arate branches. Each branch contains two fully-connected layers, which output the synthetic data of one party. In Table 6.3, we report the one-hot dimensions and the model size under the two-party setting.

Baseline Methods

Considering the objective and setting of existing works on the publication of vertically-partitioned data, we use the DPLT algorithm proposed by Tang *et al.* [142] as our baseline in the following experiments. The algorithm uses a latent tree model to represent the cross-attribute correlations in the original dataset and perturbs the tree parameters via a distributed Laplace protocol to achieve DP guarantee for *each* local dataset. Additionally, a tree index-based method TICQ can also be used to determine the minimum set of latent attribute pairs for constructing the latent tree, which helps to reduce the noise scale. The total privacy budget is consumed by three parts, namely the generation of latent attributes, quantification of latent attributes' correlations, and privatization of the tree parameters. For each dataset, we respectively compare the synthetic data utility of using the DPLT algorithm (referred to as DPLT) as well as the improved TICQ-DPLT algorithm (referred to as DPLT+). Moreover, we also present the utility of synthetic data generated under the non-private setting as a reference.

Parameter Configurations

In the following experiments, we conduct the collaborative training process for $T = 1500$ rounds. During local training, each local discriminator is updated for $E = 10$ steps with a batch size of $B = 1000$. For both the generator and discriminator, we use the RMSprop optimizer with a default learning rate of $\eta = 0.001$. Moreover, we apply the gradient perturbation when training the local discriminators, where the L_2 -clip bound C is set to 1 and the noise scale σ varies according to the target privacy budget. We choose a different privacy budget $\epsilon \in \{0.5, 1, 2, 4, 8\}$ and $\delta = 10^{-5}$ so as to explore the influence of privacy on the framework performance. The ϵ here follows the traditional DP definition (Definition 1).

Evaluation Metrics

We evaluate the performance of our VERTIGAN framework from two perspectives, namely the *utility evaluation* and the *privacy evaluation*. For the *utility evaluation*, we first compare the statistical similarity of synthetic data and real data. Then, we apply commonly used machine learning models to investigate the utility of synthetic data in AI training tasks. For the *privacy evaluation*, we investigate the capability of our framework against membership inference attacks, where an attacker aims to use the synthetic dataset to determine whether a target record is used for training theGAN model.

Table 6.4 Computation time (sec) of the proposed VERTIGAN framework and baseline DPLT+ algorithm regarding different datasets. For VERTIGAN, we perform 1500 global rounds and report the total training time.

Dataset	Web	Vehicle	Census	Twitter	HAR	Dilbert
DPLT+	1516.20	1116.81	1341.47	3954.03	19598.39	34413.66
VertiGAN	485.35	396.35	424.42	510.26	3296.08	8387.30

Computation Environments

We perform all the experiments on an NVIDIA Quadro RTX 6000 GPU. In Table 6.4, we compare the training time (sec) of our VERTIGAN framework and the baseline DPLT+ algorithm regarding all the datasets.

6.6.2 Utility: Statistical Similarity

We start our evaluation under the two-party setting, which is commonly used in existing VFL frameworks. Here, each party holds half of the attributes. We first evaluate the performance of VERTIGAN by investigating whether the generated synthetic data can preserve similar statistical properties as real data. To this end, we respectively compare the m -way joint distributions and cross-attribute correlations of the real data and synthetic data and analyze their statistical similarity.

Comparison of Joint Distributions

For the analysis of joint distributions, we used the average total variation distance (AVD) to quantify the distribution difference between the real data and synthetic data, as used in [142], which is defined as

$$AVD = \frac{1}{2} \sum_{A \subseteq \mathcal{A}} |P_{real}(A) - P_{syn}(A)|, \quad (6.9)$$

where A is one of the m -way attribute combinations in the attribute set \mathcal{A} , $P_{real}(A)$ and $P_{syn}(A)$ are joint distributions of real and synthetic data. More specifically, assume the attribute combination A has a domain size of $|\Lambda_A|$, P_{real} and P_{syn} are $|\Lambda_A|$ -dimensional vectors, where each entry is the probability of a specific value combination (namely the ratio of occurrence in the entire real or synthetic dataset). For each dataset, we randomly chose 100 m -way attribute combinations and computed the average distribution difference.

AVD Regarding the Privacy Budget ϵ In Figure 6.4, we first compare the 4-way AVD of the synthetic data generated by the VERTIGAN framework as well as the two baseline algorithms under different privacy levels. We also report the results under the fully centralized setting and the results of the proposed framework under the non-private setting as a reference. The error bars represent the 95% confidence interval (also for the remaining experimental results). It can be seen that the AVD of all the algorithms reduces with the increase of ϵ . Nonetheless, for all the datasets, the synthetic data generated by the VERTIGAN framework consistently achieve

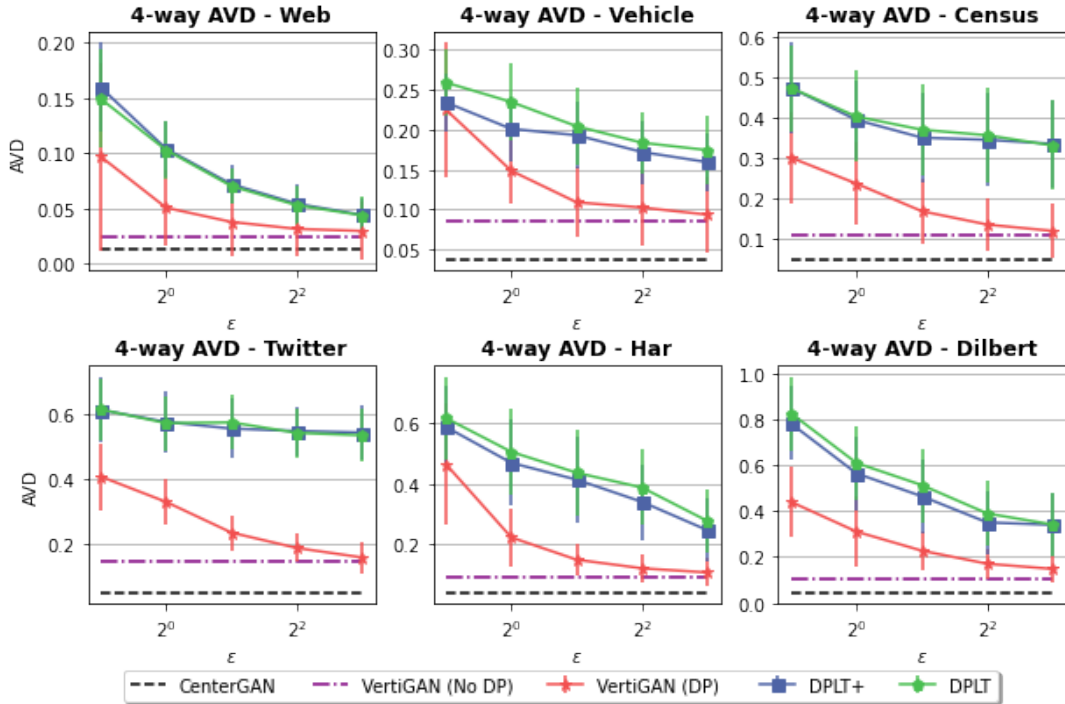


Figure 6.4 AVD of 4-way joint distributions between the real and synthetic data with respect to different privacy levels.

a smaller AVD in comparison with the baseline methods, which indicates a better capability of our VERTIGAN framework in capturing the multivariate distributions. Moreover, there is a more distinctive gap in AVD between the baseline algorithms and VERTIGAN for the datasets with a larger domain size. It can be observed that when $\epsilon \geq 4$, the AVD of the baseline algorithms is almost two to three times in comparison with VERTIGAN. This is because a larger domain size refers to more cross-attribute combinations. Since the baseline algorithms are supposed to evenly split the privacy budget to all the attribute pairs, the increase in domain size may cause each attribute pair to be allocated with an insufficient privacy budget, which may result in serious degradation of data utility. In comparison, VERTIGAN applies DP perturbation to the discriminator’s gradients and is not directly related to the domain size. Therefore, the increase in domain size does not significantly affect the utility of the synthetic data generated by VERTIGAN.

AVD Regarding the Multivariate Dimension m We further analyze the AVD with varied multivariate dimension m to gain a deeper insight into VERTIGAN’s capability in the context of complex datasets. To this end, we choose $m \in \{2, 3, 4, 5, 6\}$ and compare the m -way AVD of using VERTIGAN as well as the baseline algorithms. We present the results under $\epsilon = 2$ in Figure 6.5. Similarly, we also report the m -way AVD under the centralized setting and under the non-private VERTIGAN setting as a reference. It can be seen that for all the datasets, VERTIGAN steadily shows a smaller m -way AVD compared to the baseline algorithms. Moreover, although the baseline algorithms achieve similar AVD when m is small, the difference gets distinctively larger with an increase of m . Especially, for all the datasets, the

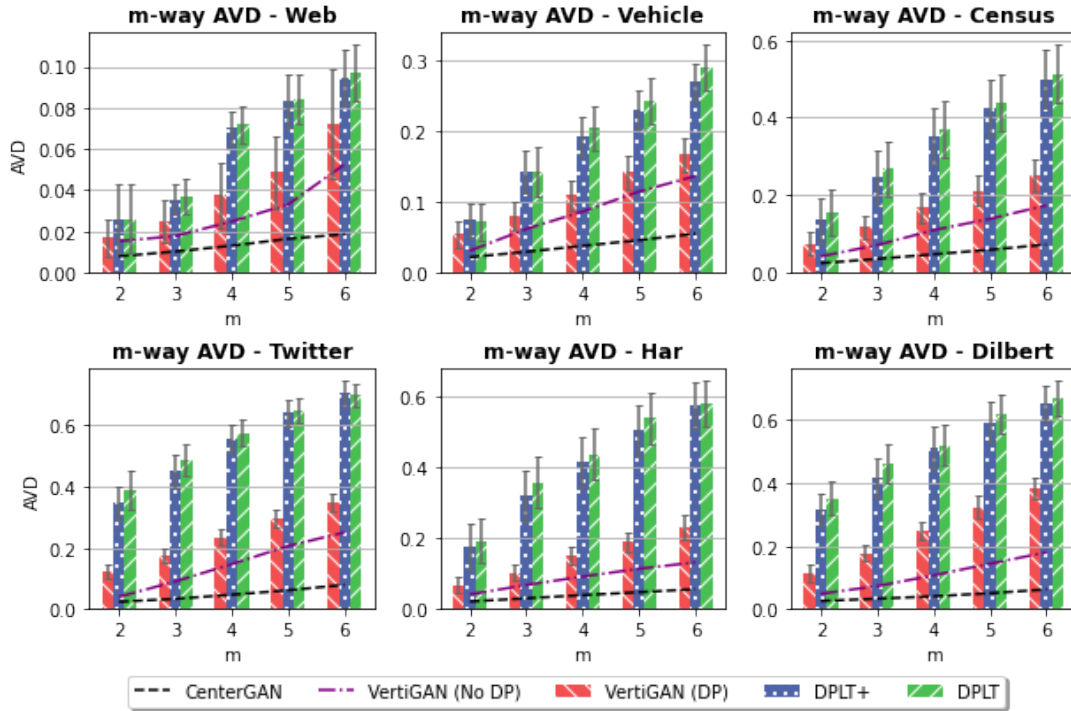


Figure 6.5 AVD of m -way joint distributions between the real and synthetic data with respect to different dimensions of the joint distribution.

5-way and 6-way AVD of the baseline algorithms are almost twice that of VERTIGAN. This indicates that our framework is more adept at capturing the information of high-dimensional joint distributions of real data.

Comparison of Correlation

We further visualize the correlation coefficient matrix of real data and synthetic data with heat maps in order to better understand the capability of our method in capturing and preserving the cross-attribute correlations. Figure 6.6 shows the comparison result of the different datasets with $\epsilon = 8$. For each dataset, we respectively select 10 attributes from each party and present the correlation matrix of the 20 attributes. From the visualization results, it can be seen that the correlation of synthetic data is similar to the correlation of real data, which further demonstrates that the synthetic data successfully preserves the attribute correlations of real data.

6.6.3 Utility: AI Training Performance

Next, we investigate the utility of synthetic data in AI training tasks. To this end, we train two classification models M_{real} and M_{syn} , respectively, with real data and synthetic data. Then, we test both models with an amount of held-out real data and compare the test accuracy,

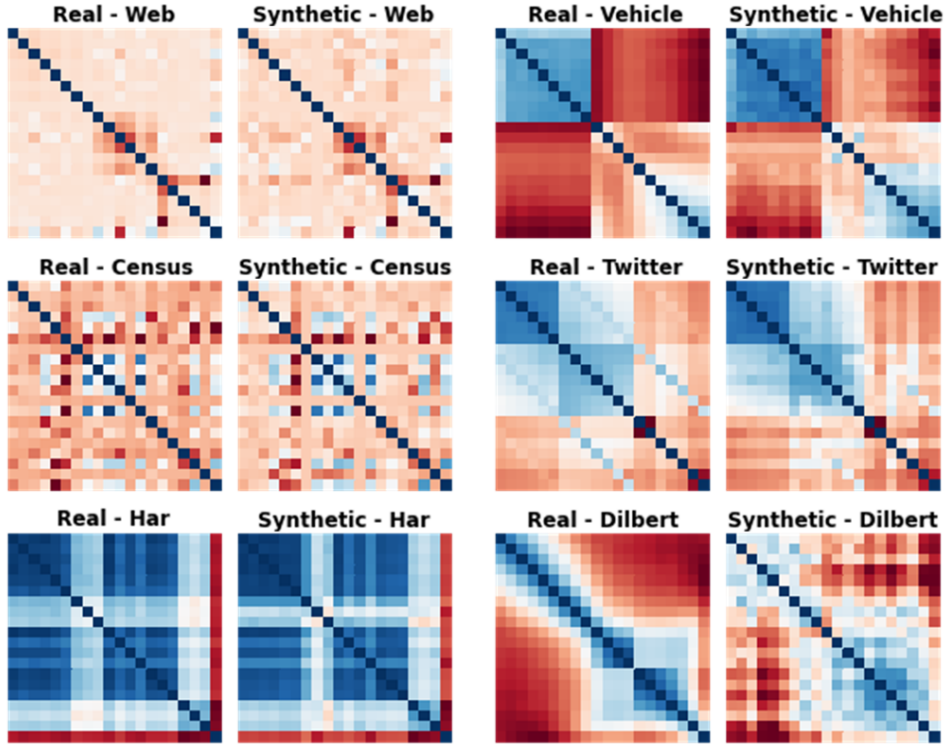


Figure 6.6 Correlation comparison between the real and synthetic data with $\epsilon = 8$. For each dataset, we present the correlations of 20 attributes, where each party contributes 10 attributes. It can be seen that the synthetic data preserves similar correlations as real data.

namely, Acc_{real} and Acc_{syn} . Intuitively, if Acc_{syn} is close to Acc_{real} , we consider the synthetic data to be of high utility which can replace real data for AI training tasks.

In the experiments, we use the Multi-layer Perceptron (MLP) classifier as the target AI model. We train both M_{real} and M_{syn} ten times and compute the averaged Acc_{real} and Acc_{syn} . In Table 6.5, we present the accuracy of the MLP classifiers evaluated on different datasets. For each dataset, we compare the Acc_{syn} of synthetic data generated under the non-private centralized and VERTIGAN setting, as well as that generated by the private DPLT and VERTIGAN frameworks with $\epsilon \in \{0.5, 2, 8\}$. It can be observed that although all the algorithms show a higher Acc_{syn} with an increase of ϵ , the accuracy of VERTIGAN is generally higher than the baselines for all privacy levels, especially for complex datasets. In particular, with $\epsilon = 8$, the synthetic data generated by VERTIGAN achieves around 2% ~ 15% improvement of Acc_{syn} compared to the baseline algorithms. The results indicate that our framework has a better capacity for preserving the hidden patterns and correlations of real data compared to the baselines. The generated synthetic data can be effectively used for data mining and AI training tasks.

Table 6.5 Classification accuracy of MLP models evaluated on synthetic data generated under different privacy settings.

Dataset	Acc. Real	Acc. Synthetic (No DP)		Acc. Synthetic (With DP)			
		Center GAN	Verti GAN	ϵ	DPLT	DPLT+	Verti GAN
Web	0.8453	0.8276	0.8079	0.5	0.7114	0.7124	0.6970
				2	0.7251	0.7238	0.7776
				8	0.7385	0.7484	0.7900
Vehicle	0.8204	0.8074	0.7984	0.5	0.7385	0.7208	0.7498
				2	0.7596	0.7373	0.7627
				8	0.7690	0.7729	0.7840
Census	0.9858	0.9820	0.9732	0.5	0.8822	0.8870	0.9088
				2	0.9027	0.9092	0.9432
				8	0.9465	0.9487	0.9655
Twitter	0.8209	0.8180	0.7871	0.5	0.7277	0.7274	0.7445
				2	0.7371	0.7397	0.7701
				8	0.7520	0.7580	0.7822
HAR	0.9532	0.8990	0.8414	0.5	0.4228	0.4570	0.5368
				2	0.5519	0.5702	0.7022
				8	0.6038	0.6284	0.7746
Dilbert	0.9394	0.8651	0.8010	0.5	0.2722	0.2988	0.5525
				2	0.4331	0.4353	0.6241
				8	0.5434	0.5672	0.7134

6.6.4 Ablation Study

We further conduct a series of ablation studies to investigate how the size of local datasets, the imbalanced splitting of attribute sets, and the increase of local parties impact the performance of the VERTIGAN framework and synthetic data utility.

Impact of the Number of Records

To start with, in the previous experiments, we assume that the local parties share data of 10^5 records. We further investigate how varying the number of local records affects the framework’s performance. To this end, we respectively vary the size of local datasets with 10^4 , 10^5 , and 10^6 records and conduct experiments under different privacy levels. In Figure 6.7, we present the 4-way AVD of the **Vehicle**, **Census**, and **HAR** datasets with $\epsilon = \{0.5, 2, 8\}$. It can be seen that using a larger number of records can significantly improve the data utility, especially in high-privacy regimes. For instance, when $\epsilon = 0.5$, for all the datasets, the AVD with 10^4 records is $2 \sim 4$ times the results with 10^5 records. This is because the privacy loss of each iteration is related to the *sampling rate* ζ , as shown in Lemma 2. Therefore, with a fixed

6 VERTIGAN for Private Sharing of Vertically-Partitioned Structured Data

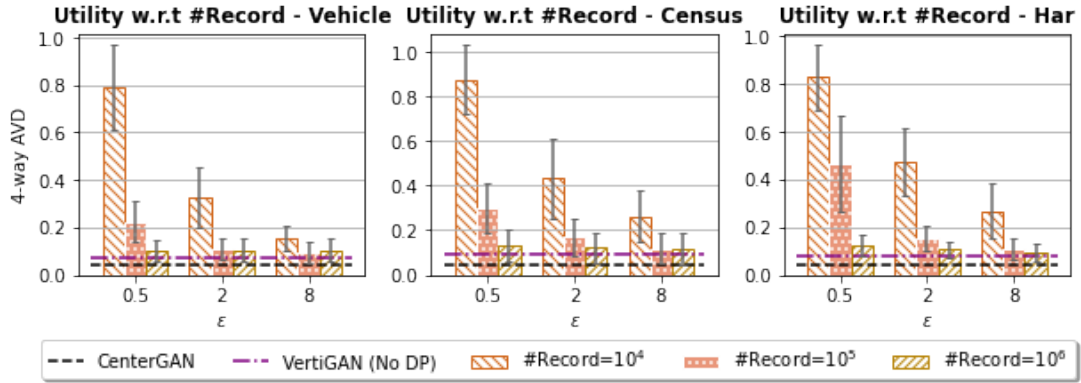


Figure 6.7 4-way AVD under the two-party settings with 10^4 , 10^5 , and 10^6 records under the privacy level $\epsilon = \{0.5, 2, 8\}$.

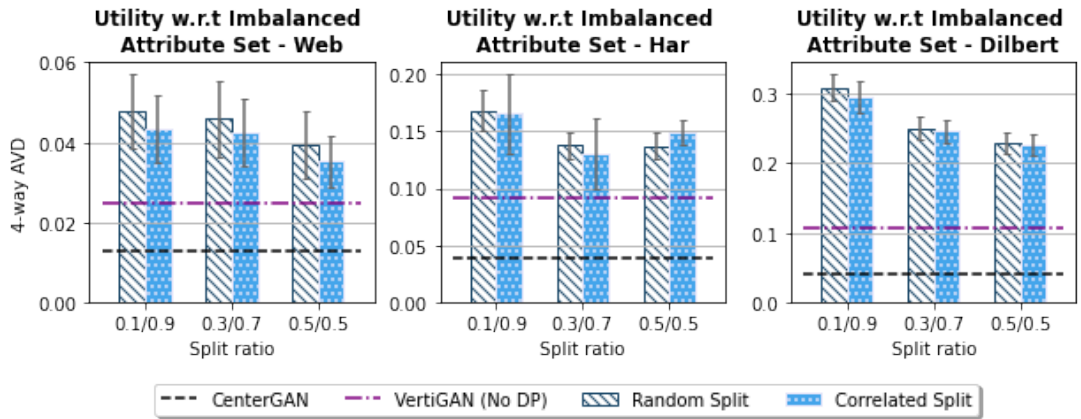


Figure 6.8 4-way AVD under the two-party settings with $\epsilon = 8$ and attribute split ratio from $\{0.1/0.9, 0.3/0.7, 0.5/0.5\}$. For each dataset, we compare the results of *random splitting* and *correlated splitting*, where the strongly correlated attributes are assigned to one of the parties.

batch size of B , increasing the total number of records leads to a decrease in privacy loss. In other words, the framework only needs to add a smaller amount of noise to achieve the same privacy level, which largely enhances the utility of synthetic data. On the other hand, for $\epsilon = 8$, the AVD with 10^6 is similar to the results with 10^5 records. This is because larger privacy budgets result in less noise being injected during training, hence the model can already converge well with 10^5 records. In this case, using larger datasets offers a comparatively smaller contribution to the utility.

Impact of Imbalanced Attribute Sets

Next, in addition to exploiting the setting where the entire attribute set is evenly split and held by two local parties, we also investigate whether the utility of the synthetic data will differ if the local parties possess an imbalanced number of attributes. To this end, we split the entire attribute set with a ratio of 0.1/0.9, 0.3/0.7, and 0.5/0.5 (i.e., an even split) and

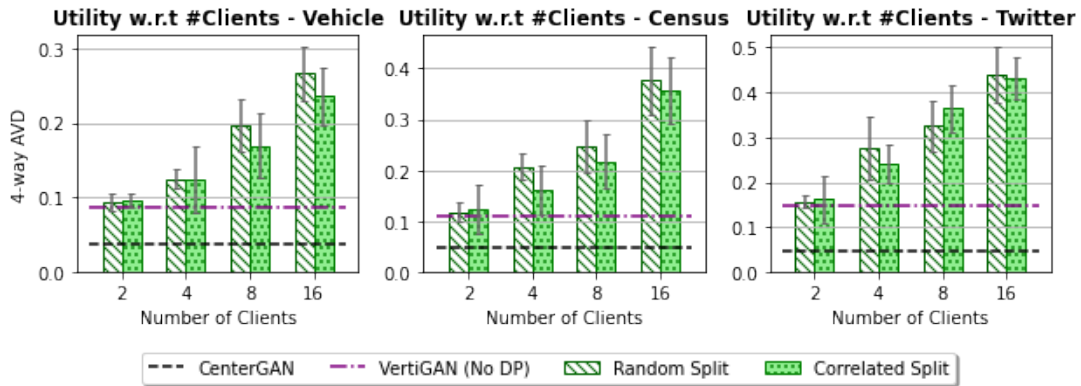


Figure 6.9 4-way AVD under the two-party settings with $\epsilon = 8$ and the number of clients from $\{2, 4, 8, 16\}$. For each dataset, we compare the results of *random splitting* and *correlated splitting*, where the strongly correlated attributes are assigned to a subset of the parties.

compare the data utility under different privacy levels. Moreover, we also explore whether the imbalanced split of strongly correlated attributes affects the data utility. To this end, we first compute the pair-wise correlation of all the attributes and apply hierarchical clustering to group the most correlated attributes. Then, we construct the imbalanced attribute sets in two ways: *random split* and *correlated split*. The former randomly splits the attribute set according to the split ratio, while the latter manually assigns the strongly correlated attributes to one of the local parties. We conduct experiments under different split ratios following both split fashions and report the results in Figure 6.8. It can be observed that an imbalanced attribute set can lead to a degradation of framework performance. In contrast, assigning the strongly correlated attributes to one of the parties slightly improves the data utility compared to the random setting. Intuitively, when the attributes belong to different generator branches, the framework may suffer from a certain information loss on the pair-wise correlations. In contrast, the correlation information can be better preserved when both attributes belong to the same branch, hence leading to higher data utility.

Impact of the Number of Local Parties

Besides the impact of imbalanced splitting, we also analyze the effects of varying the number of local parties on the framework performance. To this end, we respectively perform the data publication process using the different methods under the settings consisting of 2, 4, 8, and 16 local parties and compare the utility of synthetic data. In Figure 6.9, we present the 4-way AVD with different numbers of local parties under $\epsilon = 8$. It can be observed that the framework performance degrades with the increase of local parties. This might be because the joint distributions and correlations are more difficult to capture when the correlated attributes are spread over multiple local parties. On the other hand, similar to observations in Section 6.6.4, when assigning all the strongly-correlated attributes to a subset of local parties (*i.e.*, by using *correlated splitting*), the cross-attribute correlations can be better preserved and the data utility can be further improved.

Table 6.6 MIA accuracy under different privacy settings.

	CenterGAN		VertiGAN		
	No DP	No DP	$\epsilon = 8$	$\epsilon = 2$	$\epsilon = 0.5$
Vehicle	0.5841	0.5758	0.5538	0.5448	0.5287
Web	0.6008	0.5844	0.5633	0.5367	0.5223
Census	0.6509	0.6394	0.6171	0.5800	0.5421
Twitter	0.6746	0.6672	0.6320	0.5980	0.5676
HAR	0.5784	0.5637	0.5324	0.5241	0.5160
Dilbert	0.6044	0.5856	0.5623	0.5433	0.5386

6.6.5 Empirical Privacy Analysis

Although choosing a larger privacy budget ϵ can distinctively improve the data utility, this may lead to increased privacy leakage. In order to obtain a better understanding of the utility-privacy trade-off, we conduct a membership inference attack to empirically analyze the privacy protection capabilities of our framework under different privacy settings. We follow the black-box MIA protocol proposed in [62], which uses the distance of a target record to the synthetic dataset to infer the membership information. The intuition is that the generator tends to generate synthetic data close to the training data. Therefore, given a target record x , let $\mathcal{U}_\rho(x) = \{x' | \Gamma(x, x') \leq \rho\}$ denote the ρ -neighborhood of x with respect to the distance metric Γ . Then, we randomly generate a synthetic dataset \mathcal{X}_{syn} with n records and compute the ratio that the synthetic records fall into the neighborhood of x , namely

$$\mathcal{R}_\rho(x) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}[x_{syn}^i \in \mathcal{U}_\rho(x)], \quad (6.10)$$

where x_{syn}^i is the i^{th} synthetic record. Obviously, the higher the $\mathcal{R}_\rho(x)$, the more likely it is that x is included in the training data.

In our experiments, we construct the target dataset by randomly sampling 100 training records (denoted as \mathcal{X}_{in}) and 100 testing records (denoted as \mathcal{X}_{out}). Then, we generate a synthetic dataset \mathcal{X}_{syn} with 10^4 records and use the normalized Hamming distance to measure the minimum distance between each target record and the synthetic data. Following [62], we set ρ as the median of the minimum distance of each record. Given the ground truth label and the predicted membership probability, we compute the averaged attack accuracy under different privacy settings. The results are reported in Table 6.6. It can be observed that synthetic data generated by non-private GANs are still likely to reveal the membership information of the target record. In particular, for **Twitter** and **Census** dataset, the attack accuracy under the non-private setting is more than 65%. On the other hand, applying DP to our VERTIGAN framework can effectively reduce attack accuracy. With $\epsilon = 8$, the attack accuracy is reduced by 2% \sim 4%, while with $\epsilon = 0.5$, the attack accuracy is reduced by 5% \sim 10%. The results demonstrate that our framework is able to mitigate the risk of membership inference attacks and can provide strengthened privacy protection to the local data.

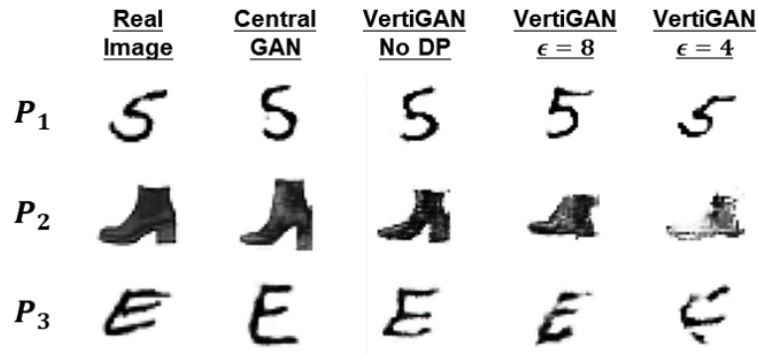


Figure 6.10 Results of image data synthesis under a three-party setting. Each row represents the synthetic images generated by one local party under different privacy settings.

6.7 Discussions and Future Work

In this section, we discuss potential extensions of our framework, current limitations, and directions for future work.

6.7.1 Extension to Other Data Types

In Section 6.6, we demonstrated that the VertiGAN framework is effective in publishing vertically-partitioned categorical datasets and achieves better data utility compared to previous statistics-based baselines. Moreover, our framework can be further extended to more complex settings where each party holds different types of data. For instance, in a healthcare scenario, a group of hospitals can use the framework to publish a joint dataset containing patients' CT images and physical symptoms for future medical research. This can be realized by modifying the structure of the models and using the advanced layers. For instance, we can respectively adopt convolution layers and recurrent layers to enhance the feature extraction on image data and time-series data. Despite the variation of the layers and model structures, the main workflow of the VertiGAN framework remains unchanged. In Figure 6.10, we further demonstrate the framework's feasibility in the context of image data. Here, we assume that there are three local parties respectively holding handwritten digits from **MNIST**, handwritten letters from **Extended MNIST**, and article images from **Fashion-MNIST**. We construct a global generator with three output branches and the corresponding local discriminators and analyze the quality of synthetic images generated under different privacy settings. Note that the synthetic data are randomly generated and hence are not identical to the real data. Nevertheless, it can still be observed that our framework is capable of jointly synthesizing all three categories of images of different local clients, and the generated data enjoys a satisfactory level of quality under a larger privacy budget.

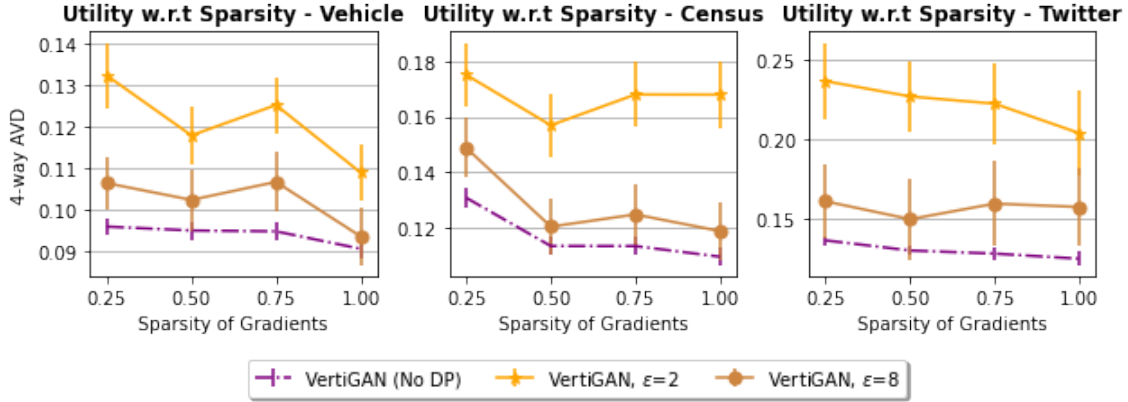


Figure 6.11 4-way AVD between the real and synthetic data with respect to different gradient sparsity ratios.

6.7.2 Reduction of Communication Cost

As described in Section 6.5.4, in each global round, the parameters and gradients of the global generator are repeatedly exchanged between the server and local parties. This may result in a high communication cost, especially for high-dimensional models. One possible approach for mitigating the upload communication cost is to process the generator’s gradients with top- k sparsification and send the sparsified gradients to the server. In Figure 6.11, we investigate how the sparsification level affects the utility of synthetic data. Here, we choose the top- k ratio from $\{0.25, 0.5, 0.75, 1\}$ and compare the corresponding 4-way AVD of the synthetic data under the privacy level of $\epsilon \in \{2, 8\}$. It can be observed that even processing the gradients with a top- k ratio of 0.25 can still achieve data utility comparable to returning the entire gradients. The results demonstrate the effectiveness of gradient sparsification in reducing the upload communication cost.

Moreover, it is also possible to further reduce the download communication cost of the framework. More specifically, before the training starts, the server broadcasts the initialized global generator to all the local parties. Then, during training, instead of broadcasting the entire global generator, the server only sends the parameters of the common layers G^0 and the corresponding branch G^i to the party \mathcal{P}^i , which is enough for \mathcal{P}^i to produce the synthetic data $\tilde{X}^i = G^i(G^0)(Z)$ on the local side (see Section 6.5.2). The improvement not only reduces the *download* communication but also prevents the local parties from inferring the inputs of the other parties.

6.7.3 Protection for the Uploaded Gradients

In this paper, we apply DP perturbation to the discriminator and enforce privacy guarantees to the generator according to the post-processing property. Nevertheless, even though the global generator is not directly trained on the local data, the gradients derived by the local discriminators may still reveal sensitive information about local data. Considering that recent studies in FL [14, 75] adopt secure multi-party computation (SMC) or local differential privacy (LDP) for encrypting or perturbing local updates, such protection techniques may also be

applicable to our framework. For instance, we can use SMC protocols to encrypt the real gradients on the local side before sending them to the server. In this way, the server cannot obtain the individual real gradients but only the sum of all the gradients after the decryption. However, the use of SMC protocols may increase the communication and computational cost of the framework due to the key generation and exchange process. On the other hand, LDP-based solutions add random noise to the local gradients, which will not largely affect efficiency. Nevertheless, it may cause significant utility loss due to the limited number of local parties under the vertical setting. Hence, how to protect the uploaded gradients regarding security, utility, and efficiency will be an important direction for future work.

6.8 Conclusion

Due to the great variety in service scenarios, user data in real-life applications are often vertically partitioned and distributed among different local parties. Although it is of great benefit for data analysts to explore the hidden correlations of attributes of all the local parties, publishing the vertically-partitioned data raises both privacy and utility concerns.

In this paper, we follow the idea of synthetic data generation and propose VERTIGAN, the first GAN-based framework for privately publishing vertically-partitioned data. Different from the prior statistics-based solutions, our framework adopts a distributed GAN architecture, where a global generator is adversarially trained with a group of local discriminators to learn the distribution of all parties' local data and used to directly generate synthetic integrated data on the server side. Moreover, we apply DP perturbation during the training process to ensure strict privacy guarantees for the local data. Experimental evaluation with real-world datasets shows that our framework significantly outperforms the statistics-based baseline algorithms for publishing high-dimensional vertically-partitioned data. The synthetic data generated by our framework preserves very similar statistical properties as real data and can replace real data for data mining and model training tasks.

Part III

Discussion and Conclusions

7 Summary of Results

In recent years, emerging privacy regulations have set strict restrictions for data analysts for directly accessing and using local user data, which has significantly changed the landscape for the development of advanced AI applications. This thesis centered around the data inaccessibility problem and seeks potential solutions for efficiently leveraging the local data without compromising individuals' privacy. The thesis first identifies a sequence of challenges encountered by both *model-to-data transmission* and *data-to-model transmission* schemes and addresses these challenges respectively through the four peer-reviewed publications embedded in Part B. This chapter presents the main results of each publication and provides a consolidated summary in Table 9.1.

The first publication (P1) focuses on addressing the challenge of improving the privacy-utility-efficiency balance in the *model-to-data transmission* scheme and proposes an improved LDP-based PPFL framework called SIGNDS-FL. The framework adopts the strategy of sign-based private dimension selection to mitigate the significant utility loss in previous LDP-FL frameworks. In addition, the work also proposes a multi-dimension selection algorithm that satisfies strict LDP guarantees and can further improve model convergence and accuracy. In comparison to the previous work under the same privacy level, the framework achieves around 4% ~ 10% improvement in accuracy on simple linear models and even 10% ~ 20% improvement on DNNs. Moreover, as each client only needs to report a few dimension indices instead of the entire model, the uplink communication cost is reduced by more than 99%. Extensive experimental results show that the SIGNDS-FL framework achieves significant improvements regarding the balance between privacy, utility, and efficiency.

Motivated by the effectiveness of the SIGNDS-FL framework proposed in P1, publication two (P2) further focuses on the *data-to-model transmission* scheme and proposes a distributed framework DP-FED-WAE to enable DL-SDG-based data sharing in cross-device scenarios. The main idea of the framework is to first train a generative WAE model under FL to learn the overall distribution of local data and then use the trained model to generate high-utility synthetic data on the server side. The adoption of the SIGNDS-FL framework not only avoids the server's direct access to the raw local data but also provides strict LDP guarantees to each data owner. The framework effectively addresses the *curse-of-dimensionality* problem in the previous LDP-based solutions and achieves significantly better utility on high-dimensional data. When using the synthetic data generated under the same privacy guarantee for training downstream AI models, the LDP-based solutions suffer from an accuracy loss of 10% ~ 30%, whereas the accuracy loss of the proposed framework is less than 3%, and at best even less than 1%. Extensive evaluation experiments show that the framework has advanced capability and efficiency in collecting high-dimensional data while striking a satisfactory utility-privacy balance.

7 Summary of Results

On the basis of P2, the third publication (P3) further explores the feasibility of applying distributed DL-SDG for sharing unstructured data in cross-device scenarios. More specifically, this work proposes a variant framework called FEDSTDG, which combines SIGNDS-FL with a recurrent generative AE for privately sharing the horizontally-partitioned time-series data held by different data owners. Moreover, this work modifies the previous SIGNDS-FL framework with an improved multi-dimension selection algorithm SUBMDS, which applies an additional dimension subsampling step to reduce the input size and achieves a distinctive improvement in the utility of the synthetic data. Additionally, the framework further adopts a novel algorithm MAGRR to adaptively adjust the learning rate during the training process and hence accelerate model convergence. Comprehensive utility evaluation results on real-world datasets show that the modified framework is capable of generating better-utility time-series data. Under the same privacy level, the synthetic data generated by using the modified framework achieves around 20% ~ 85% reduction in the downstream prediction error compared to the previous SIGNDS-FL framework. Additionally, empirical analysis using membership inference attacks and poisoning attacks further suggests the framework is capable of mitigating privacy and security risks during model training. Extensive evaluation results demonstrate that the FEDSTDG framework has superior capability and efficiency in synthesizing time-series data while achieving satisfactory privacy protection and robustness.

Finally, the fourth publication (P4) tackles challenges of the *data-to-model transmission* scheme regarding the sharing of vertically-partitioned data in cross-silo scenarios. A framework called VERTIGAN is proposed, which follows the idea of distributed DL-SDG while adopting a modified GAN architecture comprised of a single global generator and multiple local discriminators to effectively learn the joint distribution among attributes held by different data silos without the direct access to the raw local data. Moreover, it incorporated DP perturbation during the training process to provide a strict privacy guarantee to the local data. Compared to the prior statistics-based solutions, the proposed framework shows significantly better performance in capturing the joint distribution of high-dimensional data and yields around 2% ~ 15% improvement of accuracy in downstream ML tasks. Extensive evaluation results demonstrate that the framework can effectively support the private sharing of high-dimensional vertically-partitioned data with a favorable utility-privacy balance.

Table 7.1 Summary of results

No.	Title	Results
P1	SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection	<ul style="list-style-type: none"> • Proposal of SIGNDS-FL, an LDP-based PPFL framework with the idea of sign-based private dimension selection. • Proposal of an EM-based private multi-dimension selection algorithm to further improve model utility. • Evaluation on real-world data to show the capability of the framework in achieving a better balance in privacy, utility, and efficiency.
P2	Privacy-Preserving High-Dimensional Data Collection with Federated Generative Autoencoder	<ul style="list-style-type: none"> • Proposal of DP-FED-WAE, the first distributed DL-SDG-based framework for the <i>data-to-model transmission</i> of high-dimensional horizontally-partitioned structured data. • The combination of the generative AE and FL provides a promising solution for servers to learn the overall distribution of the local data without accessing the raw data, and the LDP perturbation during FL training further ensures strict privacy guarantees to the local data. • Evaluation on real-world datasets to show that the proposed framework can effectively mitigate the <i>curse-of-dimensionality</i> problem in previous LDP-based solutions and is able to achieve a satisfactory privacy-utility balance.
P3	Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning	<ul style="list-style-type: none"> • Proposal of FEDSTDG, the first distributed DL-SDG-based framework for the <i>data-to-model transmission</i> of horizontally-partitioned time-series data. • Design of a recurrent generative AE model for capturing the spatial-temporal information in the multivariate time series. • Modification of the previous SIGNDS-FL framework with an improved multi-dimension selection algorithm and an LDP adaptive learning rate adjustment algorithm which contributes to better model utility. • Evaluation on real-world datasets to show the capability of the framework in the private aggregation of time series from local silos. The generated synthetic data achieve high utility and can be applied to support downstream AI tasks.
P4	Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data	<ul style="list-style-type: none"> • Proposal of VERTIGAN, the first distributed DL-SDG-based framework for the <i>data-to-model transmission</i> of vertically-partitioned structured data. • Design of a distributed GAN architecture with a global generator and multiple local discriminators to learn the joint distribution of features from different data silos. • Design of training process with DP perturbation to ensure strict privacy guarantees to local data. • Extensive evaluation results to show the capability of the proposed framework in aggregating high-dimensional data while offering a favorable utility-privacy balance.

8 Limitations and Future Work

8.1 Inspection of Data Quality

It is well known that data quality is one of the key factors in developing effective AI products. Data with bad quality may not only render performance degradation on AI models or algorithms but also create serious security vulnerabilities (such as poisoning attacks) in the model. In the traditional centralized ML setting, the data analysts can directly aggregate the raw local data and manually perform a series of data inspection and cleaning steps on the server side to ensure the data's correctness and usefulness. However, privacy compliance issues restrict the raw local data from being directly shared with the server, making manual data inspection no longer feasible. While this thesis presents a number of solutions for data analysts to harvest insights from the local data in a distributed manner without directly aggregating these raw data, the solutions assume that the local data is benign, correct, and relevant to the target mission. To further ensure the usefulness of local data, one can conduct data inspection on the local side based on a mathematically derived *influence score* [91] or a pre-trained evaluation model [149]. Also, filtering and clipping techniques (*e.g.*, [169, 118]) that were originally proposed for defending against poisoning attacks may be utilized to limit the impact of model updates of low-quality data. **In a nutshell, how to inspect the quality of local data in a privacy-preserving manner and ensure the data have a positive impact on the models' performance is an important topic for future research.**

8.2 Optimization of Hyperparameters

In ML applications, it is universally understood that the choices of hyperparameters (*e.g.*, configurations for model structures and training algorithms) play a crucial role in the models' performance. Although techniques on hyperparameter optimization (HPO) under the centralized training setting have been extensively studied in recent years, they may encounter more challenges in the federated setting. To start with, FL includes more hyperparameters compared to the centralized setting, such as the number of participants and the update step for the global model, which results in a geometric (exponential) increase of the search space and a more complex HPO procedure. Moreover, conducting an HPO search usually requires a large number of experiments to find the hyperparameter combinations with the best model performance. This already results in a tremendous computational cost under the centralized setting and will cause even larger communication and computational expenses under the federated setting. Moreover, compared to the centralized setting, the local data under the FL setting are sometimes non-iid, which further increases the difficulty of the HPO process. Last but not least, existing HPO techniques are mostly applied in the non-private setting, yet whether the choices of hyperparameters can expose the privacy of local data remains an open question.

In summary, how to efficiently search for the optimal hyperparameters under the FL setting while preserving privacy guarantees for the local data is an important future research direction.

8.3 Performance Improvement Under High-Privacy Regimes

Although the solutions proposed in this thesis have achieved significant improvements regarding model accuracy and synthetic data utility compared to the baseline solutions, there is still a performance gap under high-privacy regimes (e.g., with a privacy budget $\epsilon \leq 1$), especially for higher-dimensional models and datasets. This is because the DP algorithms inject more random noise under the high-privacy regimes, which seriously affects model convergence. To mitigate the problem, advanced DP algorithms should be further investigated, which inject less noise during training while providing the same privacy guarantees for the local data. Moreover, searching for the optimal hyperparameters under the distributed setting, as mentioned in Section 8.2, can be another potential solution. Finally, instead of training the models from scratch, one can consider using certain auxiliary data to first pre-train the model on the server side and then fine-tune the model under the distributed setting. For instance, it is shown in Section 4.6.2 and Section 5.6.2 that pre-training the models with a small set of auxiliary data can further improve the utility of synthetic data. **In brief, how to apply different techniques to improve model accuracy and synthetic data utility under high-privacy regimes is also an essential direction for future research.**

8.4 Reduction of Download Communication Cost in SIGNDS-FL

In the original FL protocol, the local clients are responsible for conducting the local training process and returning the entire model updates to the server for updating the global model. This thesis proposed an improved SIGNDS-FL framework where local clients only need to transmit a subset of dimension indices to the server, which not only provides strict LDP guarantees to the local data but also significantly reduces the *uplink* communication cost. However, as the server still needs to broadcast the entire model weights to the local clients in each global round, the *downlink* communication may become another bottleneck as the model size increases, especially for large-scale FL scenarios. A few recent studies [19, 76, 15] have proposed solutions for reducing the size of the model broadcast to the client side. For instance, [19] proposed a dropout-based solution, which first extracts a number of small sub-models by only selecting a subset of neurons in each layer of the original high-dimensional model and then only sends the sub-models to the local side. Follow-up work in [15] further proposed an adaptive algorithm, which builds the sub-models based on an adaptive activation score map to improve the model accuracy. Besides the reduction of communication costs, training smaller models on the local side can also help decrease the computational cost on the local side. On the other hand, [76] proposed to prune the global model during the training process, which also achieves significant improvement in communication and computational efficiency. **In summary, exploring whether such techniques can be integrated into the**

current SIGNDS-FL framework to further improve communication and computational efficiency will be important future work.

8.5 Comprehensive Privacy-Utility Assessments of Synthetic Data

Finally, although SDG has been considered a promising solution for privacy-preserving ML, some recent studies also highlight that synthetic data may memorize sensitive information of training data and cause privacy breaches. The frameworks proposed in this thesis apply DP perturbation during the generative model training to provide privacy guarantees for the training data. Nevertheless, choosing a proper privacy level ϵ that achieves an optimal privacy-utility trade-off remains an open question. A number of recent works proposed different privacy assessment frameworks to evaluate the privacy risks of commonly used SDG algorithms. For instance, [64] offers an attack-based framework that measures privacy leakage of synthetic data regarding a number of membership and attribute inference attacks under different threat models. Moreover, [52] proposed to evaluate the privacy of synthetic data based on the privacy risks of *singling out*, *linkability*, and *inference* defined in [123]. **Therefore, such privacy assessment methods can also be incorporated into the current frameworks in the future to provide a comprehensive evaluation of the privacy-utility trade-off under different privacy levels and serve as a guiding tool for choosing the proper privacy levels when considering different scenarios and criteria.**

9 Conclusions

In the current digital era, data is widely recognized as business-enabling capital that helps companies and organizations to improve their operational efficiency, make informed decisions, and foster innovation and growth. By collecting and analyzing diverse and large amounts of real-world user data, data analysts are able to gain deeper insights into the target groups and create more advanced and competitive AI products. However, the growing data privacy concerns and the emerging privacy regulations have strictly restricted data analysts from directly accessing and leveraging raw local data, posing great challenges in the development of AI technologies.

Existing solutions to local data utilization problems can generally be divided into two categories, namely model-to-data transfer and data-to-model transfer. On the one hand, the model-to-data transfer solution uses distributed machine learning protocols such as FL for model training to avoid direct access to raw data. However, the previous PPFL framework failed to achieve a satisfactory balance between privacy, utility, and efficiency. On the other hand, the data-to-model transfer scheme uses state-of-the-art anonymization technology to process raw data, sharing only privatized data. However, there are still limitations in conducting private data sharing under different scenarios, different data types, and different data partition strategies.

This thesis proposed a series of solutions to address the challenges in both aforementioned schemes based on the idea of distributed ML. To start with, Chapter 3 focuses on the challenge of finding a better *privacy-utility-efficiency* balance in the model-to-data transmission scheme and introduces an improved LDP-based framework called SIGNDS-FL. The framework leverages a sign-based dimension selection strategy to effectively mitigate the large randomness incurred in previous LDP-FL frameworks and achieves a significant improvement in model utility, particularly for DNN models. The improved performance in LDP-FL fosters the generation of the distributed DL-SDG concept, which is used for addressing challenges in the *data-to-model transfer* scheme. In Chapter 4, a framework called DP-FED-WAE is introduced for addressing the *curse-of-dimensionality* challenge in aggregating horizontally-partitioned structured data. The framework trains a generative AE under the LDP-FL framework to learn the overall data distribution without direct access to the raw data and generates synthetic data with outperforming data utility compared to the prior work. Then, the concept is further applied in Chapter 5 to unstructured data, where a framework called FEDSTDG is introduced for the private aggregation of horizontally-partitioned time-series data. The framework fills the gap of privacy-preserving aggregation of unstructured data. In addition, an advanced LDP-FL framework is proposed, which adopts an improved multi-dimension selection algorithm and adaptive learning rate adjustment to further improve model performance and synthetic data utility. Finally, Chapter 6 focuses on the private aggregation of vertically-partitioned structured data and introduces a framework called VERTIGAN. By using a distributed GAN architecture comprised of a single multi-output global generator and multiple

9 Conclusions

local discriminators, the framework is capable of learning the joint distribution of attributes, and the generated synthetic data preserves utility significantly better compared to prior solutions under the same privacy guarantees.

Based on the thesis work, a number of research directions can be considered in the future. First, proper data inspection methods should be incorporated into the current frameworks to ensure the quality and correctness of the local data. Moreover, there is still room for performance enhancements in the current solutions, especially under high-privacy regimes. Potential improvement approaches include the optimization of hyperparameters, optimization of DP algorithms, and the use of pre-trained models. In addition, the *uplink* communication cost of the current FL framework can be further reduced by using dropout-based or model-pruning-based techniques. Lastly, it is necessary to build a comprehensive privacy-utility assessment module that extensively evaluates the performance of the proposed distributed DL-SDG algorithms from different perspectives. This helps improve the transparency of DL-SDG techniques and can be used as an empirical tool for determining an adequate privacy level given different scenarios and other important criteria.

