TUM School of Computation, Information and Technology
Technische Universität München

Generative Adversarial Networks for Time Series Generation and Translation

Hiba Arnout

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

Doktorin der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende:
Prof. Gudrun J. Klinker, Ph.D.

Prüfende der Dissertation:
1. Hon.-Prof. Dr. Thomas A. Runkler
2. Prof. Dr. Hans-Joachim Bungartz

Die Dissertation wurde am 22.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 14.11.2022 angenommen.

ABSTRACT

Motivated by its enormous economic benefits, many attempts have been made to integrate Artificial Intelligence (AI) in almost all industrial domains by automatizing tasks and developing smart products. However, this revolutionary technology and its speed of propagation are constrained by major challenges. One main issue is the lack of data and its poor quality. Moreover, a lot of privacy concerns are arising today regarding the protection of the data and the AI applications. To tackle these issues, a lot of researchers investigated various techniques to enable an efficient synthesis of new private and non-private data of high-quality.

In this thesis, we propose new generative approaches for time series data that leverage a considered dataset by increasing its diversity. We present a novel *Generative Adversarial Networks* (GAN) framework, *Class-specific Recurrent Generative Adversarial Networks* (ClaRe-GAN) that relies on *class specific encoders* and a *class discriminator* to extract the inter- and intra-class properties of a multi-class dataset. Our main goal is to enable a generation of diverse time series of high-quality by finding the right trade-off between sample fidelity, i.e., their similarity to the real time series, sample diversity, i.e., how different, and heterogeneous they are, and preserving the temporal relationship between the data points. Moreover, we investigate in depth the problem of mapping time series between different application domains by presenting *Disentangled Representation for Time Series Translation* (DR-TiST) a novel algorithm for time series translation. Thanks to the disentangled representation, each time series is split into functional behavior highlighting the properties of the time series and a context depicting the environmental setup and can be easily mapped to other application domains. Finally, we combine ClaRe-GAN, DR-TiST, and the state-of-the-art time series generation and translation methods with Differential Privacy (DP), a well-known technique to protect the privacy of dataset instances, to provide a collection of private time series generation and translation methods. The frameworks are fitted with a private Discriminator relying on Differentially Private Stochastic Gradient (DPSGD) and their performances in terms of finding the right trade-off between data privacy and data utility are assessed.

Abstract

Aware that assessing the quality of generated time series is of paramount importance, we test the existing GAN metrics for time series on detecting different GAN training problems. Our main purpose is to pinpoint the advantages and disadvantages of each metric and to recommend a possible combination of metrics that can be used together. Furthermore, we present a computational and visual method to accomplish this task. Particularly, we propose *Mean of incoming Variance of outgoing* (MiVo) a new evaluation metric that identifies for each synthetic time series a real one and for each real a synthetic one. We prove that this metric can detect numerous training problems simultaneously and use it in two different manners to evaluate the generated as well as the translated data. Moreover, we introduce a human-centered approach that, using Visual Analytics (VA) techniques, allows for further investigation and more precise comparison of the synthetic data to the real ones.

We show that our algorithms ClaRe-GAN and DR-TiST and their DP variants DP-ClaRe-GAN and DP-DR-TiST outperform the state-of-the-art algorithms for time series generation and translation by testing them on a set of public datasets and use cases where the translation topic is relevant. We also prove that MiVo outperforms the existing metrics in detecting common training problems. To sum up, we present in this work new approaches for time series generation and translation as well as a methodology to rigorously assess the quality of the obtained data. Our approaches constitute a significant improvement in this area and enable practitioners and researchers to deal with the lack of data problems occurring in many industrial domains by providing new standard or private data that can be freely shared between different stakeholders.

ZUSAMMENFASSUNG

Aufgrund ihrer enormen wirtschaftlichen Vorteile wurden viele Versuche unternommen, künstliche Intelligenz (KI) in fast alle industriellen Bereiche zu integrieren, indem Aufgaben automatisiert und intelligente Produkte entwickelt wurden. Diese revolutionäre Technologie und ihre schnelle Verbreitung werden jedoch durch große Herausforderungen behindert. Ein Hauptproblem ist der Mangel an Daten und deren schlechte Qualität. Außerdem gibt es heute viele Bedenken hinsichtlich des Schutzes der Daten und der KI-Anwendungen. Um diese Probleme zu lösen, haben viele Forscher verschiedene Techniken untersucht, die eine effiziente Synthese neuer privater und nicht-privater Daten von hoher Qualität ermöglichen.

In dieser Arbeit schlagen wir neue generative Ansätze für Zeitreihendaten vor, die einen betrachteten Datensatz durch Erhöhung seiner Diversität nutzen. Wir präsentieren ein neuartiges Generative Adversarial Networks (GAN) Framework, Class-specific Recurrent Generative Adversarial Networks (ClaRe-GAN), das sich auf klassenspezifische Kodierer und einen Klassendiskriminator stützt, um die Inter- und Intraklasseneigenschaften eines Mehrklassendatensatzes zu extrahieren. Unser Hauptziel ist es, die Erzeugung vielfältiger Zeitreihen von hoher Qualität zu ermöglichen, indem wir den richtigen Kompromiss zwischen Sample Fidelity, d.h. ihrer Ähnlichkeit mit den realen Zeitreihen, Sample Diversity, d.h. wie unterschiedlich und heterogen sie sind, und der Erhaltung der zeitlichen Beziehung zwischen den Datenpunkten finden. Darüber hinaus untersuchen wir eingehend das Problem der Abbildung von Zeitreihen zwischen verschiedenen Anwendungsdomänen, indem wir mit der "Disentangled Representation for Time Series Translation" (DR-TiST) einen neuartigen Algorithmus für die Translation von Zeitreihen vorstellen. Dank der Disentangled Representation wird jede Zeitreihe in ein funktionales Verhalten, das die Eigenschaften der Zeitreihe hervorhebt, und einen Kontext, der die Umgebungsbedingungen darstellt, aufgeteilt und kann leicht auf andere Anwendungsbereiche übertragen werden. Schließlich kombinieren wir ClaRe-GAN, DR-TiST und die hochmodernen Zeitreihengenerierungs- und -translationsmethoden mit Differential Privacy (DP), einer bekannten Technik zum Schutz der Privatsphäre

von Datensatzinstanzen, um eine Sammlung von privaten Zeitreihengenerierungs- und -translationsmethoden bereitzustellen. Die Algorithmen werden mit einem privaten Diskriminator ausgestattet, der auf Differentially Private Stochastic Gradient (DPSGD) beruht, und ihre Leistungen im Hinblick auf die Suche nach dem richtigen Kompromiss zwischen Datenschutz und Datennutzen werden bewertet.

Da die Bewertung der Qualität der erzeugten Zeitreihen von größter Bedeutung ist, testen wir die bestehenden GAN-Metriken für Zeitreihen auf die Erkennung verschiedener GAN-Trainingsprobleme. Unser Hauptziel ist es, die Vor- und Nachteile der einzelnen Metriken aufzuzeigen und eine mögliche Kombination von Metriken zu empfehlen, die zusammen verwendet werden können. Außerdem stellen wir eine rechnerische und visuelle Methode vor, um diese Aufgabe zu erfüllen. Insbesondere schlagen wir Mean of incoming Variance of outgoing (MiVo) vor, eine neue Bewertungsmetrik, die für jede synthetische Zeitreihe eine reale und für jede reale eine synthetische identifiziert. Wir beweisen, dass diese Metrik zahlreiche Trainingsprobleme gleichzeitig erkennen kann, und verwenden sie auf zwei verschiedene Arten, um sowohl die generierten als auch die transformierten Daten zu bewerten. Darüber hinaus führen wir einen menschenzentrierten Ansatz ein, der mit Hilfe von Visual Analytics (VA)-Techniken eine weitere Untersuchung und einen genaueren Vergleich der synthetischen Daten mit den realen Daten ermöglicht.