Bibliography

- [1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, Vienna, Austria, 2016.
- [2] Mohammad Alaggan, Mathieu Cunche, and Sébastien Gambs. Privacy-preserving Wi-Fi analytics. *Proceedings on Privacy Enhancing Technologies*, 2018(2):4–26, 2018.
- [3] Mohammad Alaggan, Sébastien Gambs, and Anne-Marie Kermarrec. BLIP: Non-interactive differentially-private similarity computation on Bloom filters. In *14th International Symposium of Stabilization, Safety, and Security of Distributed Systems*, volume 7596, pages 202–216, Toronto, Canada, 2012.
- [4] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *31st Annual Conference on Neural Information Processing Systems 2018*, pages 5977–5987, Montreal, Canada, 2018.
- [5] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *21st European Symposium on Artificial Neural Networks*, pages 437–442, Bruges, Belgium, 2013.
- [6] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *34th International Conference on Machine Learning*, volume 70, pages 214–223, Sydney, NSW, Australia, 2017.
- [7] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. Generative models for effective ML on private, decentralized datasets. In *8th International Conference on Learning Representations*, virtual, 2020.
- [8] Ching-Yuan Bai, Hsuan-Tien Lin, Colin Raffel, and Wendy Chi-wen Kan. On training sample memorization: Lessons from benchmarking generative modeling with a large-scale competition. In *27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2534–2542, Virtual Event, Singapore, 2021.
- [9] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017.

Bibliography

- [10] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. In *30th Annual Conference on Neural Information Processing Systems*, pages 2288–2296, Long Beach, CA, USA, 2017.
- [11] Gabrielle Berman, Sara de la Rosa, and Tanya Accone. Ethical considerations when using geospatial technologies for evidence generation. *Innocenti Discussion Papers*, 2018-02, 2018.
- [12] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. SIGNSGD: Compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 559–568, Stockholm, Sweden, 2018.
- [13] Abhishek Bhowmick, John C. Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *CoRR*, abs/1812.00984, 2018.
- [14] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, Dallas, TX, USA, 2017.
- [15] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. In *2021 IEEE Conference on Computer Communications Workshops*, pages 1–6, Vancouver, BC, Canada, 2021.
- [16] Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. In *37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 435–447, New York, NY, USA, 2018.
- [17] Kuntai Cai, Xiaoyu Lei, Jianxin Wei, and Xiaokui Xiao. Data synthesis via differentially private Markov random fields. *Proceedings of the VLDB Endowment*, 14(11):2190–2202, 2021.
- [18] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. “You might also like:” Privacy risks of collaborative filtering. In *32nd IEEE Symposium on Security and Privacy*, pages 231–246, Berkeley, CA, USA, 2011.
- [19] Sebastian Caldas, Jakub Konečný, H. Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *CoRR*, abs/1812.07210, 2018.
- [20] Luis M Candanedo, Véronique Feldheim, and Dominique Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81–97, 2017.
- [21] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, pages 267–284, Santa Clara, CA, USA, 2019.

- [22] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *22nd Annual Conference on Neural Information Processing Systems*, pages 289–296, Vancouver, British Columbia, Canada, 2008.
- [23] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. GS-WGAN: A gradient-sanitized approach for learning differentially private generators. In *34th International Conference on Neural Information Processing Systems*, pages 12673–12684, virtual, 2020.
- [24] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. GAN-Leaks: A taxonomy of membership inference attacks against generative models. In *2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 343–362, Virtual Event, USA, 2020. ACM.
- [25] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. Differentially private high-dimensional data publication via sampling-based inference. In *21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 129–138, New York, NY, USA, 2015. ACM.
- [26] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. VAFL: A method of vertical asynchronous federated learning. *CoRR*, abs/2007.06081, 2020.
- [27] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.
- [28] Yen-Chi Cheng, Hsin-Ying Lee, Min Sun, and Ming-Hsuan Yang. Controllable image synthesis via SegVAE. In *16th European Conference on Computer Vision (ECCV), Proceedings Part VII*, volume 12352 of *Lecture Notes in Computer Science*, pages 159–174, Glasgow, UK, 2020.
- [29] Michele Ciampi and Claudio Orlandi. Combining private set-intersection with secure two-party computation. In *11th International Conference on Security and Cryptography for Networks*, volume 11035, pages 464–482, Amalfi, Italy, 2018.
- [30] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: An extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017.
- [31] Andrew A. Cook, Goksel Misirli, and Zhong Fan. Anomaly detection for IoT time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2020.
- [32] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier’s probabilistic public-key system. In *4th International Workshop on Practice and Theory in Public Key Cryptography*, volume 1992, pages 119–136, Cheju Island, Korea, 2001.
- [33] Saverio De Vito, Ettore Massera, Marco Piga, Luca Martinotto, and Girolamo Di Francia. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, 129(2):750–757, 2008.

Bibliography

- [34] Sumit Kumar Debnath and Ratna Dutta. Secure and efficient private set intersection cardinality using Bloom filter. In *18th Information Security Conference*, volume 9290, pages 209–226, Trondheim, Norway, 2015.
- [35] Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(8), 2017.
- [36] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *30th Annual Conference on Neural Information Processing Systems*, pages 3571–3580, Long Beach, CA, USA, 2017.
- [37] Nikoli Dryden, Tim Moon, Sam Ade Jacobs, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In *2nd Workshop on Machine Learning in HPC Environments, MLHPC@SC*, pages 1–8, Salt Lake City, UT, USA, 2016.
- [38] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. <http://archive.ics.uci.edu/ml>.
- [39] Marco F. Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.
- [40] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.
- [41] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *41st Annual ACM Symposium on Theory of Computing*, pages 381–390, Bethesda, MD, USA, 2009.
- [42] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.
- [43] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *CoRR*, abs/2001.03618, 2020.
- [44] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067, Scottsdale, AZ, USA, 2014.
- [45] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. Building a Rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies*, 3:1–21, 2016.
- [46] Yahoo! Finance. Alphabet inc. (goog) stock historical prices and data. <https://finance.yahoo.com/quote/GOOG/history/>, 2004.

- [47] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *34th International Conference on ICT Systems Security and Privacy Protection*, volume 562, pages 151–164, Lisbon, Portugal, 2019.
- [48] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X. Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *31st USENIX Security Symposium*, pages 1397–1414, Boston, MA, 2022.
- [49] Sébastien Gambs, Frédéric Ladouceur, Antoine Laurent, and Alexandre Roy-Gaumond. Growing synthetic data through differentially-private vine copulas. *Proceedings on Privacy Enhancing Technologies*, 2021(3):122–141, 2021.
- [50] Gartner, Inc. Gartner identifies top five trends in privacy through 2024.
- [51] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients – How easy is it to break privacy in federated learning? In *33rd Annual Conference on Neural Information Processing Systems*, pages 16937–16947, virtual, 2020.
- [52] Matteo Giomi, Franziska Boenisch, Christoph Wehmeyer, and Borbála Tasnádi. A unified framework for quantifying privacy risk in synthetic data. *Proceedings on Privacy Enhancing Technologies*, 2:0–0, 2023.
- [53] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *27th Annual Conference on Neural Information Processing Systems*, pages 2672–2680, Montreal, Quebec, Canada, 2014.
- [54] Mehmet Emre Gursoy, Acar Tamersoy, Stacey Truex, Wenqi Wei, and Ling Liu. Secure and utility-aware data collection with condensed local differential privacy. *IEEE Transactions on Dependable and Secure Computing*, 2019.
- [55] Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, Alexander R. Statnikov, Wei-Wei Tu, and Evelyne Viegas. Analysis of the AutoML challenge series 2015-2018. In *Automated Machine Learning - Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 177–219. Springer, Berlin, Germany, 2019.
- [56] Ikhsanul Habibie, Daniel Holden, Jonathan Schwarz, Joe Yearsley, and Taku Komura. A recurrent variational autoencoder for human motion synthesis. In *British Machine Vision Conference 2017*, London, UK, 2017.
- [57] Meng Hao, Hongwei Li, Xizhao Luo, Guowen Xu, Haomiao Yang, and Sen Liu. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics*, 16(10):6532–6542, 2020.
- [58] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data

Bibliography

- via entity resolution and additively homomorphic encryption. *CoRR*, abs/1711.10677, 2017.
- [59] Florian Hartmann, Sunah Suh, Arkadiusz Komarzewski, Tim D. Smith, and Ilana Segall. Federated learning for ranking browser history suggestions. *CoRR*, abs/1911.11807, 2019.
- [60] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133–152, 2019.
- [61] Markus Herdin, Nicolai Czink, Hüseyin Özcelik, and Ernst Bonek. Correlation matrix distance, a meaningful measure for evaluation of non-stationary MIMO channels. In *IEEE 61st Vehicular Technology Conference*, volume 1, pages 136–140, Stockholm, Sweden, 2005.
- [62] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. Monte Carlo and reconstruction membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(4):232–249, 2019.
- [63] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [64] Florimond Houssiau, James Jordon, Samuel N Cohen, Owen Daniel, Andrew Elliott, James Geddes, Callum Mole, Camila Rangel-Smith, and Lukasz Szpruch. Tapas: A toolbox for adversarial privacy auditing of synthetic data. In *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*, 2022.
- [65] Huaibo Huang, Zhihang Li, Ran He, Zhenan Sun, and Tieniu Tan. Introvae: Introspective variational autoencoders for photographic image synthesis. In *31st Annual Conference on Neural Information Processing Systems*, pages 52–63, Montréal, Canada, 2018.
- [66] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *19th Annual Network and Distributed System Security Symposium*, San Diego, California, USA, 2012.
- [67] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.
- [68] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-efficient distributed SGD with sketching. In *32nd Annual Conference on Neural Information Processing Systems*, pages 13144–13154, Vancouver, BC, Canada, 2019.
- [69] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: Privacy-preserving empirical risk minimization. In *32nd Annual Conference on Neural Information Processing Systems*, pages 6343–6354, Montréal, Canada, 2018.

- [70] Jinsung Jeon, Jeonghak Kim, Haryong Song, Seunghyeon Cho, and Noseong Park. GT-GAN: General purpose time series synthesis with generative adversarial networks. In *36th Annual Conference on Neural Information Processing Systems*, 2022.
- [71] Malhar S Jere, Tyler Farnan, and Farinaz Koushanfar. A taxonomy of attacks on federated learning. *IEEE Security & Privacy*, 19(2):20–28, 2020.
- [72] Wei Jiang and Chris Clifton. A secure distributed framework for achieving k-anonymity. *The International Journal on Very Large Data Bases*, 15(4):316–333, 2006.
- [73] Xue Jiang, Xuebing Zhou, and Jens Grossklags. Comprehensive analysis of privacy leakage in vertical federated learning during prediction. *Proceedings on Privacy Enhancing Technologies*, 2022(2):263–281, 2022.
- [74] Xue Jiang, Xuebing Zhou, and Jens Grossklags. Privacy-preserving high-dimensional data collection with federated generative autoencoder. *Proceedings on Privacy Enhancing Technologies*, 2022(1):481–500, 2022.
- [75] Xue Jiang, Xuebing Zhou, and Jens Grossklags. SignDS-FL: Local differentially private federated learning with sign-based dimension selection. *ACM Transactions on Intelligent Systems and Technology*, 13(5):1–22, 2022.
- [76] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassioulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2022.
- [77] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019.
- [78] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete Gaussian mechanism for federated learning with secure aggregation. In *38th International Conference on Machine Learning*, volume 139, pages 5201–5212, Virtual Event, 2021. PMLR.
- [79] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D’Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrede Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1-2):1–210, 2021.

Bibliography

- [80] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.
- [81] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, Long Beach, CA, USA, 2019.
- [82] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.
- [83] François Kawala, Ahlame Douzal-Chouakria, Eric Gaussier, and Eustache Dimert. Prédiction d'activité dans les réseaux sociaux en ligne. In *4ième Conférence sur les Modèles et l'Analyse des Réseaux: Approches Mathématiques et Informatiques*, France, 2013.
- [84] Latif U. Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 23(3):1759–1799, 2021.
- [85] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations*, Banff, AB, Canada, 2014.
- [86] Ron Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 202–207, Portland, Oregon, USA, 1996. AAAI Press.
- [87] Florian Kohlmayer, Fabian Prasser, Claudia Eckert, and Klaus A Kuhn. A flexible approach to distributed data anonymization. *Journal of Biomedical Informatics*, 50:62–76, 2014.
- [88] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.
- [89] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.
- [90] Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 97–104, Washington, DC, USA, 2004.
- [91] Anran Li, Lan Zhang, Junhao Wang, Feng Han, and Xiang-Yang Li. Privacy-preserving efficient federated-learning model debugging. *IEEE Transactions on Parallel and Distributed Systems*, 33(10):2291–2303, 2022.
- [92] Haoran Li, Li Xiong, and Xiaoqian Jiang. Differentially private synthesization of multi-dimensional data using copula functions. In *17th International Conference on Extending Database Technology*, pages 475–486, Athens, Greece, 2014.

- [93] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-Closeness: Privacy beyond k-anonymity and l-diversity. In *IEEE 23rd International Conference on Data Engineering*, pages 106–115, 2007.
- [94] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. In *Tenth International Conference on Learning Representations*, Virtual Event, 2022.
- [95] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.
- [96] Wenqi Li, Fausto Milletari, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M. Jorge Cardoso, and Andrew Feng. Privacy-preserving federated brain tumour segmentation. In *10th International Workshop on Machine Learning in Medical Imaging*, volume 11861 of *Lecture Notes in Computer Science*, pages 133–141, 2019.
- [97] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020.
- [98] Claire Little, Mark Elliot, and Richard Allmendinger. Federated learning for generating synthetic data: A scoping review. *International Journal of Population Data Science*, 8(1), 2023.
- [99] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. FedSel: Federated SGD under local differential privacy with top-k dimension selection. In *25th International Conference on Database Systems for Advanced Applications*, volume 12112, pages 485–501, Jeju, South Korea, 2020.
- [100] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. Fedvision: An online visual object detection platform powered by federated learning. In *The 34th AAAI Conference on Artificial Intelligence*, pages 13172–13179, 2020.
- [101] Yang Liu, Zhihao Yi, and Tianjian Chen. Backdoor attacks and defenses in feature-partitioned collaborative learning. *CoRR*, abs/2007.03608, 2020.
- [102] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Feature inference attack on model predictions in vertical federated learning. In *37th IEEE International Conference on Data Engineering*, pages 181–192, Chania, Greece, 2021.
- [103] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and S Yu Philip. Privacy and robustness in federated learning: Attacks and defenses. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [104] Jing Ma, Si-Ahmed Naas, Stephan Sigg, and Xixiang Lyu. Privacy-preserving federated learning based on multi-key homomorphic encryption. *International Journal of Intelligent Systems*, 37(9):5880–5901, 2022.

Bibliography

- [105] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkatasubramanian. I-Diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1):3–es, 2007.
- [106] Stuart Madnick. The continued threat to personal data: Key factors behind the 2023 increase. Technical report, Massachusetts Institute of Technology, 12 2023.
- [107] Sourav Majumdar and Arnab Kumar Laha. Clustering and classification of time series using topological data analysis with applications to finance. *Expert Systems with Applications*, 162:113868, 2020.
- [108] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 1273–1282, Fort Lauderdale, USA, 2017.
- [109] H. Brendan McMahan and Galen Andrew. A general approach to adding differential privacy to iterative training procedures. *CoRR*, abs/1812.06210, 2018.
- [110] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.
- [111] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103, Providence, RI, USA, 2007.
- [112] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy*, pages 691–706, 2019.
- [113] Ilya Mironov. Rényi differential privacy. In *30th IEEE Computer Security Foundations Symposium*, pages 263–275, Santa Barbara, CA, USA, 2017.
- [114] Noman Mohammed, Dima Alhadidi, Benjamin CM Fung, and Mourad Debbabi. Secure two-party differentially private data release for vertically partitioned data. *IEEE Transactions on Dependable and Secure Computing*, 11(1):59–71, 2013.
- [115] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy*, pages 739–753, San Francisco, CA, USA, 2019.
- [116] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.
- [117] Dinh C. Nguyen, Quoc-Viet Pham, Pubudu N. Pathirana, Ming Ding, Aruna Seneviratne, Zihuai Lin, Octavia A. Dobre, and Won-Joo Hwang. Federated learning for smart healthcare: A survey. *ACM Computing Surveys*, 55(3):60:1–60:37, 2023.

- [118] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. FLAME: Taming backdoors in federated learning. In *31st USENIX Security Symposium*, pages 1415–1432, Boston, MA, USA, 2022.
- [119] Thông T. Nguyễn, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. Collecting and analyzing data from smart device users with local differential privacy. *CoRR*, abs/1606.05053, 2016.
- [120] Solmaz Niknam, Harpreet S Dhillon, and Jeffrey H Reed. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58(6):46–51, 2020.
- [121] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations*, Toulon, France, 2017.
- [122] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, 2018.
- [123] Article 29 Data Protection Working Party. Opinion 05/2014 on anonymization techniques. Technical report, EU, 2014.
- [124] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2017.
- [125] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft, 1998.
- [126] John C. Platt. *Fast training of support vector machines using sequential minimal optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.
- [127] Zhaozhi Qian, Bogdan-Constantin Cebere, and Mihaela van der Schaar. Synthcity: Facilitating innovative use cases of synthetic data in different data modalities. *CoRR*, abs/2301.07573, 2023.
- [128] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 192–203, Vienna, Austria, 2016.
- [129] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.
- [130] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A. McCann, and S. Yu Philip. LoPub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166, 2018.

Bibliography

- [131] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *NPJ Digital Medicine*, 3(1):119, 2020.
- [132] Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robert Sandmann, Robin Roehm, and Michael A. Hoeh. PyVertical: A vertical federated learning framework for multi-headed splitNN. *CoRR*, abs/2104.00489, 2021.
- [133] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. Technical report, University of California, San Diego; Institute for Cognitive Science, 1985.
- [134] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *15th Annual Conference of the International Speech Communication Association*, pages 1058–1062, Singapore, 2014.
- [135] Shreya Sharma, Chaoping Xing, Yang Liu, and Yan Kang. Secure and efficient federated transfer learning. In *2019 IEEE International Conference on Big Data*, pages 2569–2576, Los Angeles, CA, USA, 2019. IEEE.
- [136] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *2015 ACM SIGSAC Conference on Computer and Communications Security*, pages 1310–1321, 2015.
- [137] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy*, pages 3–18, San Jose, CA, USA, 2017.
- [138] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. Synthetic data – Anonymisation groundhog day. In *31st USENIX Security Symposium*, pages 1451–1468, Boston, MA, USA, 2022.
- [139] Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *31st Annual Conference on Neural Information Processing Systems*, pages 4452–4463, Montreal, Canada, 2018.
- [140] Yan Sun, Lihua Yin, Licai Liu, and Shuang Xin. Toward inference attacks for k-anonymity. *Personal and Ubiquitous Computing*, 18(8):1871–1880, 2014.
- [141] Latanya Sweeney. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.
- [142] Peng Tang, Xiang Cheng, Sen Su, Rui Chen, and Huaxi Shao. Differentially private publication of vertically partitioned data. *IEEE Transactions on Dependable and Secure Computing*, 18(2):780–795, 2019.
- [143] Petroc Taylor. Total data volume worldwide 2010-2025, Sep 2022.

- [144] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. In *6th International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.
- [145] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. DP-CGAN: Differentially private synthetic data and label generation. In *IEEE Conference on Computer Vision and Pattern Recognition – Workshops*, pages 98–104, Long Beach, CA, USA, 2019.
- [146] Aleksei Triastcyn and Boi Faltings. Federated generative privacy. *IEEE Intelligent Systems*, 35(4):50–57, 2020.
- [147] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11, London, UK, 2019.
- [148] Stacey Truex, Ling Liu, Ka Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. LDP-Fed: Federated learning with local differential privacy. In *3rd International Workshop on Edge Systems, Analytics and Networking, EdgeSys@EuroSys 2020*, pages 61–66, Heraklion, Greece, 2020.
- [149] Tiffany Tuor, Shiqiang Wang, Bong Jun Ko, Changchang Liu, and Kin K. Leung. Overcoming noisy and irrelevant data in federated learning. In *25th International Conference on Pattern Recognition*, pages 5020–5027, Virtual Event / Milan, Italy, 2020.
- [150] Jaideep Vaidya and Chris Clifton. Privacy preserving naïve Bayes classifier for vertically partitioned data. In *Fourth SIAM International Conference on Data Mining*, pages 522–526, Lake Buena Vista, Florida, USA, 2004.
- [151] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and Scott Patterson. Privacy-preserving decision trees over vertically partitioned data. *ACM Transactions on Knowledge Discovery from Data*, 2(3):1–27, 2008.
- [152] Gerrit van den Burg and Chris Williams. On memorization in probabilistic deep generative models. In *34th Annual Conference on Neural Information Processing Systems*, pages 27916–27928, Virtual, 2021.
- [153] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.
- [154] Tim van Erven and Peter Harremoës. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.
- [155] Curtis R. Vogel. *Computational methods for inverse problems*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 2002.
- [156] Boxin Wang, Fan Wu, Yunhui Long, Luka Rimanic, Ce Zhang, and Bo Li. Datalens: Scalable privacy preserving training via gradient compression and aggregation. In *2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2146–2168, Virtual Event, Republic of Korea, 2021.

Bibliography

- [157] Chang Wang, Jian Liang, Mingkai Huang, Bing Bai, Kun Bai, and Hao Li. Hybrid differentially private federated learning on vertically partitioned data. *CoRR*, abs/2009.02763, 2020.
- [158] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *35th IEEE International Conference on Data Engineering*, pages 638–649, Macao, China, 2019.
- [159] Shaowei Wang, Liusheng Huang, Yiwen Nie, Pengzhan Wang, Hongli Xu, and Wei Yang. Privset: Set-valued data analyses with locale differential privacy. In *2018 IEEE Conference on Computer Communications*, pages 1088–1096, Honolulu, HI, USA, 2018.
- [160] Teng Wang, Xinyu Yang, Xuebin Ren, Wei Yu, and Shusen Yang. Locally private high-dimensional crowdsourced data release based on copula functions. *IEEE Transactions on Services Computing*, 15(2):778–792, 2019.
- [161] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private frequent item-set mining. In *2018 IEEE Symposium on Security and Privacy*, pages 127–143, San Francisco, California, USA, 2018.
- [162] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled Renyi differential privacy and analytical moments accountant. In *22nd International Conference on Artificial Intelligence and Statistics*, volume 89, pages 1226–1235, Naha, Okinawa, Japan, 2019.
- [163] Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.
- [164] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. Improving the improved training of Wasserstein GANs: A consistency term and its dual effect. In *6th International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.
- [165] Wikipedia. List of data breaches, Jan 2023.
- [166] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *33rd International Conference on Very Large Data Bases*, pages 543–554, Vienna, Austria, 2007.
- [167] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. Privacy preserving vertical federated learning for tree-based models. *Proceeding of VLDB Endowment*, 13(11):2090–2103, 2020.
- [168] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.
- [169] Chulin Xie, Minghao Chen, Pin-Yu Chen, and Bo Li. CRFL: Certifiably robust federated learning against backdoor attacks. In *38th International Conference on Machine Learning*, volume 139, pages 11372–11382, Virtual Event, 2021.

- [170] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *CoRR*, abs/1802.06739, 2018.
- [171] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In *32nd Annual Conference on Neural Information Processing Systems*, pages 7333–7343, Vancouver, BC, Canada, 2019.
- [172] Dong Yang, Ziyue Xu, Wenqi Li, Andriy Myronenko, Holger R. Roth, Stephanie A. Harmon, Sheng Xu, Baris Turkbey, Evrim Turkbey, Xiaosong Wang, Wentao Zhu, Gianpaolo Carratiello, Francesca Patella, Maurizio Cariati, Hirofumi Obinata, Hitoshi Mori, Kaku Tamura, Peng An, Bradford J. Wood, and Daguang Xu. Federated semi-supervised learning for COVID region segmentation in chest CT using multi-national data from China, Italy, Japan. *Medical Image Analysis*, 70:101992, 2021.
- [173] Kai Yang, Tao Fan, Tianjian Chen, Yuanming Shi, and Qiang Yang. A quasi-Newton method based vertical federated learning framework for logistic regression. *CoRR*, abs/1912.00513, 2019.
- [174] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):12:1–12:19, 2019.
- [175] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *CoRR*, abs/1812.02903, 2018.
- [176] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, USA, 1982.
- [177] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *32th Annual Conference on Neural Information Processing Systems*, pages 5509–5519, Vancouver, BC, Canada, 2019.
- [178] Abdelhafid Zeroual, Fouzi Harrou, Abdelkader Dairi, and Ying Sun. Deep learning methods for forecasting COVID-19 time-series data: A comparative study. *Chaos, Solitons & Fractals*, 140:110121, 2020.
- [179] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. PrivBayes: Private data release via Bayesian networks. *ACM Transactions on Database Systems*, 42(4):25:1–25:41, 2017.
- [180] Longling Zhang, Bochen Shen, Ahmed Barnawi, Shan Xi, Neeraj Kumar, and Yi Wu. FedDPGAN: Federated differentially private generative adversarial networks framework for the detection of COVID-19 pneumonia. *Information Systems Frontiers*, 23(6):1403–1415, 2021.
- [181] Nevin L Zhang. Hierarchical latent class models for cluster analysis. *The Journal of Machine Learning Research*, 5:697–723, 2004.

Bibliography

- [182] Rui Zhang, Song Guo, Junxiao Wang, Xin Xie, and Dacheng Tao. A survey on gradient inversion: Attacks, defenses and future directions. In *32st International Joint Conference on Artificial Intelligence*, pages 5678–5685, Vienna, Austria, 2022.
- [183] Xinyang Zhang, Shouling Ji, and Ting Wang. Differentially private releasing via deep generative model. *CoRR*, abs/1801.01594, 2018.
- [184] Yang Zhao, Jun Zhao, Mengmeng Yang, Teng Wang, Ning Wang, Lingjuan Lyu, Dusit Niyato, and Kwok-Yan Lam. Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal*, 8(11):8836–8853, 2021.
- [185] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. Property inference attacks against GANs. In *29th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, 2022.
- [186] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, pages 14747–14756, 2019.
- [187] Tianyuan Zou, Yang Liu, Yan Kang, Wenhan Liu, Yuanqin He, Zhihao Yi, Qiang Yang, and Ya-Qin Zhang. Defending batch-level label inference and replacement attacks in vertical federated learning. *IEEE Transactions on Big Data*, 1:1–12, 2022.

Appendix: Original Version of Publications

This Appendix includes the original versions of the peer-reviewed core publications that this cumulative dissertation is based on. The works in Chapter 3, Chapter 4, and Chapter 6 have revised layouts compared to the original publications included in this Appendix. Details of the papers are summarized in order of appearance:

- **Xue Jiang**, Xuebing Zhou, and Jens Grossklags (2022) SignDS-FL: Local Differentially Private Federated Learning with Sign-Based Dimension Selection. *ACM Transactions on Intelligent Systems and Technology*, 13(5): 1–22.
- **Xue Jiang**, Xuebing Zhou, and Jens Grossklags (2022) Privacy-Preserving High-Dimensional Data Collection with Federated Generative Autoencoder. *Proceedings on Privacy Enhancing Technologies*, 2022(1): 481–500.
- **Xue Jiang**, Yufei Zhang, Xuebing Zhou, and Jens Grossklags (2023) Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data. *Proceedings on Privacy Enhancing Technologies*, 2023(2): 236–250.

Each published paper is introduced with an overview page, and followed by the relevant Creative Commons deed.

1 SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection

Authors	Xue Jiang ^{1,2} (xue.jiang@tum.de) Xuebing Zhou ² (xuebing.zhou@huawei.com) Jens Grossklags ¹ (jens.grossklags@in.tum.de)
	¹ Technical University of Munich (TUM), Boltzmannstraße 3, 85748 Garching, Germany ² Huawei Munich Research Center, Riesstraße 25, 80992 Munich, Germany
Type	Journal
Outlet	ACM Transactions on Intelligent Systems and Technology ³
Ranking	Impact Factor in 2022 ⁴ : 10.489 Rank in the field of Information Systems ⁴ : 36/163 Rank in the field of Artificial Intelligence ⁴ : 13/145
Status	Published
DOI	https://doi.org/10.1145/3517820
Citation	Jiang, X., Zhou, X., & Grossklags, J. (2022). SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection. ACM Transactions on Intelligent Systems and Technology (TIST), 13(5), 1-22.
Copyright	This work is published under a Creative Commons Attribution-NonCommercial International 4.0 (CC BY-NC 4.0) License ⁵ .
Author Contributions	Xue Jiang developed the core idea for the paper and took primary responsibility for designing the methodology, acquiring data, implementing experiments, evaluating results, and drafting the manuscript. Xuebing Zhou and Jens Grossklags provided valuable feedback and suggestions on the methodology and experiments, and assisted in reviewing and enhancing the manuscript.

³<https://dl.acm.org/journal/tist>

⁴<https://dl.acm.org/journal/tist/indexing>

⁵<https://creativecommons.org/licenses/by-nc/4.0/>



SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection

XUE JIANG, Technical University of Munich, Germany

XUEBING ZHOU, Huawei Technologies Düsseldorf GmbH, Germany

JENS GROSSKLAGS, Technical University of Munich, Germany

Federated Learning (FL) [31] is a decentralized learning mechanism that has attracted increasing attention due to its achievements in computational efficiency and privacy preservation. However, recent research highlights that the original FL framework may still reveal sensitive information of clients' local data from the exchanged local updates and the global model parameters. Local Differential Privacy (LDP), as a rigorous definition of privacy, has been applied to Federated Learning to provide formal privacy guarantees and prevent potential privacy leakage. However, previous LDP-FL solutions suffer from considerable utility loss with an increase of model dimensionality. Recent work [29] proposed a two-stage framework that mitigates the dimension-dependency problem by first selecting one "important" dimension for each local update and then perturbing the dimension value to construct the sparse privatized update. However, the framework may still suffer from utility loss because of the insufficient per-stage privacy budget and slow model convergence.

In this article, we propose an improved framework, SIGNDS-FL, which shares the concept of dimension selection with Reference [29], but saves the privacy cost for the value perturbation stage by assigning random sign values to the selected dimensions. Besides using the single-dimension selection algorithms in Reference [29], we propose an Exponential Mechanism-based Multi-Dimension Selection algorithm that further improves model convergence and accuracy. We evaluate the framework on a number of real-world datasets with both simple logistic regression models and deep neural networks. For training logistic regression models on structured datasets, our framework yields only a $\sim 1\%$ – 2% accuracy loss in comparison to a $\sim 5\%$ – 15% decrease of accuracy for the baseline methods. For training deep neural networks on image datasets, the accuracy loss of our framework is less than 8% and at best only 2%. Extensive experimental results show that our framework significantly outperforms the previous LDP-FL solutions and enjoys an advanced utility-privacy balance.

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**; • **Computing methodologies** → **Machine learning**;

Additional Key Words and Phrases: Federated learning, local differential privacy

ACM Reference format:

Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection. *ACM Trans. Intell. Syst. Technol.* 13, 5, Article 74 (June 2022), 22 pages.

<https://doi.org/10.1145/3517820>

Authors' addresses: X. Jiang and J. Grossklags, Technical University of Munich, Boltzmannstraße 3, Garching, Germany, 85748; emails: xue.jiang@tum.de, jens.grossklags@in.tum.de; X. Zhou, Huawei Technologies Düsseldorf GmbH, Riesstraße 25, Munich, Germany, 80992; email: xuebing.zhou@huawei.com.



This work is licensed under a Creative Commons Attribution-NonCommercial International 4.0 License.

© 2022 Copyright held by the owner/author(s).

2157-6904/2022/06-ART74

<https://doi.org/10.1145/3517820>

1 INTRODUCTION

Machine learning (ML) has been widely applied in solving societal challenges in recent years. Traditional centralized learning mechanisms gather all the client data for model training and therefore suffer from high computational complexity and privacy issues. Due to these problems, **federated learning (FL)** [31] has been proposed, where the ML models are jointly trained by multiple local devices (also referred to as *clients* or *users*) under the coordination of a central server. As training tasks are distributed to local devices and clients' private data are never uploaded to the server, the framework enjoys distinctive advantages in both computational efficiency and privacy protection. Federated learning is increasingly used in real-life scenarios such as health care [27], recommendation systems [47], and other mobile edge computing applications [28].

Although FL enjoys significant privacy benefits in comparison to centralized learning, recent works have demonstrated that FL is still vulnerable to various privacy attacks such as reconstruction attacks [52] and membership inference attacks [34], as the exchanged local updates and the global model parameters may reveal sensitive information of the private local data. Motivated by this, an increasing number of privacy-preserving FL frameworks with privacy-enhancing techniques such as **homomorphic encryption (HE)** [10], **secure multi-party computation (SMC)** [49], and **differential privacy (DP)** [14] have been proposed, aiming to prevent potential privacy leakage in FL. However, the cryptography-based solutions may not be practical to large-scale FL scenarios due to the massive additional communication and computation costs, as discussed in Reference [26].

In comparison, DP-FL frameworks use randomization algorithms (e.g., injecting random noise) to perturb the model updates or the model parameters and do not impose significant additional communication and computation costs. Moreover, the randomization algorithms follow a strict DP definition and can effectively prevent the attackers from inferring information from local data. The DP-FL frameworks in References [3, 32] add Gaussian noise on the server side to protect the privacy of the global model. However, they assume the presence of a trusted server. Thus, a more practical solution is to apply **local differential privacy (LDP)** to FL, which perturbs the local updates before sending them to the server. Nevertheless, as discussed in Reference [22], it is very challenging for LDP-FL frameworks to achieve a satisfactory privacy-utility balance, especially for high-dimensional models. For instance, Shokri and Shmatikov [38] propose the first LDP-FL framework using the sparse vector technique. However, its DP guarantee is per-dimension and is less effective for large models [36]. Later works, i.e., Duchi et al. [12], Wang et al. [42], and Zhao et al. [51], adopted LDP mean estimation algorithms to perturb the local updates. However, the injected noise in these solutions is in essence proportional to the model dimensionality, making them only applicable to simple ML models. A recent work proposed FEDSEL [29], a *two-stage* LDP-FL framework that includes a **dimension selection (DS)** stage and a **value perturbation (VP)** stage. Given an original local update vector, the DS stage first builds a top- k dimension set containing the dimensions of the k largest *absolute* update values and privately selects one "important" dimension from the top- k set. Then, in the VP stage, the value of the selected dimension is perturbed via the LDP algorithms in References [12, 42] and used to construct a sparse privatized local update.

Although FEDSEL [29] mitigates the dimension-dependency problem in previous works, the framework may still suffer from utility loss due to the following reasons. First, since the privacy budget is consumed by both stages, for high-privacy regimes (when the privacy budget is small), each stage may obtain only an insufficient privacy budget and cause large randomness. Moreover, the framework only selects one dimension for each local update, which may lead to slow model convergence, especially for high-dimensional models. In this article, we address the above challenges from two perspectives. First, we propose a novel LDP-FL framework, SIGNDS-FL (as shown in Figure 1), which aims to mitigate the problem of an insufficient per-stage privacy

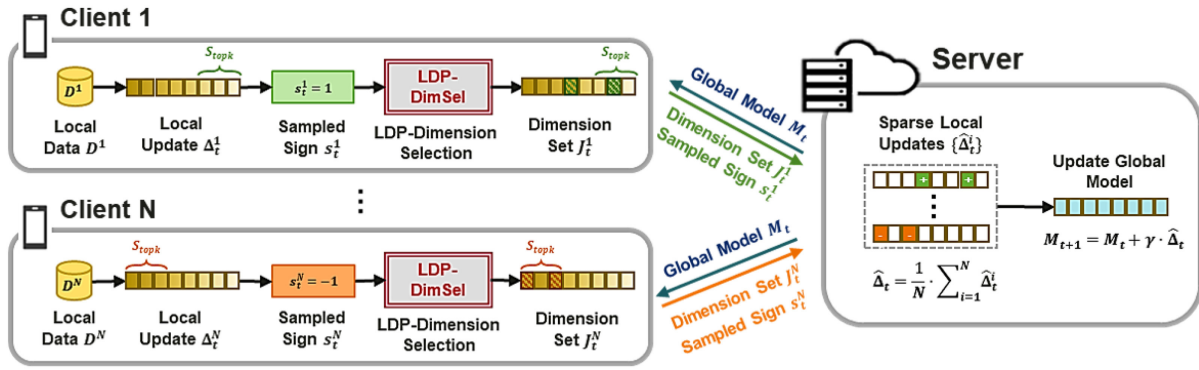


Fig. 1. Overview of the SIGN-FL framework. At each global round, the server broadcasts the current global model to the local side. Each client trains the global model with local data and computes the local model update. Then, the client randomly samples a sign value and builds the top- k dimension set: If the sign value equals 1, then the top- k set is built with the dimensions of the k largest update values; otherwise, it is built with the dimensions of the k smallest update values. An LDP-based dimension selection algorithm is then applied to select a set of “important” dimensions. The sampled sign value and the selected dimension set will be sent to the server. The server will construct sparse privatized local updates by assigning the sign value to the corresponding selected dimensions and finally use them to update the global model.

budget in the high-privacy regime. The main idea is to save the privacy budget for the VP stage in Reference [29] by building the top- k set according to the *real* local update values and assigning sign values instead of the perturbed dimension values to the selected dimensions. The sign values are randomly sampled by each client on the local side, which are used for determining the top- k set and constructing the privatized local update. Moreover, besides adopting the single-dimension selection algorithms as in Reference [29], we give the first attempt to explore multi-dimension selection algorithms. To select h dimensions under the ϵ -LDP guarantee, a naive approach is to independently perform the single-dimension selection h times. However, the privacy budget for selecting each dimension should then be less than ϵ/h , which will lead to a significant degradation in model utility. Inspired by the *exponential mechanism* [33] and its extension in frequency estimation on set-valued data [43], we propose an **Exponential Mechanism-based Multi-Dimension Selection (EM-MDS)** algorithm. In comparison to the naive approach, EM-MDS utilizes the privacy budget more efficiently and helps improve model convergence and accuracy.¹

Our major contributions can be summarized as follows:

- We propose SIGNDS-FL, a novel LDP-FL framework. Different from Reference [29], we propose to build the top- k dimension set according to *real* update values and construct the privatized updates based on sign values randomly sampled by the clients. Our solution saves the privacy budget for the VP stage in Reference [29] and achieves better model accuracy under the same privacy level.
- We further extend the single-dimension selection strategy in Reference [29] to multi-dimension selection and propose EM-MDS, a novel algorithm based on the idea of the *exponential mechanism* [33]. The algorithm can more effectively utilize the privacy budget in comparison to the naive approach and achieves better model utility.
- We evaluate the performance of our framework on a number of real-world datasets and ML models and compare the results with previous works. For the simple tasks of training logistic

¹The implementation code can be found at: https://gitee.com/mindspore/mindspore/blob/master/mindspore/lite/java/java/fl_client/src/main/java/com/mindspore/flclient/SecureProtocol.java.

regression models on structured datasets, our framework achieves an accuracy loss of only $\sim 1\%$ – 2% under a privacy level $\epsilon \geq 4$ in comparison to a $\sim 5\%$ – 15% decrease of accuracy for the baseline methods. For the complex tasks, which train deep neural networks on image datasets, the accuracy loss of our framework under a privacy level $\epsilon \geq 8$ is also less than 8% and at best only 2% . Extensive experimental results demonstrate that our framework significantly outperforms the previous LDP-FL solutions and enjoys an advanced utility-privacy balance.

The remainder of the article is organized as follows. In Section 2, we discuss prior work on privacy-preserving FL. Section 3 presents a brief background about DP and FL. The system model and the proposed methodology are introduced in detail in Section 4. The evaluation experiments and results are then presented in Section 5. In Section 6, we discuss the potential applications and the limitations of the proposed method. Finally, conclusions and avenues for future work are presented in Section 7.

2 RELATED WORK

In recent years, DP [14] has been widely used as a strict criterion for privacy protection in data analysis [13], data publishing [50], and machine learning [1, 8]. Moreover, an increasing number of studies also incorporate DP into FL to reduce potential privacy leakage and offer privacy guarantees for the framework. Previous works by McMahan et al. [32] and Augenstein et al. [3] proposed to add Gaussian noise on the server side to protect the privacy of the global model. However, such solutions cannot prevent privacy leakage from clients' local updates. Follow-up works further adopted crypto-based algorithms to strengthen local privacy. For instance, Jayaraman et al. [18] presented a framework that incorporates DP with SMC while Truex et al. [40] introduced a hybrid solution with DP and HE. Yet, the solutions require extra communication and computation costs during the key-distribution phase and cannot be applied to large-scale scenarios.

Due to the privacy and efficiency issues in the above-mentioned solutions, a more practical approach is to use LDP [23] to privatize the original local updates before sending them to the server. Considering that the local updates are numerical vectors and the global aggregation on the server side computes the average of all the local updates, a natural way is to use the LDP mean estimation algorithms. Given a local update vector $\Delta \in [-1, 1]^d$, a naive solution is to independently perturb each dimension using the Laplace mechanism [14], i.e., $\hat{\Delta} = \Delta + \text{Lap}(\frac{2d}{\epsilon})$. However, the noise scale is essentially linear to the dimension d and will result in a significant utility loss for high-dimensional models. To reduce the noise scale, Duchi et al. [12] further proposed a method based on randomized response [44] (referred to as DM) that maps each original value to two possible constants $\{-B, B\}$, which are determined by d and ϵ . Although the algorithm achieves a lower noise scale, it is relatively sophisticated and does not achieve ϵ -LDP when d is even [35]. Based on DM, Wang et al. [42] further proposed a **piecewise mechanism (PM)**, which returns a sparse privatized vector with at most m dimensional values, where $m = \max\{1, \min\{d, \lfloor \frac{\epsilon}{2.5} \rfloor\}\}$. More specifically, for each selected dimension $\Delta[j]$, the algorithm first computes the noised results x_j and let the $\hat{\Delta}[j] = \frac{d}{m}x_j$. In this case, the communication cost is reduced to $O(m)$ in comparison with $O(d)$ in DM. The authors also proposed a **hybrid mechanism (HM)**, which combines DM and PM to achieve an optimized worst-case variance. Additionally, a follow-up work by Zhao et al. [51] proposed an improved PM-SUB algorithm, which further reduced the variance when ϵ is large. Although Reference [42] and Reference [51] increase the per-dimension privacy budget to ϵ/m by only reporting the value of m random dimensions, the perturbed values are finally enlarged by d/m for an unbiased estimation, which increases the injected noise at the same time. Based on the above LDP mean estimation based solutions, a recent work by Liu et al. [29] further proposed

FEDSEL, a *two-stage* LDP-FL framework that includes a DS stage and a VP stage. In the DS stage, LDP-based dimension selection algorithms are applied to select one “important” dimension from the top- k dimension set (i.e., the set of dimensions with the k largest *absolute* update values); in the VP stage, the value of the selected dimension is perturbed via the LDP algorithms in References [12, 42]. Finally, a sparse privatized local update is constructed and returned to the server. Although [29] mitigates the *dimension-dependency* problem apparent in previous works, the privacy budget is still consumed by two stages. When the privacy budget is small, each stage may obtain only an insufficient privacy budget and cause large randomness. Also, only selecting one dimension for each local update may lead to slow model convergence, especially for high-dimensional models.

In addition to solutions following a strict LDP definition, alternative notions of LDP have also been investigated. For instance, Bhowmick et al. [5] introduced minimax differential privacy, which relaxes *local privacy* by limiting the prior knowledge of the attackers, thus allowing the algorithm to be performed under a much larger privacy budget (e.g., $\epsilon > 500$). Similarly, Truex et al. [41] proposed the α -CLDP-Fed framework based on **Condensed LDP (CLDP)** (α -CLDP) [15]. The algorithm adopts α as the privacy cost under CLDP and requires $\alpha \ll \epsilon$ to ensure meaningful privacy protection. However, they adopted $\alpha = 1$ in the evaluation experiments, which is equivalent to using a very large ϵ under the original LDP definition and results in weak privacy protection. As we mainly focus on LDP-FL under the original LDP definition, the above two solutions are out of scope for this article.

3 PRELIMINARIES

In the following, we offer background information about DP and FL.

3.1 Differential Privacy

Differential privacy [14] is a state-of-the-art data anonymization technique that provides strong privacy guarantees for data analysis. The mathematical definition of DP is as follows:

Definition 1 (DP [14]). A randomized mechanism $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{O}$ satisfies ϵ -DP if for any two adjacent datasets D, D' differing from one data sample and for any measurable subset of outputs $\mathcal{Y} \subseteq \mathcal{O}$:

$$\Pr [\mathcal{A}(D) \in \mathcal{Y}] \leq e^\epsilon \cdot \Pr [\mathcal{A}(D') \in \mathcal{Y}], \quad (1)$$

where ϵ describes the privacy loss.

Definition 1 is usually applied in centralized settings where the data have already been collected by a trusted server. However, in local settings, we aim to ensure that each client’s local data will not be accessed by the server. Thus, the definition of LDP has been proposed [23], which provides strong local privacy guarantees for each user. The definition is as follows:

Definition 2 (LDP [23]). A randomized mechanism $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{O}$ satisfies ϵ -LDP if and only if for any two inputs $x, x' \in \mathcal{D}$ and for any output $y \in \mathcal{O}$:

$$\Pr [\mathcal{A}(x) = y] \leq e^\epsilon \cdot \Pr [\mathcal{A}(x') = y], \quad (2)$$

where ϵ describes the privacy loss.

In addition, LDP also holds two widely used properties [14], namely *robustness to post-processing* and *sequential composition*. The former property states that any deterministic or randomized function defined over an LDP mechanism also satisfies LDP. The latter property states that interactively applying the LDP mechanism on the same set of data yields an accumulated privacy cost.

PROPERTY 1 (ROBUSTNESS TO POST-PROCESSING). *Let \mathcal{A} be an ϵ -LDP mechanism and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{A}$ is ϵ -LDP.*

PROPERTY 2 (SEQUENTIAL COMPOSITION). *Suppose n mechanisms $\{\mathcal{A}_1, \dots, \mathcal{A}_n\}$ respectively satisfy ϵ_i -LDP and are sequentially computed on the same set of private data D , then a mechanism formed by $(\mathcal{A}_1, \dots, \mathcal{A}_n)$ satisfies $(\sum_{i=1}^n \epsilon_i)$ -LDP.*

3.2 Federated Learning

Federated learning [31] is a decentralized learning framework that achieves computational efficiency and privacy benefits by distributing the training task to local devices. At each global round, the server distributes the current global model to a number of local clients. Each client locally updates the global model and returns the model update to the server. On the server side, all the local updates are aggregated to update the global model, which will be distributed in the next global round. Since only model parameters are exchanged during the training process, FL allows the model to be trained without accessing raw local data and thus provides enhanced privacy protection in comparison to centralized training. Nevertheless, recent contributions point out that the naive FL framework still suffers from various privacy attacks, which may take place not only during the training phase [34, 52] but also during the inference phase [19]. These privacy risks can be mainly divided into *local privacy* and *global privacy* aspects. *Local privacy* risks appear when the local updates reveal insights about local data, while *global privacy* risks represent situations when the global model memorizes local data. Clearly, the local privacy risks are more severe than the global privacy risks since they can directly reveal information of a specific client. To address this problem, an increasing number of privacy-preserving FL frameworks with privacy-enhancing techniques such as HE [40], SMC [18], and DP [29, 32] have been proposed, aiming to provide protection against potential privacy risks.

4 SIGNDS-FL FRAMEWORK

Although the LDP-FL framework in Reference [29] shows considerable performance improvements in comparison to previous works, it may still suffer from utility loss because of the insufficient per-stage privacy budget and slow model convergence. In this section, we will present our solution to address these limitations. To start with, we will describe our system model. Then, we will introduce our enhanced LDP-FL framework SIGNDS-FL, which aims to further reduce the privacy cost in Reference [29] and mitigate the utility loss in high-privacy regimes. Finally, we will present EM-MDS, a new privacy-preserving multi-dimension selection algorithm that helps improve model convergence.

4.1 System Model

A common FL scenario is considered in this article, where a machine learning model M is jointly trained by a number of local clients under the coordination of a central server. Each client i holds a local dataset D^i that contains the client's private personal data. At each global round t , the server selects a group of N clients and distributes the current global model M_t . Each client i in the group trains the global model for several gradient descent steps with his local data D^i and obtains the local model M_t^i . Then, the client computes the local update $\Delta_t^i = M_t^i - M_t$ and sends Δ_t^i back to the server. On the server side, all the local updates are aggregated and averaged, which is then used to update the global model as follows:

$$\Delta_t = \frac{1}{N} \sum_{i=1}^N \Delta_t^i, \quad M_{t+1} = M_t + \Delta_t. \quad (3)$$

The updated global model M_{t+1} is distributed again to local clients to start the next round.

Since the raw local updates Δ_t^i are derived based on clients' personal data, directly uploading Δ_t^i to the server may reveal private information of raw local data. Here, the server is assumed to be

honest-but-curious, which follows the system protocols but tries to infer sensitive information of local users from Δ_t^i . Thus, to prevent such privacy leakage, the clients use a randomization algorithm \mathcal{A} to perturb the local updates and send the privatized updates to the server. The randomization algorithm \mathcal{A} follows the strict LDP definition, ensuring that the server lacks access to the original local updates and providing formal privacy guarantees for clients' local data.

4.2 General Workflow of SIGNDS-FL

Motivated by the limitations of Reference [29], we propose SIGNDS-FL, an improved LDP-FL framework. The main idea is to substitute the VP stage in Reference [29] by assigning a constant value to the selected dimensions to save privacy costs. In this way, with the same privacy level, the proposed method can achieve less randomness and thus higher accuracy in the dimension selection stage. However, since the parameter values may have different signs, assigning the same constant value to all the selected dimensions may result in a significant change in the gradient descent direction. To address the problem, the SIGNDS-FL framework adopts an extra sign variable $s \in \{-1, 1\}$, which is used for dimension selection and for the construction of the sparse privatized local updates. The idea is inspired by the SIGNSGD algorithm proposed by Bernstein et al. [4]. The authors quantized each local update value to its sign value to reduce the communication cost and proved that the algorithm can still enjoy a satisfying convergence rate. Thus, in our algorithm, each client randomly samples a sign value s on the local side. If $s = 1$, then the dimensions of the k largest values in the local update are used to build the top- k dimension set and perform the private dimension selection process. The selected dimensions will then be assigned with $s = 1$ to construct the sparse privatized local update. Intuitively, the k largest values are highly likely to be larger than zero. Therefore, assigning the positive sign value to the selected dimension will not cause much difference in the model update direction, which mitigates the impact on model accuracy. Similarly, when $s = -1$, the dimensions of the k smallest values are used for building the top- k set. The selected dimensions are assigned with the negative sign value for constructing the sparse privatized local update.

The general training workflow of SIGNDS-FL is presented in Algorithm 1. At each global round t , the server selects a group of N clients and broadcasts the current global model M_t to the clients (Lines 2–4). On the local side, the client i trains the global model for several gradient descent epochs with his local data D^i and computes the local update Δ_t^i (Lines 13–17). Then, the client sorts Δ_t^i by *real* update values, randomly samples a sign value $s_t^i \in \{-1, 1\}$ with equal probability, and builds the top- k dimension set S_{topk} following the idea described above (Lines 18–23). Given the privacy budget ϵ , different LDP dimension selection algorithms are applied to privately select a set of dimensions J_t^i (Line 24), which, together with the sampled sign s_t^i , will be returned to the server. The server then constructs the sparse privatized local update $\hat{\Delta}_t^i$ by assigning the sign value s_t^i to all the dimensions contained in J_t^i (Lines 6 and 7). Finally, all the sparse local updates are aggregated and used to update the global model with a global learning rate γ (Lines 9 and 10). The updated global model M_{t+1} is distributed again to local clients to start the next round.

In the later sections, we will introduce two private dimension selection algorithms that provide strict ϵ -LDP guarantees to clients' local data. In addition, since the sign values are randomly sampled by the clients and are unrelated to the local data, the sparse privatized local updates constructed on the server side also satisfy ϵ -LDP.

THEOREM 1. *For the original local update Δ of any client, if the dimension selection algorithm used in Algorithm 1 satisfies ϵ -LDP, then the sparse privatized local update $\hat{\Delta}$ also satisfies ϵ -LDP.*

PROOF. According to Definition 2, for any client with two possible local updates Δ, Δ' , the privatized local update $\hat{\Delta}$ satisfies ϵ -LDP if and only if $\frac{\Pr[\hat{\Delta}|\Delta]}{\Pr[\hat{\Delta}|\Delta']} \leq \exp(\epsilon)$.

ALGORITHM 1: SIGNDS-FL

Input: $M_1 \in \mathbb{R}^d$: initial global model; T : global aggregation rounds; N : number of per-round clients; E : number of local epochs; η : local learning rate; d : local update size; k : top- k set size; h : output dimension set size; γ : global learning rate; ϵ : privacy budget

Output: Trained model M

Server executes:

```

1: for global round  $t = 1, \dots, T$  do
2:   Randomly select a group of  $N$  clients
3:   for client  $i = 1, \dots, N$  in parallel do
4:     Broadcast current global model  $M_t$ 
5:     Receive the dimension set and sign
        $J_t^i, s_t^i = \text{LocalUpdate}(M_t, E, \eta, \epsilon, k, h)$ 
6:     Initialize the sparse privatized local update
        $\hat{\Delta}_t^i = [0]^d$ 
7:     For  $j \in \{1, \dots, d\}$ , if  $j \in J_t^i$ , set  $\hat{\Delta}_t^i[j] = s_t^i$ 
8:   end for
9:   Aggregate all the sparse local updates:
        $\hat{\Delta}_t = \frac{1}{N} \sum_{i=1}^N \hat{\Delta}_t^i$ 
10:  Update global model  $M_{t+1} = M_t + \gamma \cdot \hat{\Delta}_t$ 
11: end for
12: Return Global model  $M = M_{T+1}$ 

```

LocalUpdate($M_t, E, \eta, \epsilon, k, h$):

```

// Run on the client side
13: Initialize local model  $M_t^i \leftarrow M_t$ 
14: for epoch  $e = 1, \dots, E$  do
15:    $M_t^i = M_t^i - \eta \cdot \nabla \mathcal{L}(M_t^i, D^i)$ 
16: end for
17: Calculate local update:  $\Delta_t^i = M_t^i - M_t$ 
18: Randomly sample a sign  $s_t^i \in \{1, -1\}$  with
    probability  $\Pr[s_t^i = 1] = 0.5$ 
19: if  $s_t^i = 1$  then
20:   Select dimensions of  $k$  largest values in  $\Delta_t^i$  to
    construct  $S_{topk}$ 
21: else
22:   Select dimensions of  $k$  smallest values in  $\Delta_t^i$  to
    construct  $S_{topk}$ 
23: end if
24: Obtain the private dimension set  $J_t^i =$ 
    LDP-DimSel( $S_{topk}, d, k, h, \epsilon$ )
25: Return  $J_t^i, s_t^i$ 

```

In Algorithm 1, the construction of the privatized local update can be decomposed into two steps: privately selecting a dimension set J according to the sampled sign value s and assigning each dimension $j \in J$ with s . This can be formulated as

$$\frac{\Pr[\hat{\Delta}|\Delta]}{\Pr[\hat{\Delta}|\Delta']} = \frac{\Pr[J|\Delta] \cdot \prod_{j \in J} \Pr[\hat{\Delta}[j]|\Delta[j]]}{\Pr[J|\Delta'] \cdot \prod_{j \in J} \Pr[\hat{\Delta}[j]|\Delta'[j]]}, \quad (4)$$

and since the dimension selection step satisfies the ϵ -LDP guarantee, it holds that $\frac{\Pr[J|\Delta]}{\Pr[J|\Delta']} \leq \exp(\epsilon)$. Additionally, each selected dimension is assigned with the sign value s , which is a constant and is independent from the real dimension value. Thus, $\Pr[\hat{\Delta}[j]|\Delta[j]] = \Pr[\hat{\Delta}[j]|\Delta'[j]] = 1$, where $j \in J$. To sum up the above considerations, it holds that

$$\frac{\Pr[\hat{\Delta}|\Delta]}{\Pr[\hat{\Delta}|\Delta']} = \frac{\Pr[J|\Delta] \cdot 1}{\Pr[J|\Delta'] \cdot 1} \leq \exp(\epsilon), \quad (5)$$

which completes the proof. \square

Note that the privacy guarantees achieved by Theorem 1 hold for any ϵ -LDP dimension selection algorithms and are agnostic to the model structures.

4.3 LDP Dimension Selection Algorithms

Next, we will introduce two private dimension selection algorithms used in the framework that provide strict ϵ -LDP guarantees to the local updates and local data.

ALGORITHM 2: PS for single dimension selection [29]**Input:** S_{topk} : top- k dimension set; d : local update size; k : top- k set size; ϵ : privacy budget.**Output:** J : selected dimension

```

1: Sample a Bernoulli variable  $x$  such that
   
$$\Pr[x = 1] = \frac{\exp(\epsilon) \cdot k}{d - k + \exp(\epsilon) \cdot k}$$

2: if  $x = 1$  then
   3:   Randomly sample a dimension
       
$$J \in \{a \in \{1, \dots, d\} | a \in S_{topk}\}$$

   4: else
   5:   Randomly sample a dimension
       
$$J \in \{a \in \{1, \dots, d\} | a \notin S_{topk}\}$$

   6: end if
7: Return  $J$ 

```

4.3.1 Single-dimension Selection. We start with the single-dimension selection algorithms, which only select one dimension for each local update. This has been investigated in Reference [29]. Here, we briefly introduce one of the proposed algorithms called **perturbed sampling (PS)**, which is presented in Algorithm 2. Given the top- k dimension set S_{topk} , a dimension J is randomly sampled as

$$J \in \begin{cases} \{a \in \{1, \dots, d\} | a \in S_{topk}\} & \text{with probability } p_{sin} \\ \{a \in \{1, \dots, d\} | a \notin S_{topk}\} & \text{with probability } 1 - p_{sin} \end{cases}. \quad (7)$$

Namely, the dimension J is sampled from the top- k dimension set with a probability p_{sin} and otherwise from the non-top- k dimension set. Let p_{sin} be the top- k probability. Thus, the privacy guarantee of Algorithm 2 is as follows:

LEMMA 1. *Algorithm 2 satisfies ϵ -LDP when $p_{sin} \leq \frac{\exp(\epsilon) \cdot k}{d - k + \exp(\epsilon) \cdot k}$.*

PROOF. For each client, given any two possible local updates Δ and Δ' and the sampled sign value s , let S_{topk}, S'_{topk} be the corresponding top- k dimension sets and $J \in \{1, \dots, d\}$ be any output dimension. Given the top- k probability p_{sin} , the probabilities of sampling the dimension J from the top- k set and from the non-top- k set are respectively $p_{sin} \cdot \frac{1}{k}$ and $(1 - p_{sin}) \cdot \frac{1}{d-k}$. Thus, when $p_{sin} \leq \frac{\exp(\epsilon) \cdot k}{d - k + \exp(\epsilon) \cdot k}$ it holds that

$$\frac{\Pr[J|\Delta]}{\Pr[J|\Delta']} = \frac{\Pr[J|S_{topk}]}{\Pr[J|S'_{topk}]} \leq \frac{\Pr[J|J \in S_{topk}]}{\Pr[J|J \notin S'_{topk}]} = \frac{p_{sin} \cdot \frac{1}{k}}{(1 - p_{sin}) \cdot \frac{1}{d-k}} \leq \exp(\epsilon), \quad (8)$$

which completes the proof. \square

4.3.2 Multi-dimension Selection. Although Algorithm 2 can efficiently select top- k dimensions under LDP guarantees, it only returns one dimension for each local model update, which may result in the loss of valuable parameter information and a slow convergence for high-dimensional models. This motivates us to consider whether it is possible to generalize the algorithm by returning multiple dimensions. To select h dimensions under ϵ -LDP, a naive method is to repeatedly apply Algorithm 2 for h times. However, according to the sequential composition property (Property 2), the privacy budget for selecting *each* dimension should be less than ϵ/h , which will lead to a significant decrease in the probability p_{sin} and a degradation of model accuracy. Therefore, better dimension selection algorithms are needed. Since the dimension indices here are non-numerical values, traditional LDP algorithms such as Laplace and Gaussian mechanisms [14] cannot be directly applied. In contrast, the *exponential mechanism* [33] is a widely used method for handling such non-numerical queries. Inspired by this idea, we further propose an enhanced EM-MDS algorithm to improve model convergence and utility.

Consider a top- k dimension set S_{topk} and an output set J with h elements. Let $v = |S_{topk} \cap J|$ be the number of intersections between the two sets, which is equivalent to the number of top- k dimensions contained in J . A score function $u(S_{topk}, J)$ is then defined as an indicator function to highlight whether the top- k dimensions contained in J are larger than a certain threshold v_{th} , namely

$$u(S_{topk}, J) = \mathbb{1}(|S_{topk} \cap J| \geq v_{th}) = \mathbb{1}(v \geq v_{th}), \quad (9)$$

where $0 \leq v \leq h$ and $1 \leq v_{th} \leq h$. Furthermore, let ϕ_u be the sensitivity of u , it can be derived that

$$\phi_u = \max_{J \in \mathcal{J}} \|u(S_{topk}, J) - u(S'_{topk}, J)\| = 1 - 0 = 1, \quad (10)$$

where \mathcal{J} is the domain of the output set and S_{topk} and S'_{topk} are two random top- k dimension sets. With the above definitions, the multi-dimension selection process can be defined as follows.

Definition 3 (EM-MDS). Given the top- k dimension set S_{topk} of a local update, one can randomly sample an output dimension set $J \in \mathcal{J}$ with the following probability:

$$\begin{aligned} p_{mul} &= \frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J))}{\sum_{J' \in \mathcal{J}} \exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J'))} = \frac{\exp(\epsilon \cdot \mathbb{1}(v \geq v_{th}))}{\sum_{\tau=0}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon \cdot \mathbb{1}(\tau \geq v_{th}))}, \\ &= \frac{\exp(\epsilon \cdot \mathbb{1}(v \geq v_{th}))}{\sum_{\tau=0}^{\tau=v_{th}-1} \omega_{\tau} + \sum_{\tau=v_{th}}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon)}, \end{aligned} \quad (11)$$

where v is the number of top- k dimensions contained in J , v_{th} is the threshold of the score function and $\omega_{\tau} = \binom{k}{\tau} \binom{d-k}{h-\tau}$ is the number of possible combinations in \mathcal{J} that contains τ top- k dimensions.

The next question is How do we determine an appropriate threshold v_{th} to achieve a satisfactory model utility? From Equation (11), the probability that the sampled J contains τ top- k dimensions given a threshold v_{th} can further be derived as follows:

$$p_{mul}(v = \tau | v_{th}) = \begin{cases} \omega_{\tau} / \Omega & \text{if } 0 \leq \tau < v_{th} \\ \omega_{\tau} \cdot \exp(\epsilon) / \Omega & \text{if } v_{th} \leq \tau \leq h \end{cases}, \quad (12)$$

where $\Omega = \sum_{\tau=0}^{\tau=v_{th}-1} \omega_{\tau} + \sum_{\tau=v_{th}}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon)$ is the denominator part of Equation (11). Moreover, the expectation of v given the threshold v_{th} can be calculated as

$$\mathbb{E}_{mul}[v | v_{th}] = \sum_{\tau=0}^{\tau=h} \tau \cdot p_{mul}(v = \tau | v_{th}). \quad (13)$$

Intuitively, the higher $\mathbb{E}_{mul}[v | v_{th}]$, the higher the probability that the sampled J contains more top- k dimensions and the better the model utility. Therefore, the optimum threshold v_{th}^* can be determined as the threshold that achieves the highest $\mathbb{E}_{mul}[v | v_{th}]$, namely

$$v_{th}^* = \arg \max_{v_{th} \in \{1, \dots, h\}} \mathbb{E}_{mul}[v | v_{th}]. \quad (14)$$

By summarizing all the above design considerations, the workflow of our EM-MDS algorithm is presented in Algorithm 3. Given all the input settings (d, k, h, ϵ) , the optimal threshold v_{th}^* is firstly determined based on Equations (13) and (14) (Line 1). Then, the dimension selection process is conducted as in Definition 3. Nevertheless, as the output domain \mathcal{J} contains $\binom{d}{k}$ possible combinations, directly sampling a set J from \mathcal{J} with a certain probability is computationally expensive, especially when d and k are large. To further improve the efficiency, the trick of inverse sampling is applied, which first samples a random variable β from the uniform distribution $\mathcal{U}(0, 1)$ and determines the target number of top- k dimensions v according to the cumulative distribution

ALGORITHM 3: EM-MDS for multi-dimension selection

Input: S_{topk} : top- k dimension set; d : local update size; k : top- k set size; h : output size;
 ϵ : privacy budget.

Output: J : dimension set with h elements

```

1: Determine the optimum threshold
    $v_{th}^* = \arg \max_{v_{th} \in \{1, \dots, h\}} \mathbb{E}_{mul}[v|v_{th}]$ 
2: Compute denominator
    $\Omega = \sum_{\tau=0}^{\tau=v_{th}^*-1} \omega_{\tau} + \sum_{\tau=v_{th}^*}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon)$ , where
    $\omega_{\tau} = \binom{k}{\tau} \binom{d-k}{h-\tau}$ 
3: Randomly sample  $\beta$  from uniform distribution
    $\mathcal{U}(0, 1)$ 
4: Initialize  $\tau = 0$ ,  $CDF = \omega_0/\Omega$ 
5: while  $CDF < \beta$  do
6:    $\tau = \tau + 1$ 
7:   if  $\tau < v_{th}^*$  then
8:      $CDF = CDF + \omega_{\tau}/\Omega$ 
9:   else
10:     $CDF = CDF + \omega_{\tau} \cdot \exp(\epsilon)/\Omega$ 
11:   end if
12: end while
13: Let  $v = \tau$  be the number of top- $k$  dimensions
14: Sample  $v$  dimensions from
    $\{a \in \{1, \dots, d\} | a \in S_{topk}\}$ 
   and append to  $J$ 
15: Sample  $h - v$  dimensions from
    $\{a \in \{1, \dots, d\} | a \notin S_{topk}\}$ 
   and append to  $J$ 
16: Return  $J$ 

```

of $p_{mul}(v|v_{th}^*)$ (Lines 3–13). An example of the inverse sampling trick is illustrated in Figure 2. In this case, the computational cost of Algorithm 3 is efficiently reduced from $O(\binom{d}{k})$ to $O(d)$. Finally, the output dimension set J is constructed by randomly sampling v dimensions from the top- k dimension set and $h - v$ dimensions from the non-top- k set.

Now, we present the privacy and utility analysis of the EM-MDS algorithm.

LEMMA 2. *Algorithm 3 satisfies ϵ -LDP.*

PROOF. For each client, given any two possible local updates Δ and Δ' and the sampled sign value s , let S_{topk}, S'_{topk} be the corresponding top- k dimension sets. For any output dimension set $J \in \mathcal{J}$, let $v = |S_{topk} \cap J|$, $v' = |S'_{topk} \cap J|$ be the number of intersections between J and both top- k sets. With the sampling probability defined in Equation (11) it holds that

$$\begin{aligned} \frac{\Pr[J|\Delta]}{\Pr[J|\Delta']} &= \frac{\Pr[J|S_{topk}]}{\Pr[J|S'_{topk}]} = \frac{\frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J))}{\sum_{J' \in \mathcal{J}} \exp(\frac{\epsilon}{\phi_u} \cdot u(S_{topk}, J'))}}{\frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S'_{topk}, J))}{\sum_{J' \in \mathcal{J}} \exp(\frac{\epsilon}{\phi_u} \cdot u(S'_{topk}, J'))}} = \frac{\frac{\exp(\epsilon \cdot \mathbb{1}(v \geq v_{th}))}{\sum_{\tau=0}^{\tau=v_{th}^*-1} \omega_{\tau} + \sum_{\tau=v_{th}^*}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon)}}{\frac{\exp(\epsilon \cdot \mathbb{1}(v' \geq v_{th}))}{\sum_{\tau=0}^{\tau=v_{th}^*-1} \omega_{\tau} + \sum_{\tau=v_{th}^*}^{\tau=h} \omega_{\tau} \cdot \exp(\epsilon)}} \quad (15) \\ &= \frac{\exp(\epsilon \cdot \mathbb{1}(v \geq v_{th}))}{\exp(\epsilon \cdot \mathbb{1}(v' \geq v_{th}))} \leq \frac{\exp(\epsilon \cdot 1)}{\exp(\epsilon \cdot 0)} = \exp(\epsilon), \end{aligned}$$

which completes the proof. \square

As for the utility analysis, we compare the performance of our EM-MDS algorithm with the naive method of applying Algorithm 2 for multi-dimension selection. More specifically, with the output size h , the naive way repeats the PS algorithm h times with the privacy budget ϵ/h for each iteration. Thus, the probability of sampling from the top- k set is $p_{sin} = \frac{\exp(\epsilon/h) \cdot k}{d-k + \exp(\epsilon/h) \cdot k}$. Given a sampled dimension set J , the probability distribution and the expectation of v can be derived as follows:

$$p_{sin}(v = \tau) = \binom{h}{\tau} \cdot p_{sin}^{\tau} \cdot (1 - p_{sin})^{h-\tau}, \quad \mathbb{E}_{sin}[v] = \sum_{\tau=0}^{\tau=h} \tau \cdot p_{sin}(v = \tau). \quad (16)$$

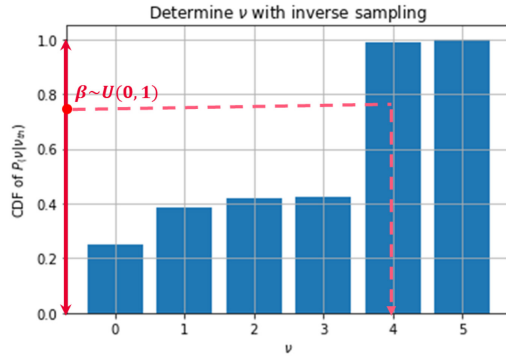


Fig. 2. Example of using the inverse sampling trick to determine v , namely the number of top- k dimensions in J .

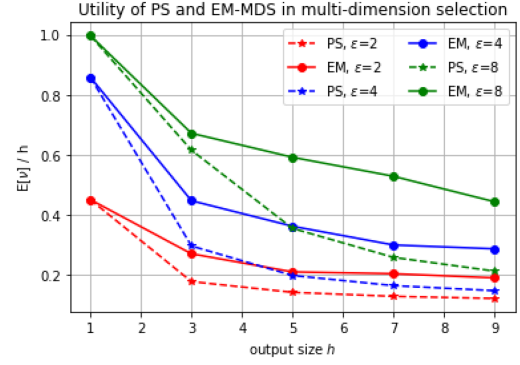


Fig. 3. Utility of using PS and EM-MDS algorithms in multi-dimension selection regarding different output size h and privacy budget ϵ .

For the EM-MDS algorithm, we first determine the optimum v_{th}^* based on Equation (14) and then calculate the expectation $\mathbb{E}_{mul}[v] = \mathbb{E}[v|v_{th}^*]$ as in Equation (13).

We select different settings of h and ϵ to empirically compare the privacy-utility tradeoff of the naive method and the proposed method. More specifically, we use $\mathbb{E}[v]/h$, the expected ratio of top- k dimensions contained in the output set, to represent the model utility. Intuitively, a higher expected ratio means that there is a higher probability to select dimensions from the top- k set, which contributes to the model utility. The comparison results are presented in Figure 3. It can be seen that both algorithms achieve the same expected ratio when $h = 1$. This can be derived from Equation (11), where the EM-MDS algorithm is equivalent to the PS algorithm when $h = 1$. Moreover, with the same privacy budget ϵ , increasing the output dimension h will lead to a lower top- k ratio and a lower model utility. This is because a large h causes each dimension allocated with less privacy budget, thus a lower probability to be sampled from the top- k set. Nonetheless, the EM-MDS algorithm achieves a distinctively higher top- k ratio in the output set compared with the naive method. In particular, the higher the privacy budget ϵ , the larger the difference. This is due to the fact that EM-MDS considers all the combinations of the output set and assigns higher probabilities to those with more top- k dimensions. As a result, the algorithm can utilize the privacy budget more efficiently in comparison to the naive method and achieve a better model utility.

5 EXPERIMENTS AND RESULTS

After implementing the proposed framework, we now report the results of comprehensive experiments with a number of open source datasets to evaluate its performance. In this section, we will first introduce the experimental settings and then discuss the evaluation results.

5.1 Experiment Setup

We first introduce the experimental settings including the datasets and models used in the experiments, baseline algorithms, and parameter configurations.

5.1.1 Datasets and Models. Six open source datasets are used for evaluating the performance of our framework. Each dataset contains multi-dimensional data records used for classification tasks:

- The **Census** dataset [11] contains records drawn from the 1990 United States census data, which include 68 personal attributes such as gender, income and marriage status. The dataset is used for a classification task to determine the duration of people’s active duty service.

Table 1. Details of Datasets and Models

Data Type	Dataset	#Records	#Features	#Classes	Model	#Parameters
Structured Data	Census	2,458,285	68	3	LR	207
	Adult	32,561	123	2	LR	248
	USPS	9,298	256	10	LR	2570
	HAR	10,299	561	6	LR	3372
Image Data	FMNIST	70,000	784	10	NN	76,330
					CNN	44,426
	EMNIST	131,600	784	47	NN	79,919
					CNN	47,571

- The **Adult** dataset [24] originally contains records with 15 personal attributes such as age, occupation, education and gender. The goal is to train a binary classifier that determines whether a person earns more than 50 thousand dollars a year. We use a processed version from [37] that converts the original attributes into 123 binary features.
- The **USPS** dataset [17] is a digit dataset provided by the U.S. Postal Service. The dataset contains 9,298 samples with 256 features, which are categorized to 10 classes.
- The **Human Activity Recognition (HAR)** [2] dataset contains 10,299 sensor data records of 30 volunteers that can be categorized into 6 daily activities. Each data record has 561 features representing different time and frequency domain variables.
- The **Fashion-MNIST (FMNIST)** [45] dataset contains 70,000 article images (from Zalando), which are categorized into 10 classes. Each record is a grey-scale image of size 28×28 .
- The **Extended-MNIST (EMNIST)** [9] dataset consists of 131,600 handwritten letters, each is a grey-scale image of size 28×28 . Here, we use the EMNIST-Balanced dataset, where the samples are evenly categorized into 47 classes.

We conduct experiments to evaluate the performance of our framework in both simple and complex training tasks. For simple training tasks, we train **logistic regression (LR)** models on the structured datasets (i.e., **Census**, **Adult**, **USPS**, **HAR**). For complex tasks, we conduct experiments on the image datasets (**FMNIST**, **EMNIST**) using **neural networks (NN)** as well as **convolutional neural networks (CNN)**. Here, we use a two-layer NN with 96 neurons on the hidden layer and LeNet [25], a widely used CNN that consists of two convolution layers (each followed by a maxpooling layer) and three fully connected layers to perform the experiments.

Details about the datasets and models are presented in Table 1, which include the number of records, features, classes for each dataset, as well as the number of parameters of the used models.

5.1.2 Baselines. Various previous works mentioned in Section 2 are used as our baselines, namely: DM [12], PM [42], HM [42], PM-SUB [51], and FEDSEL [29]. For FEDSEL, we choose the ratio of the privacy budget for the dimension selection stage to be 0.1 and 0.5, which is referred in the experiment results as FEDSEL0.1 and FEDSEL0.5.

5.1.3 Parameter Configurations. In the experiments, we set the global round $T = 500$, $N = 250$ clients participate in each round. Each client possesses 10 data records. The global learning rate is set to $\gamma = 0.05N$. For local training, each client updates the model for $E = 10$ epochs with a learning rate $\eta = 0.001$. For the LDP dimension selection algorithms, we set the size of the top- k dimension set $k = 0.1d$, where d is the size of local updates (i.e., the number of model parameters). Moreover, we vary the privacy budget ϵ to explore the influence of privacy on the performance of the framework. In our experiments, we choose $\epsilon \in \{0.5, 1, 2, 4, 8, 12\}$. Additionally, we repeat each experiment 10 times and report the averaged accuracy.

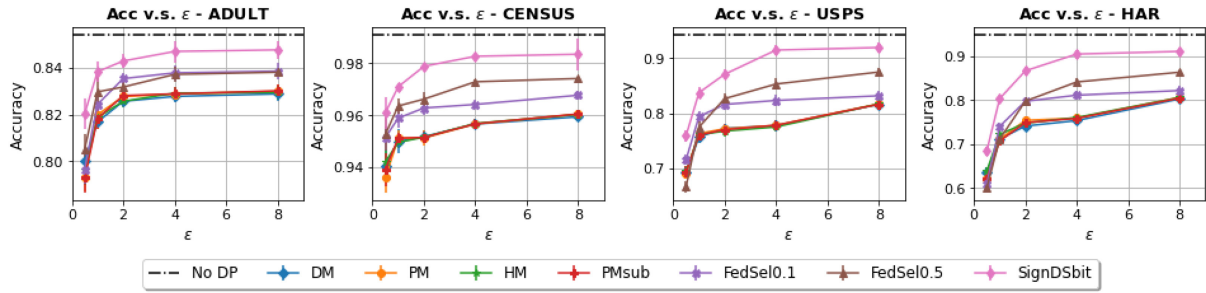


Fig. 4. Accuracy of LR models trained on structured datasets with different privacy budgets ϵ .

5.2 Performance in Simple Training Tasks

We first evaluate the performance of our SIGNDS-FL framework in simple training tasks. Here, we adopt the single-dimension selection algorithm (Algorithm 2) for a fair comparison, which is referred to as SIGNDSBIT. We train LR models on the four structured datasets and compare the model accuracy with the baselines under different settings.

5.2.1 Model Accuracy Regarding the Privacy Level. To start with, we analyze how the framework performs under different privacy budgets. We conduct the training process under different LDP-FL frameworks with $\epsilon \in \{0.5, 1, 2, 4, 8\}$ and compare the model accuracy, as shown in Figure 4. We use the error bars here and also in later results to represent the 95% confidence level. It can be observed that the model accuracy of all methods improves with an increase of ϵ . This aligns with the privacy-utility tradeoff in LDP-FL frameworks: The larger the ϵ , the less randomness added for privacy protection, and the better the model utility. Moreover, although the LDP mean estimation based baselines (DM, PM, HM, and PM-SUB) can achieve relatively satisfactory accuracy for low-dimensional datasets (**Adult**, **Census**), they suffer from an obvious accuracy loss for datasets with more features (**USPS**, **HAR**). This is because the noise scale in these algorithms is proportional to model dimensionality. Thus, for the **USPS** and **HAR** datasets, the increase of model size results in a larger noise scale and a higher accuracy loss. In comparison, FEDSEL and SIGNDS-FL can effectively mitigate the *dimension-dependency* problem. However, SIGNDS-FL can achieve even better performance than FEDSEL. This is likely due to the fact that FEDSEL splits the privacy budget for selecting the top- k dimensions and perturbing the dimension values, whereas our framework applies all the privacy budget for better dimension selection results. In addition, FEDSEL adds random noise to the dimension value, while SIGNDS-FL replaces the dimension value with the sign values. This can better preserve the real model update direction and further improves model convergence. It can be seen that SIGNDS-FL consistently outperforms the baselines for all the datasets. With $\epsilon \geq 4$, the baseline methods suffer from at minimum $\sim 5\%$ – 15% decrease of accuracy for the **USPS** and **HAR** datasets, while our framework only causes around $\sim 1\%$ – 2% accuracy loss. The results illustrate that the SIGNDS-FL framework can effectively address the *dimension-dependency* problem in baseline algorithms and achieve better model utility under the same privacy levels.

5.2.2 Model Accuracy Regarding the Group Size. We further investigate how the choice of group size N (i.e., the number of participating clients in each round) influences the training performance. We conduct experiments varying the group size $N \in \{100, 250, 500, 750\}$ under a privacy level of $\epsilon = 4$ and compare the change of model accuracy for the different LDP-FL solutions. The results in Figure 5 show that training with a larger group size can in general help improve model accuracy. In addition, our framework consistently outperforms the baselines under the same group size settings. Moreover, it can be observed that the baseline algorithms are more sensitive to the group

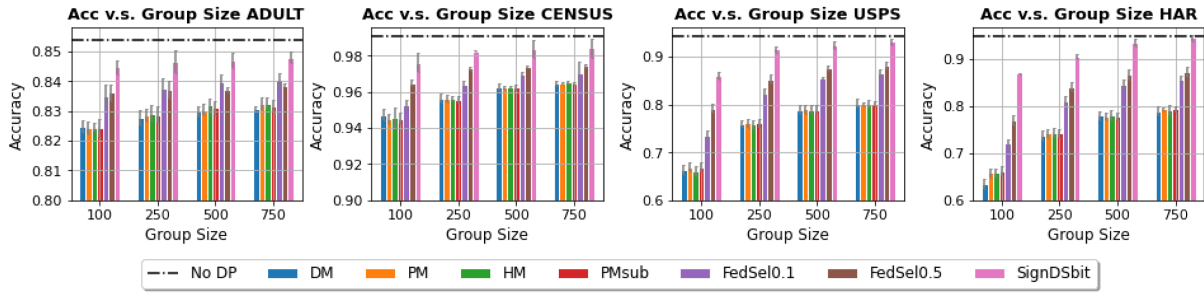


Fig. 5. Accuracy of LR models trained on structured datasets with different group sizes.

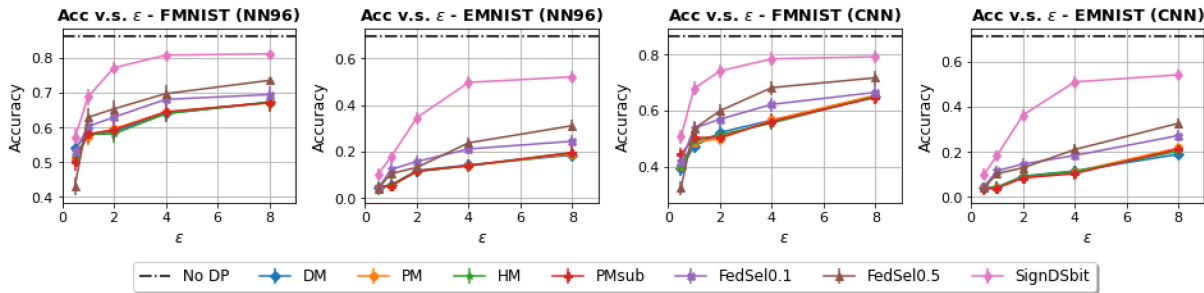


Fig. 6. Accuracy of NNs and CNNs trained on image datasets with different privacy budget ϵ .

size in comparison to SIGNDS-FL, especially with larger models. More specifically, for **USPS** and **HAR**, the accuracy change of our framework is less than 8%, but around $\sim 10\%$ – 15% for the baseline methods. This is due to the baseline algorithms adding random noise to the real dimension values, which alters the direction of the selected dimensions. Thus, the algorithms require a sufficiently large group size to reduce the impact of noise on the direction of the averaged local updates as well as on the model utility. In comparison, our SIGNDS-FL framework replaces the real dimension values with their corresponding sign values, which preserves the direction of the selected dimensions. Therefore, our framework shows higher robustness to different choices of group sizes in comparison to the baselines and can still achieve satisfactory performance even with small group sizes.

5.3 Performance in Complex Training Tasks

Next, we evaluate the performance of our framework on complex tasks. We respectively train the two-layer NNs and CNNs on both **FMNIST** and **EMNIST** datasets and compare the results with the baseline methods.

5.3.1 Model Accuracy Regarding the Privacy Level. First, we analyze the impact of the privacy budget ϵ on model accuracy. The results are presented in Figure 6, where we can see that the baseline algorithms based on LDP mean estimation cannot obtain an acceptable model accuracy for complex training tasks even with $\epsilon = 8$. This occurs since the injected noise in these algorithms gets proportionally larger for high-dimensional DNNs, which may require an extremely large ϵ to mitigate the randomness. Moreover, although FEDSEL shows a better performance, there is still a distinctive decrease of accuracy. In comparison, our framework achieves a notable accuracy improvement. For the **FMNIST** dataset, the accuracy loss is around 4% for two-layer NNs and CNNs model. For **EMNIST**, the accuracy loss is respectively around 10% and 15% for both models. It follows that saving the privacy budget for better dimension selection results and using the sign values

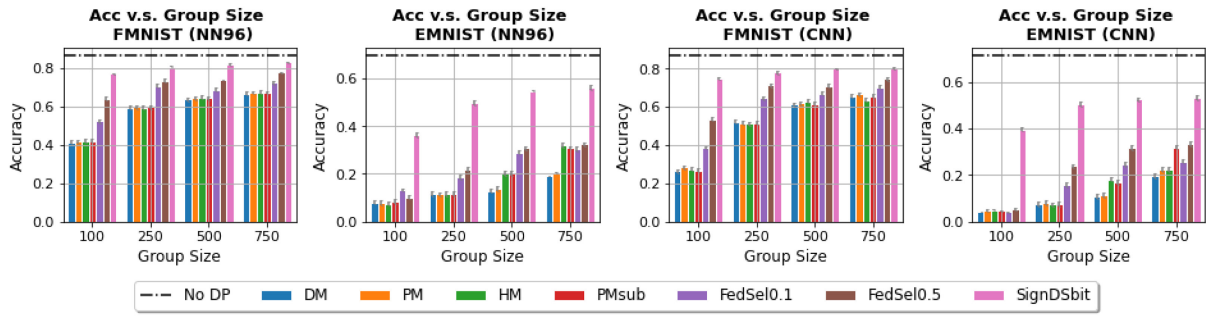


Fig. 7. Accuracy of NNs and CNNs trained on image datasets with different group size.

to preserve the model update direction can contribute to improving the model utility. Moreover, it can be observed that the models trained on **FMNIST** always have a higher accuracy than those trained on **EMNIST**. This is because **EMNIST** has 47 classes in comparison to **FMNIST**, which only has 10 classes. Thus, the increase of data variants leads to more difficulty for model training.

5.3.2 Model Accuracy Regarding the Group Size. We also analyze the impact of group size N on the accuracy of complex models. As in Section 5.2.2, we conduct experiments varying the group size $N \in \{100, 250, 500, 750\}$ under a privacy level of $\epsilon = 4$ and compare the model accuracy. The results are shown in Figure 7. It can be seen that the baseline algorithms cannot yield negligible accuracy loss for the complex models even with an increase in group size. However, our proposed method consistently outperforms the baseline methods under different group sizes. The model accuracy can be further improved by $\sim 1\%$ – 5% when increasing the group size from 250 to larger than 500. The results demonstrate that the proposed **SIGNDS-FL** framework can effectively support the complex training tasks with high-dimension DNNs.

5.4 Performance Improvement with Multi-dimension Selection

Although our framework shows significant accuracy improvements for training DNNs in comparison to the baseline methods, there is still an obvious gap in model accuracy between the private training setting and the non-private setting. This may be due to the fact that we only select one dimension for each local update, which slows down the model convergence and thus results in a decrease in model accuracy. In the following, we aim to explore whether our multi-dimension selection algorithm **EM-MDS** can further help improve model accuracy. We respectively train NNs and CNNs under **SIGNDS-FL** using the **EM-MDS** algorithm with h , namely the number of selected dimensions, to be 3 and 5. We choose different privacy budgets $\epsilon \in \{1, 4, 8, 12\}$ in the experiments. In addition, we also analyze the model accuracy when using the naive multi-dimension selection approach, which simply repeats the single dimension selection algorithm h times with a privacy budget of ϵ/h for each time (as mentioned in Section 4.3.2).

5.4.1 Comparison of Model Accuracy Regarding the Privacy Level. In Figure 8, we report the model accuracy of using different dimension selection methods under various privacy levels. It can be seen that the naive multi-dimension selection method may not contribute to model utility under high privacy regimes. In particular, when $\epsilon \leq 4$, selecting multiple dimensions via the naive method causes a distinctive decrease of accuracy in comparison to the single-dimension selection results. In contrast, the proposed **EM-MDS** algorithm can achieve similar or even slightly better accuracy under the same ϵ . This aligns with the utility analysis in Figure 3. More specifically, the naive method evenly splits the privacy budget for each output dimension. When the total ϵ is small, each dimension may be allocated with an insufficient privacy budget and the output set may only

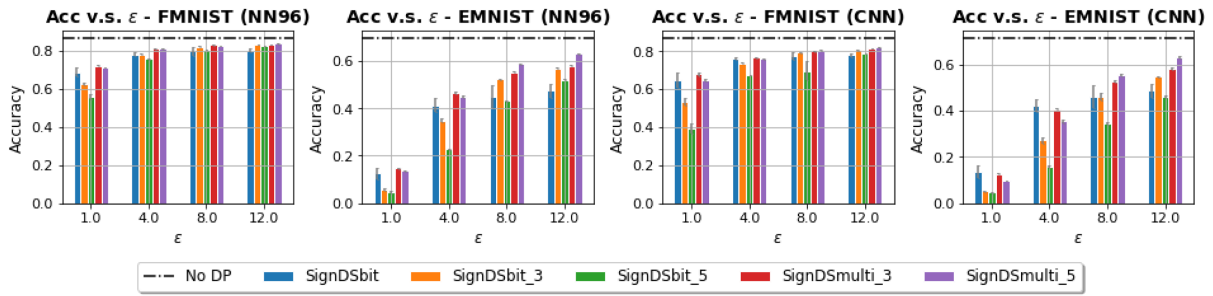


Fig. 8. Comparison of model accuracy regarding different dimension selection strategies.

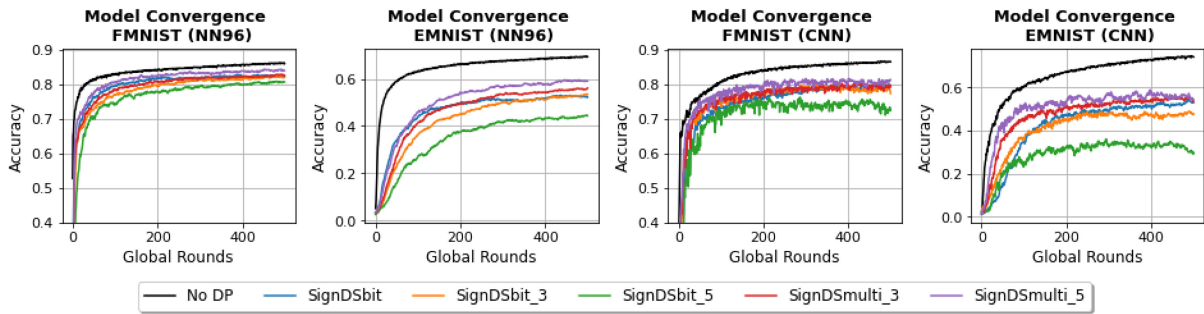


Fig. 9. Comparison of model convergence regarding different dimension selection strategies.

obtain a low top- k ratio. In comparison, EM-MDS considers all the combinations of the output set and assigns higher probabilities to the combinations with more top- k dimensions. Thus, the algorithm can achieve a higher top- k ratio in the output set and a better model utility. Moreover, there is an obvious accuracy improvement with EM-MDS as ϵ increases. For example, with $\epsilon \geq 8$, EM-MDS gets better accuracy with $h = 5$ in comparison to the single-dimension selection algorithm while the naive approach still shows a decrease in accuracy. As a result, with the same privacy guarantee, our EM-MDS can more efficiently utilize the privacy budget for selecting more dimensions in comparison to the naive approach and can further improve model utility. In comparison to the results in Section 5.3.1, with $\epsilon \geq 8$, EM-MDS achieves around a 2% increase of accuracy for the **FMNIST** dataset and more than an 8% accuracy improvement for the **EMNIST** dataset.

5.4.2 Comparison of Model Convergence. We further visualize the model accuracy regarding the global rounds to investigate whether the EM-MDS algorithm can help speed up model convergence. We compare the results of both models and datasets under the privacy level $\epsilon = 8$ and present the results of different dimension selection strategies in Figure 9. Intuitively, model convergence faces a tradeoff between the output size h and the randomness during the dimension selection process. As discussed above, a larger output size may not always contribute to model convergence since it may result in an increase in randomness in dimension selection when the privacy budget is insufficient. However, when ϵ is large enough, an increase in h will speed up model convergence since more information of the local updates is used during training. Similarly to the results in Figure 8, using the naive approach with $h = 5$ triggers an even slower convergence speed in comparison to the single-dimension selection strategy, which is because of the insufficient privacy budget for dimension selection. In contrast, EM-MDS with $h = 5$ shows the fastest convergence over all the selection strategies. This indicates that EM-MDS can help improve model convergence and thus requires fewer communication rounds for model training.

5.5 Analysis of Results

From the results in Sections 5.2 and 5.3, it can be seen that the LDP mean estimation based baselines are effective for low-dimensional models but suffer from large accuracy loss for high-dimensional models. This is because the injected noise in these algorithms grows proportionally with the model size, which causes significant utility loss. Moreover, although FEDSEL can mitigate this *dimension-dependency* problem for LR models, it still poses an obvious accuracy loss for DNN models. On the contrary, the SIGNDS-FL framework achieves better accuracy for both simple LR models and complex DNNs. In addition, we observe that the baseline algorithms are more sensitive to the group size and require a sufficiently large group size for mitigating the impact of noise on model convergence. In comparison, SIGNDS-FL can still obtain acceptable model accuracy even with small group sizes. The results demonstrate the viability of SIGNDS-FL in real-life applications. On the one hand, the framework can effectively support not only simple LR models but also complex DNNs models. On the other hand, the framework can also be extended to scenarios with only a small number of local clients.

The results in Section 5.4 illustrate how the proposed EM-MDS algorithm can further improve the accuracy of complex models. The results show that the naive multi-dimension selection algorithm may have a negative impact on model accuracy, especially when ϵ is small. This is because the naive approach evenly splits the privacy budget across each output dimension. This causes each dimension to be allocated with an insufficient privacy budget and thus a low probability to be selected from the top- k dimensions. In contrast, with the same privacy budget, our EM-MDS algorithm follows the idea of exponential mechanism [33] and assigns higher probabilities to the output combinations with higher top- k ratios. Therefore, the top- k dimensions will be more likely to be selected, which contributes to a better model utility. With the increase of ϵ , the model accuracy of using EM-MDS is better than single-dimension SIGNDS-FL. The results demonstrate that the EM-MDS algorithm can be applied to improve the performance of SIGNDS-FL in training high-dimensional models, which further strengthens the framework's practicality in real-world applications.

6 DISCUSSION

In this section, we will first present potential applications of the proposed framework. Then, we will discuss the limitations and directions for future work.

6.1 Potential Applications

In this article, we introduced an efficient and privacy-preserving LDP-FL framework with a satisfactory privacy-utility balance. The framework enjoys significantly lower communication and computation costs than crypto-based solutions and thus can be applied in a variety of large-scale privacy-sensitive applications. For example, in mobile edge computing applications, one can use the framework for privately training user profile models for providing keyboard suggestions [48] or ranking browser history suggestions [16]. In computer vision applications [30], the framework can be applied to train object detection models while ensuring the privacy of the local images. Besides the existing applications of supervised learning tasks, it is also possible to extend the framework to support other ML tasks, such as unsupervised learning, reinforcement learning, and semi-supervised learning [22]. For instance, one can apply the framework to privacy-preserving distributed synthetic data generation scenarios [20, 39], where the generative models are trained privately using the proposed framework to learn the distribution of local data and then to generate synthetic data on the server side. This can be considered as an alternative approach for privacy-preserving data collection. In addition, the framework can also be integrated into private

semi-supervised learning applications, where both labeled and unlabeled local data are utilized for updating the global model [46]. It should be noted that despite the large variety of possible applications and ML model scenarios, the main idea of the LDP-FL protocol remains unchanged.

6.2 Limitations and Future Work

Although our proposed framework shows notable performance improvements in comparison to the baseline methods, there are still limitations regarding model utility and communication efficiency.

As for data utility, while our proposed SIGNDS-FL framework shows significantly better model utility compared to the baseline algorithms, there is still an obvious accuracy loss for complex models. Therefore, it is desirable to further improve the proposed framework to better support complex and high-dimensional models. On the one hand, an important future work is to conduct comprehensive theoretical convergence analysis to have a better understanding of the performance of the current algorithm. On the other hand, one can also consider whether existing LDP mechanisms for frequency estimation and item-set mining can be applied for the private dimension selection process. Moreover, in real-life implementations, it might be helpful to first use auxiliary data to pre-train the model on the server side and then to fine-tune the model parameters using the LDP-FL framework, so as to mitigate the impact of the randomness during FL training.

As for communication efficiency, in comparison to the original FL approach, our proposed framework significantly reduces the *uplink* communication cost by only transmitting a set of dimension indices and the corresponding sign value to the server. However, the *downlink* communication may also become a bottleneck as the model size increases, especially for large-scale FL scenarios. A few recent studies [6, 7, 21] have proposed dropout-based and model pruning-based solutions for reducing the size of the model broadcast to the client side. Meanwhile, training the smaller models on the local side can also help decrease the computational cost of the client devices. Therefore, exploring whether such techniques can be integrated into our framework to further improve communication and computational efficiency will be an important future work.

7 CONCLUSION

Federated learning has recently attracted increased attention due to its computational efficiency and privacy benefits. However, the naive FL framework is still vulnerable to a number of privacy attacks as the exchanged local updates and the final model can still reveal sensitive information of clients' local data. LDP, as a strong notion of privacy, has been recently applied to the local side of federated learning to provide privacy guarantees for clients' local data. However, previous LDP-FL solutions cannot provide satisfactory outcomes for high-dimensional models.

In this article, we propose SIGNDS-FL, an efficient and privacy-preserving federated learning framework based on local differentially-private dimension selection. The main idea is to privately select a set of "important" dimensions of the local updates under strict LDP guarantees and to construct sparse privatized local updates using sign values, which are randomly determined by the clients. Moreover, we propose EM-MDS, an efficient multi-dimension selection algorithm that can better utilize the privacy budget and contribute to improving model convergence and accuracy. We evaluate the framework on many real-world structured and image datasets using simple LR models as well as DNNs. Extensive experimental results demonstrate that our framework significantly outperforms the previous LDP-FL solutions and enjoys a favorable utility-privacy balance.

ACKNOWLEDGEMENTS

We thank the reviewers and editors for their constructive comments and valuable suggestions for improving this manuscript.

REFERENCES

- [1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery. 308–318.
- [2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21st European Symposium on Artificial Neural Networks*. 437–442.
- [3] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. 2020. Generative models for effective ML on private, decentralized datasets. In *Proceedings of the 8th International Conference on Learning Representations*.
- [4] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. 2018. SIGNSGD: Compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. PMLR, 559–568.
- [5] Abhishek Bhowmick, John C. Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. 2018. Protection against reconstruction and its applications in private federated learning. *CoRR* abs/1812.00984.
- [6] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. 2021. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. In *Proceedings of the IEEE Conference on Computer Communications Workshops*. IEEE, 1–6.
- [7] Sebastian Caldas, Jakub Konečný, H. Brendan McMahan, and Ameet Talwalkar. 2018. Expanding the reach of federated learning by reducing client resource requirements. *CoRR* abs/1812.07210.
- [8] Kamalika Chaudhuri and Claire Monteleoni. 2008. Privacy-preserving logistic regression. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems*. Curran Associates, Inc., 289–296.
- [9] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: An extension of MNIST to handwritten letters. *CoRR* abs/1702.05373.
- [10] Ivan Damgård and Mads Jurik. 2001. A generalisation, a simplification and some applications of paillier’s probabilistic public-key system. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, Lecture Notes in Computer Science, Vol. 1992. Springer, 119–136.
- [11] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. Retrieved from <http://archive.ics.uci.edu/ml>.
- [12] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2018. Minimax optimal procedures for locally private estimation. *J. Am. Stat. Assoc.* 113, 521 (2018), 182–201.
- [13] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. 2009. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*. ACM, 381–390.
- [14] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (2014), 211–407.
- [15] Mehmet Emre Gursoy, Acar Tamersoy, Stacey Truex, Wenqi Wei, and Ling Liu. 2021. Secure and utility-aware data collection with condensed local differential privacy. *IEEE Trans. Depend. Sec. Comput.* 18, 5 (2021), 2365–2378.
- [16] Florian Hartmann, Sunah Suh, Arkadiusz Komarzewski, Tim D. Smith, and Ilana Segall. 2019. Federated learning for ranking browser history suggestions. *CoRR* abs/1911.11807.
- [17] Jonathan J. Hull. 1994. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.* 16, 5 (1994), 550–554.
- [18] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. 2018. Distributed learning without distress: Privacy-preserving empirical risk minimization. In *Advances in Neural Information Processing Systems*. Curran Associates Inc., 6343–6354.
- [19] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. Comprehensive analysis of privacy leakage in vertical federated learning during prediction. *Proc. Priv. Enhanc. Technol.* 2022, 2 (2022), 263–281.
- [20] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. Privacy-preserving high-dimensional data collection with federated generative autoencoder. *Proc. Priv. Enhanc. Technol.* 2022, 1 (2022), 481–500.
- [21] Yuang Jiang, Shiqiang Wang, Bong-Jun Ko, Wei-Han Lee, and Leandros Tassioulas. 2019. Model pruning enables efficient federated learning on edge devices. *CoRR* abs/1909.12326.
- [22] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D’Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancréde Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha

- Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* 14, 1–2 (2021), 1–210.
- [23] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.
- [24] Ron Kohavi. 1996. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the 2nd SIGKDD Conference on Knowledge Discovery and Data Mining*. AAAI Press, 202–207.
- [25] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* 1, 4 (1989), 541–551.
- [26] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Sign. Process. Mag.* 37, 3 (2020), 50–60.
- [27] Wenqi Li, Fausto Milletari, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M. Jorge Cardoso, and Andrew Feng. 2019. Privacy-preserving federated brain tumour segmentation. In *Proceedings of the 10th International Workshop on Machine Learning in Medical Imaging*, Lecture Notes in Computer Science, Vol. 11861. Springer, 133–141.
- [28] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated learning in Mobile edge networks: A comprehensive survey. *IEEE Commun. Surv. Tutor.* 22, 3 (2020), 2031–2063.
- [29] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. 2020. FedSel: Federated SGD under local differential privacy with top-k dimension selection. In *Proceedings of the 25th International Conference on Database Systems for Advanced Applications*, Lecture Notes in Computer Science, Vol. 12112. Springer, 485–501.
- [30] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. 2020. FedVision: An online visual object detection platform powered by federated learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. AAAI Press, 13172–13179.
- [31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, 1273–1282.
- [32] Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning differentially private recurrent language models. In *Proceedings of the 6th International Conference on Learning Representations*.
- [33] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 94–103.
- [34] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 739–753.
- [35] Thông T. Nguyễn, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. 2016. Collecting and analyzing data from smart device users with local differential privacy. *CoRR* abs/1606.05053.
- [36] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data. In *Proceedings of the 5th International Conference on Learning Representations*.
- [37] John C. Platt. 1999. *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*. MIT Press, Cambridge, MA, 185–208.
- [38] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1310–1321.
- [39] Aleksei Triastcyn and Boi Faltings. 2020. Federated generative privacy. *IEEE Intell. Syst.* 35, 4 (2020), 50–57.
- [40] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. ACM, 1–11.
- [41] Stacey Truex, Ling Liu, Ka Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the 3rd International Workshop on Edge Systems, Analytics and Networking, (EdgeSys@EuroSys’20)*. ACM, 61–66.
- [42] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and analyzing multidimensional data with local differential privacy. In *Proceedings of the 35th IEEE International Conference on Data Engineering*. IEEE, 638–649.
- [43] Shaowei Wang, Liusheng Huang, Yiwen Nie, Pengzhan Wang, Hongli Xu, and Wei Yang. 2018. PrivSet: Set-valued data analyses with locale differential privacy. In *Proceedings of the IEEE Conference on Computer Communications*. IEEE, 1088–1096.

- [44] Stanley L. Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *J. Am. Stat. Assoc.* 60, 309 (1965), 63–69.
- [45] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *CoRR* abs/1708.07747.
- [46] Dong Yang, Ziyue Xu, Wenqi Li, Andriy Myronenko, Holger R. Roth, Stephanie A. Harmon, Sheng Xu, Baris Turkbey, Evrim Turkbey, Xiaosong Wang, Wentao Zhu, Gianpaolo Carrafiello, Francesca Patella, Maurizio Cariatì, Hirofumi Obinata, Hitoshi Mori, Kaku Tamura, Peng An, Bradford J. Wood, and Daguang Xu. 2021. Federated semi-supervised learning for COVID region segmentation in chest CT using multi-national data from China, Italy, Japan. *Med. Image Anal.* 70 (2021), 101992.
- [47] Liu Yang, Ben Tan, Vincent W. Zheng, Kai Chen, and Qiang Yang. 2020. Federated recommendation systems. In *Federated Learning—Privacy and Incentive*. Lecture Notes in Computer Science, Vol. 12500. Springer, 225–239.
- [48] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: Improving Google keyboard query suggestions. *CoRR* abs/1812.02903.
- [49] Andrew Chi-Chih Yao. 1982. Protocols for secure computations (extended abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 160–164.
- [50] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via Bayesian networks. *ACM Trans. Datab. Syst.* 42, 4 (2017), 25:1–25:41.
- [51] Yang Zhao, Jun Zhao, Mengmeng Yang, Teng Wang, Ning Wang, Lingjuan Lyu, Dusit Niyato, and Kwok-Yan Lam. 2021. Local differential privacy-based federated learning for Internet of Things. *IEEE IoT J.* 8, 11 (2021), 8836–8853.
- [52] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems*. Curran Associates, Inc., 14747–14756.

Received March 2021; revised January 2022; accepted February 2022

 **CC BY-NC 4.0**

ATTRIBUTION- NONCOMMERCIAL 4.0 INTERNATIONAL

Deed

Canonical URL : <https://creativecommons.org/licenses/by-nc/4.0/>

[See the legal code](#)

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give **appropriate credit**, provide a link to the license, and **indicate if changes were made**. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for **commercial purposes** .

No additional restrictions — You may not apply legal terms or **technological measures** that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable **exception or limitation** .

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as **publicity, privacy, or moral rights** may limit how you use the material.

Notice

This deed highlights only some of the key features and terms of the actual license. It is not a license and has no legal value. You should carefully review all of the terms and conditions of the actual license before using the licensed material.

Creative Commons is not a law firm and does not provide legal services. Distributing, displaying, or linking to this deed or the license that it summarizes does not create a lawyer-client or any other relationship.

Creative Commons is the nonprofit behind the open licenses and other legal tools that allow creators to share their work. Our legal tools are free to use.

- [Learn more about our work](#)
- [Learn more about CC Licensing](#)
- [Support our work](#)

- [Use the license for your own material.](#)
- [Licenses List](#)
- [Public Domain List](#)

Footnotes

appropriate credit — If supplied, you must provide the name of the creator and attribution parties, a copyright notice, a license notice, a disclaimer notice, and a link to the material. CC licenses prior to Version 4.0 also require you to provide the title of the material if supplied, and may have other slight differences.

- [More info](#)

indicate if changes were made — In 4.0, you must indicate if you modified the material and retain an indication of previous modifications. In 3.0 and earlier license versions, the indication of changes is only required if you create a derivative.

- [Marking guide](#)
- [More info](#)

commercial purposes — A commercial use is one primarily intended for commercial advantage or monetary compensation.

- [More info](#)

technological measures — The license prohibits application of effective technological measures, defined with reference to Article 11 of the WIPO Copyright Treaty.

- [More info](#)

exception or limitation — The rights of users under exceptions and limitations, such as fair use and fair dealing, are not affected by the CC licenses.

- [More info](#)

publicity, privacy, or moral rights — You may need to get additional permissions before using the material as you intend.

- [More info](#)

[Contact](#)

[Newsletter](#)

[Privacy](#)

[Policies](#)

[Terms](#)

CONTACT US

Creative Commons PO Box 1866,
Mountain View, CA 94042

info@creativecommons.org

+1 415 429 6753

SUBSCRIBE TO OUR NEWSLETTER

Your email

SUBSCRIBE

Except where otherwise **noted**, content on this site is licensed under a **Creative Commons Attribution 4.0 International license**.
Icons by **Font Awesome**.

SUPPORT OUR WORK

Our work relies on you! Help us keep the Internet free and open.

DONATE NOW

2 Privacy-Preserving High-dimensional Data Collection with Federated Generative Autoencoder

Authors	Xue Jiang ^{1,2} (xue.jiang@tum.de) Xuebing Zhou ² (xuebing.zhou@huawei.com) Jens Grossklags ¹ (jens.grossklags@in.tum.de)
	¹ Technical University of Munich (TUM), Boltzmannstraße 3, 85748 Garching, Germany ² Huawei Munich Research Center, Riesstraße 25, 80992 Munich, Germany
Type	Journal
Outlet	Proceedings on Privacy Enhancing Technologies, Volume 2022 ³
Ranking	CORE 2021 ⁴ : A
Status	Published
DOI	https://doi.org/10.2478/popets-2022-0024
Citation	Jiang, X., Zhou, X., & Grossklags, J. (2022). Privacy-Preserving High-dimensional Data Collection with Federated Generative Autoencoder. Proceedings on Privacy Enhancing Technologies, 2022(1), 481-500.
Copyright	This work is published under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 (CC BY-NC-ND 3.0) License ⁵ .
Author Contributions	Xue Jiang developed the core idea for the paper and took primary responsibility for designing the methodology, acquiring data, implementing experiments, evaluating results, and drafting the manuscript. Xuebing Zhou and Jens Grossklags provided valuable feedback and suggestions on the methodology and experiments, and assisted in reviewing and enhancing the manuscript.

³<https://www.petsymposium.org/popets/2022/>

⁴<http://portal.core.edu.au/conf-ranks/1442/>

⁵<https://creativecommons.org/licenses/by-nc-nd/3.0/>

Xue Jiang*, Xuebing Zhou, and Jens Grossklags

Privacy-Preserving High-dimensional Data Collection with Federated Generative Autoencoder

Abstract: Business intelligence and AI services often involve the collection of copious amounts of multi-dimensional personal data. Since these data usually contain sensitive information of individuals, the direct collection can lead to privacy violations. Local differential privacy (LDP) is currently considered a state-of-the-art solution for privacy-preserving data collection. However, existing LDP algorithms are not applicable to high-dimensional data; not only because of the increase in computation and communication cost, but also poor data utility.

In this paper, we aim at addressing the *curse-of-dimensionality* problem in LDP-based high-dimensional data collection. Based on the idea of machine learning and data synthesis, we propose DP-FED-WAE, an efficient privacy-preserving framework for collecting high-dimensional categorical data. With the combination of a generative autoencoder, federated learning, and differential privacy, our framework is capable of privately learning the statistical distributions of local data and generating high utility synthetic data on the server side without revealing users' private information. We have evaluated the framework in terms of data utility and privacy protection on a number of real-world datasets containing 68–124 classification attributes. We show that our framework outperforms the LDP-based baseline algorithms in capturing joint distributions and correlations of attributes and generating high-utility synthetic data. With a local privacy guarantee $\epsilon = 8$, the machine learning models trained with the synthetic data generated by the baseline algorithm cause an accuracy loss of 10% ~ 30%, whereas the accuracy loss is significantly reduced to less than 3% and at best even less than 1% with our framework. Extensive experimental results demonstrate the capability and efficiency of our framework in synthesizing high-dimensional data while striking a satisfactory utility-privacy balance.

Keywords: high-dimensional data collection, local differential privacy, federated learning, generative models

DOI 10.2478/popets-2022-0024

Received 2021-05-31; revised 2021-09-15; accepted 2021-09-16.

1 Introduction

With the rapid development of network and computer technologies, large and diverse quantities of multi-dimensional person-specific data are frequently generated on local devices such as smartphones and IoT sensors. These data usually contain rich information of univariate and multivariate (joint) distributions describing user profiles, which is valuable for data analysts to explore the hidden correlations and patterns of data from different perspectives and to obtain a better understanding of the characteristics of user groups. For instance, a digital healthcare application may utilize users' physical information (*i.e.*, temperature, blood pressure, activity signals, *etc.*) for health monitoring and disease predictions, while an online shopping website may take users' age, gender, and purchase history for providing suitable product recommendations. In principle, the more dimensions the data consist of, the more information can be used for describing an individual user; thus, the more accurate the decision-making system can be. Therefore, the collection of multi-dimensional data can be of significant help for companies and organizations in designing and building effective business intelligence & AI services.

However, since the data are generated based on individuals' ongoing behaviors, the direct collection can reveal sensitive information about them and lead to severe privacy problems (see, for example, [7, 12]). Local differential privacy (LDP) [31], as a state-of-the-art data anonymization mechanism, has been recently deployed

***Corresponding Author: Xue Jiang:** Technische Universität München, Huawei Technologies Düsseldorf GmbH, E-mail: xue.jiang@tum.de

Xuebing Zhou: Huawei Technologies Düsseldorf GmbH, E-mail: Xuebing.Zhou@huawei.com

Jens Grossklags: Technische Universität München, E-mail: jens.grossklags@tum.de

by major technology organizations such as Apple [14], Google [23], and Microsoft [15] for privacy-preserving data collection. By locally randomizing the user data before sending it to the server, the LDP algorithms ensure that the server cannot access the original user data, but is able to learn the population’s overall statistics. However, prior research on LDP-based data collection mainly focuses on one-dimensional statistics, such as frequency estimation [14, 23], heavy-hitter identification [6, 11], and itemset mining [43, 54], *etc.* But since the attributes in multi-dimensional data are usually correlated, the server is particularly interested in learning the correlations and joint distributions among attributes.

Directly applying the above-mentioned LDP algorithms for estimating the joint distributions of multi-dimensional data faces a foundational problem: the *curse-of-dimensionality*. The domain size increases exponentially with data dimensionality, which will lead to extremely large communication cost and storage complexity, as well as a significant degradation in data utility. To reduce the large communication overhead, Fanti et al. [24] proposed to separately collect data of each dimension under LDP and to estimate the joint distributions using expectation maximization (EM). However, the algorithm only supports estimates of the joint distribution of two attributes. Further, Ren et al. [44] introduced LOPUB, which splits the c -dimensional data into k -dimensional clusters ($k < c$) using dependence graphs and estimates k -way joint distributions via an EM-based and Lasso regression-based approach. However, the algorithm still suffers from high computational complexity and low data utility when k is large. Based on these facts, alternative solutions for privacy-preserving high-dimensional data collection are still greatly needed.

Recently, data synthesis has been considered a promising approach for addressing data privacy issues in business intelligence & AI services. With the strong capabilities of characterizing the joint distributions and correlations of high-dimensional data, deep generative models are increasingly used for generating high-utility and low-sensitivity synthetic data. In this paper, we follow the idea of data synthesis and propose DP-FED-WAE, a privacy-preserving framework for high-dimensional categorical data collection. Different from prior work on differentially private synthetic data generation algorithms [41, 49, 57], which mainly focuses on the centralized setting where the real data are already collected by the server, our framework conducts the data synthesis *without* collecting real local data. The main idea is to train a (generative) Wasserstein Autoencoder [48] (WAE) under the federated learning [38] (FL) set-

ting to learn the distributions of the high-dimensional local data and then to generate high-quality synthetic data on the cloud server. Moreover, we propose a novel local randomization algorithm SIGNDS, which is applied on a client’s local updates to prevent potential privacy leakages in FL. The algorithm provides a strict ϵ -LDP privacy guarantee for any client’s local dataset.

In comparison with previous data collection approaches, our framework shows significant advantages in both *data utility* and *privacy protection*. As for data utility, the WAE model has a strong capability in capturing correlations and joint distributions of high-dimensional data and generating high-utility synthetic data. The generated synthetic data can be easily scaled up to replace the real data for data analysis and AI training tasks. As for privacy, the generated data are fully synthetic, which effectively reduce risks of re-identification attacks or attribute disclosure [41]. Moreover, training the WAE model under the LDP-FL setting not only avoids the collection of raw user data but also provides comprehensive privacy guarantees to the framework. Our contributions can be summarized as follows:

- We propose DP-FED-WAE, an efficient and privacy-preserving framework that effectively combines a generative autoencoder, FL, and DP for collecting high-dimensional categorical data. Based on the idea of data synthesis, the framework effectively solves the *curse-of-dimensionality* problem in LDP-based data collection solutions. The synthetic data preserves high utility and can replace real data for data mining and AI training tasks.
- We further propose a novel local randomization algorithm SIGNDS, which perturbs clients’ local updates and prevents potential privacy leakages in FL. We prove that the algorithm follows a strict ϵ -LDP definition and provides a strong local privacy guarantee to any client’s local data.
- We have implemented our framework and evaluated the performance in terms of data utility and privacy protection using real-world datasets containing 68–124 classification attributes. Through comparison with the LDP-based algorithms, we show that the synthetic data generated by our framework always preserve much closer joint distributions and correlations to real data. Also, the accuracy loss of the model trained with synthetic data generated by our framework is significantly reduced in comparison to the baseline method. With a local privacy guarantee $\epsilon = 8$, we reduced the accuracy loss from 10% ~ 30% to less than 3% and at best even less than 1%. Extensive evaluation experiments show

that our framework has outperforming capability and efficiency in collecting high-dimensional data while striking a satisfactory utility-privacy balance.

2 Problem Statement

In this paper, we consider a scenario where a large number of local users hold high-dimensional personal data. A central server aims to estimate the joint distributions of these high-dimensional data and to generate similar synthetic data for data analysis or designing new AI services. Here, we assume the server to be *honest-but-curious*, who follows the system protocols but tries to infer sensitive information of local users. Thus, to protect local privacy, we require the server not to have access to raw local data but only anonymized versions of data or their feature representations.

Assume that there are N local users, each holding one or more data records, which have c attributes $\mathcal{W} = \{w_i | i = 1, \dots, c\}$. Each attribute w_i has a domain Ω_i of possible values. The full domain for the c -dimensional data record is denoted as $\Omega = \Omega_1 \times \dots \times \Omega_c$, where \times is the Cartesian product. The total domain size is $|\Omega| = \prod_{i=1}^c |\Omega_i|$, which increases exponentially with data dimensionality c . Based on the notation above, the problem can then be formulated as follows: given a c -dimensional private dataset X distributed among N local users, a central server aims to generate a synthetic dataset X' without access to raw data X . The synthetic dataset X' has the same attributes \mathcal{W} , and preserves similar joint distributions of X , namely

$$P_{X'}(w_1 \dots w_c) \approx P_X(w_1 \dots w_c). \quad (1)$$

It can be further derived that the synthetic data X' also preserves m -way joint distributions as X . Namely, given an m -way attribute combination $\omega \subseteq \mathcal{W}$, we have

$$P_{X'}(\omega) \approx P_X(\omega). \quad (2)$$

3 Proposed Solution

As discussed previously, existing LDP algorithms are impractical for collecting high-dimensional data due to both high computation and communication cost and poor data utility (e.g., [23, 24, 44]). In order to solve the curse-of-dimensionality problem, we propose DP-FED-WAE, an efficient privacy-preserving framework for collecting high-dimensional categorical data. The frame-

work contains three main components: a generative autoencoder, FL, and DP. Following the idea of recent data synthesis techniques, the framework utilizes generative autoencoders to learn the statistical distributions and correlations of high-dimensional user data and then to generate high-utility synthetic data on the server side. Different from existing works of synthetic data generation where the real data are already available to the server (e.g., [41, 49, 57]), our framework focuses on the scenario where the real data are distributed on local devices. Therefore, we propose to train the generative autoencoder under the FL setting, which only exchanges model parameters during the training process and keeps the raw user data inaccessible to the server. Furthermore, we incorporate DP during the training process in order to prevent potential privacy leakages in FL. In comparison to the previous DP-FL frameworks that add DP noise on the server side [5, 39], we propose a novel local randomization algorithm that perturbs the local updates before uploading them to the server. This ensures that the server cannot gain access to the real local updates and efficiently prevents local privacy leakages. We prove that the randomization algorithm follows a strict LDP definition and provides a strong *local privacy guarantee* to each client's local dataset.

The overall workflow is presented in Figure 1, which is processed in the following sequence:

1. The local clients first process the original categorical data into a numerical form, which can be used for training the generative autoencoder. At the same time, the server defines the structure of the generative autoencoder based on the dimensionality of local data and initializes the model.
2. The generative autoencoder is then collaboratively trained under the FL mechanism that is incorporated with LDP to achieve strict privacy guarantees.
3. After the model gets trained, the decoder is extracted for generating synthetic data. The generated data will be finally converted back to categorical form and used for data mining and building machine learning models.

3.1 Data Pre-Processing and Design of the Generative Model

Since the original data are categorical and cannot be directly processed by machine learning models, we first convert the data into numerical form. Here, we use a *one-hot encoding* to encode each categorical attribute into a binary vector. Each entry in the binary vector

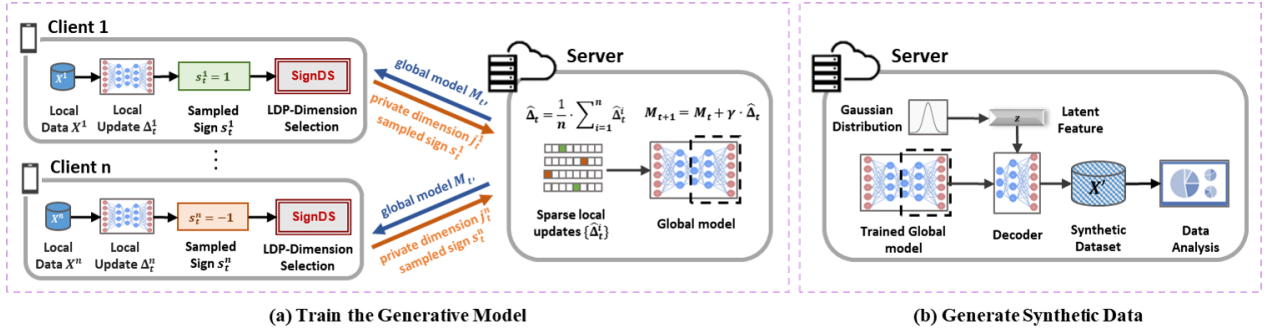


Fig. 1. Overview of the DP-FED-WAE framework. The generative Wasserstein Autoencoder is first trained under the federated setting, which learns the distributions of real local data. An LDP algorithm SIGNDS is applied to the local updates to provide strict local privacy guarantees. After the model is trained, the *decoder* part is used to generate high-utility synthetic data. The generated data will be used for data mining and building AI services.

stands for a unique attribute value and the entry of the given value is set to 1 while all the others are set to 0. Finally, we concatenate all the binary vectors into one vector as the input data for the generative model.

In this paper, we have chosen the Wasserstein Autoencoder (WAE) as the generative model in our framework, which provides better data synthesis capability in comparison to the Variational Autoencoder (VAE) [33] and less training difficulty than the Generative Adversarial Network (GAN) [26]. As a variant from the family of autoencoders, WAE preserves the encoder-decoder architecture. The encoder Q_ϕ compresses the original high-dimensional input $x \sim P_x$ into the low-dimensional latent feature $z = Q_\phi(x)$ and the decoder G_θ maps z to the reconstructed output $x' = G_\theta(z)$, which is the same shape as x . The distance between the original input and the reconstructed output can be presented as $D_r(x, G_\theta(Q_\phi(x)))$. In addition, a regularizer term $D_z(q_z, p_z)$ is applied to measure the distance between the latent space distribution q_z and certain prior distribution p_z . The final objective function of the WAE model can thus be formulated as follows:

$$\mathcal{L}_{WAE} = \mathbb{E}_{x \sim P_x} [D_r(x, G_\theta(Q_\phi(x)))] + \lambda \cdot D_z(q_z, p_z), \quad (3)$$

where λ is a hyperparameter for balancing the two terms. The goal of training is to find an optimal set of parameters, which minimizes the distance between the inputs and outputs while restricting the latent space to follow the prior distribution.

We designed the WAE models with fully-connected hidden layers. We apply the *relu* activation on the output of each hidden layer for better training performance. Moreover, since the inputs are binary vectors, we use the *sigmoid* activation on the output layer, which restricts the output value within $[0,1]$. Then, we calculate

the binary cross-entropy of each input/output dimension and compute the average as the reconstruction distance $D_r(x, G_\theta(Q_\phi(x)))$. For the latent space distance $D_z(q_z, p_z)$, we use the standard Gaussian distribution as the prior distribution p_z and use the maximum mean discrepancy (MMD) to measure the distance between the latent space distribution q_z and p_z , as in [48]. Given a batch of data sampled from the two distributions, *i.e.*, $\{q^1, \dots, q^n\} \sim q_z$ and $\{p^1, \dots, p^n\} \sim p_z$, $D_z(q_z, p_z)$ can be empirically estimated as

$$D_z(q_z, p_z) = \frac{1}{n(n-1)} \sum_{i \neq j} \mathcal{K}(p^i, p^j) - \frac{2}{n^2} \sum_{i,j} \mathcal{K}(p^i, q^j) + \frac{1}{n(n-1)} \sum_{i \neq j} \mathcal{K}(q^i, q^j), \quad (4)$$

where $\mathcal{K}(x, y) = \frac{\kappa}{\kappa + \|x - y\|_2^2}$. Given d_z as the dimension of latent layer and σ_z as the scale of the prior distribution, $\kappa = 2d_z\sigma_z^2$. We choose λ equal to 1.

3.2 Training the Generative Model

Previous LDP-FL frameworks (*e.g.*, [19, 21, 52]) evenly split the privacy budget across dimensions and apply the perturbation independently. However, the per-dimension privacy budget becomes extremely small for high-dimensional models, which results in a significant increase of noise. A recent work [37] proposed a *two-stage* LDP-FL framework, which splits the privacy budget into a *dimension selection* (DS) stage and a *value perturbation* (VP) stage. In the DS stage, the local update is sorted by *absolute* value and one "important" dimension is privately selected from the top- k dimensions; in the VP stage, the value of the selected di-

Algorithm 1: SIGNDS

Input: $\Delta \in \mathbb{R}^d$: local update; k : size of the top- k set; ϵ : privacy budget; s : sampled sign

Output: j : selected dimension index

- 1: **if** $s = 1$ **then**
- 2: Select dimensions of k largest values in Δ to build the top- k dimension set \mathcal{S}_{topk}
- 3: **else**
- 4: Select dimensions of k smallest values in Δ to build the top- k dimension set \mathcal{S}_{topk}
- 5: **end if**
- 6: Sample a Bernoulli variable x such that $\Pr[x = 1] = \frac{e^{\epsilon \cdot k}}{d - k + e^{\epsilon \cdot k}}$
- 7: **if** $x = 1$ **then**
- 8: Randomly sample a dimension $j \in \{a \in \{1, \dots, d\} | a \in \mathcal{S}_{topk}\}$
- 9: **else**
- 10: Randomly sample a dimension $j \in \{a \in \{1, \dots, d\} | a \notin \mathcal{S}_{topk}\}$
- 11: **end if**
- 12: **Return** j

mension is perturbed. Finally, a sparse local update is constructed and returned to the server. Although [37] mitigated the dimension-dependency problem by only selecting one "important" dimension, the privacy budget is still consumed by the two stages. In high-privacy scenarios (where the privacy budget is small), each stage may therefore obtain only an insufficient privacy budget and cause large randomness.

Motivated by the limitations of [37], we propose a *sign-based dimension selection* algorithm SIGNDS, as presented in Algorithm 1. The main idea is to substitute the VP stage by assigning a constant value to the selected dimension. Since the parameter values may have different signs, we introduce an extra variable $s \in \{-1, 1\}$, which is randomly sampled by the client with equal probability. Then, given each local update Δ , we build the top- k dimension set \mathcal{S}_{topk} according to Δ 's real values and the sampled sign s : if $s = 1$, \mathcal{S}_{topk} is built with the dimensions of the k largest values; otherwise, it is built with the dimensions of the k smallest values. We refer the dimensions included in \mathcal{S}_{topk} as *top- k dimensions* and the rest as *non-top- k dimensions*. Then, a dimension index j is randomly sampled as follows:

$$j \in \begin{cases} \{a \in \{1, \dots, d\} | a \in \mathcal{S}_{topk}\} & w.p. \quad p \\ \{a \in \{1, \dots, d\} | a \notin \mathcal{S}_{topk}\} & w.p. \quad 1 - p \end{cases}, \quad (5)$$

Namely, the index j is sampled from the *top- k dimensions* with a probability of p and otherwise from the *non-top- k dimensions* with a probability of $1 - p$. We refer to p as the *top- k probability*. Finally, the dimension index j and the sampled sign value s are returned to the server. Since our algorithm does not return the dimension value to the server, we save the privacy budget for the value perturbation stage in [37]. With the same privacy level, we can now achieve less randomness and thus higher accuracy in dimension selection.

In the following, we provide the privacy guarantee and utility analysis of Algorithm 1.

Lemma 1. *Algorithm 1 satisfies ϵ -LDP when the top- k probability $p \leq \frac{e^{\epsilon \cdot k}}{d - k + e^{\epsilon \cdot k}}$.*

Proof. For each client, given the sampled sign s and any output dimension $j \in \{1, \dots, d\}$, let $\mathcal{S}_{topk}, \mathcal{S}'_{topk}$ be the top- k dimension set of any two possible local update vectors Δ and Δ' . Given the top- k probability p , the probability of sampling a dimension j from the top- k set and the non-top- k set are respectively $p \cdot \frac{1}{k}$ and $(1 - p) \cdot \frac{1}{d - k}$. Thus, when $p \leq \frac{e^{\epsilon \cdot k}}{d - k + e^{\epsilon \cdot k}}$ we have

$$\begin{aligned} \frac{\Pr[j|\Delta]}{\Pr[j|\Delta']} &= \frac{\Pr[j|\mathcal{S}_{topk}]}{\Pr[j|\mathcal{S}'_{topk}]} \leq \frac{\Pr[j|j \in \mathcal{S}_{topk}]}{\Pr[j|j \notin \mathcal{S}'_{topk}]} \\ &= \frac{p \cdot \frac{1}{k}}{(1 - p) \cdot \frac{1}{d - k}} \leq e^{\epsilon} \end{aligned} \quad (6)$$

which completes the proof. \square

In addition to the privacy guarantee, we are also interested in how to choose proper k and ϵ in order to achieve a certain top- k probability p . Let $\alpha = k/d$ be the ratio of the top- k parameters regarding the total number of parameters. An $\alpha = 1$ means to randomly select one dimension from the entire dimension group. Intuitively, the smaller α , the closer the parameter values of top- k dimensions to the real largest (or smallest) value and the better the model utility. We derive relations among ϵ , p , and α as follows:

Corollary 1. *With a fixed privacy budget ϵ , in order to achieve a probability p , α should satisfy $\alpha \geq \frac{p}{e^{\epsilon \cdot (1-p)} + p}$.*

Corollary 2. *With a fixed top- k ratio α , in order to achieve a probability p , ϵ should satisfy $\epsilon \geq \log \frac{p \cdot (1-\alpha)}{(1-p) \cdot \alpha}$.*

Proof. From Lemma 1, we have $p \leq \frac{e^{\epsilon \cdot k}}{d - k + e^{\epsilon \cdot k}} = \frac{e^{\epsilon \cdot \alpha}}{1 - \alpha + e^{\epsilon \cdot \alpha}}$. Thus, with a fixed ϵ , we have $\alpha \geq \frac{p}{e^{\epsilon \cdot (1-p)} + p}$; with a fixed α , we have $\epsilon \geq \log \frac{p \cdot (1-\alpha)}{(1-p) \cdot \alpha}$ \square

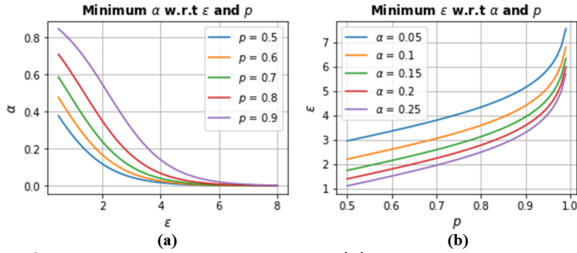


Fig. 2. Relations among ϵ , p , and α . (a): given privacy budget ϵ and the expected top- k probability p , the minimum top- k ratio α required. (b): given the expected top- k ratio α and top- k probability p , the minimum privacy budget ϵ required.

Corollary 1 states that with a fixed privacy budget ϵ , a smaller top- k ratio α leads to a decrease of top- k probability p . Moreover, given an expected top- k probability p and a predefined top- k ratio α , the minimum required privacy budget ϵ can be calculated using Corollary 2. We further visualize the relations of ϵ , p , and α in Figure 2. As shown in Figure 2a, in high-privacy scenarios (e.g., $\epsilon \leq 2$), the required top- k ratio α differs distinctly with the choices of p . Namely, we have to choose a large α in order to ensure that the index is more likely to be sampled from top- k dimensions. As ϵ increases (e.g., $\epsilon \geq 6$), α does not differ much regarding p . In other words, we can always achieve a high top- k probability even with a small top- k ratio. In Figure 2b, we further present the minimum ϵ under various α and p .

We now describe the overall training process presented in Algorithm 2. At each global round t , the server selects a group of n clients and broadcasts the current global model M_t . On the local side, each client i in the group trains the global model for several epochs with his local data X^i and computes the local update Δ_t^i . Then, the client randomly samples a sign $s_t^i \in \{-1, 1\}$ with equal probability and uses it along with the predefined privacy budget ϵ_r to privately select a dimension index j_t^i of the local update. Finally, s_t^i and j_t^i are returned to the server. After receiving the dimension j_t^i and the sampled sign s_t^i , the server builds a sparse local update $\hat{\Delta}_t^i$ and assigns s_t^i to the selected dimension. Since the selected dimension j_t^i satisfies the ϵ -LDP guarantee and the assigned sign value s_t^i is unrelated to the local data, according to DP's robustness to post-processing (Property 1 in Appendix A.2), the sparse local updates also satisfy ϵ -LDP. Finally, the server aggregates all the sparse local updates and updates the global model with a global learning rate γ . The updated global model M_{t+1} is distributed to local clients to start the next round.

Note that according to the sequential composition property (Property 2 in Appendix A.2), if the same client repeatedly participates in the training and sub-

Algorithm 2: Training the Generative Model

Input: $M_1 \in \mathbb{R}^d$: initial global model; n : number of per-round clients; E : number of local epochs; η : local learning rate; k : number of parameters in the top- k set of each local update; T : number of global aggregation rounds; γ : global learning rate; ϵ_r : per-round privacy budget

Output: Trained WAE model M

Server executes:

- 1: **for** global round $t = 1, \dots, T$ **do**
- 2: Randomly select a group of n clients
- 3: **for** client $i = 1, \dots, n$ in parallel **do**
- 4: Broadcast current global model M_t
- 5: Receive sampled sign and dimension $s_t^i, j_t^i = \mathbf{LocalUpdate}(M_t, E, \eta, \epsilon_r, k)$
- 6: Build sparse local update $\hat{\Delta}_t^i = \{0\}^d$ and set $\hat{\Delta}_t^i[j_t^i] = s_t^i$
- 7: **end for**
- 8: Aggregate local updates: $\hat{\Delta}_t = \frac{1}{n} \sum_{i=1}^n \hat{\Delta}_t^i$
- 9: Update global model $M_{t+1} = M_t + \gamma \cdot \hat{\Delta}_t$
- 10: **end for**
- 11: **Return** Global model $M = M_{T+1}$

LocalUpdate($M_t, E, \eta, \epsilon_r, k$):

// Run on the client side

- 13: Initialize local model $M_t^i \leftarrow M_t$
 - 14: **for** epoch $e = 1, \dots, E$ **do**
 - 15: $M_t^i = M_t^i - \eta \cdot \nabla \mathcal{L}(M_t^i, X^i)$
 - 16: **end for**
 - 17: Calculate local update: $\Delta_t^i = M_t^i - M_t$
 - 18: Randomly sample a sign $s_t^i \in \{1, -1\}$ with probability $\Pr[s_t^i = 1] = 0.5$
 - 19: Dimension selection $j_t^i = \mathbf{SignDS}(\Delta_t^i, k, \epsilon_r, s_t^i)$
 - 20: **Return** s_t^i, j_t^i
-

mits the local update for multiple global rounds, the overall privacy guarantee for his local data will be accumulated. Assume each client is allowed to participate in at most t_r global rounds. In order to ensure an overall privacy guarantee of ϵ -LDP for each client's local data after the whole training process, the per-round privacy guarantee should satisfy $\epsilon_r \leq \epsilon/t_r$. Moreover, during the training process, we monitor the number of rounds each client participates in. If a client has reached the maximum participating rounds (which is t_r here), he is not allowed to participate in the later training process.

3.3 Generating Synthetic Data and Data Post-Processing

Once the model has been trained, the server can use the decoder part to generate synthetic data. Recall that the latent space features are enforced to follow the standard Gaussian distribution p_z . Therefore, we can simply generate random latent features from p_z and feed them into the decoder. The decoder output has the same length as the encoded input described in Section 3.1, where each dimension is a numerical value between 0 and 1.

Finally, we need to convert the synthetic data back to categorical form. Given an output vector, we first split it into pieces of short vectors, each representing one categorical attribute. Then, for each short vector, we choose the entry with the maximum value as the attribute value. In the end, we concatenate all the categorical labels into one vector as the final synthetic data. The synthetic data will be used for data analysis and training of machine learning models.

4 Experiments and Results

We implemented the proposed framework and performed comprehensive experiments with a number of open-source datasets to evaluate its performance.¹ In this section, we introduce the experimental settings and discuss the evaluation results.

4.1 Experiment Setup

4.1.1 Datasets and WAE Models

We used four open-source datasets for evaluating the performance of our framework. Each dataset contains multi-dimensional data records, which were used for classification tasks:

- The **Census** dataset [17] contains records drawn from the 1990 United States census data, which include 68 personal attributes such as gender, income, and marriage status. We used the dataset for a classification task to determine the duration of people’s active duty service.
- The **Twitter** dataset [32] contains records with 77 attributes such as the number of discussions, aver-

Table 1. Datasets details

Dataset	Type	Num. Records	Num. Attributes	Domain Size
Census	Integer	2458285	68	2^{150}
Twitter	Integer	140707	78	2^{181}
Vehicle	Binary	98528	101	2^{101}
Adult	Binary	32561	124	2^{124}

Table 2. Structure of WAE models

Dataset	Num.Params	Model Structure
Census	76524	Input-Dense(96, relu)-Dense(24) -Dense(96, relu)-Output(sigmoid)
Twitter	94961	Input-Dense(128, relu)-Dense(36) -Dense(128, relu)-Output(sigmoid)
Vehicle	13093	Input-Dense(64, relu)-Dense(16) -Dense(64, relu)-Output(sigmoid)
Adult	16060	-Dense(64, relu)-Output(sigmoid)

age discussion length, and the number of authors, which are used to predict the number of active discussions, namely the popularity magnitude of each instance. In our experiment, we quantified the values of each attribute into five bins. The goal was to classify the level of popularity of each instance.

- The **Vehicle** dataset [18] contains data collected in wireless distributed sensor networks. Each record has 100 attributes representing data collected from different acoustic and seismic sensors. The goal was to train a classifier for vehicle type classification.
- The original **Adult** dataset [34] contains records with 15 personal attributes such as age, occupation, education, and gender. The goal was to train a binary classifier which determines whether a person earns more than 50K a year. We used the processed version from [42], which converted the original attributes into binary features.

We present details of each dataset in Table 1, which include the number of records and attributes, the length of the *one-hot* encoded input, and the total domain size. Since the number of user data should be large in order to preserve data utility (which will be discussed in Section 6.1), in the following experiments we simulated the large-scale distributed scenario by assuming there were 5×10^4 clients, each holding two data records. Hence, we randomly sampled 10^5 records for each dataset. For datasets with more than 10^5 records (*i.e.*, **Census** and **Twitter**), we did the sampling *without* replacement.

We varied the structure of the WAE models to fit the input size of different datasets. Details of the WAE models can be found in Table 2. For binary datasets

¹ The code will be available at <https://gitee.com/mindspore/mindspore/tree/r1.3/tests/st/fl/mobile/>.

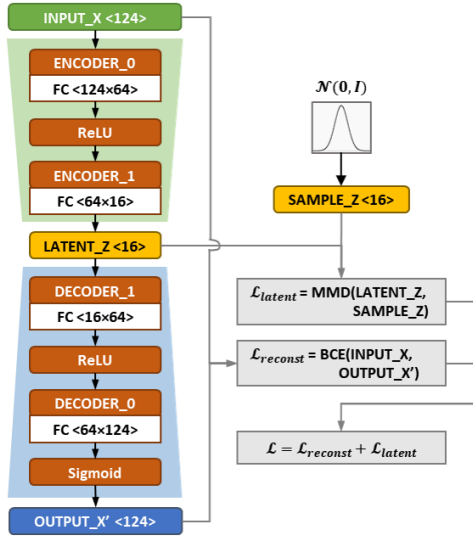


Fig. 3. Structure of the WAE model used for **Adult** dataset.

(*i.e.*, **Vehicle** and **Adult**), small WAE models with a latent-layer size of 16 were already sufficient to achieve satisfactory data synthesis performance. On the other hand, for the other two complex datasets that had distinctively higher domain sizes, we used larger models with a higher latent-layer size to better capture the hidden distribution and cross-attribute correlations. Moreover, it is also possible to use auxiliary data to further optimize the model structure, which we will discuss in Section 5.2. We also provide an example structure of the WAE model used for the **Adult** dataset (Figure 3), where FC represents fully-connected layers, BCE represents the binary cross-entropy and MMD represents the MMD penalty.

4.1.2 Baseline Methods

In the following experiments, we have used LOPUB [44] and LoCOP [53] as our baseline algorithms. Both algorithms apply LDP directly on the *local data* and send the randomized data to the server. The local randomization follows the RAPPOR algorithm [23]. As derived in [44], given c as the dimension of local data, h as the number of hash functions and f as the flip probability, the overall privacy for each individual client is

$$\epsilon = 2 \cdot c \cdot h \cdot \ln((2 - f)/f). \quad (7)$$

Then, the randomized data will be aggregated on the server side for estimating the joint distributions and attribute dependencies. Such information will then be finally used for constructing the synthetic dataset:

LOPUB generates the dependency graph based on a dependence threshold ϕ and estimates k -way joint distributions to generate the synthetic data; LoCOP leverages multivariate Gaussian copula to determine attribute dependencies and generates synthetic data by only using one- and two-way joint distributions. For both algorithms, we used the Lasso-based regression for estimating the joint distributions. In addition, we followed [44] to choose the number of hash function $h = 4$ and the dependence threshold $\phi = 0.4$.

4.1.3 Evaluation Metrics

We evaluated the performance of our framework from two perspectives, namely the *data utility evaluation* and the *privacy evaluation*:

- For the data utility evaluation, we first compared the statistical distributions of synthetic data and real data. Then, we used different machine learning models to investigate the utility of synthetic data in AI training tasks. Intuitively, synthetic data with high utility should show similar statistical properties and model accuracy as real data.
- For the privacy evaluation, we investigated the capability of our framework against membership inference attacks, where an attacker aimed to use the synthetic dataset to determine whether a target data was used for training the WAE model.

4.1.4 Parameter Configurations

In the experiments, we assumed there were 5×10^4 clients. We set the global round $T = 5000$, and $n = 10$ clients were sampled to train the WAE model in each global round; namely, each client was sampled once during the whole training process. We set the global learning rate $\gamma = 1$ due to the good empirical performance. For local training, each client updated the model for $E = 10$ epochs. We used the Adam optimizer with a default learning rate $\eta = 0.001$ for all the WAE models. For the local randomization, we chose the top- k ratio α from $\{0.05, 0.1, 0.25\}$ and the privacy budget $\epsilon \in \{0.5, 1, 2, 4, 6, 8\}$ to explore the influence of privacy on the framework performance.

It should be noted that ϵ here was the *overall* local privacy budget for each client. As mentioned in Section 3.2, if each client participated in t_r global training rounds, the per-round privacy budget should satisfy $\epsilon_r \leq \epsilon/t_r$. Since we assumed that each client only partic-

Table 3. Computation time of model training and synthetic data generation

Dataset		Adult	Vehicle	Census	Twitter
Training	Client	1.06 s	0.93 s	1.28 s	1.25 s
	Server	3.51 ms	3.05 ms	4.95 ms	4.92 ms
Data Generation		7.38 s	7.37 s	9.44 s	10.47 s

ipated once during the whole training process, we have $t_r = 1$ and the per-round privacy budget is equal to the overall privacy budget. Moreover, we would like to emphasize that the selected ϵ values are *reasonable* local privacy guarantees for collecting c -dimensional data. Consider the privacy guarantee of the baseline algorithms (Equation (7)), with the number of hash function $h = 1$ and a flipping probability $f = 0.5$, we already have $\epsilon = 150$ for the **Census** dataset with $c = 68$. For the **Adult** dataset with $c = 124$, the overall ϵ is even 272, which is significantly larger than our setting.

4.1.5 Computation Environments

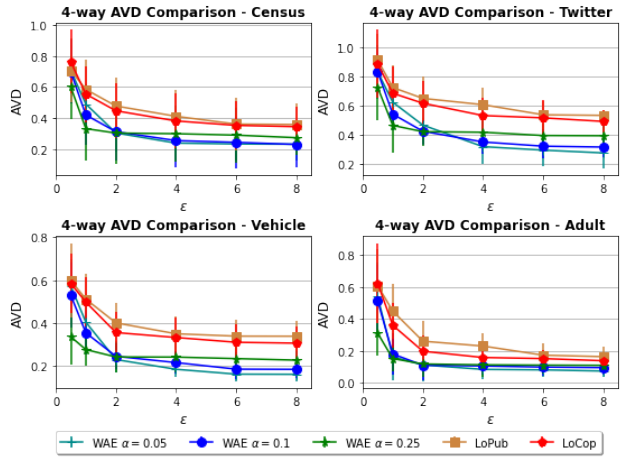
We performed all the experiments on a server with Intel E5-2470 2.40GHz CPU. In Table 3, we report the computational time of 1) 10 epochs of local training on each client; 2) one round of local updates aggregation and global model update on the server side; 3) generation of 10^5 synthetic data records on the server.

4.2 Evaluation for Data Utility

In this section, we evaluate the utility of the synthetic data generated using our framework in comparison to the baseline. The evaluations can be generally divided into *statistical comparison* and *AI training performance*.

4.2.1 Statistical Comparison

For the statistical comparison, the goal is to investigate whether the synthetic data generated by our framework can preserve the joint distributions and correlations of real data. Intuitively, synthetic data with high utility should show similar statistical properties as real data. We have respectively compared m -way joint distributions and the cross-attribute correlations to analyze the utility of the synthetic data.


Fig. 4. Average total variation distance (AVD) of four-way joint distribution between the real and synthetic data with respect to different privacy levels.

4.2.1.1 Comparison of Joint Distributions

For the analysis of joint distributions, we used the Average Variant Distance (AVD) to quantify the distribution difference between the real data and synthetic data, as suggested in [44], which is defined as

$$AVD = \frac{1}{2} \sum_{\omega \in \Omega} |P_{real}(\omega) - P_{syn}(\omega)|, \quad (8)$$

where $P_{real}(\omega)$ and $P_{syn}(\omega)$ are m -way joint distributions of real data and synthetic data. More specifically, given an m -way attribute combination ω with a domain size of $|\omega|$, P_{real} and P_{syn} are $|\omega|$ -dimensional vectors, where each entry is the probability of a specific value combination (namely the ratio of occurrence in the entire real or synthetic dataset). For each dataset, we randomly chose 100 combinations of m attributes and calculated the average distribution difference.

We first analyzed the AVD of all three algorithms with respect to the privacy level ϵ . For each dataset, we respectively compared the AVD of the synthetic data generated by the baseline algorithms and by our framework. In Figure 4, we present the results for the four-way joint distribution with different privacy budgets. The error bars represent the 95% confidence interval (also for the remaining experimental results). It can be seen that the AVD of all the algorithms decreases with the increase of ϵ . For all datasets, the synthetic data generated by our framework (referred to as *WAE*) have smaller AVD in comparison to the baseline methods (referred to as *LoPub* and *LoCop*), indicating that the synthetic data generated by our framework preserves better multivariate distributions than the baseline methods. Also, we notice that the non-binary datasets **Census**

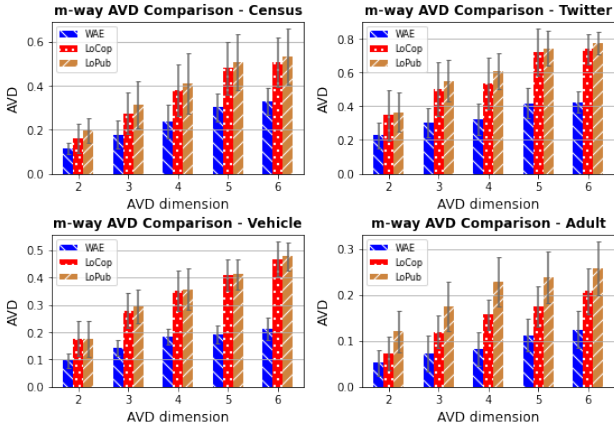


Fig. 5. Average total variation distance (AVD) of m -way joint distributions between the real and synthetic data with respect to different dimension of joint distribution.

and **Twitter** usually show larger AVD in comparison to the other two binary datasets. This is due to the fact that the non-binary datasets have a larger domain size, which leads to lower frequencies of the potential attribute combinations. Therefore, it is more difficult for the generative models to find meaningful mappings between the original input space and the compact latent space, which results in a comparatively larger difference between the synthetic data and real data. Moreover, we observe that for our solution, when the privacy budget ϵ is small, the synthetic data with larger top- k ratio have smaller AVD; while for larger ϵ , the synthetic data with smaller α show better utility. This complies with the discussion in Section 3.2. By intuition, the smaller α is, the better model performance is. However, when ϵ is small, the decrease of α leads to a significant decrease in top- k probability p , which increases the randomness of dimension selection and affects the model convergence. As ϵ increases, p is always relatively high and does not differ much regarding to α . In this case, a smaller α enhances the model performance and thus improves the utility of the synthetic data.

We further analyzed the AVD of all the algorithms with regard to the dimension of joint distributions m , in order to get a deeper insight into our framework’s capability on complex statistics. For each dataset, we tested the m -way AVD where $m \in \{2, 3, 4, 5, 6\}$ and present the results under $\epsilon = 4$ in Figure 5. It can be seen that for all the datasets, the AVD increases with a larger m . In addition, our proposed solution consistently outperforms the baseline algorithms. More specifically, the AVD of the baseline algorithms is close to our framework when m is small, yet gets distinctively larger with

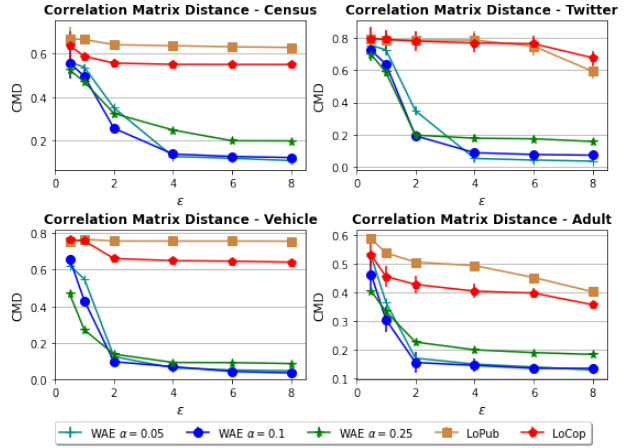


Fig. 6. Averaged correlation error (CMD) between the real and synthetic data with different privacy levels.

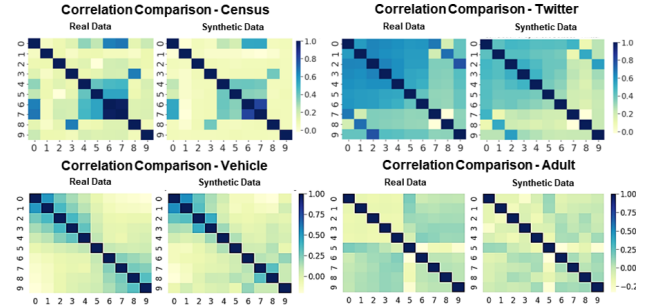


Fig. 7. Correlation comparison between the real and synthetic data with $\epsilon = 8$ and $\alpha = 0.1$. For each dataset, we present the correlations of the first 10 attributes. It can be seen that the synthetic data preserves similar correlations as real data.

an increase of m . This indicates that our framework can effectively capture the information of high-dimensional joint distributions of real data.

4.2.1.2 Comparison of Correlation

For the comparison of correlation, we have respectively computed the Pearson correlation coefficient of the real and synthetic dataset and used the Correlation Matrix Distance (CMD) [27] to measure the distance between the two correlations, which is defined as follows:

$$CMD = 1 - \frac{\text{tr}\{R_{real}R_{syn}\}}{\|R_{real}\|_2\|R_{syn}\|_2}, \quad (9)$$

where R_{real} and R_{syn} are correlation coefficient matrices of real and synthetic data, $\text{tr}(\cdot)$ is the matrix trace, $\|\cdot\|_2$ is the Frobenius norm. The CMD is bounded by $[0, 1]$, where zero means the two correlation matrices are identical.

For each dataset, we calculated the CMD of the synthetic data generated by both the baseline algorithms and our framework under different privacy levels and compare the results in Figure 6. It can be seen that with the same ϵ , the baseline algorithms always show a much larger CMD in comparison to the results of our framework. Although increasing the ϵ helps to reduce the CMD, it is still insufficient for preserving the multivariate correlations of real data. On the other hand, the synthetic data generated by our framework shows a distinctive decrease with the increase of ϵ . In particular, the CMD is close to zero when $\epsilon \geq 4$, indicating that the synthetic data have similar cross-attribute correlations as real data.

We further visualized the correlation coefficient matrix of real data and synthetic data with heat maps in order to better understand the capability of our method in capturing and preserving the cross-attribute correlations. Figure 7 shows the comparison result of the different datasets with $\epsilon = 8$ and $\alpha = 0.1$. For each dataset, we present the correlations of the first 10 attributes. From the visualization results, it can be seen that the correlation of synthetic data is similar to the correlation of real data, indicating that the synthetic data successfully preserves the attribute correlations of real data.

4.2.2 AI Training Performance

Next, we used different machine learning models to evaluate the utility of synthetic data in different AI training tasks. More specifically, we trained two classification models M_{real} , M_{syn} , respectively, with real data and synthetic data, and tested both models with an amount of held-out real data. Then, we compared the test accuracy Acc_{real} and Acc_{syn} , which represent the test accuracy of M_{real} , M_{syn} . If Acc_{syn} was close to Acc_{real} , we considered that the synthetic data are of high utility.

For each dataset, we used a two-layer Neural Network (NN) and Random Forest (RF) as the classification models. We trained each classification model 10 times and calculated the averaged Acc_{syn} . In Figure 8 and Figure 12, we present the results of Acc_{real} as well as Acc_{syn} evaluated on the synthetic data generated by all three methods under different privacy levels. It can be seen that the Acc_{syn} of the baselines shows, in general, distinctive distance from Acc_{real} on both evaluation models and only has slight improvement with larger privacy budget ϵ . In comparison, the Acc_{syn} of our method consistently outperforms the baselines for both classification algorithm. With an increase of ϵ , the

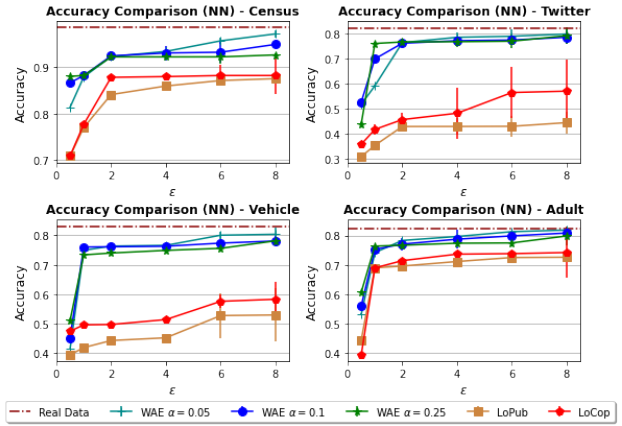


Fig. 8. Classification accuracy of the neural network (NN) trained with real data (*Real Data*) and synthetic data generated by our framework (*WAE*) as well as by the baseline algorithms (*LoPub*, *LoCop*) under different privacy levels.

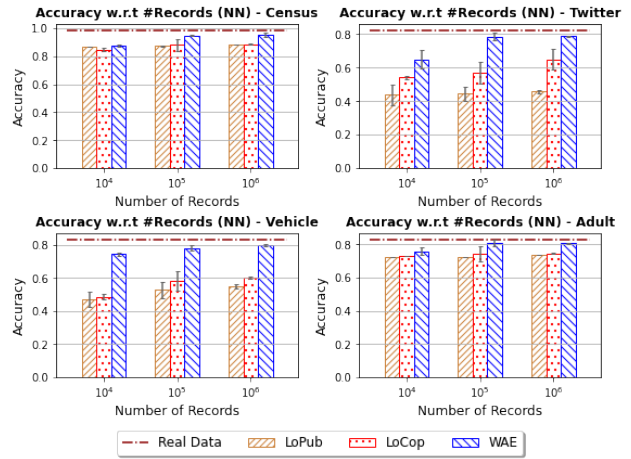


Fig. 9. Classification accuracy of the neural network (NN) with different number of records under the privacy level of $\epsilon = 8$.

Acc_{syn} gradually gets close to Acc_{real} . Moreover, we observe higher Acc_{syn} with the decrease of top- k ratio α . In particular, with $\epsilon = 8$ and $\alpha = 0.05$, the reduction of Acc_{syn} is less than 1% for the **Census** and **Adult** dataset and less than 3% for the other two datasets. The results above further indicate that the synthetic data generated by our framework largely preserves the joint distributions and hidden correlations of real data, and can replace real data for AI training tasks.

4.2.3 Impact of the Number of Records

In the above experiments, we assumed a group size of 5×10^4 clients and in total 10^5 records. We further investigated how the number of records impacts the util-

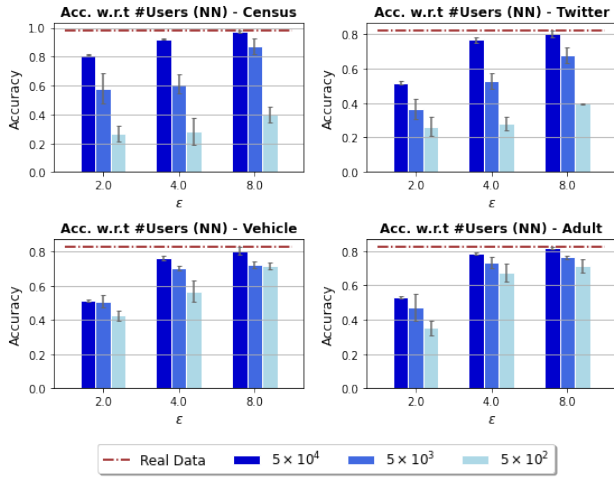


Fig. 10. Classification accuracy of the neural network (NN) with different number of users under different privacy levels.

ity of synthetic data. We varied the number of records among $\{10^4, 10^5, 10^6\}$ (thus the total number of clients is respectively $\{5 \times 10^3, 5 \times 10^4, 5 \times 10^5\}$). Similar to previous experiments, we assumed that each client held two data records and only participated once during the whole training process. For the experiments with 10^4 , we set the total global rounds $T = 500$ with $n = 10$ clients for each round. For the experiments with 10^6 records, we set the total global rounds $T = 5000$ with $n = 100$ clients for each round.

We have evaluated the accuracy of both classification models (*i.e.*, two-layer NN and RF) with respect to the number of records and present the results in Figure 9 and Figure 13. Here we compare the results under a privacy of $\epsilon = 8$ (and $\alpha = 0.1$ for our method). Although both algorithms show higher classification accuracy with a larger number of records, the baseline algorithms still cannot achieve significant improvements even with the largest number of records. In comparison, the classification accuracy of our method constantly outperforms the baseline algorithms. In addition, we notice that the classification accuracy in the experiments with 10^4 records is distinctively lower than others. This is because the generative model is underfitted when trained on a limited number of records and thus cannot generate high-utility synthetic data. On the other hand, although a larger number of local data (*e.g.*, 10^6) ensures the generative model be fully-trained, the model performance does not improve much after achieving convergence and thus cannot reach much improvement regarding classification accuracy.

4.2.4 Impact of the Number of Users

In previous experiments, we assumed that there were a large number of users (*e.g.*, 5×10^4) who each only participated once during the entire training process. We further extended the scenario by assuming there were fewer clients and each client was selected multiple times for model training. To this end, we respectively assumed there were 5×10^4 , 5×10^3 , 5×10^2 clients, each with 2, 20, and 200 data records. Each client participated in 1, 10, and 100 global training rounds and the corresponding per-round privacy budget ϵ_r equals ϵ , $\epsilon/10$ and $\epsilon/100$ (ϵ is the total privacy cost).

For each dataset, we conducted experiments with the total privacy budget $\epsilon \in \{2, 4, 8\}$ and evaluated the accuracy of both classification models regarding the number of users. The results are shown in Figure 10 and Figure 14. It can be generally seen that participating in multiple training rounds can cause a distinctive impact on the framework performance and data utility, especially for datasets with larger generative models (**Census** and **Twitter**). This is because with a fixed total privacy budget the per-round privacy budget is inversely proportional to the participating rounds. Therefore, having each client participating in multiple training rounds will significantly increase the randomness injected during model training and affect the model convergence. Thus, we need to further increase the total privacy budget ϵ to achieve satisfying data utility.

4.3 Evaluation for Privacy Protection

Although a larger privacy budget ϵ has a distinctive contribution to data utility, this may be at the expense of privacy. Currently, there is no concrete understanding of how to choose an appropriate ϵ in practice for a satisfactory utility-privacy trade-off. In this section, we have empirically analyzed the privacy protection capabilities of our framework against the membership inference attack (MIA). We followed the MIA protocol of [46]. The protocol assumed that the attacker holds a reference dataset that shares similar distribution as the real training data. The attacker respectively trains a pair of generative models G_{in} and G_{out} using the reference data *with* and *without* the target record. Then, an attack model is trained to distinguish the synthetic data generated by G_{in} and G_{out} , which can be considered as a binary classification task. Finally, given the published synthetic dataset, the attacker can use the attack model to test whether the synthetic data is generated by a

Table 4. Accuracy of membership inference attack

Dataset	Census	Twitter	Vehicle	Adult
No Privacy	0.735	0.698	0.637	0.642
$\epsilon = 8$	0.574	0.547	0.546	0.535
$\epsilon = 2$	0.548	0.529	0.513	0.519
$\epsilon = 0.5$	0.529	0.524	0.507	0.506

model trained with the target record, namely whether the target record is included in the generative model’s training dataset.

We randomly picked 30 records as the target record. For each target record, we trained generative models under different privacy settings and repeated the attack 10 times. In each attack trial, we used a list of machine learning models such as SVMs, logistic regression models, KNNs, RFs, and NNs as the attacker and picked the highest attack accuracy over all the attackers. Finally, we computed the averaged attack accuracy against all the target records under different privacy settings and present the results in Table 4. It can be seen that synthetic data generated by the non-private generative model is more likely to reveal the membership information of the target record. The attack accuracy of all the datasets is more than 60% and even up to 73.5% for the **Census** dataset. On the other hand, applying DP can effectively reduce the attack accuracy. With $\epsilon = 0.5$, the attack accuracy is reduced by 13% ~ 20%. Even with $\epsilon = 8$, the attack accuracy can still be reduced by 8% ~ 16% and is close to 50%, namely the accuracy of a random guess. The results demonstrate that our framework is able to reduce the risk of membership inference attacks and provide privacy protection to the local data.

5 Discussion

5.1 Extension to Other Data Types

In this paper, we demonstrated that our framework performs well on high-dimensional categorical data. In order to do so, we converted the categorical data into numerical form for training the WAE model and then reversed the model’s numerical outputs back to categorical form. In future studies, it is also possible to modify the current model structure and the loss function in order to extend the framework for supporting other data types, image data and text data. For instance, for image data, we can further apply convolution layers to

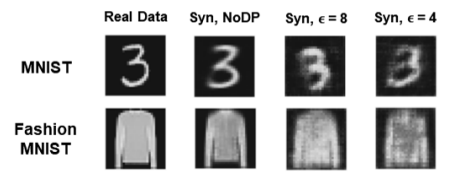

Fig. 11. Results of data synthesis on image datasets.

Table 5. Classification accuracy of synthetic data generated by WAE models with (w) and without (w/o) pre-training

Dataset		Accuracy - NN		Accuracy - RF	
		w/o	w	w/o	w
Census	$\epsilon = 8$	0.948	0.960	0.952	0.965
	$\epsilon = 4$	0.929	0.935	0.932	0.940
Twitter	$\epsilon = 8$	0.786	0.798	0.788	0.796
	$\epsilon = 4$	0.771	0.782	0.778	0.785
Vehicle	$\epsilon = 8$	0.781	0.795	0.788	0.794
	$\epsilon = 4$	0.762	0.789	0.763	0.782
Adult	$\epsilon = 8$	0.808	0.815	0.812	0.820
	$\epsilon = 4$	0.787	0.798	0.775	0.789

enhance the feature extraction capability and use the mean squared error instead of the cross-entropy to measure the reconstruction distance. Despite the variation of the generative models, the main idea of training the model under privacy-preserving federated learning and generating synthetic data remains unchanged. In Figure 11, we give the synthesis results evaluated on the **MNIST** [35] and **Fashion-MNIST** [56] dataset. For each dataset, we show the synthetic data produced by generated models trained under different privacy settings, namely, non-private, with privacy of $\epsilon = 8$ and $\epsilon = 4$. Note that the synthetic data are randomly generated and may look different from real data. However, it can be observed that our framework is also capable of synthesizing image datasets, and generated images have better quality with an increase of the privacy budget.

5.2 Auxiliary Data for Pre-Training

Before applying the WAE model for collecting local user data, the server needs to design the model structure. An appropriate model structure helps to enhance the capability of capturing the local data distributions and thus the utility of synthetic data. In our scenario, the server only knows the *basic properties* of the data to be collected, such as the number of attributes and the domain of each attribute. The server can thus use some auxiliary data to optimize the model structure. The auxiliary data here refer to certain public datasets or the random

data generated by uniformly sampling from the domain of the local data. The server can use such data to simulate the data collection process and tune the model structure by evaluating the utility of the synthetic data. Moreover, the auxiliary data can also be used for pre-training the WAE model before applying the model in the data collection process, so as to improve the model convergence and the utility of synthetic data.

In Table 5, we compare the utility of synthetic data generated by WAE models with (w) and without (w/o) pre-training under the setting of $\alpha = 0.1$ and $\epsilon \in \{4, 8\}$. For each dataset, we have randomly generated an auxiliary dataset only using the *basic properties* of real data, as mentioned before. We used the auxiliary dataset to pre-train the WAE model and applied the pre-trained model to the data collection process. We respectively evaluated the utility of synthetic data generated in both scenarios based on the classification accuracy of NNs and RFs. For both types of models, we observe that the synthetic data generated by pre-trained WAE models achieve 1 ~ 2% increase in classification accuracy. The results demonstrate that using auxiliary data to pre-train the WAE model is feasible to enhance the model convergence in the data collection process and further improve the synthetic data utility.

5.3 Limitations and Future Work

5.3.1 Utility Loss Under High Privacy Regimes

Although our proposed framework shows significant performance improvement in comparison to the baseline methods, the synthetic data still suffer from obvious utility loss when $\epsilon \leq 1$. Therefore, improving the framework performance under high-privacy regimes is one of the essential future research directions. Besides fine-tuning the hyperparameters such as the number of per-round participants n and the top- k ratio α , one of the other potential solutions is to pre-train the model with auxiliary data before the FL training procedure, as discussed in Section 5.2. In addition, the current SignDS algorithm can be further extended to support the selection of multiple top- k indices, which will also help improve the model convergence.

5.3.2 Communication Cost of Download Link

In comparison to the traditional LDP-based data collection approaches, our framework trains the generative

models under the federated learning setting, where each client needs to download the global model and upload the model updates at once. Although our framework significantly reduces the communication cost of the upload link by only transmitting one dimension index and the corresponding sign value, the communication cost of downloading the global model may also become a bottleneck as the model size increases, especially for large-scale FL scenarios. As discussed in Section 6.3.1, a few recent studies propose dropout-based and model pruning-based solutions for reducing the download link communication cost. The basic idea is to reduce the size of the model broadcast to the client side by extracting sub-models or pruning redundant weights from the global model. Meanwhile, using smaller models for local training can also help to reduce the computational cost of the client devices. Therefore, exploring whether such techniques can be integrated into our framework to further improve communication and computational efficiency will be an important future work.

6 Related Work

6.1 Data Collection Under LDP

Differential privacy (DP) [21], as a strong mathematical formalization of privacy, has been used as a criterion for privacy protection in data publishing, data mining, and machine learning ([1, 20, 57]). However, traditional DP assume a trusted server (data curator), who first collects the original user data, then performs data analysis under differential privacy. In order to eliminate the assumptions of trustworthy servers, local differential privacy (LDP) [31] has been proposed, which provides strong privacy guarantees to local data. By utilizing local randomization algorithms, the server cannot infer any individual's original data, but can learn the overall statistics of the whole population. However, prior research on LDP mainly focuses on collecting one-dimensional statistics, such as frequency estimation ([14, 23]), heavy-hitter identification ([6, 11]), and itemset mining ([43, 54]). Regarding scenarios with multi-dimensional data, Alaggar et al. [2, 3] proposed using Bloom filters to encode local data and analyze aggregated statistics. However, their works do not involve the estimation of joint distributions and cross-attribute correlations, which differs from our objectives.

Directly applying the above LDP-based algorithms to estimate complex statistics of high-dimensional data

will cause extremely large communication overhead as well as a degradation in data utility. Consider, for example, RAPPOR [23], a state-of-the-art LDP-based data collection algorithm. For c -dimensional binary data with $c = 32$, we have domain size $|\Omega| = 2^{32} \approx 4.3 \times 10^{10}$. Directly applying RAPPOR consumes a communication cost and a storage space of $O(|\Omega|)$ [44]. Also, for high-dimensional input domains, it is common for each user to have a unique feature combination. Therefore, it is essential to collect a large number of data in order to cover all the possible combinations in the feature domain. Given a domain size $|\Omega|$, as a general rule of thumb [23], the number of user data N should follow $\sqrt{N}/10 \geq |\Omega|$. In the above example, $N \geq 100 \cdot 2^{64} \approx 1.8 \times 10^{21}$. All of these requirements are impractical for real-world applications. In subsequent research, Fanti et al. [24] proposed to separately collect data of each dimension under RAPPOR and estimate the joint distributions using expectation maximization (EM). Although the algorithm significantly reduces the communication overhead between clients and the server, it only supports to estimate the joint distribution of two attributes. Based on [24], Ren et al. proposed LOPUB [44], which reduces c -dimensional data to k -dimensional clusters ($k < c$) using dependence graphs and estimates k -way joint distributions with an EM-based and Lasso regression-based approach. However, the algorithm still suffers from high computational complexity and low data utility when k is large. An improved scheme, LOCOP [53] was further proposed, which leverages multivariate Gaussian copula to estimate cross-attribute dependencies and to construct synthetic data.

Instead of directly randomizing the local data, our framework uses deep generative models to learn the data distributions and to generate synthetic data without accessing real data, which effectively enhances data utility. In our experiments, we used LOPUB and LOCOP as the baselines to compare the frameworks' performance in terms of data utility.

6.2 DP Synthetic Data Generation

Differentially private synthetic data generation has been extensively studied over recent years as an alternative solution to privacy-preserving data publishing. Previous works ([36, 57]) analyzed statistical distributions of original data under differential privacy and used them to generate synthetic data. Later works have proposed using differentially private generative models ([41, 49]) to directly generate high-utility synthetic data. However,

these works only focus on the centralized setting, where the server has already collected the real data and uses them to generate private synthetic data. In contrast, our approach is practical for a distributed setting, where the server cannot access the real data, but is interested in learning statistical information about the data.

Recent works by Augenstein et al. [5] and Triastcyn et al. [50] also investigate the synthetic data generation under the distributed setting. [5] aims to use generative models to detect errors and bugs in local data. However, as they claimed, such applications do not require high-fidelity generation. On the other hand, although [50] focused on generating and publishing synthetic data, their method is only limited to image data. In addition, they adopted a weaker measure of privacy for preserving the model performance. In comparison to both works, our framework is able to generate synthetic data with high utility and fidelity, which can replace real data in data mining and AI training tasks. Moreover, we apply strict LDP randomization on the client side, which provides strong privacy guarantees for clients' local privacy.

6.3 Efficiency and Privacy in FL

6.3.1 Communication Efficiency in FL

It is widely acknowledged that communication cost can be a bottleneck for FL, especially when training high-dimensional models. Recently, a number of *upload link* and *download link* compression methods have been proposed to alleviate the communication cost in FL.

The *upload link* compression applies quantization or sparsification on the model updates to reduce the communication cost from clients to the server. The main idea of quantization is to reduce the number of bits of update values. For instance, Seide *et al.* [45] proposed the 1-BIT SGD, which quantizes the update values larger than a pre-defined threshold to 1 and the rest to 0. Similarly, Bernstein *et al.* [8] proposed SIGNSGD and SIGNUM, where the update values quantized to its sign value. The study theoretically proved that SIGNSGD can effectively reduce the communication cost while enjoying a satisfying convergence rate. In contrast, sparsification aims to transmit only a subset of update values. For instance, the top- k mechanisms (*e.g.*, [4, 16]) only keep the top- k largest magnitude values of each model update and set the others to 0. Stich *et al.* [47] proposed a similar scheme with memory. Ivkin *et al.* [28] further proposed to use the count-sketch algorithm to approximately select the top- k updates.

On the other hand, reducing the communication cost of the *download link* has recently also gained increasing attention. Caldas *et al.* [13] gave the first attempt of the *download link* compression and proposed FEDERATED DROPOUT, which extracts small sub-models from the original high-dimensional and send to the clients local training. Based on this idea, Bouacida *et al.* further proposed [10] an adaptive federated dropout algorithm, which builds the sub-models using an adaptive activation score map. In addition, Jiang *et al.* [30] proposed to gradually prune the global model during the training process to achieve better communication and computational efficiency without loss of accuracy.

6.3.2 Privacy Protection in FL

Although FL enjoys significant privacy benefits in comparison to centralized learning, recent works showed that FL is still vulnerable to various privacy attacks [25, 40] against the exchanged local updates and the global model. Thus, an increasing number of studies propose to incorporate differential privacy (DP) into the FL framework. Some works (*e.g.*, [5, 39]) add Gaussian noise on the server side to protect the privacy of the global model. However, such solutions cannot prevent the privacy leakages from the local updates. Thus, other works propose hybrid frameworks (*e.g.*, [22, 29, 51]), which use crypto-based solutions such as homomorphic encryption (HE) and secure multi-party computation (SMC) to further achieve local privacy protection. However, these solutions require extra communication and computation costs during the key-distribution phase and thus, may not be practical to large-scale scenarios.

Considering the privacy and efficiency issues in the above-mentioned solutions, a more practical solution is to apply local differential privacy (LDP) to FL, which perturbs the local updates before sending them to the server. Previous LDP-FL frameworks (*e.g.*, [19, 52, 58]) perturb the local updates using private mean estimation algorithms. However, these algorithms evenly split the privacy budget across dimensions and the injected noise is proportional to the model dimension, making them only applicable to simple ML models. A recent work proposed FEDSEL [37], a *two-stage* LDP-FL framework that includes a *dimension selection* (DS) stage and a *value perturbation* (VP) stage. The DS stage first sorts the local update by *absolute value* and then privately selects one "important" dimension from the top- k dimension set (namely, the set of k dimensions with the largest absolute values). Then, in the VP stage, the value of

the selected dimension is perturbed via the LDP algorithms in [52] and used to construct a sparse privatized local update. Although [37] mitigates the dimension-dependency problem by only selecting one "important" dimension, the privacy budget is still consumed by the two stages. In high-privacy scenarios, each stage may therefore obtain only an insufficient privacy budget and cause large randomness.

Inspired by the effectiveness of SIGNSGD [8] and top- k sparsification ([4, 16, 37]), we propose a novel local randomization algorithm called SIGNDS. The main idea is to save the the privacy budget for the VP stage in [37] by assigning sign values instead of the perturbed dimension values to the selected dimensions. With the same privacy budget, we can achieve less randomness and thus higher model utility.

7 Conclusion

Building effective business and AI services requires the collection of personal data, which often introduces challenges related to an insufficient amount of data on the one hand, and privacy violations on the other hand. Recently, deep generative model-based data synthesis techniques have created opportunities for addressing these challenges: the generative models have strong capabilities in capturing the cross-attribute correlations of real data and can easily generate large-scale high-utility data; in addition, since the generated data are fully synthetic and cannot be linked to any particular individual, re-identification attacks or attribute disclosure becomes almost impossible.

In this paper, we have followed the idea of data synthesis and proposed DP-FED-WAE, a privacy-preserving framework for high-dimensional data collection. The framework utilizes a (generative) Wasserstein autoencoder to learn the joint distributions and correlations of high-dimensional user data and generate high-utility synthetic data on the server side. Moreover, we applied a novel LDP-FL framework for training the autoencoder, which not only avoids the collection of real local data but also provides strong local privacy guarantees. Experimental evaluation with real-world datasets shows that our framework significantly outperforms the LDP-based baseline algorithms for high-dimensional data collection and synthesis. The synthetic data generated by our framework preserves very similar statistical properties as real data and can replace real data for data mining and model training tasks.

8 Acknowledgements

This research was fully funded by Huawei. We thank the anonymous reviewers for their constructive comments for improving this paper. In particular, we thank our Shepherd, Thomas Humphries, for his valuable suggestions during the revision.

References

- [1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, New York, NY, USA, 2016. Association for Computing Machinery.
- [2] Mohammad Alaggan, Mathieu Cunche, and Sébastien Gams. Privacy-preserving Wi-Fi analytics. *Proceedings on Privacy Enhancing Technologies*, 2018(2):4–26, 2018.
- [3] Mohammad Alaggan, Sébastien Gams, and Anne-Marie Kermarrec. BLIP: Non-interactive differentially-private similarity computation on Bloom filters. In Andréa W. Richa and Christian Scheideler, editors, *Stabilization, Safety, and Security of Distributed Systems - 14th International Symposium*, volume 7596 of *Lecture Notes in Computer Science*, pages 202–216, Toronto, Canada, 2012. Springer.
- [4] Dan Alistarh, Torsten Hoefer, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, pages 5977–5987, Montreal, Canada, 2018.
- [5] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. Generative models for effective ML on private, decentralized datasets. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, virtual, 2020. OpenReview.net.
- [6] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2288–2296, Long Beach, CA, USA, 2017. Curran Associates Inc.
- [7] Gabrielle Berman, Sara de la Rosa, and Tanya Accone. Ethical considerations when using geospatial technologies for evidence generation. Technical report, Innocenti Research Briefs, 2018.
- [8] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. SIGNSGD: Compressed optimisation for non-convex problems. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 559–568. PMLR, 2018.
- [9] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.
- [10] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. *CoRR*, abs/2011.04050, 2020.
- [11] Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 435–447, New York, NY, USA, 2018. Association for Computing Machinery.
- [12] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. “You might also like”: Privacy risks of collaborative filtering. In *2011 IEEE Symposium on Security and Privacy (S&P)*, pages 231–246, Berkeley, California, USA, 2011. IEEE Computer Society.
- [13] Sebastian Caldas, Jakub Konečný, H. Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *CoRR*, abs/1812.07210, 2018.
- [14] Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(8), 2017.
- [15] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580, Long Beach, CA, USA, 2017. Curran Associates Inc.
- [16] Nikoli Dryden, Tim Moon, Sam Ade Jacobs, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In *2nd Workshop on Machine Learning in HPC Environments, MLHPC@SC*, pages 1–8, Salt Lake City, UT, USA, 2016. IEEE Computer Society.
- [17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
- [18] Marco F. Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.
- [19] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.
- [20] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pages 381–390, Bethesda, MD, USA, 2009. ACM.
- [21] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.
- [22] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *CoRR*, abs/2001.03618, 2020.

- [23] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067, Scottsdale, AZ, USA, 2014. ACM.
- [24] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. Building a Rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies*, 3:1–21, 2016.
- [25] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients – How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, virtual, 2020. Curran Associates Inc.
- [26] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, Montreal, Quebec, Canada, 2014. Curran Associates Inc.
- [27] Markus Herdin, Nicolai Czink, Hüseyin Özcelik, and Ernst Bonek. Correlation matrix distance, a meaningful measure for evaluation of non-stationary MIMO channels. In *2005 IEEE 61st Vehicular Technology Conference*, volume 1, pages 136–140, Stockholm, Sweden, 2005. IEEE.
- [28] Nikita Ivkina, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-efficient distributed SGD with sketching. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*, pages 13144–13154, Vancouver, BC, Canada, 2019.
- [29] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: Privacy-preserving empirical risk minimization. In *Advances in Neural Information Processing Systems*, pages 6343–6354. Curran Associates Inc., 2018.
- [30] Yuang Jiang, Shiqiang Wang, Bong-Jun Ko, Wei-Han Lee, and Leandros Tassioulas. Model pruning enables efficient federated learning on edge devices. *CoRR*, abs/1909.12326, 2019.
- [31] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, Jan 2011.
- [32] François Kawala, Ahlame Douzal-Chouakria, Eric Gaussier, and Eustache Dimert. Prédiction d'activité dans les réseaux sociaux en ligne. In *4ième conférence sur les modèles et l'analyse des réseaux : Approches mathématiques et informatiques*, page 16, France, 2013.
- [33] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, Banff, AB, Canada, 2014. OpenReview.net.
- [34] Ron Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 202–207, Portland, Oregon, USA, 1996. AAAI Press.
- [35] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.
- [36] Haoran Li, Li Xiong, and Xiaoqian Jiang. Differentially private synthesis of multi-dimensional data using copula functions. In *Advances in Database Technology: Proceedings of the International Conference on Extending Database Technology*, volume 2014, pages 475–486, Athens, Greece, 2014. NIH Public Access, OpenProceedings.org.
- [37] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. FedSel: Federated SGD under local differential privacy with top-k dimension selection. In *Database Systems for Advanced Applications - 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24-27, 2020, Proceedings, Part I*, volume 12112 of *Lecture Notes in Computer Science*, pages 485–501. Springer, 2020.
- [38] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, Fort Lauderdale, FL, USA, 2017. PMLR.
- [39] Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *Proceedings of the International Conference on Learning Representations*, Vancouver, BC, Canada, 2018. OpenReview.net.
- [40] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (S&P)*, pages 739–753, San Francisco, CA, USA, 2019. IEEE.
- [41] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, 2018.
- [42] John C. Platt. *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*, page 185–208. MIT Press, Cambridge, MA, USA, 1999.
- [43] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 192–203, Vienna, Austria, 2016. ACM.
- [44] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and S Yu Philip. LoPub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166, 2018.
- [45] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In Haizhou Li, Helen M. Meng, Bin Ma, Engsiung Chng, and Lei Xie, editors, *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association*, pages 1058–1062, Singapore, 2014. ISCA.
- [46] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. Synthetic data - A privacy mirage. *CoRR*, abs/2011.07018, 2020.
- [47] Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, pages 4452–

- 4463, Montreal, Canada, 2018.
- [48] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations (ICLR 2018)*, Vancouver, BC, Canada, 2018. OpenReview.net.
- [49] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. DP-CGAN: Differentially private synthetic data and label generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition – Workshops*, pages 98–104, Long Beach, CA, USA, 2019. Computer Vision Foundation / IEEE.
- [50] Aleksei Triastcyn and Boi Faltings. Federated generative privacy. *IEEE Intelligent Systems*, 35(4):50–57, 2020.
- [51] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11, 2019.
- [52] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *Proceedings of the 35th IEEE International Conference on Data Engineering*, pages 638–649, 2019.
- [53] Teng Wang, Xinyu Yang, Xuebin Ren, Wei Yu, and Shusen Yang. Locally private high-dimensional crowdsourced data release based on copula functions. *IEEE Transactions on Services Computing*, pages 1–1, 2019.
- [54] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy*, pages 127–143, San Francisco, California, USA, 2018. IEEE Computer Society.
- [55] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 2512–2520, Paris, France, 2019. IEEE.
- [56] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.
- [57] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via Bayesian networks. *ACM Transactions on Database Systems*, 42(4):25:1–25:41, 2017.
- [58] Yang Zhao, Jun Zhao, Mengmeng Yang, Teng Wang, Ning Wang, Lingjuan Lyu, Dusit Niyato, and Kwok-Yan Lam. Local differential privacy based federated learning for internet of things. *IEEE Internet of Things Journal*, 2020.

A Preliminaries

In the following, we offer additional background information about federated learning and differential privacy.

A.1 Federated Learning

Federated learning [38] (FL) is a decentralized learning mechanism which achieves computational efficiency and privacy benefits by distributing the training task to local devices. At each global round, the server distributes the current global model to a number of local clients. Each client locally updates the global model and returns the model update to the server. On the server side, all the local updates are aggregated to update the global model, which will be distributed in the next global round. Since only model parameters are exchanged during the training process, FL allows the model trained without accessing raw local data.

Although FL provides enhanced privacy protection in comparison to centralized training, recent contributions (e.g., [25, 40, 55]) point out that the mechanism still has privacy risks. In the context of FL, privacy risks can be mainly divided into *local privacy* and *global privacy* aspects. *Local privacy* risks appear when the local updates reveal insights about local data, while *global privacy* risks represent situations when the global model memorizes local data. Motivated by this, privacy enhancing techniques such as secure multi-party computation (MPC) [9] or differential privacy (DP) [39] are incorporated into the original FL mechanism, providing protections against global and local privacy risks.

A.2 Differential Privacy

Differential Privacy (DP) [21] is a state-of-the-art data anonymization technique which provides strong privacy guarantees for data analysis. The mathematical definition of DP is as follows:

Definition 1 (Differential Privacy [21]). *A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{O}$ satisfies ϵ -DP if for any two adjacent datasets D, D' differing in one data sample and for any measurable subset of outputs $\mathcal{Y} \subseteq \mathcal{O}$ we have*

$$\Pr[\mathcal{M}(D) \in \mathcal{Y}] \leq e^\epsilon \cdot \Pr[\mathcal{M}(D') \in \mathcal{Y}], \quad (10)$$

where ϵ describes the privacy loss.

The Definition 1 is usually applied in centralized settings where the data have already been collected by a trusted server. However, in the local settings, we aim to ensure that each client’s local data will not be accessed by the server. Thus, the definition of *local differential privacy* (LDP) has been proposed [31], which provides

strong local privacy guarantees for each user. The definition is as follows:

Definition 2 (Local Differential Privacy [31]). *A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{O}$ satisfies ϵ -LDP if and only if for any two inputs $x, x' \in \mathcal{D}$ and for any output $y \in \mathcal{O}$ we have*

$$\Pr [\mathcal{M}(x) = y] \leq e^\epsilon \cdot \Pr [\mathcal{M}(x') = y], \quad (11)$$

where ϵ describes the privacy loss.

In addition, LDP also holds two widely-used properties [21], namely *Robustness to Post-Processing* and *Sequential Composition*. The former property states that any deterministic or randomized function defined over an LDP mechanism also satisfies LDP. The latter states that interactively applying the LDP mechanism on the same set of data yields an accumulated privacy cost.

Property 1 (Robustness to Post-Processing). *Let \mathcal{M} be an ϵ -LDP mechanism and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{M}$ is ϵ -LDP.*

Property 2 (Sequential Composition). *Suppose there are n mechanisms $\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$ that respectively satisfy ϵ_i -LDP and are sequentially computed on the same set of private data D , then a mechanism formed by $(\mathcal{M}_1, \dots, \mathcal{M}_n)$ satisfies $(\sum_{i=1}^n \epsilon_i)$ -LDP.*

B Additional Experiment Results

In this section, we provide additional experiment results of the evaluation of AI training performance (Section 4.2.2), including the classification accuracy of random forests trained with synthetic data under different privacy levels (Figure 12), as well as the impact of accuracy regarding the number of records (Figure 13) and the number of clients (Figure 14).

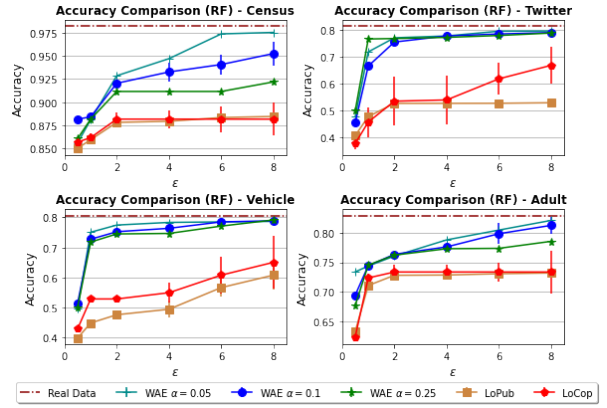


Fig. 12. Classification accuracy of the random forest (RF) trained with real data (*Real Data*) and synthetic data generated by our framework (*WAE*) as well as by the baseline algorithms (*LoPub*, *LoCop*) under different privacy levels.

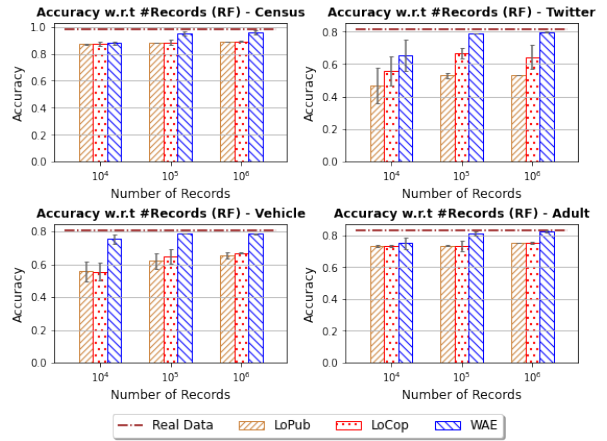


Fig. 13. Classification accuracy of the random forest (RF) with different number of records under the privacy level of $\epsilon = 8$.

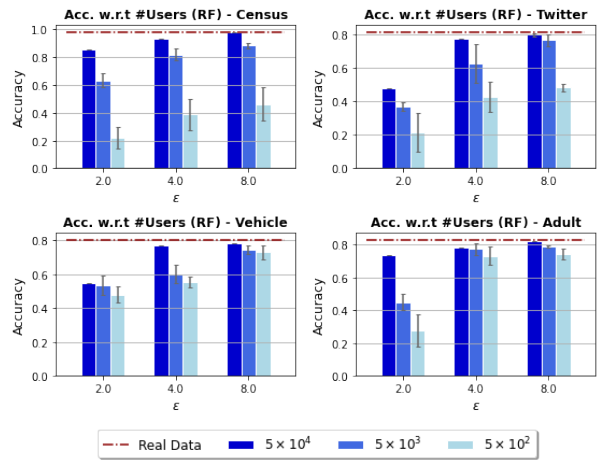



Fig. 14. Classification accuracy of the random forest (RF) with different number of users under different privacy levels.

 **CC BY-NC-ND 3.0**

ATTRIBUTION- NONCOMMERCIAL- NODERIVS 3.0 UNPORTED

Deed

Notice

This is an older version of this license. Compared to previous versions, the 4.0 versions of all CC licenses are **more user-friendly and more internationally robust**. If you are **licensing your own work**, we strongly recommend the use of the 4.0 license instead: **Deed - Attribution-NonCommercial-NoDerivatives 4.0 International**

Canonical URL : <https://creativecommons.org/licenses/by-nc-nd/3.0/>

[See the legal code](#)

You are free to:

Share — copy and redistribute the material in any medium or format

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give **appropriate credit** , provide a link to the license, and **indicate if changes were made** . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for **commercial purposes** .

NoDerivatives — If you **remix, transform, or build upon** the material, you may not distribute the modified material.

No additional restrictions — You may not apply legal terms or **technological measures** that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable **exception or limitation** .

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as **publicity, privacy, or moral rights** may limit how you use the material.

Creative Commons is the nonprofit behind the open licenses and other legal tools that allow creators to share their work. Our legal tools are free to use.

- [Learn more about our work](#)
- [Learn more about CC Licensing](#)
- [Support our work](#)
- [Use the license for your own material.](#)
- [Licenses List](#)
- [Public Domain List](#)

Footnotes

appropriate credit — If supplied, you must provide the name of the creator and attribution parties, a copyright notice, a license notice, a disclaimer notice, and a link to the material. CC licenses prior to Version 4.0 also require you to provide the title of the material if supplied, and may have other slight differences.

- [More info](#)

indicate if changes were made — In 4.0, you must indicate if you modified the material and retain an indication of previous modifications. In 3.0 and earlier license versions, the indication of changes is only required if you create a derivative.

- [Marking guide](#)
- [More info](#)

commercial purposes — A commercial use is one primarily intended for commercial advantage or monetary compensation.

- [More info](#)

remix, transform, or build upon — Merely changing the format never creates a derivative.

- [More info](#)

technological measures — The license prohibits application of effective technological measures, defined with reference to Article 11 of the WIPO Copyright Treaty.

- [More info](#)

exception or limitation — The rights of users under exceptions and limitations, such as fair use and fair dealing, are not affected by the CC licenses.

- [More info](#)

publicity, privacy, or moral rights — You may need to get additional permissions before using the material as you intend.

- [More info](#)

[Contact](#)
[Newsletter](#)
[Privacy](#)
[Policies](#)
[Terms](#)

CONTACT US

Creative Commons PO Box 1866,
Mountain View, CA 94042

info@creativecommons.org

+1-415-429-6753

SUBSCRIBE TO OUR NEWSLETTER

SUBSCRIBE

Except where otherwise noted, content on this site is licensed under a [Creative Commons Attribution 4.0 International license](#).
Icons by [Font Awesome](#).

SUPPORT OUR WORK

Our work relies on you! Help us keep the Internet free and open.

**DONATE
NOW**

3 Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning

Title	Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning
Authors	Xue Jiang ^{1,2} (xue.jiang@tum.de) Xuebing Zhou ² (xuebing.zhou@huawei.com) Jens Grossklags ¹ (jens.grossklags@in.tum.de) ¹ Technical University of Munich (TUM), Boltzmannstraße 3, 85748 Garching, Germany ² Huawei Munich Research Center, Riesstraße 25, 80992 Munich, Germany
Type	Working Paper
Status	Under review at time of thesis submission.
Copyright	Authors retain copyright of this working paper.
Author Contributions	Xue Jiang developed the core idea for the paper and took primary responsibility for designing the methodology, acquiring data, implementing experiments, evaluating results, and drafting the manuscript. Xuebing Zhou and Jens Grossklags provided valuable feedback and suggestions on the methodology and experiments, and assisted in reviewing and enhancing the manuscript.

Note: Since submission of the thesis, this working paper (in slightly modified form) has been accepted and published as: Xue Jiang, Xuebing Zhou, & Jens Grossklags (2024) Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning. *IEEE Access*, 12:157067–157082. **DOI:** <https://doi.org/10.1109/ACCESS.2024.3481652>. The published paper is licensed under a Creative Commons Attribution - Non-Commercial - NoDerivatives 4.0 License.

Distributed Synthetic Time-Series Data Generation with Local Differentially Private Federated Learning

Xue Jiang
xue.jiang@tum.de
Technical University of Munich
Garching, Germany

Xuebing Zhou
xuebing.zhou@huawei.com
Huawei Munich Research Center
Munich, Germany

Jens Grossklags
jens.grossklags@in.tum.de
Technical University of Munich
Garching, Germany

ABSTRACT

Developing effective data analysis tools and AI models for time-series data usually faces data insufficiency and privacy issues. While synthetic data generation (SDG) has been considered a promising technique for addressing these challenges, prior works assume that the central server can directly collect the real client data. However, this may not always be achievable in real-life scenarios, as the clients may be unwilling to share their private data.

In this paper, we consider a more realistic setting where the clients' local data are *inaccessible* to the server. We propose FED-STDG, the first privacy-preserving framework for *distributed* multivariate time-series synthesis. With the combination of federated learning (FL), local differential privacy (LDP), and a recurrent generative autoencoder, our framework facilitates the learning of spatial-temporal distributions of local multivariate time series *without* collecting the real data. Instead, the trained autoencoder will be used to generate high-utility synthetic data on the server side. We further propose an improved LDP-FL framework which achieves a remarkable utility improvement compared to previous algorithms. Experimental results on four open-source datasets demonstrate the capability and efficiency of our framework in generating high-quality time-series data. Under the same privacy level, the synthetic data generated using the proposed LDP-FL algorithm yields around 20% ~ 85% reduction in the downstream prediction error compared to the baselines. Additionally, empirical analyses of privacy and adversarial attacks show that our framework can effectively improve privacy protection and robustness in the FL process.

KEYWORDS

Time-series synthesis, federated learning, differential privacy.

1 INTRODUCTION

With the rapid development of network and computer technologies, large and diverse quantities of person-specific data are frequently generated on local devices such as smartphones and IoT sensors. Besides the widely-explored tabular data and image data, there has also been an increasing number of studies in modeling time-series data in finance [25], healthcare [41], and IoT [5] applications. These data usually consist of a series of measurements gathered periodically over time. Data analysts can use the rich information contained in these time-series data to explore the hidden temporal distributions and correlations among attributes from different perspectives and develop algorithms for classification and forecasting tasks. For instance, a digital healthcare application may utilize users' physical information for health monitoring and disease predictions, while a location-based service may use the drivers' historical trajectory to forecast potential future locations. However, the development

of such AI services usually faces two major challenges: data insufficiency and privacy issues. On the one hand, building reliable machine learning (ML) models usually requires adequate training data to prevent overfitting and achieve satisfactory performance. On the other hand, directly using real personal data for training the ML models may cause severe privacy problems and may even violate legal requirements (e.g., in the context of the GDPR).

In recent years, synthetic data generation (SDG) has been considered one of the potential techniques for addressing the above challenges in business intelligence & AI services. Extensive studies have proposed to use generative models to generate different types of synthetic data, such as tabular data [28], images [29], and time series [40] for a variety of real-world applications. Such techniques enjoy advantages in both data utility and privacy protection. The well-designed generative models have strong capabilities for capturing the joint distributions and hidden correlations of real data and can flexibly produce high-fidelity synthetic data for data analysis and building accurate ML models at a large scale. Also, the generated data are fully synthetic, which effectively reduces the risks of re-identification attacks or attribute disclosure [28]. Some later works [18, 39] also incorporate differential privacy (DP) into the training process to provide formal privacy guarantees to the algorithms. Nevertheless, these previous works only consider the SDG under the centralized setting, where the real data are already collected by a data curator (referred to as the *cloud server*) and will be used to train the generative models for privacy-preserving data publishing. This may not always be realistic since the data owners (referred to as *local clients*) may be reluctant to share their personal data with untrusted servers. To address this problem, some recent works [1, 16, 32] propose solutions for *distributed* SDG. The generative models are trained using the federated learning (FL) mechanism [26], which only exchanges model parameters and keeps the real data on the local side. However, existing *distributed* SDG solutions only focus on structured data and images (see [23], Table 2), which cannot be directly applied to time-series data.

In this paper, we offer the first attempt at *distributed* SDG of time-series data. Inspired by prior work conducted by Jiang et al. [16], we propose FEDSTDG, an efficient and privacy-preserving framework that achieves synthesizing the clients' local time-series data *without* the collection of real data. Our framework shows several advantages compared to [16]. To start with, we extend the Wasserstein Autoencoder (WAE) architecture used in [16] into a recurrent Wasserstein Autoencoder (RWAE), which can better capture the temporal distributions and correlations of multivariate time-series data. The model is then trained under the federated learning setting to learn the distribution of local data without collecting the

raw data. Finally, the trained model can be used to generate high-fidelity synthetic data on the server side. In addition, [16] proposed a local differentially private (LDP) algorithm, SIGNDS, to prevent privacy leakage for the local model updates. However, for each local update vector, the algorithm only selects one dimension index of an "important" parameter to send to the server, which suffers from a particularly slow convergence speed with large models. A follow-up study [17] introduced EM-MDS, a privacy-preserving algorithm that employs the exponential mechanism to select multiple dimensions under LDP guarantees. In this paper, we propose enhancements to EM-MDS from two perspectives. Firstly, we introduce an improved multi-dimensional selection algorithm called SUBMDS, which applies parameter subsampling to enable the selection of more dimension indices under the same privacy guarantee. Additionally, we present MAGRR, which adaptively adjusts the FL learning rate in a private manner and facilitates a rapid and stable model convergence. By incorporating both algorithms, our framework demonstrates superior model convergence and synthetic data utility in comparison to prior methods.

Our major contributions can be summarized as follows:

- We propose FEDSTDG, a privacy-preserving framework for the distributed synthesis of multivariate time-series data. With the combination of LDP, FL, and a recurrent autoencoder (RWAE), our framework enables the learning of local time-series data distributions and the generation of high-fidelity synthetic data *without* the need for centralized data collection. To the best of our knowledge, this is the first work of synthetic time-series data generation under a *distributed* setting.
- We introduce two innovative approaches, SUBMDS and MAGRR, as improvements to existing LDP-FL algorithms. The former applies parameter subsampling to enable the selection of more dimension indices, while the latter adaptively adjusts the learning rate to achieve a rapid and stable model convergence. The integration of both algorithms leads to superior synthetic data utility compared to existing methods.
- We use a number of real-world time-series datasets to evaluate the performance of our framework. Extensive evaluation results demonstrate that our framework has superior capability and efficiency in synthesizing time-series data while achieving satisfactory privacy protection and robustness.

2 RELATED WORK

2.1 DP Synthetic Data Generation

Differentially private synthetic data generation has been extensively studied over recent years as an alternative solution to privacy-preserving data publishing. Previous works [22, 42] analyzed statistical distributions of original structured data under differential privacy and used them to generate synthetic data. Later works proposed to use differentially private generative models such as generative adversarial networks [12] and autoencoders [21, 31] to directly generate high-utility synthetic data, which can be flexibly applied to tabular data [18], images [4], and time series [10]. However, prior SDG solutions mainly focus on the centralized setting, where the server has already collected the real clients' data for

generating private synthetic data. This may not always be realistic since the clients may refuse to share their personal local data with untrusted servers. In order to address the problem, some recent works [1, 32] introduced solutions for distributed SDG. The generative models are trained using the FL mechanism, which only exchanges model parameters and keeps the real data on the local side. However, existing distributed SDG solutions only focus on structured data and image data. In this paper, we fill this research gap and propose the first framework for synthetic *time series* data generation. The framework learns the spatial-temporal distributions of raw local data and generates synthetic time series on the server side to support downstream data analysis tasks.

2.2 Privacy-Preserving FL

Although FL enjoys significant privacy benefits in comparison to centralized learning, recent works showed that FL is still vulnerable to various privacy attacks against the local model updates [11] and the final global model [27]. A number of existing works propose to incorporate crypto-based solutions such as homomorphic encryption (HE) [33] and secure multi-party computation (SMC) [15] into FL for protecting the local model updates. However, these solutions cannot be well-scaled due to the extra communication and computation costs during the encryption and key-distribution phase. Some other works proposed to apply LDP to FL, where the local updates are perturbed before being sent to the server. However, previous private mean estimation-based LDP-FL frameworks (*e.g.*, [7, 36, 43]) easily suffer from significant utility loss, as the injected noise in these algorithms is, in essence, proportional to the number of model parameters. To address the issue, some recent works [16, 24] proposed dimension selection-based solutions, which only select one "important" dimension index for each local update vector under LDP guarantees and use the shared indices on the server side to construct corresponding sparse updates to update the global model. Nevertheless, both algorithms only select one dimension for each local update, which may lead to slow model convergence, especially for high-dimensional models. A follow-up work [17] introduced EM-MDS, an advanced LDP multi-dimension selection algorithm. Rather than repetitively conducting single-dimension selection with evenly divided privacy budgets, the algorithm treats each group of indices as a complete entity and adopts the exponential mechanism to assign higher probabilities to subsets containing more "important" indices. In this paper, we further enhance the existing algorithm with a reduced input size and plan an adaptive learning rate, which achieves better model utility under the same privacy guarantees.

3 PRELIMINARIES

3.1 Differential Privacy

DP [8] is a formal notion of privacy that is widely used in privacy-preserving data analysis applications. The definition of DP is as follows:

DEFINITION 1 (DP [8]). *A randomized mechanism $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ satisfies ϵ -DP if for any two adjacent datasets X, X' differing in one data sample and for any measurable subset of outputs $Y \subseteq \mathcal{Y}$ we*

have:

$$\Pr [\mathcal{M}(X) \in Y] \leq e^\epsilon \cdot \Pr [\mathcal{M}(X') \in Y], \quad (1)$$

where ϵ describes the privacy loss.

The Definition 1 is usually applied in centralized settings where the data have already been collected by a trusted server. However, in the local settings, we aim to ensure that each client's local data will not be accessed by the server. Thus, the definition of *local differential privacy* (LDP) has been proposed [19], which provides strong local privacy guarantees for each user. The definition is as follows:

DEFINITION 2 (LDP [19]). A randomized mechanism $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ satisfies ϵ -LDP if and only if for any two inputs $x, x' \in \mathcal{X}$ and for any output $y \in \mathcal{Y}$ we have:

$$\Pr [\mathcal{M}(x) = y] \leq e^\epsilon \cdot \Pr [\mathcal{M}(x') = y], \quad (2)$$

where ϵ describes the privacy loss.

In addition, LDP also holds two widely-used properties [8], namely *Sequential Composition* and *Robustness to Post-Processing*. The former property states that any deterministic or randomized function defined over an LDP mechanism also satisfies LDP. The latter states that interactively applying the LDP mechanism on the same set of data yields an accumulated privacy cost.

PROPERTY 1 (ROBUSTNESS TO POST-PROCESSING). Let \mathcal{M} be an ϵ -LDP mechanism and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{M}$ is ϵ -LDP.

PROPERTY 2 (SEQUENTIAL COMPOSITION). Suppose n mechanisms $\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$ respectively satisfy ϵ_i -LDP, and are sequentially computed on the same set of private data D , then a mechanism formed by $(\mathcal{M}_1, \dots, \mathcal{M}_n)$ satisfies $(\sum_{i=1}^n \epsilon_i)$ -LDP.

3.2 Autoencoders

The Autoencoder (AE) [30] is a type of neural network that is used to learn efficient and compressed feature representations in an unsupervised manner. An AE model typically consists of two primary components: an *encoder* Q_ψ and a *decoder* G_θ . The encoder reduces the dimensionality of the original high-dimensional input $x \sim P_x$ to a lower-dimensional latent feature $z = Q_\psi(x)$, and the decoder maps z back to the reconstructed output $x' = G_\theta(z)$, which has the same shape as x . The goal of training is to find an optimized pair of encoder and decoder that minimizes the reconstruction error between x and $x' = G_\theta(Q_\psi(x))$. This can be formulated as the following equation:

$$\mathcal{L}_{AE} = \mathbb{E}_{x \sim P_x} [\mathcal{L}_{rec}(x, G_\theta(Q_\psi(x)))], \quad (3)$$

Here, $\mathcal{L}_{rec}(\cdot, \cdot)$ is a metric used to measure the difference between two vectors. Building upon this, several variants have been proposed for synthetic data generation, such as the Variational Autoencoder (VAE) [21] and the Wasserstein Autoencoder (WAE) [31]. These variations introduce additional penalty terms to enforce the latent space features to follow certain prior distributions. For example, the objective function of WAE can be expressed as follows:

$$\mathcal{L}_{WAE} = \mathbb{E}_{x \sim P_x} [\mathcal{L}_{rec}(x, G_\theta(Q_\psi(x)))] + \lambda \cdot \mathcal{L}_{lat}(q_z, p_z), \quad (4)$$

where the latent distance \mathcal{L}_{lat} is measured using the maximum mean discrepancy (MMD) between the real latent distribution $q(z)$

and the standard Gaussian distribution $p(z)$. Given a batch of data sampled from the two distributions, i.e., $\{q^1, \dots, q^N\} \sim q_z$ and $\{p^1, \dots, p^N\} \sim p_z$, $\mathcal{L}_{lat}(q_z, p_z)$ can be empirically estimated as:

$$\begin{aligned} \mathcal{L}_{lat}(q_z, p_z) = & \frac{1}{N(N-1)} \sum_{i \neq j} \kappa(p^i, p^j) - \frac{2}{N^2} \sum_{i,j} \kappa(p^i, q^j) \\ & + \frac{1}{N(N-1)} \sum_{i \neq j} \kappa(q^i, q^j), \end{aligned} \quad (5)$$

where $\mathcal{K}(x, y) = \frac{\kappa}{\kappa + \|x - y\|_2^2}$. Given d_z as the dimension of latent layer and σ_z as the scale of the prior distribution, $\kappa = 2d_z\sigma_z^2$.

3.3 Federated Learning

Federated learning (FL) is a decentralized learning approach that offers computational efficiency and privacy advantages [26] compared to the centralized setting. In this paper, we focus on a typical FL scenario where a central server coordinates the training of a machine learning model M by multiple local clients. Each client has its private dataset X^i . At each global round t , the server selects a group of N clients and distributes the current global model M_t . Each client i trains the model for a few gradient descent steps using its local data and returns the local model update Δ_t^i to the server. On the server side, the average of all the local updates is first computed as $\Delta_t = \frac{1}{N} \sum_{i=1}^N \Delta_t^i$ and then used to update the global model parameters as $M_{t+1} = M_t + \gamma \cdot \Delta_t$. The updated model M_{t+1} is then distributed to the clients for the next round.

3.3.1 SignDS-FL. Although FL has successfully addressed the privacy concern of uploading raw data, studies [11, 27, 37] show that privacy risks still exist in model updates and final global models. Existing crypto-based FL protocols [2, 20] usually introduce computational and communication overheads, while LDP-based mean estimation algorithms affect model accuracy. To improve upon this, recent work [16, 17] proposed SIGNDS-FL, a dimension selection-based protocol that enhances model utility compared to previous LDP-FL protocols. In SIGNDS-FL, a top- k set S_k is constructed with the dimension indices indicating the k most significantly changed model parameters for each local update. A set of dimension indices is then selected from S_k under LDP guarantees. These dimension indices are then sent to the server to construct sparse updates and update the global model. Moreover, the protocol introduces a random sign value s to determine the construction of the S_k . If $s = 1$, S_k is constructed using the indices with the k largest updates. Otherwise, if $s = -1$, S_k is constructed using the indices with the k smallest updates. As the sign value s is further used to replace real magnitude for building the sparse update, the dynamic construction of the top- k set ensures that the sparse updates maintain a similar direction to the original updates.

3.3.2 Exponential Mechanism-based Multi-Dimension Selection (EM-MDS). In addition to [16], where only one dimension index is selected for each local update, [17] further proposed an improved algorithm EM-MDS that selects multiple dimension indices to boost model convergence. Given a local update vector $\Delta \in \mathbb{R}^d$ and the corresponding top- k set S_k , the algorithm defines $v = |S_k \cap J| \in [0, h]$ is the number of top- k dimensions contained in J . Intuitively, a larger v means that the selected indices better represent the updated

parameters and the sparse update is closer to the real update. Hence, the algorithm adopts an indicator function $u(S_k, J) = \mathbb{1}(v \geq v_{th})$ as the utility function, which assigns higher probabilities to the output sets containing more than v_{th} top- k dimensions. Here, $v_{th} \in [1, h]$ is a pre-defined threshold of v . Then, given a privacy budget, the algorithm samples an output set J with h elements from the output domain $\mathcal{J} = \{1, \dots, d\}^h$ with the following probability:

$$p = \frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S_k, J))}{\sum_{J' \in \mathcal{J}} \exp(\frac{\epsilon}{\phi_u} \cdot u(S_k, J'))} = \frac{\exp(\epsilon \cdot \mathbb{1}(v \geq v_{th}))}{\sum_{\tau=0}^h \omega_\tau \cdot \exp(\epsilon \cdot \mathbb{1}(\tau \geq v_{th}))},$$

$$= \frac{\exp(\epsilon \cdot \mathbb{1}(v \geq v_{th}))}{\sum_{\tau=0}^{v_{th}-1} \omega_\tau + \sum_{\tau=v_{th}}^h \omega_\tau \cdot \exp(\epsilon)}, \quad (6)$$

where ϕ_u is the sensitivity of $u(S_k, J)$ that equals to 1, $\omega_\tau = \binom{k}{\tau} \binom{d-k}{h-\tau}$ is the number of output sets $J' \in \mathcal{J}$ that contain τ top- k dimensions. Unlike evenly splitting the privacy budget and repeating the single-dimension selection process multiple times, the EM-MDS algorithm treats each potential output set as one individual entity. This allows for a more efficient utilization of the privacy budget when selecting multiple dimensions, resulting in improved model utility. Instead of dividing the privacy budget evenly and repeating the process of selecting a single dimension multiple times, the EM-MDS algorithm handles each potential output set as a separate entity. This allows for a more efficient utilization of the privacy budget and results in improved model utility.

4 PROBLEM STATEMENT

In this paper, we consider a scenario where a number of clients hold multivariate time-series data on the local side. Such data can be, for instance, clients' daily activities collected by wearable devices, or vehicle trajectories collected during driving. A central server aims to investigate the distribution and correlations of these time-series data and generate similar synthetic data for data analysis and designing AI services. Here, we assume the server to be *honest-but-curious*, who follows the system protocols but tries to infer sensitive information of local users. Hence, to protect local privacy, we require the server *not to* have direct access to raw local data.

The problem can be formulated as follows: given a private multivariate time-series dataset $X_{1:C}$ with A attributes and C time steps, each sample $\mathbf{x}_{1:C}^i \in X_{1:C}$ is denoted as

$$\mathbf{x}_{1:C}^i = \begin{bmatrix} F_1^i \\ \vdots \\ F_C^i \end{bmatrix} = \begin{bmatrix} f_{1,1}^i & \cdots & f_{1,A}^i \\ \vdots & \ddots & \vdots \\ f_{C,1}^i & \cdots & f_{C,A}^i \end{bmatrix}, \quad (7)$$

where F_c^i represents the vector of multivariate features of the c^{th} timestamp and $f_{c,a}^i$ is the feature of the a^{th} attribute. Note that each record has the same number of A attributes. Assume the private dataset is distributed among N local users. A central server aims to generate a synthetic dataset $\tilde{X}_{1:C}$ *without* accessing the real local data. The synthetic dataset $\tilde{X}_{1:C}$ should have the same number of attributes as $X_{1:C}$. Moreover, $\tilde{X}_{1:C}$ can preserve the temporal distributions as in $X_{1:C}$, *i.e.*,

$$P(X_{1:C}) \approx P(\tilde{X}_{1:C}). \quad (8)$$

The above objective can be further simplified with a conditional distribution among time steps as follows:

$$P(X_c | X_{1:c-1}) \approx P(\tilde{X}_c | \tilde{X}_{1:c-1}), \quad (9)$$

which states that the synthetic data can properly approximate the real data at any time c .

5 FEDSTDG FRAMEWORK

In this paper, we propose FEDSTDG, a privacy-preserving framework for distributedly synthesizing the multivariate local time series *without* collecting raw data. The overall workflow of our FEDSTDG is presented in Figure 1. More specifically, our framework first trains a time-series WAE under the FL setting to learn the spatial-temporal distributions of local data and then generates high-utility synthetic data on the server side, which can be used for downstream data mining and model training tasks. Moreover, on the basis of the sign-based dimension selection concept in [16, 17], we propose an improved LDP-FL algorithm that achieves a better privacy-utility balance. We first introduce an enhanced algorithm SUBMDS to support private multi-dimension selection. The algorithm effectively mitigates the privacy leakage from upload updates while enjoying a better model convergence compared to existing solutions. Additionally, we introduce the MAGRR algorithm, which enables adaptive learning rate adjustments during the training process. We prove that both SUBMDS and MAGRR satisfy a strict LDP definition. By integrating these key components, our framework eliminates the need for a trusted server and ensures strict privacy protection for local data. In the following, we will first introduce the RWAE model and LDP-FL algorithms used in our framework and then describe the overall workflow in detail.

5.1 Structure of the Time-Series Autoencoder

Although [16] proposed to use WAE [31] for the data synthesis, the previous model only applies to structured data. In this paper, we extend the model by incorporating long short-term memory (LSTM) [14] layers (a type of recurrent layer) in both encoder and decoder to better capture the spatial-temporal information of the original time-series data. The structure of our RWAE model is presented in Figure 2. For the encoder, the time-series data are sequentially input into the LSTM layer. Each cell in the LSTM layer takes in input at a given time step and combines it with information from previous time steps to compute the output of the current step, which is used as input for future time steps. The information passed from the previous cells enables the layer to learn the temporal correlations of input time series. Moreover, the layer adopts a forget gate, an input gate, and an output gate to compute the cell output, which helps to mitigate the gradient vanishing problem compared to the traditional RNN layers. The outputs of all the cells are then concatenated and input into a fully connected layer to compute the latent features. The decoder is constructed with a symmetric structure as the encoder, where the latent features are mapped back to the input space using another fully connected layer and an LSTM layer. Each cell in the LSTM layer of the decoder outputs features of a specific time step, and the combination of all these outputs forms the final synthetic data. Our RWAE model follows the same objective function as WAE models, which consists of a reconstruction distance measured using

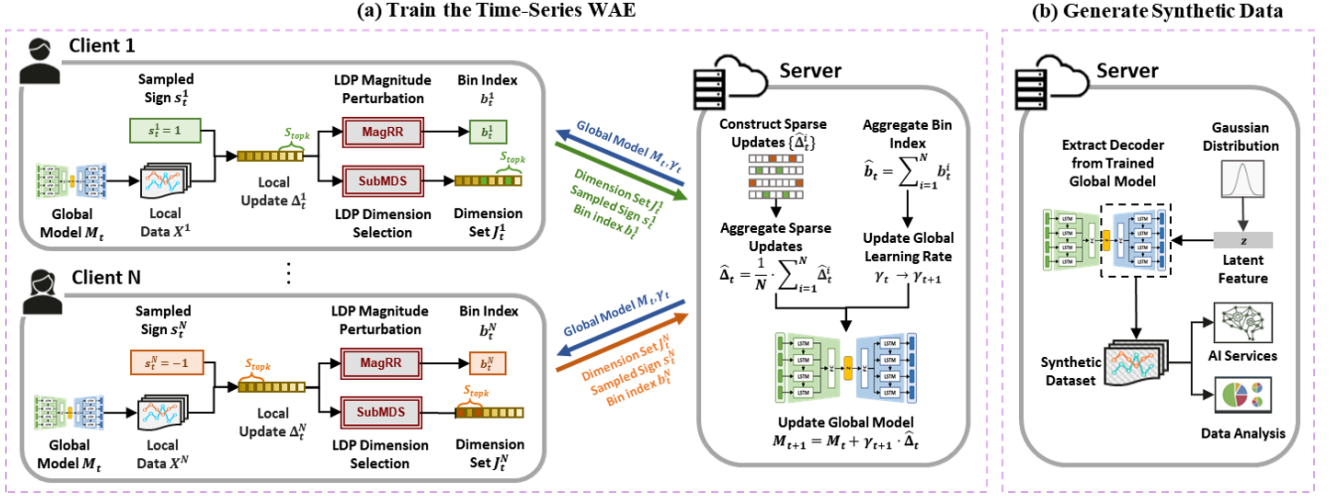


Figure 1: Overview of the FEDSTDG framework. The time-series autoencoder is first trained under the federated setting, which learns the distributions of real local data. Two LDP algorithms, SubMDS and MagRR, are respectively applied for privatizing the local updates and averaged update magnitude. After the model is trained, the decoder part is used to generate high-utility synthetic data. The generated data will be used for data analysis and building AI services.

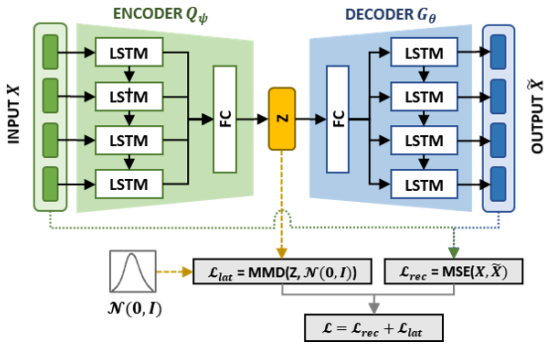


Figure 2: Structure of the RWAE model.

MSE and a latent space distance measured using MMD. Moreover, we set the balancing parameter $\lambda = 1$.

5.2 Training with LDP-FL Algorithm

5.2.1 Subset Multi-Dimension Selection (SubMDS). Upon a closer examination of the EM-MDS algorithm [17] (Section 3.3.2), we note that the optimal threshold $v_{th}^*(h^*)$ and the corresponding optimal output size h^* can be automatically determined. First, for each fixed output size h , we search the optimal threshold $v_{th}^*(h)$. More specifically, the expectation of v in the sampled output set J given a certain threshold $v_{th}(h)$ can be derived as follows:

$$\begin{aligned} \mathbb{E}[v|v_{th}(h)] &= \sum_{\tau=0}^h \tau \cdot p(v = \tau|v_{th}(h)) \\ &= \sum_{\tau=0}^{v_{th}-1} \frac{\tau \cdot \omega_\tau}{\Omega} + \sum_{\tau=v_{th}}^h \frac{\tau \cdot \omega_\tau \cdot \exp(\epsilon)}{\Omega}, \end{aligned} \quad (10)$$

where $\Omega = \sum_{\tau=0}^{v_{th}(h)-1} \omega_\tau + \sum_{\tau=v_{th}(h)}^h \omega_\tau \cdot \exp(\epsilon)$ as the denominator of Equation (6). Intuitively, the larger $\mathbb{E}[v|v_{th}(h)]$, the more likely that the sampled J contains more top- k dimension indices and the better the model utility. Therefore, the optimal threshold $v_{th}^*(h)$ can be determined as the threshold that achieves the maximum $\mathbb{E}[v|v_{th}(h)]$, namely

$$v_{th}^*(h) = \operatorname{argmax}_{v_{th} \in [1, h]} \mathbb{E}[v|v_{th}(h)]. \quad (11)$$

Next, we calculate *output top- k ratio*, namely the proportion of top- k dimension indices within the output set, as $\zeta_h = \mathbb{E}[v|v_{th}^*(h)]/h$. Intuitively, a larger output size h means more indices are selected, while a higher ratio ζ_h suggests selected indices are more likely to be top- k indices. To achieve a good balance between model convergence and utility, we set a minimum acceptable *output top- k ratio* ζ^* . Then, we respectively compute the output top- k ratio ζ_h for each output size h and choose the largest size that meets the requirement of ζ^* to be the optimal size h^* . Namely,

$$h^* = \operatorname{argmin}_{h \in [1, \text{inf}]} \zeta_h = \operatorname{argmin}_{h \in [1, \text{inf}]} \frac{\mathbb{E}[v|v_{th}^*(h)]}{h} \quad \text{s.t. } \zeta_h > \zeta^*. \quad (12)$$

Then, we use Equation (11) to compute the corresponding optimal threshold $v_{th}^*(h^*)$.

Through empirical tests, we discovered that the optimal output size h^* depends not only on the privacy budget ϵ but also on the *input top- k ratio* k/d , which is the ratio of the top- k set S_k to the total number of local update parameters. To illustrate this, let's assume the original local update has $d = 20000$ parameters and the top- k set contains the indices of $k = 100$ parameters. Keeping k fixed, we select the actual input size \tilde{d} from $\{0.01d, 0.05d, 0.1d\}$, resulting in *input top- k ratio* k/\tilde{d} of $\{0.5, 0.1, 0.05\}$. Next, we calculate the optimal output size h^* for different *output top- k ratio* ζ^* and present the results in Figure 3. We use different colors to represent

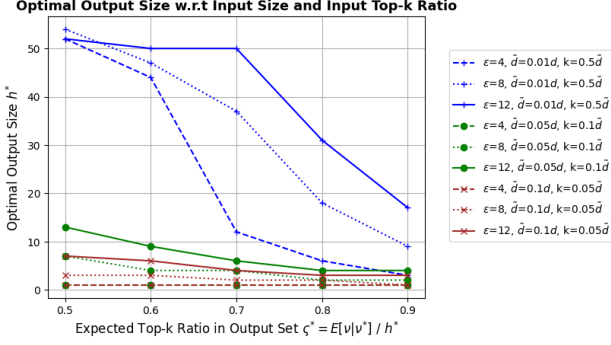


Figure 3: Comparison of the optimal output size under different privacy budgets and input top-k ratio. Given the original input size $d = 20000$ and top-k size $k = 100$, we vary the actual input size among $\tilde{d} = \{0.01d, 0.05d, 0.1d\}$, resulting in an input top-k ratio k/\tilde{d} among $\{0.5, 0.1, 0.05\}$. Then, we compare the corresponding optimal output size h^* regarding different output top-k ratio ζ^* .

various input top-k ratio and distinct line patterns to differentiate the privacy budget levels. The results show that, under the same privacy level, a larger input top-k ratio k/d leads to a larger h^* . This suggests that reducing the input size \tilde{d} may accelerate model convergence and enhance model utility.

The insightful result motivates us to introduce a new algorithm, SUBMDS. The algorithm aims to enhance model performance by using a reduced input size \tilde{d} . The process of SUBMDS is illustrated in Algorithm 1. Given the original local update $\Delta \in \mathbb{R}^d$, we first randomly select $\tilde{d} = \rho \cdot d$ model parameters and construct the input set S_{in} using the corresponding dimension indices. Then, a sign value s is sampled and used to determine the construction of the top-k set S_k . Next, with a predefined privacy budget ϵ and an expected output top-k ratio ζ^* , we compute the optimal output size h^* and threshold v_{th}^* based on Equation (12) and Equation (11). These values are then used to calculate the probability defined in Equation (6). It should be noted that in our algorithm, the input set S_{in} only contains \tilde{d} dimension indices. Hence, the output domain is shrunk to $\mathcal{J} = \{i | i \in S_{in}\}^h$ and the number of output sets containing v top-k dimensions is reduced to $\omega_v = \binom{k}{v} \binom{\tilde{d}-k}{h^*-v}$. Following [17], we use the inverse sampling technique [35] to build the output set. Specifically, the number of top-k dimensions included in J is determined by a random variable β drawn from the uniform distribution $U(0, 1)$ and the cumulative distribution function $\mathcal{F}(v|v_{th}^*)$. Finally, J is constructed by randomly selecting v indices from S_k and $h - v$ indices from the remaining indices set $S_{in} \setminus S_k$.

As the subset of parameters is randomly sampled and is irrelevant to the original local update, we can prove that our SUBMDS algorithm satisfies ϵ -LDP guarantees.

LEMMA 1. *The SUBMDS algorithm satisfies ϵ -LDP.*

PROOF. For each client, given any two arbitrary local updates Δ, Δ' and an input set S_{in} containing \tilde{d} randomly sampled dimension indices, let S_k and S'_k be the top-k sets constructed from S_{in} .

Algorithm 1: SUBMDS

Input: $\Delta \in \mathbb{R}^d$: local update with d parameters; k : size of top-k set; ρ : subsampling ratio; ϵ : privacy budget;

- 1: Randomly sample $\tilde{d} = \rho \cdot d$ parameters from Δ to build Δ_{in}
 - 2: Use the dimension indices of parameters in Δ_{in} to build an input set S_{in}
 - 3: Randomly sample a sign value s from $\{1, -1\}$
 - 4: **if** $s = 1$ **then**
 - 5: Build a top-k set S_k with the indices of k largest parameters in Δ_{in}
 - 6: **else**
 - 7: Build a top-k set S_k with the indices of k smallest parameters in Δ_{in}
 - 8: **end if**
 - 9: Given \tilde{d} and k , compute the optimal output size h^* and threshold v_{th}^* according to Equation (11) and Equation (12)
 - 10: Compute denominator $\Omega = \sum_{\tau=0}^{v_{th}^*-1} \omega_\tau + \sum_{\tau=v_{th}^*}^{h^*} \omega_\tau \cdot \exp(\epsilon)$,
where $\omega_\tau = \binom{k}{\tau} \binom{\tilde{d}-k}{h^*-\tau}$
 - 11: Randomly sample $\beta \sim U(0, 1)$
 - 12: Initialize $v = 0$ and $\mathcal{F}(v|v_{th}^*) = \omega_0/\Omega$
 - 13: **while** $\mathcal{F}(\tau) < \beta$ **do**
 - 14: $v = v + 1$
 - 15: **if** $v < v_{th}^*$ **then** $\mathcal{F}(v|v_{th}^*) = \mathcal{F}(v|v_{th}^*) + \omega_v/\Omega$;
 else $\mathcal{F}(v|v_{th}^*) = \mathcal{F}(v|v_{th}^*) + \exp(\epsilon) \cdot \omega_v/\Omega$
 - 16: **end while**
 - 17: Construct J by sampling v indices from S_k and $h^* - v$ indices from the remaining set $S_{in} \setminus S_k$
 - 18: **Return** J
-

For any output set $J \in \mathcal{J}$, where $\mathcal{J} = \{i | i \in S_{in}\}^h$ being the output domain, let $v = |S_k \cap J|$, $v' = |S'_k \cap J|$ be the number of intersected indices between J and both top-k sets. With the sampling probability defined in Equation (6) it holds that

$$\begin{aligned} \frac{\Pr[J|\Delta]}{\Pr[J|\Delta']} &= \frac{\Pr[J|S_k, S_{in}]}{\Pr[J|S'_k, S_{in}]} = \frac{\frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S_k, J))}{\sum_{J' \in \mathcal{J}} \exp(\frac{\epsilon}{\phi_u} \cdot u(S_k, J'))}}{\frac{\exp(\frac{\epsilon}{\phi_u} \cdot u(S'_k, J))}{\sum_{J' \in \mathcal{J}} \exp(\frac{\epsilon}{\phi_u} \cdot u(S'_k, J'))}} \\ &= \frac{\frac{\exp(\epsilon \cdot \mathbb{1}(v \geq v_{th}))}{\sum_{\tau=0}^{v_{th}-1} \omega_\tau + \sum_{\tau=v_{th}}^h \omega_\tau \cdot \exp(\epsilon)}}{\frac{\exp(\epsilon \cdot \mathbb{1}(v' \geq v_{th}))}{\sum_{\tau=0}^{v_{th}-1} \omega_\tau + \sum_{\tau=v_{th}}^h \omega_\tau \cdot \exp(\epsilon)}} = \frac{\exp(\epsilon \cdot \mathbb{1}(v \geq v_{th}))}{\exp(\epsilon \cdot \mathbb{1}(v' \geq v_{th}))} \quad (13) \\ &\leq \frac{\exp(\epsilon \cdot 1)}{\exp(\epsilon \cdot 0)} = \exp(\epsilon), \end{aligned}$$

which completes the proof. \square

5.2.2 MAGRR for Adaptive Global Learning Rate. After receiving clients' output index sets and sampled sign values, the server constructs corresponding sparse updates and uses their average to update the global model parameters. However, both previous works [16, 17] used a constant global learning rate γ throughout training, which impacts the model's utility. A large γ results in fast convergence but may cause a significant discrepancy between the real and

Algorithm 2: MAGRR

Input: $\Delta \in \mathbb{R}^d$: local update with d parameters; γ : global learning rate; \hat{b} : aggregated decay flag; r : decay rate; ϵ : privacy budget; T_{acc} : patience threshold; t_{acc} : number of rounds requiring a decayed learning rate.

LocalPert($\Delta, \gamma, \epsilon_{mag}$):

// Run on the client side

- 1: Get the largest magnitude value m from Δ
- 2: Compute the decay flag b :
if $m \in [0, r \cdot \gamma)$, set $b = 1$; **elif** $m \in [r \cdot \gamma, \infty)$, set $b = 0$
- 3: Perturb the decay flag b :

$$\hat{b} = \begin{cases} b & w.p. \frac{\exp(\epsilon)}{\exp(\epsilon)+1} \\ 1-b & w.p. \frac{1}{\exp(\epsilon)+1} \end{cases} \quad (14)$$

- 4: **Return** \hat{b}

ServerAggr($\hat{b}, \gamma, r, T_{acc}, t_{acc}$):

// Run on the server side

- 1: **if** $\hat{b} \geq 0.5$ **then**
- 2: $t_{acc} = t_{acc} + 1$
- 3: **if** $t_{acc} = T_{acc}$ **then** $\gamma' = r \cdot \gamma$, $t_{acc} = 0$; **else** $\gamma' = \gamma$
- 4: **else**
- 5: $\gamma' = \gamma$, $t_{acc} = 0$
- 6: **end if**
- 7: **Return** γ' , t_{acc}

the sparse local updates as training keeps on. Conversely, a small γ requires more rounds for convergence. To address this, an ideal approach is to apply an adaptive γ that starts large and progressively decreases during training. Therefore, we introduce a novel algorithm MAGRR to adapt the global learning rate in a private manner. The main idea is to collect the largest magnitude of each client's local update and use the aggregated result to determine the global learning rate for the next training round. In addition, as the real update magnitude cannot be directly published due to privacy issues, the algorithm randomizes the real magnitude under LDP guarantees and only shares the privatized value. More specifically, with the current round's global learning rate γ , each client computes the average top- k magnitude m (where $m \geq 0$). Subsequently, we quantize m into two bins, namely, $[0, r \cdot \gamma)$ and $[r \cdot \gamma, \infty)$, where r is the decay rate. We use the bin index b as a flag to indicate whether the global learning rate should be decayed. Then, we apply a binary randomized response (RR) [38] to flip the index. Given a privacy budget ϵ , we have:

$$\hat{b} = \begin{cases} b & w.p. \frac{\exp(\epsilon)}{\exp(\epsilon)+1} \\ 1-b & w.p. \frac{1}{\exp(\epsilon)+1} \end{cases}, \quad (15)$$

The randomized bin index will be sent to the server. Here, we present the privacy analysis of the MAGRR algorithm.

LEMMA 2. *The MAGRR algorithm satisfies ϵ -LDP.*

PROOF. For each client, given two arbitrary real bin indices b , b' and the perturbed index as \hat{b} , with the flip probability defined in

Equation (15) it holds that

$$\frac{\Pr[\hat{b}|b]}{\Pr[\hat{b}|b']} \leq \frac{\Pr[\hat{b} = b|b]}{\Pr[\hat{b} = 1 - b'|b']} = \frac{\frac{\exp(\epsilon)}{\exp(\epsilon)+1}}{\frac{1}{\exp(\epsilon)+1}} = \exp(\epsilon), \quad (16)$$

which completes the proof. \square

After receiving the perturbed flags, the server will update the global learning rate accordingly. If more than half of the flags are 1, meaning the majority of local clients' update magnitude is less than $r \cdot \gamma$, we adjust the learning rate to $r \cdot \gamma$; otherwise, we keep γ unchanged. However, adjusting the learning rate in every round may lead to unstable training due to variations in update magnitudes across clients and the randomness in flag reporting. To address this, we introduce a patience threshold T_{acc} and only modify the global learning rate when the condition of majority flags being 1 persists for more than T_{acc} rounds. In this paper, we use default values of $r = 0.5$ and $T_{acc} = 50$.

5.3 Overall Workflow of FEDSTDG

We now describe the overall workflow of our FEDSTDG framework presented in Figure 1.

First, we train the RWAE within a federated setting, as depicted in Algorithm 3. At each global round t , the server randomly selects a subset of N clients and broadcasts the current global model M_t and the global learning rate γ_t . Each local client i trains the global model using its private data X^i and calculates the local update Δ_t^i . Then, given a privacy budget ϵ_{idx} , the client utilizes the SUBMDS algorithm to sample the random sign value s_t^i and compute the private index set J_t^i . Additionally, with a privacy budget ϵ_{mag} , the client computes the decay flag based on the local update magnitudes. Then, the client sends the respective b_t^i , s_t^i , and J_t^i to the server. Note that for each local update, we sequentially apply the SUBMDS and MAGRR algorithms to obtain the private index set and the decay flag of the learning rate. According to the DP's sequential composition theorem (Property 2), the total privacy guarantee to each local update is $\epsilon = \epsilon_{idx} + \epsilon_{mag}$.

After receiving the information from local clients, the server constructs the sparse update using each pair of s_t^i and J_t^i and computes the aggregated model update $\hat{\Delta}_t$. Moreover, the server also calculates an aggregated decay flag \hat{b}_t based on the local decay flags and modifies the global learning rate γ_t accordingly. Finally, the server employs $\hat{\Delta}_t$ and \hat{b}_t to update the global model parameters, and the new global model M_{t+1} is distributed to local clients for the next training round.

Once the model has been trained, the server can use the decoder G_θ to produce synthetic time-series data. Recall that the latent space features are enforced to follow the standard Gaussian distribution p_z . Therefore, we can simply sample random latent features z from p_z and feed them into G_θ to generate high-utility synthetic time-series data, which can be further used for data analysis and building AI services.

Algorithm 3: Training the RWAE

Input: $M \in \mathbb{R}^d$: global model with d parameters; T : number of aggregation rounds; N : number of clients in each round; E : number of local epochs; η : local learning rate; k : size of top- k set; ϱ : subsampling ratio; γ : global learning rate; t_{acc} : accumulated number of decay rounds; T_{acc} : threshold for decay rounds; r : decay rate; ϵ_{idx} , ϵ_{mag} : privacy budgets.

Output: Trained WAE model M

Server executes:

- 1: Initialize the global model M_1 and the global learning rate γ_1
- 2: **for** global round $t = 1, \dots, T$ **do**
- 3: Randomly select a group of N clients
- 4: **for** client $i = 1, \dots, N$ in parallel **do**
- 5: Broadcast the global model M_t and the global learning rate γ_t to the local side
- 6: $b_t^i, s_t^i, J_t^i = \text{LocalUpdate}(M_t, E, \eta, \epsilon_{idx}, \epsilon_{mag}, \gamma_t, k, \varrho, r)$
- 7: Construct $\hat{\Delta}_t^i = [0, \dots, 0]^d$; for $j \in J_t^i$, set $\hat{\Delta}_t^i[j] = s_t^i$
- 8: **end for**
- 9: Compute the aggregated model update $\hat{\Delta}_t = \frac{1}{N} \sum_{i=1}^N \hat{\Delta}_t^i$
- 10: Compute the aggregated decay flag $\hat{b}_t = \frac{1}{N} \sum_{i=1}^N b_t^i$
- 11: Update global learning rate
 $\gamma_{t+1}, t_{acc} = \text{MagRR.ServerAggr}(\hat{b}_t, \gamma_t, r, T_{acc}, t_{acc})$
- 12: Update global model $M_{t+1} = M_t + \gamma_{t+1} \cdot \hat{\Delta}_t$
- 13: **end for**
- 14: **Return** Global model $M = M_{T+1}$

LocalUpdate($M_t, E, \eta, \epsilon_{idx}, \epsilon_{mag}, \gamma_t, k, \varrho, r$):

// Run on the client side

- 15: Initialize local model $M_t^i \leftarrow M_t$
- 16: **for** epoch $e = 1, \dots, E$ **do**
- 17: $M_t^i = M_t^i - \eta \cdot \nabla \mathcal{L}(M_t^i, X^i)$
- 18: **end for**
- 19: Compute local update: $\Delta_t^i = M_t^i - M_t$
- 20: Get the output set $J_t^i, s_t^i = \text{SubMDS}(\Delta_t^i, k, \varrho, \epsilon_{idx})$
- 21: Get the decay flag $b_t^i = \text{MagRR.LocalPert}(\Delta_t^i, r, \gamma_t, \epsilon_{mag})$
- 22: **Return** b_t^i, s_t^i, J_t^i

Table 1: Details of Datasets and Model Size

	#Records	#Features	#Steps	#Parameters
Sine	50,000	5	30	29,269
Stock	4,400	6	24	26,358
Energy	19,735	27	24	29,729
Air	9,333	13	24	27,485

6 EXPERIMENTS

We implemented the proposed framework using the TensorFlow platform and conducted comprehensive experiments with real-world datasets to evaluate its performance. We conduct the proposed experiments on an Intel 1.8 GHz Core i7 CPU. In this section, we will introduce the experimental settings and discuss the evaluation results.

6.1 Experiment Setup

6.1.1 Datasets and Model Architectures. We use four real-world multivariate time-series datasets for evaluating the performance of our framework:

Sine [40]: The dataset contains 50,000 multivariate sinusoidal sequences simulated using the open-source code provided in the original paper¹. Each record contains five attributes, representing the sequence with different frequencies and phase.

Stock [9]: The dataset contains daily historical Google stocks data starting from August 2004, which sums to approximately 4400 records. Each record contains six attributes such as volume, opening prices, and closing prices. The goal is to predict future stock prices and volume.

Energy [3]: The dataset contains 19,735 records, each with 27 attributes presenting the energy usage as well as temperature and humidity in different areas of each period of time. The goal is to predict energy usage in the future.

Air [6]: The dataset contains 9358 hourly records of an Air Quality Multisensor Device, including the hourly concentrations and sensor responses of different types of gas. The goal is to predict the future trend of gas concentration. We remove the date and time attributes in the original data and replace the missing values with zeros.

For all the datasets, we normalize the record values to [0,1]. For the encoder part of the RWAE models, we use a single-layer LSTM with 32 hidden units followed by a fully connected layer with 16 units. The layers in the decoder are of the same hidden units as the encoder but in reversed order. Furthermore, we use the *sigmoid* activation for the output layer to restrict the reconstructed values to be within [0,1]. Details of the datasets and the corresponding model size are presented in Table 1.

6.1.2 Evaluation Metrics. We evaluate the performance of our framework considering three aspects, namely *data utility*, *privacy protection*, and *robustness*. For the evaluation of data utility, we follow the evaluation approaches in [40] and assess the quality of synthetic data from three perspectives:

- *Fidelity*: The synthetic data should preserve the correlations and distributions of real data.
- *Diversity*: The synthetic data should be diversely distributed and cover most of the variety of real data.
- *Usefulness*: The synthetic data should have similar performance as real data in AI training tasks.

In this study, we utilize the MMD distance Equation (5), TSNE analysis [34], and mean absolute error (MAE) for next step prediction to assess the *fidelity*, *diversity*, and *usefulness*, respectively. To evaluate privacy protection, we examine the effectiveness of our framework against membership inference attacks (MIA) by comparing the attack accuracy under various privacy levels. Finally, we evaluate the robustness of our framework against untargeted poisoning attacks.

6.1.3 Baselines. In the following experiments, we investigate the performance of our framework under different privacy settings. We

¹https://bitbucket.org/mvdschaar/mlforhealthlabpub/src/master/alg/timegan/data_loading.py

also compare the results *with* and *without* adaptive global learning rate adjustment, which are respectively referred to as SUBMDS and ADASUBMDS. We also compare the performance of the RWAE model trained with baseline LDP-FL algorithms, including the SIGNDS with single-dimension selection [16] (referred to as SIGNDS-s) and with multi-dimension selection [17] (referred to as SIGNDS-M). We also include a baseline that evenly splits the privacy budget for multi-dimension selection (referred to as SIGNDS-E). Moreover, we also compare the model trained under the non-private centralized setting and federated setting and the DP centralized setting.

6.1.4 Hyperparameter Configurations. In each experiment, we conduct the FL training for 1000 global rounds, where $N = 50$ clients are sampled in each round. For the local training, we assume each local client has 10 time-series records and trains the RWAE model for $E = 10$ epochs using the *Adam* optimizer with a default learning rate $\eta = 0.001$. We further choose various privacy budgets $\epsilon \in \{0.5, 2, 4, 6, 8\}$ to explore the influence of privacy on the framework performance. For all the algorithms, we keep $k = 0.05d$, where d is the number of model parameters and k is the number of indices in S_{topk} . For SUBMDS, we set the subsampling rate ρ to 0.2. Hence, the input top- k ratio for SIGNDS-S, SIGNDS-M, SIGNDS-E is 0.05, while for SUBMDS is $0.05/0.1 = 0.5$. Moreover, for all three algorithms, we set the expected output top- k ratio ζ^* to 0.8. In experiments *without* adaptive global learning rate, we set ϵ_{idx} to ϵ and γ to 5. In experiments *with* adaptive global learning rate (ADASUBMDS), we set ϵ_{idx} to 0.9ϵ and ϵ_{mag} to 0.1ϵ . The initial value of γ is set to 25, the decay rate r is 0.5, and the patience threshold T_{pat} is 50.

6.2 Evaluation of Data Utility

We first evaluate the utility of the synthetic data under different privacy levels regarding the *fidelity*, *diversity*, and *usefulness*.

6.2.1 Analysis of Fidelity. To start with, we investigate the fidelity of synthetic data in comparison to real data. To this end, we use the MMD distance in Equation (5) to measure the distribution difference between real data and synthetic data. Intuitively, the smaller the MMD, the more similar the synthetic data is to real data. For each dataset, we respectively compare the MMD of the RWAE models trained under private and non-private centralized settings and FL settings and present the results in Figure 4. It can be seen that, for the RWAE models trained with LDP-FL algorithms, the increase of the privacy budget ϵ generally leads to a decrease in MMD, as the model is perturbed by less noise during the training process. Moreover, our SUBMDS and ADASUBMDS algorithms achieve a distinctive improvement compared to the baseline algorithms, particularly with smaller privacy budgets. Specifically, when $\epsilon = 0.5$, the SUBMDS algorithm results in approximately 30% – 40% reduction in MMD, and the ADASUBMDS algorithm yields a substantial reduction of nearly 45% – 80%. The results suggest that reducing the input dimensions and adopting a larger top- k ratio can effectively accelerate model convergence and help improve synthetic data utility.

6.2.2 Analysis of Diversity. Next, we qualitatively assess the diversity of the synthetic data. Here, we conduct a T-SNE analysis [34]

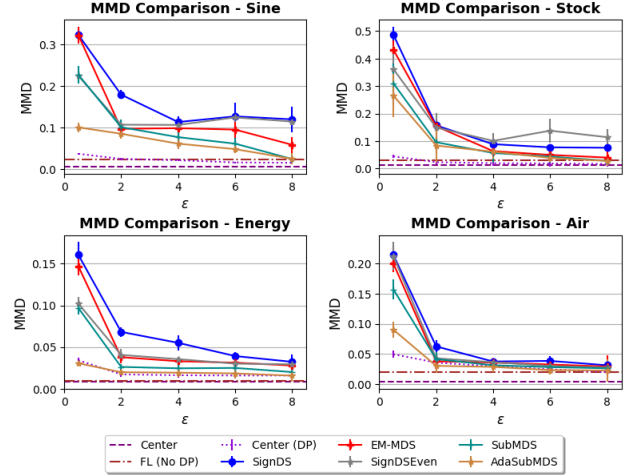


Figure 4: Comparison of MMD distance between real and synthetic data of all the datasets. For each dataset, we compare the results of model trained under non-private centralized and FL settings, as well as private centralized and FL settings under different privacy levels.

to visually depict the distribution of real and synthetic data from different datasets in a two-dimensional space. Intuitively, synthetic data with good utility should be characterized by a distribution with high diversity and cover the distribution of real data. In Figure 5, we present the visualization results of the four datasets generated by the RWAE model under the non-private centralized and federated settings, as well as using different LDP-FL algorithms with $\epsilon = 8$. For each dataset, we respectively compare the distribution between the real data (red) and the synthetic data (blue). Moreover, we also use the Synthcity library [?] ² to compute the coverage ratio between real and synthetic data. It can be seen that the synthetic data generated by the two baseline algorithms fail to cover the entirety of the real distribution. In contrast, the synthetic data generated using our SUBMDS and ADASUBMDS algorithms can better overlap with real data and achieve a coverage ratio similar to that of the non-private setting. The results illustrate that our framework can capture the distributions of different variants of real data and generate diversified synthetic data.

6.2.3 Analysis of Usefulness. We further analyze the usefulness of synthetic data in AI training tasks. We respectively use the real and synthetic data to train a one-layer LSTM model for *next-step prediction*. Then, we evaluate the model performance using a *held-out* set of real data. Intuitively, if the LSTM models trained with synthetic data demonstrate similar performance as those trained with real data, we say that the synthetic data is of high usefulness and can replace real data for downstream AI training tasks. Here, we use the mean absolute error (MAE) to measure the performance of the LSTM models.

The predictive MAE of all datasets under various privacy settings is presented in Table 2. The MAE of models trained with real data

²<https://github.com/vanderschaarlab/synthcity>

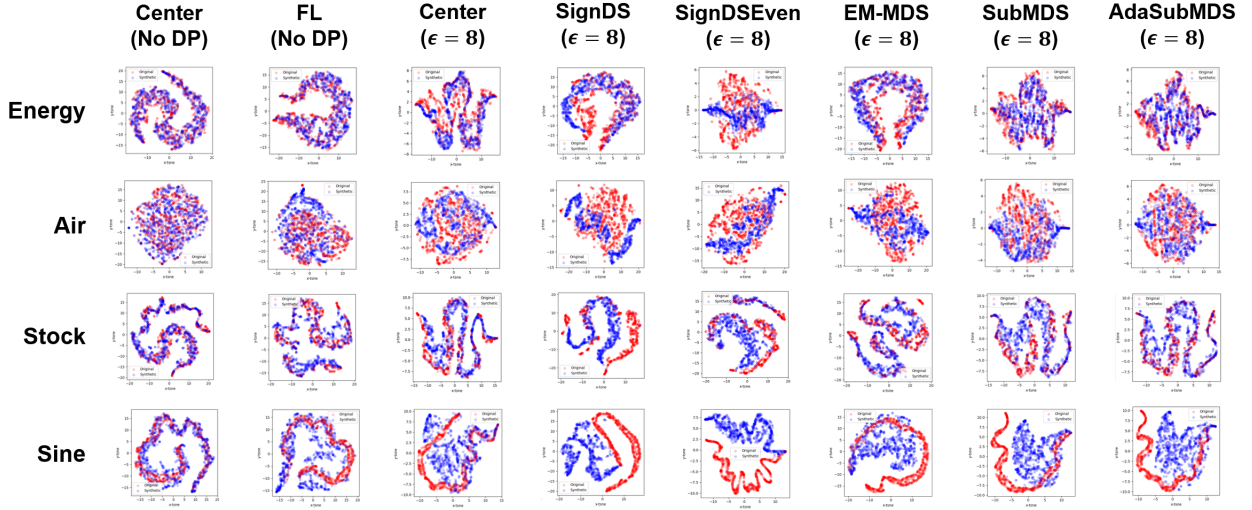


Figure 5: T-SNE visualization of the distribution of real and synthetic data on the four datasets. Each row represents the results for one dataset. Each column provides the distribution of synthetic data generated under non-private centralized and FL settings, as well as DP centralized setting and using different LDP-FL algorithms with $\epsilon = 8$. Red denotes the real data and blue denotes synthetic data.

is also included as a reference. It is evident that both SUBMDS and ADASUBMDS exhibit significantly lower predictive errors compared to the other two baselines, particularly for small privacy budgets. Specifically, with $\epsilon = 0.5$, the ADASUBMDS algorithm reduces the MAE by at least 20% compared to the SIGNDS and EM-MDS algorithms, and achieves a maximum reduction of 85% on the **Stock** dataset. This further demonstrates the effectiveness of our multi-dimensional selection algorithms in enhancing the performance of RWAEs and the quality of synthetic data. Furthermore, as ϵ increases, the MAE of the SUBMDS and ADASUBMDS algorithms gradually approaches that of the real data. With $\epsilon = 8$, the increase in MAE for both algorithms is less than 0.09 and only 0.002 at best when compared to models trained with real data. These results demonstrate that the synthetic data generated by our framework has higher usefulness compared to the baselines and can serve as a replacement for real data in AI training tasks.

6.2.4 Ablation Study. We further conduct a series of ablation studies under the ADASUBMDS setting to investigate how the split ratio of privacy budgets and the number of per-round clients will impact the framework’s performance.

Impact of the Split of Privacy Budgets. In previous experiments, we found that allocating 90% of the total privacy budget to dimension selection enhances the performance of ADASUBMDS compared to baselines under the same privacy budget. In this section, we further investigate how the different split ratios of privacy budgets impact the model performance. To this end, we conduct experiments under $\epsilon_{total} = \{2, 4, 8\}$ with ϵ_{idx} being 10%, 50%, and 90%, respectively, and compare the predictive MAE of synthetic data generated under different privacy settings. The results of **Energy** and **Air** datasets are shown in Figure 6. It can be observed that

for the same total privacy value ϵ , choosing a larger split ratio for dimension selection typically results in a lower MAE, namely better data utility. This suggests the model’s update direction is more crucial for achieving a satisfactory convergence in comparison to the update magnitude. Given that the dimension selection process significantly impacts the update direction, it should receive a higher proportion of privacy to reduce randomness during training. Additionally, for the magnitude perturbation, since we implement a quantization on the real value and only modify the global learning rate following T_{pat} rounds, it can withstand greater randomness and therefore be assigned less privacy.

Impact of the Per-round Clients. In addition to the split ratio of privacy budgets, we also analyze the impact of the number of participating clients in each training round on the framework’s performance. We choose 50, 100, and 200 per-round clients respectively, and train the model with $\epsilon \in \{2, 4, 8\}$. The MMD of the **Sine** and **Stock** datasets under different privacy settings is shown in Figure 7. For both datasets, it is evident that increasing the number of per-round clients further reduces the MAE. With $\epsilon \leq 4$, increasing the per-round clients from 50 to 200 results in around a 50% ~ 75% decrease in MMD. This outcome is due to the aggregation of privatized local updates on the server side. Consequently, when ϵ is small, using more per-round clients helps mitigate the impact of local randomization on the aggregated global update, thereby improving the convergence of the global model. Conversely, when ϵ is large, the randomness caused by the local protection process is already significantly reduced. Thus, the number of per-round clients does not significantly enhance the utility of synthetic data.

Impact of the Decay Rate and Patience Threshold. Finally, we study the impact of r and T_{pat} on the model convergence. We train the

Table 2: Comparison of synthetic data utility in AI training tasks. For each dataset, we respectively use real and synthetic data to train one-layer LSTM models for *next-step prediction* and evaluate the models’ predictive MAE on a *held-out* set of real data. Here we compare the MAE of synthetic data generated under different privacy settings.

Datasets	Sine			Stock			Energy			Air		
	Real	Center	FL	Real	Center	FL	Real	Center	FL	Real	Center	FL
Non-Private Setting	0.020	0.027	0.038	0.012	0.013	0.014	0.031	0.043	0.043	0.038	0.043	0.047
Central-DP Setting	$\epsilon=0.5$	$\epsilon=2$	$\epsilon=8$	$\epsilon=0.5$	$\epsilon=2$	$\epsilon=8$	$\epsilon=0.5$	$\epsilon=2$	$\epsilon=8$	$\epsilon=0.5$	$\epsilon=2$	$\epsilon=8$
	0.124	0.103	0.051	0.018	0.015	0.014	0.060	0.056	0.050	0.083	0.081	0.074
FL Setting	$\epsilon=0.5$	$\epsilon=2$	$\epsilon=8$	$\epsilon=0.5$	$\epsilon=2$	$\epsilon=8$	$\epsilon=0.5$	$\epsilon=2$	$\epsilon=8$	$\epsilon=0.5$	$\epsilon=2$	$\epsilon=8$
SignDS [16]	0.211	0.141	0.130	0.131	0.022	0.017	0.118	0.066	0.054	0.117	0.087	0.079
SignDSEven	0.210	0.145	0.127	0.130	0.021	0.017	0.116	0.065	0.054	0.116	0.087	0.078
EM-MDS [17]	0.196	0.141	0.110	0.118	0.021	0.015	0.108	0.062	0.053	0.113	0.085	0.071
SubMDS	0.178	0.122	0.107	0.033	0.016	0.014	0.084	0.054	0.051	0.097	0.082	0.070
AdaSubMDS	0.134	0.108	0.110	0.020	0.015	0.014	0.058	0.052	0.050	0.087	0.083	0.070

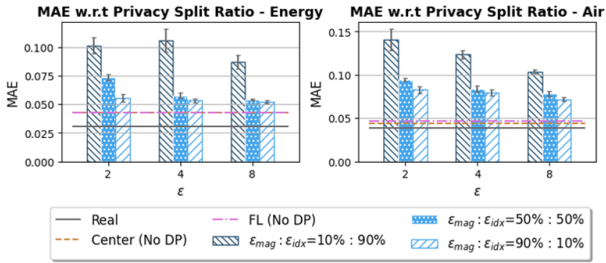


Figure 6: Analysis of the impact of privacy split ratio. For each dataset, we train the RWAE model with ϵ_{mag} being 10%, 50%, 90% of the total privacy ϵ and compare the MAE of synthetic data under different privacy settings.

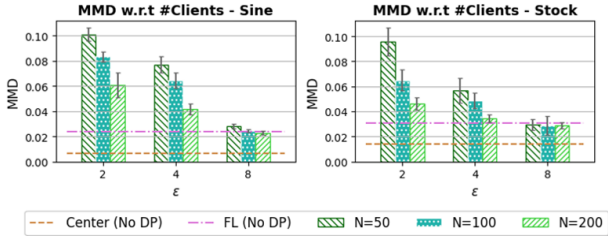


Figure 7: Analysis of the impact of per-round clients. For each dataset, we train the RWAE model with 50, 100, and 200 per-round clients and compare the MMD of synthetic data under different privacy settings.

RWAE with $r \in \{0.5, 0.9\}$ and $T_{pat} \in \{10, 50, 100\}$, and use the MMD of synthetic data after each round of aggregation for analysis. Intuitively, a smaller r leads to a faster decay of the learning rate, while a smaller T_{pat} implies a more frequent decay of the learning rate. Figure 8 illustrates the performance of the model on **Stock** and **Air** datasets with privacy budget $\epsilon = 6$. We observe that $r = 50$ and $T_{pat} = 10$ result in slower convergence and poorer model performance, as the learning rate reduces quickly at the beginning of training. Conversely, $r = 0.9$ and $T_{pat} = 100$ exhibit unstable

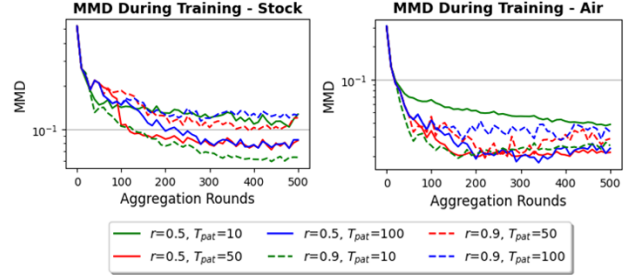


Figure 8: Analysis of the impact of decay rate r and patience threshold T_{pat} . For each dataset, we respectively use decay rate $r \in \{0.5, 0.9\}$ and patience threshold $T_{pat} \in \{10, 50, 100\}$ to train the RWAE model for 500 rounds under $\epsilon = 6$. Results show the MMD of synthetic data during the training process.

convergence, as the learning rate deviates significantly from the true magnitude, leading to a greater discrepancy in sparse updates. Furthermore, we notice that the ideal combination of r and T_{pat} may vary across datasets, suggesting potential improvement by tuning hyperparameters.

6.3 Evaluation of Privacy Protection

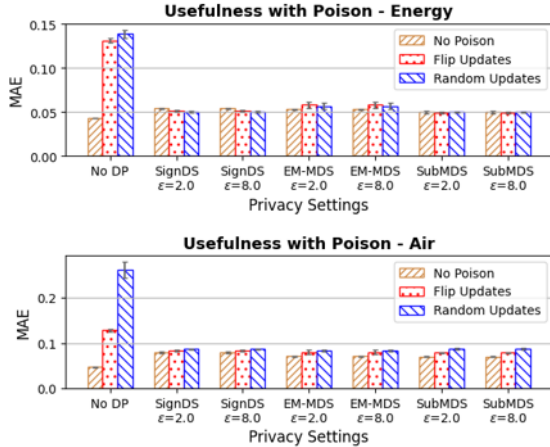
As for the privacy evaluation, we empirically analyze the performance of our framework in defending the membership inference attack (MIA). We follow the black-box MIA proposed in [13], where the attacker uses the distance between a target record and a published synthetic dataset to infer whether the record is used to train the generative model. Let x be a target record and \mathcal{X}_{syn} be the published synthetic dataset, we denote $\mathcal{U}_\rho(x) = \{x' | \Gamma(x, x') \leq \rho\}$ as the ρ -neighborhood of x under a certain distance metric Γ . The attacker then computes the ratio of synthetic records that fall into the neighborhood of x , namely

$$\mathcal{R}_\rho = \frac{1}{|\mathcal{X}_{syn}|} \sum_{x_{syn} \in \mathcal{X}_{syn}} \mathbb{1}[x_{syn} \in \mathcal{U}_\rho(x)], \quad (17)$$

where x_{syn} is an individual synthetic record in \mathcal{X}_{syn} and $|\mathcal{X}_{syn}|$ is the size of the synthetic dataset. Based on the intuition that the

Table 3: Averaged attack accuracy of the MIA attack under different privacy settings.

Datasets	FL (No DP)	$\epsilon = 8$	$\epsilon = 4$	$\epsilon = 0.5$
Sine	0.666	0.566	0.548	0.514
Stock	0.724	0.613	0.559	0.501
Energy	0.655	0.579	0.531	0.509
Air	0.706	0.621	0.587	0.539

**Figure 9: The effectiveness in defending against model poisoning attacks evaluated on Energy and Air datasets. For each dataset, we present the predictive MAE of synthetic data generated under different poisoning attacks and privacy settings.**

generative models tend to produce synthetic data that are more similar to the training data, the higher \mathcal{R}_ρ , the more likely that x is in the training data.

In our experiments, we create a target dataset with 100 training records and 100 testing records. For each experiment, we generate a synthetic dataset \mathcal{X}_{syn} with 10^5 records. We then use the Euclidean distance as metric Γ and calculate the ratio $\mathcal{R}_\rho(x)$ for each target record. Following [13], we set ρ as the median of the minimum distance for each target record. We sort the $\mathcal{R}_\rho(x)$ values for all target records, choose the 100 records with the highest $\mathcal{R}_\rho(x)$ to be the predicted members and compute the attack accuracy. We repeat each experiment 5 times and present the average attack accuracy under different privacy settings in Table 3. It is evident that synthetic data generated by the non-private RWAEs still reveal the membership information of the target record. The attack accuracy for all datasets is above 0.65, and it even exceeds 0.7 for **Stock** and **Air**. On the other hand, training the RWAEs with DP helps mitigate information leakage. With $\epsilon = 0.5$, the attack accuracy decreases by 0.15 ~ 0.22 and approaches 0.5, which is the performance of random guesses. Even with $\epsilon = 8$, the attack accuracy can still be reduced by 0.08 ~ 0.12. These results demonstrate that our framework reduces the risk of MIA and provides privacy protection for local data.

6.4 Evaluation of Robustness

Finally, we investigate whether the proposed SUBMDS algorithm can also help prevent poison attacks and improve the robustness of FL training. Here we focus on two untargeted attacks, namely *random updates* and *sign flip*, where a number of malicious local clients modify the real local updates in order to disturb the model convergence. More specifically, *random updates* replaces the real update with a random update, while *sign flip* changes the signs of update values. In the experiments, we presume a presence of 20% malicious clients and train the RWAE models using the non-private setting as well as the LDP-FL algorithms, including SIGNDS, EM-MDS and SUBMDS, with $\epsilon \in \{2, 8\}$. Then, we evaluate the predictive MAE of synthetic data under both model poisoning attacks. In Figure 9 we report the predictive MAE of the **Energy** and **Air** datasets. Noticeably, for RWAEs trained in the non-private setting, a significant increase in predictive MAE is observed on applying both poisoning attacks, suggesting that the training process is easily influenced by the attacks. In contrast, training RWAEs with LDP-FL algorithms effectively reduces the MAE. For both datasets, the MAE under the *sign flip* attack is generally reduced by 30% ~ 60%, while the MAE of the *random updates* attack is reduced by 60% ~ 70%. This is because these LDP-FL algorithms only select a subset of dimensions for each local update, which limits the impact of poisoned updates on model convergence and improves the robustness of FL.

7 DISCUSSION AND FUTURE WORK

In this section, we will discuss the limitations of this work and potential future research directions.

While our framework produces high-quality synthetic data compared to the baseline FL algorithms, there is still a utility gap in the non-private setting. Results in Section 6.2.4 show that increasing the number of per-round clients and choosing proper hyperparameters can enhance the utility of synthetic data. Additionally, exploring advanced LDP-FL algorithms and generative models can further enhance data utility while adhering to the main training workflow outlined in Section 5.3.

In addition, we empirically show in Section 6.3 and Section 6.4 that our SIGNMDS algorithm successfully defends against membership and model poisoning attacks. A comprehensive assessment of the algorithm’s resilience to other privacy and adversarial attacks would greatly contribute to the development of trustworthy federated learning frameworks in the future.

Finally, it is important to note that SUBMDS and MAGRR are not limited to specific model architectures or data types. They have a broad applicability as privacy-preserving algorithms in various federated learning scenarios.

8 CONCLUSION

In this paper, we propose FEDSTDG, an effective framework for synthesizing local multivariate time-series data with comprehensive privacy guarantees. The framework utilizes a time-series autoencoder to learn the distributions and correlations of real local data and generate high-fidelity synthetic data on the server side. Moreover, we improve state-of-the-art LDP-FL algorithms with two novel approaches, namely SUBMDS and MAGRR, which achieve a better

model and synthetic data utility under the same privacy guarantees. Extensive evaluation with real-world datasets demonstrates the capability and efficiency of our framework in synthesizing time-series data under strong privacy protection. The synthetic data preserves similar statistical properties as real data and can be easily scaled up for future data mining and AI training tasks.

REFERENCES

- [1] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. 2020. Generative Models for Effective ML on Private, Decentralized Datasets. In *8th International Conference on Learning Representations*. OpenReview.net, virtual.
- [2] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA, 1175–1191.
- [3] Luis M Candanedo, Véronique Feldheim, and Dominique Deramaix. 2017. Data Driven Prediction Models of Energy Use of Appliances in a Low-Energy House. *Energy and Buildings* 140 (2017), 81–97.
- [4] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. 2020. GS-WGAN: A Gradient-Sanitized Approach for Learning Differentially Private Generators. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Curran Associates, Inc., virtual, 12673–12684.
- [5] Andrew A. Cook, Goksel Misirli, and Zhong Fan. 2020. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet of Things Journal* 7, 7 (2020), 6481–6494.
- [6] Saverio De Vito, Ettore Massera, Marco Piga, Luca Martinotto, and Girolamo Di Francia. 2008. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical* 129, 2 (2008), 750–757.
- [7] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2018. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association* 113, 521 (2018), 182–201.
- [8] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3–4 (2014), 211–407.
- [9] Yahoo! Finance. 2004. Alphabet Inc. (GOOG) Stock Historical Prices and Data. <https://finance.yahoo.com/quote/GOOG/history/>.
- [10] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. 2019. Differentially Private Generative Adversarial Networks for Time Series, Continuous, and Discrete Open Data. In *International Conference on ICT Systems Security and Privacy Protection*, Vol. 562. Springer, Lisbon, Portugal, 151–164.
- [11] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting Gradients – How easy is it to break privacy in federated learning?. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Curran Associates, Inc., virtual, 16937–16947.
- [12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Montreal, Canada, 2672–2680.
- [13] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. 2019. Monte Carlo and Reconstruction Membership Inference Attacks against Generative Models. *Proceedings on Privacy Enhancing Technologies* 2019, 4 (2019), 232–249.
- [14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.
- [15] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. 2018. Distributed learning without distress: Privacy-preserving empirical risk minimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Montréal, Canada, 6343–6354.
- [16] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. Privacy-Preserving High-dimensional Data Collection with Federated Generative Autoencoder. *Proceedings on Privacy Enhancing Technologies* 2022, 1 (2022), 481–500.
- [17] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. SignDS-FL: Local differentially private federated learning with sign-based dimension selection. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 5 (2022), 1–22.
- [18] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. 2019. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *7th International Conference on Learning Representations*. OpenReview.net, New Orleans, USA.
- [19] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2011. What Can We Learn Privately? *SIAM Journal on Computing* 40, 3 (2011), 793–826.
- [20] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang. 2018. Secure Logistic Regression based on Homomorphic Encryption. *IACR Cryptol. ePrint Arch.* 2018 (2018), 74.
- [21] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations*. OpenReview.net, Banff, AB, Canada.
- [22] Haoran Li, Li Xiong, and Xiaoqian Jiang. 2014. Differentially private synthesis of multi-dimensional data using copula functions. In *Proceedings of the 17th International Conference on Extending Database Technology*, Vol. 2014. OpenProceedings.org, Athens, Greece, 475–486.
- [23] Claire Little, Mark Elliot, and Richard Allmendinger. 2023. Federated Learning for Generating Synthetic Data: A Scoping Review. *International Journal of Population Data Science* 8, 1 (2023).
- [24] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. 2020. FedSel: Federated SGD Under Local Differential Privacy with Top-k Dimension Selection. In *25th International Conference of Database Systems for Advanced Applications*. Springer, Jeju, South Korea, 485–501.
- [25] Sourav Majumdar and Arnab Kumar Laha. 2020. Clustering and classification of time series using topological data analysis with applications to finance. *Expert Systems with Applications* 162 (2020), 113868.
- [26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Vol. 54. PMLR, Fort Lauderdale, USA, 1273–1282.
- [27] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *2019 IEEE Symposium on Security and Privacy*. IEEE, San Francisco, CA, USA, 739–753.
- [28] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1071–1083.
- [29] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *4th International Conference on Learning Representations*. Yoshua Bengio and Yann LeCun (Eds.). OpenReview.net, San Juan, Puerto Rico.
- [30] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1985. *Learning internal representations by error propagation*. Technical Report. University of California, San Diego; Institute for Cognitive Science.
- [31] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. 2018. Wasserstein Auto-Encoders. In *6th International Conference on Learning Representations*. OpenReview.net, Vancouver, BC, Canada.
- [32] Aleksei Triastcyn and Boi Faltings. 2020. Federated Generative Privacy. *IEEE Intelligent Systems* 35, 4 (2020), 50–57.
- [33] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. ACM, London, UK, 1–11.
- [34] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008), 2579–2605.
- [35] Curtis R. Vogel. 2002. *Computational Methods for Inverse Problems*. Society for Industrial and Applied Mathematics, Philadelphia, USA.
- [36] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and Analyzing Multidimensional Data with Local Differential Privacy. In *35th IEEE International Conference on Data Engineering*. IEEE, Macao, China, 638–649.
- [37] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. In *2019 IEEE Conference on Computer Communications*. IEEE, Paris, France, 2512–2520.
- [38] Susan Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association* 60, 309 (1965), 63–69.
- [39] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially Private Generative Adversarial Network. *CoRR* abs/1802.06739 (2018).
- [40] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019. Time-series Generative Adversarial Networks. In *33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Vancouver, BC, Canada, 5509–5519.
- [41] Abdelhafid Zeroual, Fouzi Harrou, Abdelkader Dairi, and Ying Sun. 2020. Deep learning methods for forecasting COVID-19 time-series data: A Comparative study. *Chaos, Solitons & Fractals* 140 (2020), 110121.
- [42] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private data release via Bayesian networks. *ACM Transactions on Database Systems* 42, 4 (2017), 25:1–25:41.
- [43] Yang Zhao, Jun Zhao, Mengmeng Yang, Teng Wang, Ning Wang, Lingjuan Lyu, Dusit Niyato, and Kwok-Yan Lam. 2020. Local differential privacy based federated learning for Internet of Things. *IEEE Internet of Things Journal* 8, 11 (2020), 8836–8853.

4 Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data

Authors	Xue Jiang ^{1,2} (xue.jiang@tum.de) Yufei Zhang ¹ (yufei.zhang@tum.de) Xuebing Zhou ² (xuebing.zhou@huawei.com) Jens Grossklags ¹ (jens.grossklags@in.tum.de)
	¹ Technical University of Munich (TUM), Boltzmannstraße 3, 85748 Garching, Germany ² Huawei Munich Research Center, Riesstraße 25, 80992 Munich, Germany
Type	Journal
Outlet	Proceedings on Privacy Enhancing Technologies, Volume 2023 ³
Ranking	CORE 2021 ⁴ : A
Status	Published
DOI	https://doi.org/10.56553/popets-2023-0050
Citation	Jiang, X., Zhang, Y, Zhou, X., & Grossklags, J. (2023). Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data. Proceedings on Privacy Enhancing Technologies, 2023(2), 236-250.
Copyright	This work is published under a Creative Commons Attribution 4.0 (CC BY 4.0) License ⁵ .
Author Contributions	Xue Jiang developed the core idea for the paper and took primary responsibility for designing the methodology. Yufei Zhang was responsible for acquiring data, implementing experiments and evaluating results. The manuscript was drafted by Xue Jiang, and Xuebing Zhou and Jens Grossklags provided valuable feedback and suggestions on the methodology and experiments, and assisted in reviewing and enhancing the manuscript.

³<https://www.petsymposium.org/popets/2023/>

⁴<http://portal.core.edu.au/conf-ranks/1442/>

⁵<https://creativecommons.org/licenses/by/4.0/>

Distributed GAN-Based Privacy-Preserving Publication of Vertically-Partitioned Data

Xue Jiang

xue.jiang@tum.de

Technical University of Munich
Garching, Germany

Xuebing Zhou

xuebing.zhou@huawei.com

Huawei Munich Research Center
Munich, Germany

Yufei Zhang

yufei.zhang@tum.de

Technical University of Munich
Garching, Germany

Jens Grossklags

jens.grossklags@in.tum.de

Technical University of Munich
Garching, Germany

ABSTRACT

In the era of big data, user data are often vertically partitioned and stored at different local parties. Exploring the data from all the local parties would enable data analysts to gain a better understanding of the user population from different perspectives. However, the publication of vertically-partitioned data faces a dilemma: on the one hand, the original data cannot be directly shared by local parties due to privacy concerns; on the other hand, independently privatizing the local datasets before publishing may break the potential correlation between the cross-party attributes and lead to a significant utility loss. Prior solutions compute the privatized multivariate distributions of different attribute sets for constructing a synthetic integrated dataset. However, these algorithms are only applicable for low-dimensional structured data and may suffer from large utility loss with the increase in data dimensionality.

Following the idea of synthetic data generation, we propose VERTIGAN, the first framework based on a generative adversarial network (GAN) for publishing vertically-partitioned data with privacy protection. The framework adopts a GAN model comprised of one multi-output global generator and multiple local discriminators. The generator is collaboratively trained by the server and local parties to learn the distribution of all parties' local data and is used to generate a high-utility synthetic integrated dataset on the server side. Additionally, we apply differential privacy (DP) during the training process to ensure strict privacy guarantees for the local data. We evaluate the framework's performance on a number of real-world datasets containing 68–1501 classification attributes and show that our framework is more capable of capturing joint distributions and cross-attribute correlations compared to statistics-based baseline algorithms. Moreover, with a privacy guarantee of $\epsilon = 8$, our framework achieves around a 2% ~ 15% improvement in classification accuracy compared to the baseline algorithms. Extensive experimental results demonstrate the capability and efficiency of our framework in synthesizing vertically-partitioned data while striking a satisfactory utility-privacy balance.

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit <https://creativecommons.org/licenses/by/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Proceedings on Privacy Enhancing Technologies 2023(2), 236–250

© 2023 Copyright held by the owner/author(s).

<https://doi.org/10.56553/popets-2023-0050>



KEYWORDS

Differential privacy, vertically-partitioned data, synthetic data

1 INTRODUCTION

With the rapid development of network and computer technologies, large and diverse quantities of user data have been extensively collected and stored by different companies and institutes (referred to as local parties). These data usually contain rich information characterizing user profiles, which is valuable for data mining and building AI services. Due to the variety in service scenarios, the user data are often vertically partitioned and distributed among these local parties. That is, the local dataset held by each party usually contains different attributes of the same group of users. Considering that the more attributes the data consist of, the more information can be used for describing an individual user, it is practical for local parties to collaborate with each other and publish an integrated dataset with all the attributes for better decision making or building high-accuracy services. For instance, in a healthcare scenario, a group of specialist hospitals could publish a joint dataset to study potential correlations between different types of illnesses such as cancer, and heart and lung diseases. Similarly, in a smart finance scenario, a loan company could use a dataset jointly published by a bank and an e-commerce company to more deeply explore the key attributes that may result in higher default risk. More generally, integrating and analyzing these vertically-partitioned datasets enables data analysts to explore the hidden correlations of attributes from different perspectives and thus obtain a better understanding of the characteristics of user groups. This can be of significant help in designing optimized data mining algorithms and machine learning models.

However, publishing vertically-partitioned datasets has to be cognizant of the restrictions of data protection regulations such as the GDPR and users' privacy concerns. On the one hand, since the local data are generated based on users' ongoing behaviors and may contain sensitive information of individual users, directly sharing the original local datasets with an untrusted third party may lead to serious privacy leakage (see, for example, [5, 7]). On the other hand, the local parties can use state-of-the-art privacy-enhancing techniques, such as differential privacy (DP) [18], to process the real data and only share the privatized datasets. Nevertheless, each party individually privatizing the local data may break the correlations and joint distributions among attributes held by

different parties and lead to distinctive utility loss in the published dataset. Therefore, solutions for publishing vertically-partitioned data under a satisfactory privacy-utility balance are greatly needed.

In comparison to the substantial attention given to privacy-preserving data mining and machine learning under a vertical setting, algorithms for publishing the vertically-partitioned data are still barely studied. Prior works [26, 36] proposed two-party publication protocols under k -anonymity guarantees [52]. Unfortunately, later studies [51, 62] pointed out that k -anonymity models are vulnerable to various privacy attacks and cannot provide sufficient privacy protection. Follow-up work [44] proposed the first algorithm for publishing vertically-partitioned data under DP guarantees. However, the algorithm is limited to two-party scenarios and requires pre-defined taxonomy trees for all categorical attributes. Recent work by Tang *et al.* [53] proposed to use a latent tree model [70] to represent the cross-attribute distributions in the original dataset and privatizes the latent tree parameters via a distributed Laplace protocol to achieve ϵ -DP for each local dataset. Although the work by Tang *et al.* [53] effectively improves data utility and efficiency compared to [44], the algorithm evenly splits the privacy budget to all the attribute pairs. Therefore, the noise scale may increase exponentially with the data dimensionality and cause significant utility loss. Moreover, the algorithm is limited to discrete structured datasets and cannot support other data types.

In recent years, data synthesis has increasingly been considered a useful approach for addressing data insufficiency problems in developing AI applications. With the strong capabilities of characterizing the correlations and distributions of high-dimensional data, deep generative models such as generative adversarial networks (GANs) are increasingly used for generating high-utility and low-sensitivity synthetic data. Although some recent works (*e.g.*, [28, 54]) also proposed training the generative models under the federated learning (FL) framework to avoid the direct collection of real local data, the solutions all focus on the horizontal setting, which cannot be directly applied to vertically-partitioned data.

In this paper, we address this research gap and propose VERTIGAN, the first GAN-based framework for privacy-preserving publication of vertically-partitioned data. The framework adopts a distributed GAN architecture, comprised of a global generator and multiple local discriminators. By using a collaborative training strategy, the global generator is trained without accessing the real local data. Moreover, we adopt a multi-output structure for the generator, which enables the model to directly learn the correlations and distributions of the attributes held by different local parties and generate synthetic integrated data. Finally, we inject DP perturbation during the training process, which ensures that the generator and the synthetic data satisfy strict DP guarantees for each local party. The main contributions of our approach are as follows:

- We propose VERTIGAN, an efficient and privacy-preserving framework for publishing vertically-partitioned data. The framework trains a multi-output global generator to directly learn the distribution of all parties' local data and to generate high-utility synthetic integrated data on the server side. To the best of our knowledge, this is the first framework based on a deep generative model for private data publication under the vertical setting.

- We introduce a distributed training strategy, where the global generator is updated based on the gradients calculated by the local discriminators. The strategy eliminates the need to access real local data when training the global generator. Moreover, we apply DP perturbation during the training process to provide a strict privacy guarantee for each local dataset.
- We implement our framework and evaluate the performance on a number of real-world datasets containing 68–1501 classification attributes. Through comparison with the previous statistics-based algorithms, we show that the synthetic data generated by our framework always preserve much closer joint distributions and correlations to real data. Moreover, with a local privacy guarantee $\epsilon = 8$, we achieve around 2% ~ 15% improvement in classification accuracy compared to the baseline algorithms. Extensive evaluation experiments show that our framework has outperforming capability and efficiency in collecting high-dimensional data while offering a favorable utility-privacy balance.

2 RELATED WORK

2.1 Data Analysis on Vertically-Partitioned Data

In recent decades, data analysis on vertically-partitioned data has attracted increasing attention. Different from the horizontal setting, vertical partitioning refers to the scenario that local parties collect different attributes of the same set of users. Existing applications on vertically-partitioned data include, for instance, jointly training ML models using attributes of all the local parties, or publishing an integrated dataset for future data mining.

2.1.1 Machine Learning Under Vertical Setting. In the context of ML, prior studies by Vaidya *et al.* proposed a series of secure multi-party computation (SMC) protocols [66] for training different models on vertically-partitioned data, including Bayes classifier [55], and decision trees [56], etc. Hardy *et al.* [22] proposed a vertical federated learning (VFL) framework, which trained LR models using homomorphic encryption (HE) [14]. Yang [65] further applied the quasi-Newton method in VFL to reduce the number of communication rounds. Some other works [12, 63] also proposed solutions for tree-based models and neural networks [48]. Besides using crypto-based technologies such as HE and SMC to ensure security in VFL, recent works [11, 59] further proposed to incorporate DP into the training process to provide strict privacy guarantees for local data.

On the other hand, some recent works also investigate potential privacy attacks against VFL, which include label inference attacks and feature reconstruction attacks. In the label inference attacks, the parties without ground-truth labels aim to use the back-propagated gradients to infer the sample labels. Several existing attacks proposed to explore the difference of the gradient norms [39] or the sign of the last-layer gradients [40, 73]. Other research [19] also proposed a semi-supervised learning approach that first estimated the bottom-layer parameters and then used the “completed” model to “generate” the label of arbitrary samples. Apart from the label leakage, some other works [27, 41] also studied the feature leakage

in VFL, where the party obtaining the model predictions tries to reconstruct the input features of other parties. Nevertheless, existing attacks against VFL only focused on classification models, where the attackers either try to infer the ground-truth labels or need to use the model predictions to reconstruct local features. In contrast, in this paper, we use the GAN model for data synthesis, which does not involve such label (or prediction) information. Hence, the above-mentioned attacks in VFL are no more applicable.

2.1.2 Data Publication Under Vertical Setting. Compared to the extensive set of studies on machine learning under the vertical setting, there are still only limited prior works on publishing vertically-partitioned data. Prior works in [26, 36] proposed SMC-based protocols for two-party data publication under k -anonymity guarantees [52]. Nevertheless, later studies [51, 62] pointed out that k -anonymity models are vulnerable to various privacy attacks and cannot provide sufficient privacy protection. In contrast, DP [18] is considered as a more principled approach for private data publication. Mohammed *et al.* proposed DistDiffGen [44], the first algorithm for publishing vertically-partitioned data under DP guarantees. DistDiffGen first generalizes the raw data using a distributed exponential mechanism and then adds noise to the distributions to ensure ϵ -DP. However, the algorithm is limited to two-party scenarios and requires pre-defined taxonomy trees for all categorical attributes, which may not always be available in practice. Later work by Tang *et al.* [53] proposed an improved differentially private latent tree (DPLT) algorithm, which first uses a latent tree model [70] to represent the cross-attribute distributions in the original dataset and then privatizes the latent tree parameters via a distributed Laplace protocol to achieve ϵ -DP for each local dataset. The latent tree model will then be used for generating a synthetic dataset. Although [53] significantly improves the data utility and efficiency in comparison to [44], it is still limited to discrete attributes. Moreover, since the privacy budget is evenly split over all the attribute pairs, the noise scale may increase exponentially with the increased data dimensionality and cause a large utility loss.

In this paper, we propose a distributed GAN-based protocol for publishing vertically partitioned data in a private manner. Compared to previous works, our solution can support the publication of high-dimensional datasets with strict DP guarantees. Moreover, the framework can be further extended to support other types of data such as numerical data and images.

2.2 Differentially-Private Data Synthesis

DP data synthesis has been extensively studied over recent years as one of the solutions for privacy-preserving data publishing. Previous statistics-based works [38, 68] computed joint distributions of original structured data under DP guarantees and used them to generate synthetic datasets. However, these methods can only be applied to structured data and may suffer from a significant utility loss with the increase in data dimensionality.

Inspired by the rapid evolution of deep learning, later works proposed to directly train generative models such as autoencoders [3, 35] and generative adversarial networks (GANs, [20]) and to generate high-utility synthetic data. Nevertheless, simply training these generative models without protection may still lead to privacy leakage. For instance, prior work [4, 57] showed that GANs may

unintentionally memorize the training data. Moreover, Hayes *et al.* [23] proposed different membership inference attacks against the trained generator and discriminators. Later works also demonstrated that the membership information can be revealed from the generated synthetic data [10, 24, 50]. In addition, Zhou *et al.* [72] performed a property inference attack, which uses the released synthetic data to infer the macro-level information of training data (e.g., the ratio of samples regarding a certain property).

DP has been considered one of the countermeasures against such privacy attacks. Existing DP data synthesis algorithms are generally divided into two categories, namely by using differentially-private stochastic gradient descent (DPSGD, [1]) or private aggregation of teacher ensembles (PATE, [45]). The DPSGD-based algorithms [64, 71] perturb the model gradients in each iteration by clipping and adding Gaussian noise to ensure DP guarantees. The PATE-based algorithms [31, 58] first train a group of teacher models (e.g., the discriminator in GAN) on non-overlapping subsets of original data and then use the noisy predictions from the teacher group to train the student model (e.g., the generator). Nevertheless, previous data synthesis algorithms mainly focus on the centralized setting, where the server has already collected the clients' real data. This may not always be realistic since the clients may refuse to share their personal local data with untrusted servers. Therefore, some recent works also proposed to train the generative autoencoders [28] and GANs [54, 69] under the FL framework to avoid the collection of original data. However, existing solutions only focus on the horizontal setting, where the local data shares the same set of attributes. In contrast, in this paper, we conduct the first attempt at the GAN-based DP data synthesis for vertically-partitioned data.

3 BACKGROUND

3.1 Differential Privacy

DP [18] is a state-of-the-art anonymization technique that provides rigorous privacy guarantees for data analysis. The classic definition of DP is as follows:

DEFINITION 1 ((ϵ, δ) -DP [18]). *A randomized mechanism \mathcal{M} satisfies (ϵ, δ) -DP if for any two adjacent datasets $\mathcal{X}, \mathcal{X}'$ differing in one data sample and any measurable subset of outputs $\mathcal{Y} \subseteq \text{range}(\mathcal{M})$ we have*

$$\Pr [\mathcal{M}(\mathcal{X}) \in \mathcal{Y}] \leq e^\epsilon \cdot \Pr [\mathcal{M}(\mathcal{X}') \in \mathcal{Y}] + \delta, \quad (1)$$

where ϵ is the privacy loss and δ is the probability of privacy leakage. When $\delta = 0$, we have ϵ -DP.

The original DP defined an upper bound of the privacy cost. Recent works further proposed various relaxations of DP to achieve tighter bounds for the privacy cost, especially for iterative algorithms. One of the widely used definitions is Rényi DP (RDP) [43], which uses the Rényi divergence to measure the distance between two probabilities. The definition of RDP is as follows:

DEFINITION 2 ($(\alpha, \epsilon(\alpha))$ -RDP [43]). *A randomized mechanism \mathcal{M} satisfies $(\alpha, \epsilon(\alpha))$ -RDP if for any two adjacent datasets $\mathcal{X}, \mathcal{X}'$ differing in one data sample, the Rényi α -divergence between $\mathcal{M}(\mathcal{X})$ and $\mathcal{M}(\mathcal{X}')$ satisfies*

$$\mathcal{D}_\alpha(\mathcal{M}(\mathcal{X})||\mathcal{M}(\mathcal{X}')) \triangleq \frac{1}{\alpha - 1} \log \mathbb{E} \left[\left(\frac{\mathcal{M}(\mathcal{X})}{\mathcal{M}(\mathcal{X}')} \right)^\alpha \right] \leq \epsilon. \quad (2)$$

Similar to DP, a Gaussian mechanism can also be used to achieve $(\alpha, \epsilon(\alpha))$ -RDP:

DEFINITION 3 (GAUSSIAN MECHANISM). For a real-valued function $f : \mathcal{X} \rightarrow \mathbb{R}^d$ with l_2 sensitivity Δ_f defined as

$$\Delta_f = \max_{\mathcal{X}, \mathcal{X}'} \|f(\mathcal{X}) - f(\mathcal{X}')\|_2 \quad (3)$$

over all adjacent datasets \mathcal{X} and \mathcal{X}' . The following Gaussian mechanism \mathcal{M}_σ satisfies $(\alpha, \epsilon(\alpha))$ -RDP:

$$\mathcal{M}_\sigma(x) = f(x) + \mathcal{N}(0, \sigma^2 I), \text{ where } \epsilon(\alpha) = \frac{\Delta_f^2 \alpha}{2\sigma^2}. \quad (4)$$

Moreover, RDP also preserves the composition property for accumulating the privacy cost over a sequence of mechanisms. Namely:

THEOREM 1 (COMPOSITION PROPERTY). Suppose n mechanisms $\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$ respectively satisfy $(\alpha, \epsilon_i(\alpha))$ -RDP, and are sequentially computed on the same set of private data \mathcal{X} , then a mechanism formed by $(\mathcal{M}_1, \dots, \mathcal{M}_n)$ satisfies $(\alpha, \sum_{i=1}^n \epsilon_i(\alpha))$ -RDP.

THEOREM 2 (ROBUSTNESS TO POST-PROCESSING). Let \mathcal{M} be an $(\alpha, \epsilon(\alpha))$ -RDP mechanism and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{M}$ also satisfies $(\alpha, \epsilon(\alpha))$ -RDP.

Additionally, the accumulated privacy cost under RDP can be further amplified by the subsampled mechanism:

LEMMA 1 (RDP FOR SUBSAMPLED MECHANISM [60]). Given a dataset of n points drawn from a domain \mathcal{X} and a randomized mechanism \mathcal{M} that takes an input from \mathcal{X}^m for $m \leq n$, let the randomized algorithm $\mathcal{M} \circ \text{subsample}$ be defined as: (1) subsample: subsample without replacement m data points of the dataset (sampling parameter $\gamma = m/n$), and (2) apply \mathcal{M} : a randomized algorithm taking the subsampled dataset as the input. For all integers $\alpha \geq 2$, if \mathcal{M} obeys $(\alpha, \epsilon(\alpha))$ -RDP, then the new randomized algorithm $\mathcal{M} \circ \text{subsample}$ obeys $(\alpha, \epsilon'(\alpha))$ -RDP where

$$\begin{aligned} \epsilon'(\alpha) \leq & \frac{1}{\alpha-1} \log \left(1 + \gamma^2 \binom{\alpha}{2} \min \{4(e^{\epsilon(2)} - 1), \right. \\ & \left. e^{\epsilon(2)} \min \{2, (e^{\epsilon(\infty)} - 1)^2\} \right) \\ & + \sum_{j=3}^{\alpha} \gamma^j \binom{\alpha}{j} e^{(j-1)\epsilon(j)} \min \{2, (e^{\epsilon(\infty)} - 1)^j\}. \end{aligned} \quad (5)$$

Finally, the privacy guarantees under RDP can be converted to the original DP guarantees:

LEMMA 2 (RDP TO DP [43]). If a mechanism \mathcal{M} satisfies $(\alpha, \epsilon(\alpha))$ -RDP, then \mathcal{M} satisfies $(\epsilon(\alpha) + \frac{\log 1/\delta}{\alpha-1}, \delta)$ -DP for any $\delta \in (0, 1)$.

3.2 Generative Adversarial Network

The GAN [20] is a class of unsupervised learning algorithms that have been extensively studied in the last decade due to its strong capability in generating high-fidelity synthetic data. A GAN model usually consists of a generator G and a discriminator D . The generator G takes as input a random noise z from a certain latent distribution P_z and generates synthetic data $\tilde{x} = G(z)$. The discriminator D learns to distinguish between data drawn from the real

distribution $x \sim P_r$ and from the synthetic distribution $\tilde{x} \sim P_g$, where P_g is determined by G and P_z . This can be considered a binary classification task. Both models are trained simultaneously through an adversarial process, where the generator keeps improving the quality of the synthetic data to fool the discriminator while the discriminator tries to discriminate between real and synthetic data with high accuracy. The ultimate goal is to approximate the real distribution P_r with the synthetic distribution P_g such that the discriminator cannot correctly distinguish between the real and the synthetic data. The problem can be formulated as a min-max training process with the following objective [20]:

$$\mathcal{L}_{GAN} = \mathbb{E}_{x \sim P_r} [\log D(x)] + \mathbb{E}_{\tilde{x} \sim P_g} [\log(1 - D(\tilde{x}))], \quad (6)$$

where P_r is the distribution of real data and P_g is the distribution of synthetic data $\tilde{x} = G(z)$ with $z \sim P_z$.

By utilizing different generator and discriminator structures, GANs have been adjusted to generate various types of synthetic data such as tabular data [46], images [33], and time-series data [67]. Nevertheless, the original GAN models usually suffer problems such as training instability and failure to converge. Therefore, some other works proposed to modify the loss function to improve the model convergence. The Wasserstein GAN (WGAN) [3, 61] is one of the well-known improved GANs. In comparison with the original loss function, WGAN-GP uses the Wasserstein-1 distance with an additional gradient norm penalty to achieve Lipschitz continuity. Given the real data x , the input noise $z \sim P_z$ and the synthetic data $\tilde{x} = G(z)$, the gradient penalty term can be written as

$$(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2, \text{ where } \hat{x} = \mu x + (1 - \mu)\tilde{x}. \quad (7)$$

Here \hat{x} is a weighted average between the real and synthetic data and $\mu \sim \mathcal{U}(0, 1)$ is a randomly sampled weight. Thus, the loss function for the generator and discriminator is formulated as follows:

$$\mathcal{L}_G = D(\tilde{x}) = D(G(z)), \quad (8)$$

$$\mathcal{L}_D = D(x) - D(\tilde{x}) + \lambda (\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2, \quad (9)$$

where λ is the weight for the gradient penalty.

In this paper, we choose P_z to follow the standard Gaussian distribution $\mathcal{N}(0, I)$ and $\lambda = 10$ for the gradient penalty. Similar to Equation (6), the loss function of WGAN can be formulated as:

$$\mathcal{L}_{WGAN} = \mathbb{E}_{\tilde{x} \sim P_g} [D(\tilde{x})] - \mathbb{E}_{x \sim P_r} [D(x)] + \lambda \mathbb{E}_{\hat{x} \sim P_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2]. \quad (10)$$

4 PROBLEM STATEMENT

In this paper, we focus on the scenario where the user data are vertically partitioned and distributed over multiple local parties. Each party possesses a different set of attributes of the same group of samples. A central server aims to integrate these local datasets in a private manner and publish a joint dataset containing all the attributes. The joint dataset will be further used by external data analysts for downstream data mining and model training tasks.

An illustration of the system setting is shown in Figure 1. We assume there are M local parties $\mathcal{P}_1, \dots, \mathcal{P}_M$. Each party \mathcal{P}_i has a local dataset containing a *different* set of attributes $A^i = \{a_1^i, \dots, a_{|A^i|}^i\}$. Here, the attribute sets can be either partially overlapping or non-overlapping. Moreover, each party may hold samples not

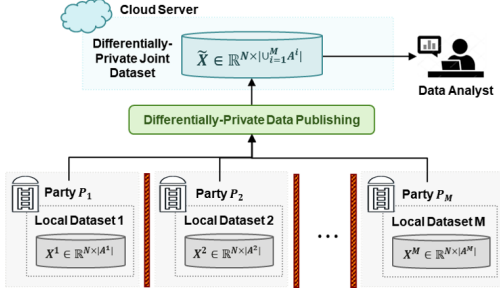


Figure 1: Overview of the system model.

covered by other parties. Therefore, we assume that the local data have certain alignable sample IDs (e.g., ID number, cellphone number, etc.). The local parties can use private set intersection (PSI) protocols (e.g., [13, 15, 25]) to determine the intersecting sample IDs without exposing the non-intersect samples. Then, each party sorts the common samples according to their IDs and obtains the final training dataset $X^i \in \mathbb{R}^{N \times |A^i|}$, where N is the number of samples and $|A^i|$ is the number of attributes.

The goal of the task is to design a privacy-preserving framework, where a central server can collaborate with all the local parties and publish a private joint dataset $\tilde{X} \in \mathbb{R}^{N \times |\cup_{i=1}^M A^i|}$ that contains the full set of attributes. The joint dataset \tilde{X} preserves both single-party and cross-party attribute correlations. More specifically, consider local parties \mathcal{P}_i and \mathcal{P}_j respectively holding local datasets $X^i \in \mathbb{R}^{N \times |A^i|}$ and $X^j \in \mathbb{R}^{N \times |A^j|}$, then the distribution of \tilde{X} should satisfy

$$P_{\tilde{X}}(A^i) \approx P_{X^i}(A^i), \quad P_{\tilde{X}}(A^i, A^j) \approx P_{X^i, X^j}(A^i, A^j). \quad (11)$$

Following previous works, we assume that the local parties and the central server are *honest-but curious*, who correctly follow the protocols but try to infer sensitive information of other local datasets. Moreover, we also consider the threat posed by external data analysts, who aim to use the published joint dataset to re-identify sensitive information of specific users. Based on the considerations above, it is required that there is no information exchange among local parties and each party does not know the attribute set of other parties. Moreover, we assume that the server cannot directly access the raw local data but is aware of the full attribute set and the size of the training dataset. Finally, the published dataset should satisfy strict DP guarantees and not reveal the privacy of individual users in the local datasets.

5 PROPOSED FRAMEWORK

Although previous works proposed statistics-based algorithms for publishing vertically-partitioned data under DP guarantees, the solutions are only limited to low-dimensional structured data and may suffer from large utility loss with the increase in domain size. Following the idea of data synthesis, we propose VERTIGAN, the first GAN-based framework for differentially-private publication of vertically-partitioned data. The overall workflow of the framework is presented in Figure 2, which consists of two phases, namely the *collaborative training process* and the *synthetic data generation*

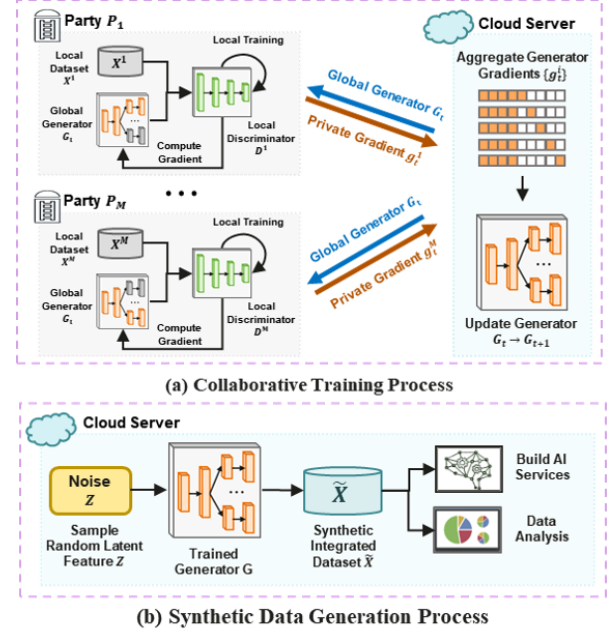


Figure 2: General workflow of the VERTIGAN framework.

process. In the first process, a GAN model is collaboratively trained by the server and all the local parties to learn the correlations and distributions of all the local datasets in a private manner. In the second phase, the generator part is used to directly generate synthetic integrated data that contains attributes held by all the local parties. The synthetic data preserves similar statistical properties to real data and can be alternatively used for downstream data analysis and AI training tasks.

Nevertheless, training the GAN model on distributed vertically-partitioned data faces several challenges. To start with, in this paper, we focus on the scenario where the real data are distributed on the local side and cannot be directly shared with the server. Hence, the model cannot be simply trained as in the centralized setting due to *data inaccessibility*. Moreover, in the vertical setting, the attribute sets held by the local parties are usually different from each other, which is referred to as *attribute inconsistency* in this paper. This causes existing solutions that train GANs in the horizontal FL framework to be inapplicable. Finally, recent contributions (e.g., [9, 49]) point out that the ML models may memorize information in training data and suffer from different privacy attacks. Therefore, *privacy protection* techniques should be applied during model training to prevent potential privacy leakage. We apply corresponding solutions in the VERTIGAN framework to address the above-mentioned challenges. In the following sections, we will respectively introduce each solution in detail.

5.1 Distributed GAN Against Data Inaccessibility

Different from other generative models, GANs are usually built with two independent networks, namely a generator and a discriminator.

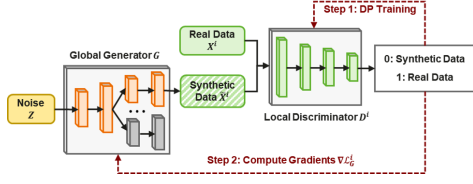


Figure 3: Workflow of the local training process.

The two networks are trained in an adversarial manner to improve their own performance. By taking advantage of GANs' separate generator-discriminator architecture, the VERTIGAN framework applies a distributed training strategy to address *data inaccessibility* problems. More specifically, the framework deploys a global generator G on the server side and multiple discriminators $\{D^1, \dots, D^M\}$ on the local side. The global generator takes in random latent features and outputs synthetic data for each local party, while the local discriminators are trained on the local side to distinguish between real data and synthetic data. The ultimate goal of the framework is to obtain a well-trained global generator on the server side that is capable of producing high-utility synthetic data without violating the privacy of real local data.

The training process is conducted in cooperation with the server and all the local parties, as shown in Figure 2. During each global training round, the server broadcasts the current global generator to all the local parties for generating synthetic data. Each party first uses its real local data and the corresponding part of synthetic data to train its local discriminator and then uses the trained discriminator to compute the generator's gradient. Finally, the gradients from all the local parties will be aggregated on the server side and used to update the global generator. In Figure 3, we also present a detailed illustration of the local training process. It can be seen that the local data are only used for training the local discriminator D^i , and the global generator G is only updated based on the gradient computed by the trained discriminators. Moreover, only the information (weights and gradients) of the generator is exchanged between the local and server side, while the discriminators and the real data are always kept on the local side. In this way, the framework can facilitate the training of the global generator without direct access to the real local data.

5.2 Multi-Output Generator Against Attribute Inconsistency

Moreover, in this paper, we consider the scenario where the user data are vertically-partitioned and distributed among M local parties. Since the local parties under this setting may hold different sets of attributes, the conventional single-output generators are not applicable for the framework. In order to address the *attribute inconsistency* problem, we propose a multi-output structure for the global generator. The generator consists of several common layers (denoted as G^0) and M separate follow-up branches (denoted as $\{G^1, \dots, G^M\}$). Each branch G^i produces synthetic data with attributes of one local party \mathcal{P}_i . Given a batch of input feature Z , the global generator is capable of concurrently producing synthetic data $\{\tilde{X}^1, \dots, \tilde{X}^M\}$ for all the local parties. Here, $\tilde{X}^i = G^i(G^0(Z))$ corresponds to the data generated from the i -th branch.

We follow the optimization approach of WGAN introduced in Section 3.2 to iteratively train the global generator and local discriminators in the proposed framework. On the one hand, the training of the discriminators on the local side is conducted as under the centralized setting. Here, the loss function for the i -th local discriminator D^i is:

$$\mathcal{L}_D^i = D^i(x^i) - D^i(\tilde{x}^i) + \lambda(\|\nabla_{\tilde{x}^i} D(\tilde{x}^i)\| - 1)^2, \quad (12)$$

where x^i is the real data of the i -th local party, $\tilde{x}^i = G^i(G^0(z))$ is the synthetic data generated by the i -th branch of G , \tilde{x}^i is the gradient penalty as defined in Equation (7), and λ is the weight for the gradient penalty. Once the discriminators have been trained for several iterations, they will be used to compute the gradient of the global generator. The loss function for the global generator G can be computed as the sum of the loss regarding all the local discriminators, each of which is derived following Equation (8):

$$\mathcal{L}_G = \sum_{i=1}^M \mathcal{L}_G^i = \sum_{i=1}^M D^i(G^i(G^0(z))). \quad (13)$$

The generator's gradient $\nabla \mathcal{L}_G$ can be further derived as

$$\begin{aligned} \nabla \mathcal{L}_G &= \frac{\partial \sum_{i=1}^M \mathcal{L}_G^i}{\partial G} = \sum_{i=1}^M \frac{\partial \mathcal{L}_G^i}{\partial [G^0, G^1, \dots, G^M]} \\ &= \left[\frac{\partial \mathcal{L}_G^1}{\partial G^0}, \frac{\partial \mathcal{L}_G^1}{\partial G^1}, 0, \dots, 0 \right] + \dots + \left[\frac{\partial \mathcal{L}_G^M}{\partial G^0}, 0, 0, \dots, \frac{\partial \mathcal{L}_G^M}{\partial G^M} \right] \\ &= \left[\sum_{i=1}^M \frac{\partial \mathcal{L}_G^i}{\partial G^0}, \frac{\partial \mathcal{L}_G^1}{\partial G^1}, \dots, \frac{\partial \mathcal{L}_G^M}{\partial G^M} \right] \end{aligned} \quad (14)$$

which is the sum of the generator gradients from all the local parties. Hence, by aggregating all the returned generator gradients, the server achieves to use the sum of the gradients to update the global generator. It can be seen from Equation (14) that the parameters of each branch G^i are updated based on the gradients from party \mathcal{P}_i , while the parameters of the common layers G^0 are updated by the gradients from all the local parties. Therefore, the multi-output structure enables the global generator to automatically capture the correlations and distributions of attributes across local parties during the training process and directly generate synthetic integrated data with the entire attribute set.

5.3 Collaborative Training with DP

In the previous sections, we illustrate how the VERTIGAN framework enables a global generator to learn the hidden correlations of attributes across all the local parties without actually accessing the real local data. Nevertheless, recent studies (e.g., [9, 49]) showed that the trained generator may reveal sensitive information of real local data under various privacy attacks. In order to mitigate potential privacy risks, we further apply DP during the training process, which provides strict privacy guarantees to the local datasets.

Considering the global generator does not directly access real local data, we follow previous DP-GAN algorithms [64, 71] and only perturb the gradients of local discriminators to achieve privacy protection. Specifically, in each update step of the discriminator, we first sample a batch of real local data and synthetic data, and then

compute the corresponding gradients $\{g_D^{i,b}\}_{b \in B}$. Each gradient $g_D^{i,b}$ is then clipped by a pre-defined L_2 -norm bound C , namely

$$\tilde{g}_D^{i,b} = \text{clip}(g_D^{i,b}, C) = g_D^{i,b} / \max(1, \|g_D^{i,b}\|_2 / C). \quad (15)$$

Next, we sum up all the clipped gradients, add random Gaussian noise $\mathcal{N}(0, \sigma^2 C^2 I)$, and divide the perturbed gradient by the batch size B as shown below:

$$\tilde{g}_D^i = \frac{1}{B} \left(\sum_{b=1}^B \tilde{g}_D^{i,b} + \mathcal{N}(0, \sigma^2 C^2 I) \right). \quad (16)$$

The gradient \tilde{g}_D^i is used to update the discriminator parameters.

Since the local discriminator is repeatedly updated during the training process, according to the composition property, the total privacy cost should be accumulated. Considering that RDP achieves a much tighter privacy estimation in comparison to the traditional DP (as mentioned in Section 3.1), we first compute the overall privacy cost under the RDP definition and then convert it back to the traditional DP definition. To start with, the privacy cost of each gradient perturbation under RDP is derived as follows:

COROLLARY 1. *With a noise scale $\mathcal{N}(0, \sigma^2 C^2)$, the perturbed gradient \tilde{g}_D^i satisfies $(\alpha, \alpha/2\sigma^2)$ -RDP.*

PROOF. Let $f = \sum_{b=1}^B \tilde{g}_D^b = \sum_{b=1}^B \text{clip}(g_D^b, C)$ be the sum of all the gradients clipped by an L_2 -norm bound of C . The sensitivity of f can be derived as:

$$\Delta_f = \max_{\mathcal{D}, \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2 \leq C. \quad (17)$$

Furthermore, the gradient perturbation process can be denoted as $\mathcal{M}_f = f + \mathcal{N}(0, \sigma^2 C^2 I)$. Based on Section 3.1, the privacy cost of \mathcal{M}_f under the order α is

$$\epsilon(\alpha) = \frac{(\Delta_f)^2 \cdot \alpha}{2 \cdot \sigma^2 C^2} = \frac{C^2 \cdot \alpha}{2 \cdot \sigma^2 C^2} = \frac{\alpha}{2\sigma^2}. \quad (18)$$

As shown in Equation (16), the perturbed gradient will be divided by a batch size B , and the result \tilde{g}_D^i will be actually used to update the discriminator. Since B is unrelated to the real data, according to the post-processing property (Theorem 2), the final discriminator update \tilde{g}_D^i also satisfies $(\alpha, \epsilon(\alpha))$ -RDP. \square

According to Lemma 1, the privacy guarantee can be further amplified by subsampling. Given N as the total number of training data and B as the batch size, we compute the sampling rate as $\gamma = N/B$ and derive the amplified privacy cost $\epsilon'(\alpha)$ following Equation (5). Next, assume the discriminator has updated for T steps during the entire training process, then the overall privacy cost is $(\alpha, T \cdot \epsilon'(\alpha))$ -RDP. We further convert privacy cost back to the traditional (ϵ, δ) -DP definition according to Lemma 2. Finally, since the global generator is trained on the local discriminators, according to the post-processing property (Theorem 2), the global generator also satisfies (ϵ, δ) -DP for the corresponding local dataset.

5.4 Overall Training Process

With the above design considerations, we now describe the overall training process presented in Algorithm 1 and Algorithm 2.

Before the training starts, the server initializes the global generator G . On the local side, each party also initializes its local discriminator D^i . Moreover, considering that the local parties may

Algorithm 1: VERTIGAN - Workflow of Server

Input: G : global generator; M : number of local parties; T_{global} : global training rounds; η : learning rate; OPT : optimizer for the GAN model.

Output: Trained global generator G

Server executes:

- 1: Initialize global generator G
 - 2: **for** each local party $i = 1, \dots, M$ **do**
 - 3: **LocalInitialization()** // Run on the local side
 - 4: **end for**
 - 5: **for** each global round $t = 1, \dots, T_{global}$ **do**
 - 6: Sample random seed τ
 - 7: **for** local party $i = 1, \dots, M$ **do**
 - 8: Distribute τ and G to the local party i
 - 9: Get local gradient $g_G^i = \text{LocalUpdate}(\tau, G)$
 - 10: **end for**
 - 11: Aggregate local gradients $g_G = \sum_{i=1}^M g_G^i$
 - 12: Update generator $G \leftarrow OPT.update(G, g_G, \eta)$
 - 13: **end for**
 - 14: **return** G
-

have personalized privacy requirements, we let each local party individually compute the noise scale σ^i . With the universally configured batch size B , global rounds T_{global} , and local steps T_d , the discriminator's total update step is derived as $T = T_{global} \cdot T_d$. Following the privacy accounting process described in Section 5.3, the required σ^i under the target privacy budget (ϵ^i, δ^i) can be determined accordingly. Finally, since each local party holds different attributes of the same group of samples, the local training data should be sample-wise aligned during each global round. A naive solution is to let the server randomly sample multiple batches of data indices for selecting the real data as well as input features for generating the synthetic data, and then broadcast all the information to the local side. However, this may cause extra communication costs, especially for large training batches. To address the issue, our framework applies a pseudorandom number generator (PRNG) Φ^i at each local party to realize the data alignment. Following prior works [6, 42], we use secure PRNGs to achieve comprehensive security guarantees. Moreover, we require that all the local PRNGs use the same algorithm and are deployed with the same configuration. Therefore, according to the reproducibility of PRNG, given the same random seed, each Φ^i is able to produce the same sequence of indices of real data or input features sampled from the standard Gaussian distribution. By using the PRNG, the server only needs to randomly sample a random seed and broadcast it to all the local parties in each global round, which significantly improves communication efficiency. Also, considering that existing secure PRNGs based on standard cryptographic primitives can have an output rate of gigabytes per second on modern CPUs [32], their computation cost is negligible compared to the local training time.

In each global training round, the server broadcasts the current global generator G as well as the random seed τ to all the local parties. Each party \mathcal{P}^i first sets Φ^i with the random seed τ and then updates the local discriminator D^i for T_d steps using the real data X^i and the synthetic data $\tilde{X}^i = G^i(G^0(Z))$ sampled by Φ^i . We

Algorithm 2: VERTIGAN - Workflow of Local Party i

Input: G : global generator; D^i : party i 's local discriminator;
 X^i : party i 's local data; Φ^i : party i 's local PRNG;
 T_{global} : global training rounds; T_d : discriminator's
local update steps; B : batch size; η : learning rate; C :
 L_2 clipping bound; (ϵ^i, δ^i) : party i 's privacy budget;
 OPT : optimizer for the GAN model.

LocalInitialization():

- 1: Initialize local discriminator D^i , local PRNG Φ^i
- 2: Given the target (ϵ^i, δ^i) and the pre-defined (B, T, T_d) ,
compute the required noise scale σ^i

LocalUpdate(τ, G):

- 3: Get local data X^i , set $\Phi^i.set_seed(\tau)$
// Train local discriminator
- 4: **for** $t = 1, \dots, T_d$ **do**
- 5: Sample indices $J = \Phi^i.random_choice(size=B)$
- 6: Sample input noise $Z = \Phi^i.random_normal(size=B)$
- 7: **for** $b = 1, \dots, B$ **do**
- 8: Let $x = X^i[J[b]]$, $\tilde{x} = G^i(G^0(Z[b]))$
- 9: Compute $\mathcal{L}_D^{i,b}(x, \tilde{x})$ and $g_D^{i,b} = \nabla \mathcal{L}_D^{i,b}(x, \tilde{x})$
- 10: Clip gradient $\tilde{g}_D^{i,b} = g_D^{i,b} / \max(1, \|g_D^{i,b}\|_2 / C)$
- 11: **end for**
- 12: Aggregate gradients and add noise
 $\tilde{g}_D^i = \frac{1}{B} (\sum_{b=1}^B \tilde{g}_D^{i,b} + \mathcal{N}(0, \sigma^i{}^2 C^2 I))$
- 13: Update discriminator $D^i \leftarrow OPT.update(D^i, \tilde{g}_D^i, \eta)$
- 14: **end for**
// Compute generator gradient
- 15: Sample input noise $Z = \Phi^i.random_normal(size=B)$
- 16: Compute $g_G^i = \frac{1}{B} \sum_{b=1}^B \nabla \mathcal{L}_G(G^i(G^0(Z[b])))$
- 17: **return** g_G^i

apply the DP perturbation in each update step, where the batch of gradients is clipped by L_2 bound C and perturbed with random Gaussian noise $\mathcal{N}(0, \sigma^i{}^2 C^2 I)$. The noise scale σ^i is determined in the initialization process. Then, the local discriminator is used to compute the gradient g_G^i of the current global generator, which will be returned to the server for updating the parameters of the global generator parameters. The global training process is conducted for T_{global} rounds. Once the training completes, the server can use the global generator G to directly generate the synthetic dataset with attributes of all the local parties.

6 EXPERIMENTS AND RESULTS

We implemented the proposed framework using the Tensorflow library and performed comprehensive experiments with a number of open-source datasets to evaluate its performance. In this section, we first introduce the experimental settings and then discuss the evaluation results.

6.1 Experiment Setup

6.1.1 Datasets and Models. We used six multi-dimensional classification datasets for evaluating the performance of the VERTIGAN framework:

Table 1: Datasets details

Dataset	Type	Num.	Num.	Domain
		Records	Attributes	Size
Census	Integer	2,458,285	68	2^{150}
Twitter	Integer	140,707	78	2^{181}
Web	Binary	36,974	124	2^{124}
Vehicle	Binary	98,528	101	2^{101}
HAR	Binary	10,299	561	2^{561}
Dilbert	Binary	10,000	1501	2^{1501}

Table 2: One-hot dimensions and the number of model parameters under the two-party setting

Dataset	Party 1		Party 2		Server
	One-hot Dim.	#Param. D^1	One-hot Dim.	#Param. D^2	#Param. G
Census	137	9,592	145	10,732	53,760
Twitter	195	19,307	174	15,313	94,768
Web	124	7,813	123	7,751	39,416
Vehicle	100	5,101	102	5,305	26,857
HAR	562	158,485	566	160,745	812,949
Dilbert	1,500	788,551	1,505	794,190	2,681,446

Web [47] contains records with 124 binary attributes extracted from each web page. The goal was to train a classifier to determine whether the web page belongs to a category.

Vehicle [17] contains data collected in wireless distributed sensor networks. Each record has 100 binary attributes representing data collected from different acoustic and seismic sensors. The goal was to train a classifier for vehicle type classification.

Census [16] contains records drawn from the 1990 United States census data, including 68 personal attributes such as gender, income, and marital status. We used the dataset to classify the duration of people's active duty service.

Twitter [34] contains records with 77 attributes such as the number of discussions and average discussion length, which are used to predict the popularity magnitude of each instance. In our experiment, we quantified the values of each attribute into five bins. The goal was to classify the level of popularity of each instance.

Activity [2] contains sensor records describing six daily activities. Each data record has 561 attributes representing different time and frequency domain variables. We normalize each attribute and convert the data to binary form.

Dilbert was originally provided in [37] for object recognition. We use the processed version in [21], where the records are categorized to five classes. We take the first 1500 attributes from the processed data to exclude the irrelevant variables mentioned in [21]. Then, we normalize each attribute and convert the data to binary form.

Details of each dataset are presented in Table 1, including the data type, the number of records and attributes, and the domain size. In the experiments, we assume that each party holds 10^5 data records. To this end, we randomly sample 10^5 records from each original dataset and partition the datasets by feature. If the original dataset

contains fewer records, the data are sampled with replacements. We further use one-hot encoding to convert the original categorical attributes to the numerical form for model training.

We design the global generator and local discriminators as multi-layer neural networks (NNs) and determine their layer size according to the one-hot dimension of the local datasets. The local discriminators are two-layer NNs whose output is a scalar between 0 and 1. The global generator is a multi-output model, which has two common layers followed by a number of separate branches. Each branch contains two fully-connected layers, which outputs the synthetic data of one party. In Table 2, we report the one-hot dimensions and the model size under the two-party setting.

6.1.2 Baseline Methods. Considering the objective and setting of existing works on the publication of vertically-partitioned data, we use the DPLT algorithm proposed by Tang *et al.* [53] as our baseline in the following experiments. The algorithm uses a latent tree model to represent the cross-attribute correlations in the original dataset and perturbs the tree parameters via a distributed Laplace protocol to achieve DP guarantee for *each* local dataset. Additionally, a tree index based method TICQ can also be used to determine the minimum set of latent attribute pairs for constructing the latent tree, which helps to reduce the noise scale. The total privacy budget is consumed by three parts, namely the generation of latent attributes, quantification of latent attributes' correlations, and privatization of the tree parameters. For each dataset, we respectively compare the synthetic data utility of using the DPLT algorithm (referred to as DPLT) as well as the improved TICQ-DPLT algorithm (referred to as DPLT+). Moreover, we also present the utility of synthetic data generated under the non-private setting as a reference.

6.1.3 Parameter Configurations. In the following experiments, we conduct the collaborative training process for $T = 1500$ rounds. During local training, each local discriminator is updated for $T_d = 10$ steps with a batch size of $B = 1000$. For both the generator and discriminator, we use the RMSprop optimizer with a default learning rate of $\eta = 0.001$. Moreover, we apply the gradient perturbation when training the local discriminators, where the L_2 -clip bound C is set to 1 and the noise scale σ varies according to the target privacy budget. We choose a different privacy budget $\epsilon \in \{0.5, 1, 2, 4, 8\}$ and $\delta = 10^{-5}$ so as to explore the influence of privacy on the framework performance. The ϵ here follows the traditional DP definition (Definition 1).

6.1.4 Evaluation Metrics. We evaluate the performance of our VERTIGAN framework from two perspectives, namely the *utility evaluation* and the *privacy evaluation*. For the *utility evaluation*, we first compare the statistical similarity of synthetic data and real data. Then, we apply commonly-used machine learning models to investigate the utility of synthetic data in AI training tasks. For the *privacy evaluation*, we investigate the capability of our framework against membership inference attacks, where an attacker aims to use the synthetic dataset to determine whether a target record is used for training the GAN model.

6.1.5 Computation Environments. We perform all the experiments on a NVIDIA Quadro RTX 6000 GPU. In Table 3, we compare the training time (sec) of our VERTIGAN framework and the baseline DPLT+ algorithm regarding all the datasets.

Table 3: Computation time (sec) of the proposed VERTIGAN framework and baseline DPLT+ algorithm regarding different datasets. For VERTIGAN, we perform 1500 global rounds and report the total training time.

Dataset	Web	Vehicle	Census	Twitter	HAR	Dilbert
DPLT+	1516.20	1116.81	1341.47	3954.03	19598.39	34413.66
VertiGAN	485.35	396.35	424.42	510.26	3296.08	8387.30

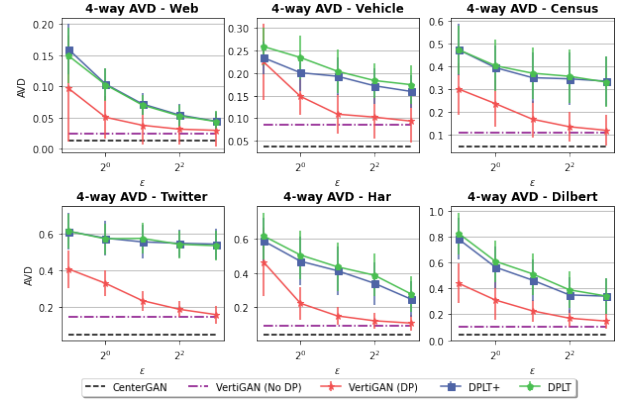


Figure 4: Average total variation distance (AVD) of four-way joint distributions between the real and synthetic data with respect to different privacy levels.

6.2 Utility: Statistical Similarity

We start our evaluation under the two-party setting, which is commonly used in existing VFL frameworks. Here, each party holds half of the attributes. We first evaluate the performance of VERTIGAN by investigating whether the generated synthetic data can preserve similar statistical properties as real data. To this end, we respectively compare the k -way joint distributions and cross-attribute correlations of the real data and synthetic data and analyze their statistical similarity.

6.2.1 Comparison of Joint Distributions. For the analysis of joint distributions, we used the Average Variant Distance (AVD) to quantify the distribution difference between the real data and synthetic data, as used in [53], which is defined as

$$AVD = \frac{1}{2} \sum_{\omega \in \Omega} |P_{real}(\omega) - P_{syn}(\omega)|, \quad (19)$$

where Ω is the domain of all the k -way attribute combinations, ω is one of the combinations, $P_{real}(\omega)$ and $P_{syn}(\omega)$ are joint distributions of real and synthetic data. More specifically, assume the attribute combination ω has a domain size of $|\omega|$, P_{real} and P_{syn} are $|\omega|$ -dimensional vectors, where each entry is the probability of a specific value combination (namely the ratio of occurrence in the entire real or synthetic dataset). For each dataset, we randomly chose 100 k -way attribute combinations and compute the average distribution difference.

AVD Regarding the Privacy Budget ϵ . In Figure 4, we first compare the four-way AVD of the synthetic data generated by the

VERTIGAN framework as well as the two baseline algorithms under different privacy levels. We also report the results under the fully-centralized setting and the results of the proposed framework under the non-private setting as a reference. The error bars represent the 95% confidence interval (also for the remaining experimental results). It can be seen that the AVD of all the algorithms reduces with the increase of ϵ . Nonetheless, for all the datasets, the synthetic data generated by the VERTIGAN framework consistently achieve a smaller AVD in comparison with the baseline methods, which indicates a better capability of our VERTIGAN framework in capturing the multivariate distributions. Moreover, there is a more distinctive gap in AVD between the baseline algorithms and VERTIGAN for the datasets with a larger domain size. It can be observed that when $\epsilon \geq 4$, the AVD of the baseline algorithms is almost two to three times in comparison with VERTIGAN. This is because a larger domain size refers to more cross-attribute combinations. Since the baseline algorithms are supposed to evenly split the privacy budget to all the attribute pairs, the increase in domain size may cause each attribute pair being allocated with an insufficient privacy budget, which may result in serious degradation of data utility. In comparison, VERTIGAN applies DP perturbation to the discriminator's gradients and is not directly related to the domain size. Therefore, the increase of domain size does not significantly affect the utility of the synthetic data generated by VERTIGAN.

AVD Regarding the Multivariate Dimension k . We further analyze the AVD with varied multivariate dimension k to gain a deeper insight into VERTIGAN's capability in the context of complex datasets. To this end, we choose $k \in \{2, 3, 4, 5, 6\}$ and compare the k -way AVD of using VERTIGAN as well as the baseline algorithms. We present the results under $\epsilon = 2$ in Figure 5. Similarly, we also report the k -way AVD under the centralized setting and under the non-private VERTIGAN setting as a reference. It can be seen that for all the datasets, VERTIGAN steadily shows a smaller k -way AVD compared to the baseline algorithms. Moreover, although the baseline algorithms achieve similar AVD when k is small, the difference gets distinctively larger with an increase of k . Especially, for all the datasets, the 5-way and 6-way AVD of the baseline algorithms are almost twice that of VERTIGAN. This indicates that our framework is more adept at capturing the information of high-dimensional joint distributions of real data.

6.2.2 Comparison of Correlation. We further visualize the correlation coefficient matrix of real data and synthetic data with heat maps in order to better understand the capability of our method in capturing and preserving the cross-attribute correlations. Figure 6 shows the comparison result of the different datasets with $\epsilon = 8$. For each dataset, we respectively select 10 attributes from each party and present the correlation matrix of the 20 attributes. From the visualization results, it can be seen that the correlation of synthetic data is similar to the correlation of real data, which further demonstrates that the synthetic data successfully preserves the attribute correlations of real data.

6.3 Utility: AI Training Performance

Next, we investigate the utility of synthetic data in AI training tasks. To this end, we train two classification models M_{real} and

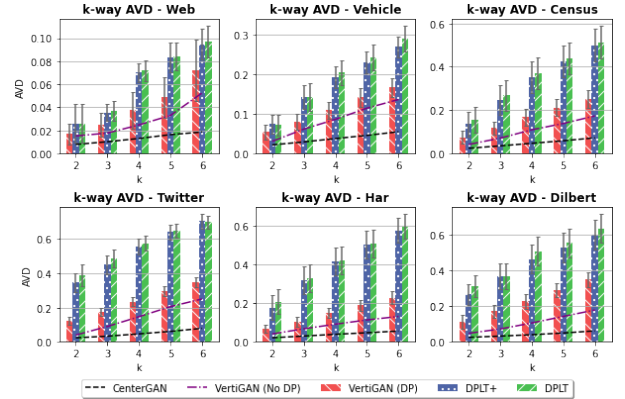


Figure 5: Average total variation distance (AVD) of k -way joint distributions between the real and synthetic data with respect to different dimensions of the joint distribution.

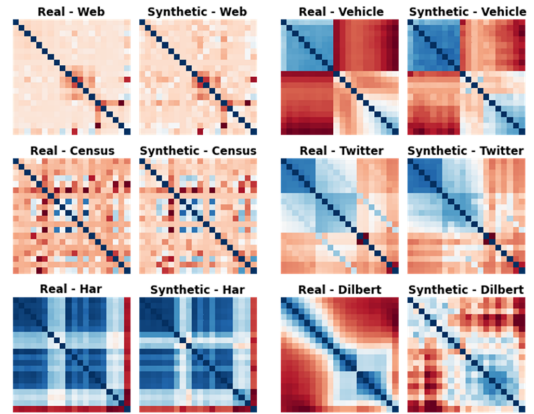


Figure 6: Correlation comparison between the real and synthetic data with $\epsilon = 8$. For each dataset, we present the correlations of 20 attributes, where each party contributes 10 attributes. It can be seen that the synthetic data preserves similar correlations as real data.

M_{syn} , respectively, with real data and synthetic data. Then, we test both models with an amount of held-out real data and compare the test accuracy, namely, Acc_{real} and Acc_{syn} . Intuitively, if Acc_{syn} is close to Acc_{real} , we consider the synthetic data to be of high utility which can replace real data for AI training tasks.

In the experiments, we use the Multi-layer Perceptron (MLP) classifier as the target AI model. We train both M_{real} and M_{syn} ten times and compute the averaged Acc_{real} and Acc_{syn} . In Table 4, we present the accuracy of the MLP classifiers evaluated on different datasets. For each dataset, we compare the Acc_{syn} of synthetic data generated under the non-private centralized and VERTIGAN setting, as well as that generated by the private DPLT and VERTIGAN frameworks with $\epsilon \in \{0.5, 2, 8\}$. It can be observed that although all the algorithms show a higher Acc_{syn} with an increase of ϵ , the accuracy of VERTIGAN is generally higher than the baselines for

Table 4: Classification accuracy of MLP models evaluated on synthetic data generated under different privacy settings.

Dataset	Acc. Real	Acc. Synthetic (No DP)		Acc. Synthetic (With DP)			
		CenterGAN	VertiGAN	ϵ	DPLT	DPLT+	VertiGAN
Web	0.8453	0.8276	0.8079	0.5	0.7114	0.7124	0.6970
				2	0.7251	0.7238	0.7776
				8	0.7385	0.7484	0.7900
Vehicle	0.8204	0.8074	0.7984	0.5	0.7385	0.7208	0.7498
				2	0.7596	0.7373	0.7627
				8	0.7690	0.7729	0.7840
Census	0.9858	0.9820	0.9732	0.5	0.8822	0.8870	0.9088
				2	0.9027	0.9092	0.9432
				8	0.9465	0.9487	0.9655
Twitter	0.8209	0.8180	0.7871	0.5	0.7277	0.7274	0.7445
				2	0.7371	0.7397	0.7701
				8	0.7520	0.7580	0.7822
HAR	0.9532	0.8990	0.8414	0.5	0.4228	0.4570	0.5368
				2	0.5519	0.5702	0.7022
				8	0.6038	0.6284	0.7746
Dilbert	0.9394	0.8651	0.8010	0.5	0.2722	0.2988	0.5525
				2	0.4331	0.4353	0.6241
				8	0.5434	0.5672	0.7134

all privacy levels, especially for complex datasets. In particular, with $\epsilon = 8$, the synthetic data generated by VERTIGAN achieves around 2% ~ 15% improvement of Acc_{syn} compared to the baseline algorithms. The results indicate that our framework has a better capacity for preserving the hidden patterns and correlations of real data compared to the baselines. The generated synthetic data can be effectively used for data mining and AI training tasks.

6.4 Ablation Study

We further conduct a series of ablation studies to investigate how the size of local datasets, the imbalanced splitting of attribute sets, and the increase of local parties impact the performance of the VERTIGAN framework and synthetic data utility.

6.4.1 Impact of the Number of Records. To start with, in the previous experiments, we assume that the local parties share data of 10^5 records. We further investigate how varying the number of local records affects the framework’s performance. To this end, we respectively vary the size of local datasets with 10^4 , 10^5 , and 10^6 records and conduct experiments under different privacy levels. In Figure 7, we present the 4-way AVD of the **Vehicle**, **Census**, and **HAR** datasets with $\epsilon = \{0.5, 2, 8\}$. It can be seen that using a larger number of records can significantly improve the data utility, especially in high-privacy regimes. For instance, when $\epsilon = 0.5$, for all the datasets, the AVD with 10^4 records is 2 ~ 4 times the results with 10^5 records. This is because the privacy loss of each iteration is related to the *sampling rate* γ , as shown in Lemma 1. Therefore, with a fixed batch size of B , increasing the total number of records

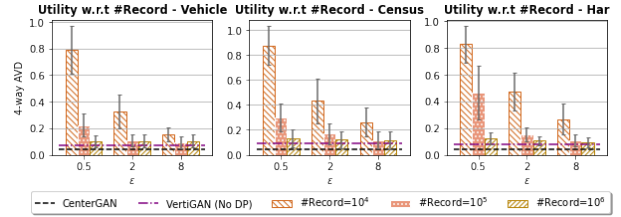


Figure 7: 4-way AVD under the two-party settings with 10^4 , 10^5 , and 10^6 records under the privacy level $\epsilon = \{0.5, 2, 8\}$.

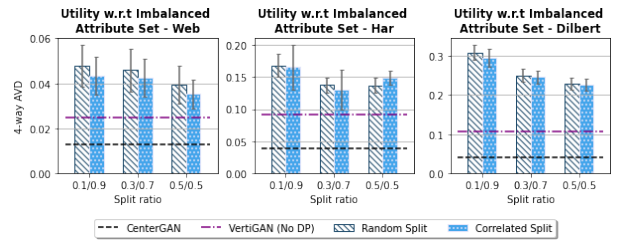


Figure 8: 4-way AVD under the two-party settings with $\epsilon = 8$ and attribute split ratio from $\{0.1/0.9, 0.3/0.7, 0.5/0.5\}$. For each dataset, we compare the results of *random splitting* and *correlated splitting*, where the strongly-correlated attributes are assigned to one of the parties.

leads to a decrease in privacy loss. In other words, the framework only needs to add a smaller amount of noise to achieve the same privacy level, which largely enhances the utility of synthetic data. On the other hand, for $\epsilon = 8$, the AVD with 10^6 is similar to the results with 10^5 records. This is because larger privacy budgets result in less noise being injected during training, hence the model can already converge well with 10^5 records. In this case, using larger datasets offers a comparatively smaller contribution to the utility.

6.4.2 Impact of Imbalanced Attribute Sets. Next, in addition to exploiting the setting where the entire attribute set is evenly split and held by two local parties, we also investigate whether the utility of the synthetic data will differ if the local parties possess an imbalanced number of attributes. To this end, we split the entire attribute set with a ratio of 0.1/0.9, 0.3/0.7, and 0.5/0.5 (i.e., an even split) and compare the data utility under different privacy levels. Moreover, we also explore whether the imbalanced split of strongly-correlated attributes affects the data utility. To this end, we first compute the pair-wise correlation of all the attributes and apply hierarchical clustering to group the most correlated attributes. Then, we construct the imbalanced attribute sets in two ways: *random split* and *correlated split*. The former randomly splits the attribute set according to the split ratio, while the latter manually assigns the strongly correlated attributes to one of the local parties. We conduct experiments under different split ratios following both split fashions and report the results in Figure 8. It can be observed that an imbalanced attribute set can lead to a degradation of framework performance. In contrast, assigning the strongly-correlated attributes to one of the parties slightly improves the data utility

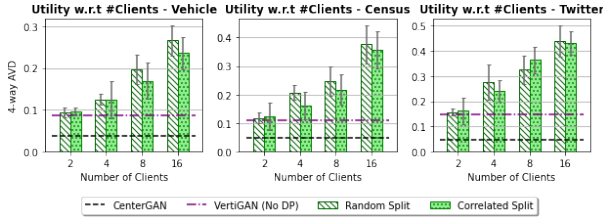


Figure 9: 4-way AVD under the two-party settings with $\epsilon = 8$ and the number of clients from $\{2, 4, 8, 16\}$. For each dataset, we compare the results of *random splitting* and *correlated splitting*, where the strongly-correlated attributes are assigned to a subset of the parties.

compared to the random setting. Intuitively, when the attributes belong to different generator branches, the framework may suffer from a certain information loss on the pair-wise correlations. In contrast, the correlation information can be better preserved when both attributes belong to the same branch, hence leading to higher data utility.

6.4.3 Impact of the Number of Local Parties. Besides the impact of imbalanced splitting, we also analyze the effects of varying the number of local parties on the framework performance. To this end, we respectively perform the data publication process using the different methods under the settings consisting of 2, 4, 8, and 16 local parties and compare the utility of synthetic data. In Figure 9, we present the 4-way AVD with different numbers of local parties under $\epsilon = 8$. It can be observed that the framework performance degrades with the increase of local parties. This might be because the joint distributions and correlations are more difficult to be captured when the correlated attributes are spread over multiple local parties. On the other hand, similar to observations in Section 6.4.2, when assigning all the strongly-correlated attributes to a subset of local parties (*i.e.*, by using *correlated splitting*), the cross-attribute correlations can be better preserved and the data utility can be further improved.

6.5 Empirical Privacy Analysis

Although choosing a larger privacy budget ϵ can distinctively improve the data utility, this may lead to increased privacy leakage. In order to obtain a better understanding of the utility-privacy trade-off, we conduct a membership inference attack to empirically analyze the privacy protection capabilities of our framework under different privacy settings. We follow the black-box MIA protocol proposed in [24], which uses the distance of a target record to the synthetic dataset to infer the membership information. The intuition is that the generator tends to generate synthetic data close to the training data. Therefore, given a target record x , let $U_\tau(x) = \{x' | d(x, x') \leq \tau\}$ denote the τ -neighborhood of x with respect to the distance metric d . Then, we randomly generate a synthetic dataset \mathcal{X}_{syn} with n records and compute the ratio that the synthetic records fall into the neighborhood of x , namely

$$\hat{f}_\tau(x) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}[x_{syn}^i \in U_\tau(x)], \quad (20)$$

Table 5: MIA accuracy under different privacy settings.

	CenterGAN		VertiGAN		
	No DP	No DP	$\epsilon=8$	$\epsilon=2$	$\epsilon=0.5$
Vehicle	0.5841	0.5758	0.5538	0.5448	0.5287
Web	0.6008	0.5844	0.5633	0.5367	0.5223
Census	0.6509	0.6394	0.6171	0.5800	0.5421
Twitter	0.6746	0.6672	0.6320	0.5980	0.5676
HAR	0.5784	0.5637	0.5324	0.5241	0.5160
Dilbert	0.6044	0.5856	0.5623	0.5433	0.5386

where x_{syn}^i is the i^{th} synthetic record. Obviously, the higher the $\hat{f}_\tau(x)$, the more likely it is that x is included in the training data.

In our experiments, we construct the target dataset by randomly sampling 100 training records (denoted as \mathcal{X}_{in}) and 100 testing records (denoted as \mathcal{X}_{out}). Then, we generate a synthetic dataset \mathcal{X}_{syn} with 10^4 records and use the normalized Hamming distance to measure the minimum distance between each target record and the synthetic data. Following [24], we set τ as the median of the minimum distance of each record. Given the ground truth label and the predicted membership probability, we compute the averaged attack accuracy under different privacy settings. The results are reported in Table 5. It can be observed that synthetic data generated by non-private GANs are still likely to reveal the membership information of the target record. In particular, for **Twitter** and **Census** dataset, the attack accuracy under the non-private setting is more than 65%. On the other hand, applying DP to our VERTIGAN framework can effectively reduce attack accuracy. With $\epsilon = 8$, the attack accuracy is reduced by 2% ~ 4%, while with $\epsilon = 0.5$, the attack accuracy is reduced by 5% ~ 10%. The results demonstrate that our framework is able to mitigate the risk of membership inference attacks and can provide strengthened privacy protection to the local data.

7 DISCUSSIONS AND FUTURE WORK

In this section, we discuss potential extensions of our framework, current limitations, and directions for future work.

7.1 Extension to Other Data Types

In Section 6, we demonstrated that the VertiGAN framework is effective in publishing vertically-partitioned categorical datasets and achieves better data utility compared to previous statistics-based baselines. Moreover, our framework can be further extended to more complex settings where each party holds different types of data. For instance, in a healthcare scenario, a group of hospitals can use the framework to publish a joint dataset containing patients' CT images and physical symptoms for future medical research. This can be realized by modifying the structure of the models and using the advanced layers. For instance, we can respectively adopt convolution layers and recurrent layers to enhance the feature extraction on image data and time-series data. Despite the variation of the layers and model structures, the main workflow of the VertiGAN framework remains unchanged. In Figure 10, we further demonstrate the framework's feasibility in the context of image data. Here, we assume that there are three local parties respectively

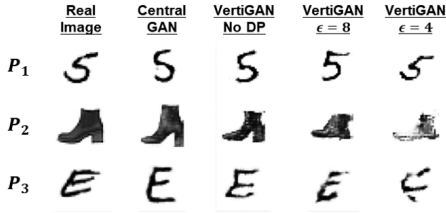


Figure 10: Results of image data synthesis under a three-party setting. Each row represents the synthetic images generated by one local party under different privacy settings.

holding handwritten digits from MNIST, handwritten letters from Extended MNIST, and article images from Fashion-MNIST. We construct a global generator with three output branches and the corresponding local discriminators and analyze the quality of synthetic images generated under different privacy settings. Note that the synthetic data are randomly generated and hence are not identical to the real data. Nevertheless, it can still be observed that our framework is capable of jointly synthesizing all three categories of images of different local clients, and the generated data enjoys a satisfactory level of quality under a larger privacy budget.

7.2 Reduction of Communication Cost

As described in Section 5.4, in each global round, the parameters and gradients of the global generator are repeatedly exchanged between the server and local parties. This may result in a high communication cost, especially for high-dimensional models. One possible approach for mitigating the upload communication cost is to process the generator’s gradients with top- k sparsification and send the sparsified gradients to the server. In Figure 11, we investigate how the sparsification level affects the utility of synthetic data. Here, we choose the top- k ratio from {0.25, 0.5, 0.75, 1} and compare the corresponding 4-way AVD of the synthetic data under the privacy level of $\epsilon \in \{2, 8\}$. It can be observed that even processing the gradients with a top- k ratio of 0.25 can still achieve data utility comparable to returning the entire gradients. The results demonstrate the effectiveness of gradient sparsification in reducing the upload communication cost. On the other hand, a few recent studies also proposed to use dropout [8] and model pruning [30] to reduce the size of the broadcast global model. Our framework can be further improved following this idea: before the training starts, the server broadcasts the initialized global generator to all the local parties. Then, during training, instead of broadcasting the entire global generator, the server only sends the parameters of the common layers G^0 and the corresponding branch G^i to the party \mathcal{P}^i , which is enough for \mathcal{P}^i to produce the synthetic data $\tilde{X}^i = G^i(G^0)(Z)$ on the local side (see Section 5.2). The improvement not only reduces the download communication but also prevents the local parties from inferring the inputs of the other parties.

7.3 Protection for the Uploaded Gradients

In this paper, we apply DP perturbation to the discriminator and enforce privacy guarantees to the generator according to the post-processing property. Nevertheless, even though the global generator

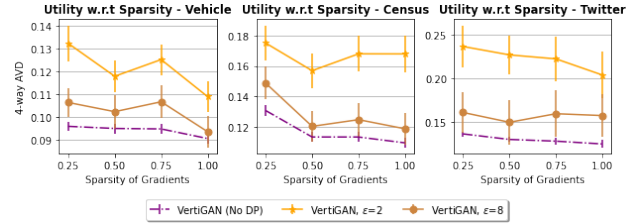


Figure 11: Four-way AVD between the real and synthetic data with respect to different gradient sparsity ratios.

is not directly trained on the local data, the gradients derived by the local discriminators may still reveal sensitive information about local data. Considering that recent studies in FL [6, 29] adopt SMC or local differential privacy (LDP) for encrypting or perturbing local updates, such protection techniques may also be applicable to our framework. For instance, we can use SMC protocols to encrypt the real gradients on the local side before sending them to the server. In this way, the server cannot obtain the individual real gradients but only the sum of all the gradients after the decryption. However, the use of SMC protocols may increase the communication and computational cost of the framework due to the key generation and exchange process. On the other hand, LDP-based solutions add random noise to the local gradients, which will not largely affect efficiency. Nevertheless, it may cause significant utility loss due to the limited number of local parties under the vertical setting. Hence, how to protect the uploaded gradients regarding security, utility, and efficiency will be an important direction for future work.

8 CONCLUSION

Due to the great variety in service scenarios, user data in real-life applications are often vertically partitioned and distributed among different local parties. Although it is of great benefit for data analysts to explore the hidden correlations of attributes of all the local parties, publishing the vertically-partitioned data raises both privacy and utility concerns.

In this paper, we follow the idea of synthetic data generation and propose VERTIGAN, the first GAN-based framework for privately publishing vertically-partitioned data. Different from the prior statistics-based solutions, our framework adopts a distributed GAN architecture, where a global generator is adversarially trained with a group of local discriminators to learn the distribution of all parties’ local data and used to directly generate synthetic integrated data on the server side. Moreover, we apply DP perturbation during the training process to ensure strict privacy guarantees for the local data. Experimental evaluation with real-world datasets shows that our framework significantly outperforms the statistics-based baseline algorithms for publishing high-dimensional vertically-partitioned data. The synthetic data generated by our framework preserves very similar statistical properties as real data and can replace real data for data mining and model training tasks.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments for improving this paper.

REFERENCES

- [1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Vienna, Austria, 308–318.
- [2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A Public Domain Dataset for Human Activity Recognition using Smartphones. In *21st European Symposium on Artificial Neural Networks*. Ciaco-6doc.com, Bruges, Belgium, 437–442.
- [3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *34th International Conference on Machine Learning*, Vol. 70. PMLR, Sydney, NSW, Australia, 214–223.
- [4] Ching-Yuan Bai, Hsuan-Tien Lin, Colin Raffel, and Wendy Chi-wen Kan. 2021. On Training Sample Memorization: Lessons from Benchmarking Generative Modeling with a Large-scale Competition. In *27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, Virtual Event, Singapore, August 14–18, 2021, 2534–2542.
- [5] Gabrielle Berman, Sara de la Rosa, and Tanya Accone. 2018. Ethical Considerations When Using Geospatial Technologies for Evidence Generation. *Innocenti Discussion Papers* 2018-02 (2018).
- [6] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Dallas, TX, USA, 1175–1191.
- [7] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. 2011. “You Might Also Like:” Privacy Risks of Collaborative Filtering. In *2011 IEEE Symposium on Security and Privacy*. IEEE, Berkeley, CA, USA, 231–246.
- [8] Sebastian Caldas, Jakub Konečný, H. Brendan McMahan, and Ameet Talwalkar. 2018. Expanding the Reach of Federated Learning by Reducing Client Resource Requirements. *CoRR* abs/1812.07210 (2018).
- [9] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *28th USENIX Security Symposium*. USENIX Association, Santa Clara, CA, USA, 267–284.
- [10] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. 2020. GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models. In *2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Virtual Event, USA, 343–362.
- [11] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. 2020. VAFL: A Method of Vertical Asynchronous Federated Learning. *CoRR* abs/2007.06081 (2020).
- [12] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. 2021. SecureBoost: A Lossless Federated Learning Framework. *IEEE Intelligent Systems* 36, 6 (2021), 87–98.
- [13] Michele Ciampi and Claudio Orlandi. 2018. Combining Private Set-Intersection with Secure Two-Party Computation. In *11th International Conference on Security and Cryptography for Networks*, Vol. 11035. Springer, Amalfi, Italy, 464–482.
- [14] Ivan Damgård and Mads Jurik. 2001. A Generalisation, a Simplification and Some Applications of Paillier’s Probabilistic Public-Key System. In *4th International Workshop on Practice and Theory in Public Key Cryptography*, Vol. 1992. Springer, Cheju Island, Korea, 119–136.
- [15] Sumit Kumar Debnath and Ratna Dutta. 2015. Secure and Efficient Private Set Intersection Cardinality Using Bloom Filter. In *18th Information Security Conference*, Vol. 9290. Springer, Trondheim, Norway, 209–226.
- [16] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. <http://archive.ics.uci.edu/ml>.
- [17] Marco F. Duarte and Yu Hen Hu. 2004. Vehicle Classification in Distributed Sensor Networks. *Journal of Parallel and Distributed Computing* 64, 7 (2004), 826–838.
- [18] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.
- [19] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X. Liu, and Ting Wang. 2022. Label Inference Attacks Against Vertical Federated Learning. In *31st USENIX Security Symposium*. USENIX Association, Boston, MA, 1397–1414.
- [20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *27th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Montreal, Quebec, Canada, 2672–2680.
- [21] Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, Alexander R. Statnikov, Wei-Wei Tu, and Evelyne Viegas. 2019. Analysis of the AutoML Challenge Series 2015–2018. In *Automated Machine Learning - Methods, Systems, Challenges*. Springer, Berlin, Germany, 177–219.
- [22] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2017. Private Federated Learning on Vertically Partitioned Data via Entity Resolution and Additively Homomorphic Encryption. *CoRR* abs/1711.10677 (2017).
- [23] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2019. LOGAN: Membership Inference Attacks Against Generative Models. *Proceedings on Privacy Enhancing Technologies* 2019, 1 (2019), 133–152.
- [24] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. 2019. Monte Carlo and Reconstruction Membership Inference Attacks against Generative Models. *Proceedings on Privacy Enhancing Technologies* 2019, 4 (2019), 232–249.
- [25] Yan Huang, David Evans, and Jonathan Katz. 2012. Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?. In *19th Annual Network and Distributed System Security Symposium*. The Internet Society, San Diego, California, USA.
- [26] Wei Jiang and Chris Clifton. 2006. A Secure Distributed Framework for Achieving k-Anonymity. *The International Journal on Very Large Data Bases* 15, 4 (2006), 316–333.
- [27] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. Comprehensive Analysis of Privacy Leakage in Vertical Federated Learning During Prediction. *Proceedings on Privacy Enhancing Technologies* 2022, 2 (2022), 263–281.
- [28] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. Privacy-Preserving High-dimensional Data Collection with Federated Generative Autoencoder. *Proceedings on Privacy Enhancing Technologies* 2022, 1 (2022), 481–500.
- [29] Xue Jiang, Xuebing Zhou, and Jens Grossklags. 2022. SignDS-FL: Local Differentially Private Federated Learning with Sign-based Dimension Selection. *ACM Transactions on Intelligent Systems and Technology* 13, 5, Article 74 (2022), 22 pages.
- [30] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. 2022. Model Pruning Enables Efficient Federated Learning on Edge Devices. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–13.
- [31] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. 2019. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *7th International Conference on Learning Representations*. OpenReview.net, New Orleans, LA, USA.
- [32] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.
- [33] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International Conference on Learning Representations*. OpenReview.net, Vancouver, BC, Canada.
- [34] François Kawala, Ahlame Douzal-Chouakria, Eric Gaussier, and Eustache Dimert. 2013. Prédications d’Activité dans les Réseaux Sociaux en Ligne. In *4ième Conférence sur les Modèles et l’Analyse des Réseaux: Approches Mathématiques et Informatiques*. France.
- [35] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations*. OpenReview.net, Banff, AB, Canada.
- [36] Florian Kohlmayer, Fabian Prasser, Claudia Eckert, and Klaus A Kuhn. 2014. A Flexible Approach to Distributed Data Anonymization. *Journal of Biomedical Informatics* 50 (2014), 62–76.
- [37] Yann LeCun, Fu Jie Huang, and Léon Bottou. 2004. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, Washington, DC, USA, 97–104.
- [38] Haoran Li, Li Xiong, and Xiaoqian Jiang. 2014. Differentially Private Synthesis of Multi-dimensional Data Using Copula Functions. In *Proceedings of the 17th International Conference on Extending Database Technology*, Vol. 2014. OpenProceedings.org, Athens, Greece, 475–486.
- [39] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. 2022. Label Leakage and Protection in Two-party Split Learning. In *Tenth International Conference on Learning Representations*. OpenReview.net, Virtual Event.
- [40] Yang Liu, Zhihao Yi, and Tianjian Chen. 2020. Backdoor Attacks and Defenses in Feature-partitioned Collaborative Learning. *CoRR* abs/2007.03608 (2020).
- [41] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2021. Feature Inference Attack on Model Predictions in Vertical Federated Learning. In *37th IEEE International Conference on Data Engineering*. IEEE, Chania, Greece, 181–192.
- [42] H. Brendan McMahan and Galen Andrew. 2018. A General Approach to Adding Differential Privacy to Iterative Training Procedures. *CoRR* abs/1812.06210 (2018).
- [43] Ilya Mironov. 2017. Rényi Differential Privacy. In *30th IEEE Computer Security Foundations Symposium*. IEEE, Santa Barbara, CA, USA, 263–275.
- [44] Noman Mohammed, Dima Alhadidi, Benjamin CM Fung, and Mourad Debbabi. 2013. Secure Two-party Differentially Private Data Release for Vertically Partitioned Data. *IEEE Transactions on Dependable and Secure Computing* 11, 1 (2013), 59–71.

- [45] Nicolas Papernot, Martin Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2017. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *5th International Conference on Learning Representations*. OpenReview.net, Toulon, France.
- [46] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data Synthesis Based on Generative Adversarial Networks. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1071–1083.
- [47] John Platt. 1998. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. Technical Report MSR-TR-98-14. Microsoft.
- [48] Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robert Sandmann, Robin Roehm, and Michael A. Hoeh. 2021. PyVertical: A Vertical Federated Learning Framework for Multi-headed SplitNN. *CoRR abs/2104.00489* (2021).
- [49] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy*. IEEE, San Jose, CA, USA, 3–18.
- [50] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. 2022. Synthetic Data – Anonymisation Groundhog Day. In *31st USENIX Security Symposium*. ACM, Boston, MA, USA, 1451–1468.
- [51] Yan Sun, Lihua Yin, Licai Liu, and Shuang Xin. 2014. Toward Inference Attacks for k-Anonymity. *Personal and Ubiquitous Computing* 18, 8 (2014), 1871–1880.
- [52] Latanya Sweeney. 2002. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 5 (2002), 557–570.
- [53] Peng Tang, Xiang Cheng, Sen Su, Rui Chen, and Huaxi Shao. 2019. Differentially Private Publication of Vertically Partitioned Data. *IEEE Transactions on Dependable and Secure Computing* 18, 2 (2019), 780–795.
- [54] Aleksei Triastcyn and Boi Faltings. 2020. Federated Generative Privacy. *IEEE Intelligent Systems* 35, 4 (2020), 50–57.
- [55] Jaideep Vaidya and Chris Clifton. 2004. Privacy Preserving Naive Bayes Classifier for Vertically Partitioned Data. In *Fourth SIAM International Conference on Data Mining*. SIAM, Lake Buena Vista, Florida, USA, 522–526.
- [56] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and Scott Patterson. 2008. Privacy-preserving Decision Trees Over Vertically Partitioned Data. *ACM Transactions on Knowledge Discovery from Data* 2, 3 (2008), 1–27.
- [57] Gerrit van den Burg and Chris Williams. 2021. On Memorization in Probabilistic Deep Generative Models. In *Annual Conference on Neural Information Processing Systems 2021*. OpenReview.net, Virtual, 27916–27928.
- [58] Boxin Wang, Fan Wu, Yunhui Long, Luka Rimanic, Ce Zhang, and Bo Li. 2021. DataLens: Scalable Privacy Preserving Training via Gradient Compression and Aggregation. In *2021 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Virtual Event, Republic of Korea, 2146–2168.
- [59] Chang Wang, Jian Liang, Mingkai Huang, Bing Bai, Kun Bai, and Hao Li. 2020. Hybrid Differentially Private Federated Learning on Vertically Partitioned Data. *CoRR abs/2009.02763* (2020).
- [60] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled Rényi Differential Privacy and Analytical Moments Accountant. In *22nd International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 89)*. PMLR, Naha, Okinawa, Japan, 1226–1235.
- [61] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. 2018. Improving the Improved Training of Wasserstein GANs: A Consistency Term and its Dual Effect. In *6th International Conference on Learning Representations*. OpenReview.net, Vancouver, BC, Canada.
- [62] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. 2007. Minimality Attack in Privacy Preserving Data Publishing. In *33rd International Conference on Very Large Data Bases*. ACM, Vienna, Austria, 543–554.
- [63] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. 2020. Privacy Preserving Vertical Federated Learning for Tree-based Models. *Proceedings of the VLDB Endowment* 13, 11 (2020), 2090–2103.
- [64] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially Private Generative Adversarial Network. *CoRR abs/1802.06739* (2018).
- [65] Kai Yang, Tao Fan, Tianjian Chen, Yuanming Shi, and Qiang Yang. 2019. A Quasi-Newton Method Based Vertical Federated Learning Framework for Logistic Regression. *CoRR abs/1912.00513* (2019).
- [66] Andrew Chi-Chih Yao. 1982. Protocols for Secure Computations (Extended Abstract). In *23rd Annual Symposium on Foundations of Computer Science*. IEEE, Chicago, Illinois, USA, 160–164.
- [67] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019. Time-series Generative Adversarial Networks. In *32th International Conference on Neural Information Processing Systems*. Curran Associates, Inc., Vancouver, BC, Canada, 5509–5519.
- [68] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. *ACM Transactions on Database Systems* 42, 4 (2017), 25:1–25:41.
- [69] Longling Zhang, Bochen Shen, Ahmed Barnawi, Shan Xi, Neeraj Kumar, and Yi Wu. 2021. FedDPGAN: Federated Differentially Private Generative Adversarial Networks Framework for the Detection of COVID-19 Pneumonia. *Information Systems Frontiers* 23, 6 (2021), 1403–1415.
- [70] Nevin L. Zhang. 2004. Hierarchical Latent Class Models for Cluster Analysis. *The Journal of Machine Learning Research* 5 (2004), 697–723.
- [71] Xinyang Zhang, Shouling Ji, and Ting Wang. 2018. Differentially Private Releasing via Deep Generative Model. *CoRR abs/1801.01594* (2018).
- [72] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. 2022. Property Inference Attacks Against GANs. In *29th Annual Network and Distributed System Security Symposium*. The Internet Society, San Diego, CA, USA.
- [73] Tianyuan Zou, Yang Liu, Yan Kang, Wenhan Liu, Yuanqin He, Zhihao Yi, Qiang Yang, and Ya-Qin Zhang. 2022. Defending Batch-Level Label Inference and Replacement Attacks in Vertical Federated Learning. *IEEE Transactions on Big Data* 1 (2022), 1–12.

  **CC BY 4.0**

ATTRIBUTION 4.0 INTERNATIONAL

Deed

Canonical URL : <https://creativecommons.org/licenses/by/4.0/>

[See the legal code](#)

You are free to:

Share — copy and redistribute the material in any medium or format for any purpose, even commercially.

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give **appropriate credit**, provide a link to the license, and **indicate if changes were made**. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or **technological measures** that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable **exception or limitation**.

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as **publicity, privacy, or moral rights** may limit how you use the material.

Notice

This deed highlights only some of the key features and terms of the actual license. It is not a license and has no legal value. You should carefully review all of the terms and conditions of the actual license before using the licensed material.

Creative Commons is not a law firm and does not provide legal services. Distributing, displaying, or linking to this deed or the license that it summarizes does not create a lawyer-client or any other relationship.

Creative Commons is the nonprofit behind the open licenses and other legal tools that allow creators to share their work. Our legal tools are free to use.

- [Learn more about our work](#)
- [Learn more about CC Licensing](#)
- [Support our work](#)
- [Use the license for your own material.](#)
- [Licenses List](#)
- [Public Domain List](#)

Footnotes

appropriate credit — If supplied, you must provide the name of the creator and attribution parties, a copyright notice, a license notice, a disclaimer notice, and a link to the material. CC licenses prior to Version 4.0 also require you to provide the title of the material if supplied, and may have other slight differences.

- [More info](#)

indicate if changes were made — In 4.0, you must indicate if you modified the material and retain an indication of previous modifications. In 3.0 and earlier license versions, the indication of changes is only required if you create a derivative.

- [Marking guide](#)
- [More info](#)

technological measures — The license prohibits application of effective technological measures, defined with reference to Article 11 of the WIPO Copyright Treaty.

- [More info](#)

exception or limitation — The rights of users under exceptions and limitations, such as fair use and fair dealing, are not affected by the CC licenses.

- [More info](#)

publicity, privacy, or moral rights — You may need to get additional permissions before using the material as you intend.

- [More info](#)

[Contact](#)
[Newsletter](#)
[Privacy](#)
[Policies](#)
[Terms](#)

CONTACT US

Creative Commons PO Box 1866,
Mountain View, CA 94042

info@creativecommons.org

+1 415 429 6753

SUBSCRIBE TO OUR NEWSLETTER

SUBSCRIBE

SUPPORT OUR WORK

Our work relies on you! Help us keep the Internet free and open.

**DONATE
NOW**

Except where otherwise noted, content on this site is licensed under a [Creative Commons Attribution 4.0 International license](#).
Icons by [Font Awesome](#).