Wir zeigen, dass unsere Algorithmen ClaRe-GAN und DR-TiST sowie ihre DP-Varianten DP-ClaRe-GAN und DP-DR-TiST die modernsten Algorithmen für die Erzeugung und Translation von Zeitreihen übertreffen, indem wir sie an einer Reihe von öffentlichen Datensätzen und Anwendungsfällen testen, bei denen das Translationsthema relevant ist. Wir beweisen auch, dass MiVo die bestehenden Metriken bei der Erkennung häufiger Trainingsprobleme übertrifft. Zusammenfassend stellen wir in dieser Arbeit neue Ansätze für die Erzeugung und Translation von Zeitreihen sowie eine Methodik zur Bewertung der Qualität der gewonnenen Daten vor. Unsere Ansätze stellen eine erhebliche Verbesserung in diesem Bereich dar und ermöglichen es Praktikern und Forschern, das Problem des Datenmangels, das in vielen industriellen Bereichen auftritt, zu lösen, indem sie neue Standard- oder private Daten bereitstellen, die von verschiedenen Beteiligten frei geteilt werden können.

ACKNOWLEDGEMENT

Foremost, I would like to thank Prof. Dr. Thomas A. Runkler for all his efforts and his support. Thomas listened to me when I needed it and pushed me to my limits. He really helped me to go further and to improve my skills through exciting and detailed conversations where I had to deepen my knowledge and investigate my ideas so, I can meet up the level. It is an honor to write this thesis by his side. Thank you for your precious advice, and help.

Moreover, I would like to thank my Siemens coworkers and mentors Johanna Bronner and Johannes Kehrer for their priceless help, their patience, and all the meetings and discussions, especially at the beginning. We all know that the hardest thing is, to begin with, the thesis topic. Also, a dedicated thank you goes to all my Siemens colleagues and especially to Ariane Sutor for giving me the opportunity to do my Ph.D. in the research group and the freedom to work on my research topics and interesting industrial projects. I would also like to thank Regine Meunier, Axel Reitinger, and Stefan Hagen Weber for their guidance, advice, and good vibes.

On another level, a particular thank you to my special aunt Asma and her husband Zouheir for their infinite support and love. Then comes people, that I would like but can not thank enough my parents Saloua and Yassine for everything they have done for me since, well... forever. And certainly, during these hard years of study. Thank you for believing in me and for supporting me morally and financially. I am also grateful for Nour, Ghassen and my brother Ali for the support, incentives, and thesis reviews. The last word of gratitude goes to my fiance Wassim who listened to me and encouraged me during all these years, particularly in the difficult moments. Thank you for being here.

Last but not least, the most important one, Grandma Bahia to whom I dedicate this thesis. You left us a few months ago, I wish you were here but I'm sure wherever you are, you are proud. To you, from all my heart.

CONTENTS

ACRONYMS

AI	Artificial Intelligence.
C-RNN-GAN	Continuous Recurrent Neural Networks Generative Adversarial Networks.
cGAN	Conditional Generative Adversarial Networks.
ClaRe-GAN	Class-Specific Recurrent Generative Adversarial Networks.
CNN	Convolutional Neural Networks.
CycleGAN	Cycle-consistent Generative Adversarial Networks.
CycleGAN-VC	Cycle-Consistent Generative Adversarial Networks based Voice Conversion.
DC	Direct Current.
DCGAN	Deep Convolutional Generative Adversarial Networks.
DL	Deep Learning.
DP	Differential Privacy.
DP-C-RNN-GAN	Differentially Private Continuous Recurrent Neural Networks Generative Adversarial Networks.
DP-ClaRe-GAN	Differentially Private Class-specific Recurrent Generative Adversarial Networks.
DP-CycleGAN-VC	Differentially Private Cycle-consistent Adversarial Networks based Voice Conversion.
DP-DR-TiST	Differentially Private Disentangled Representation for Time Series Translation.

DP-TimeGAN	Differentially Private Time-series Generative Adversarial Networks.
DPSGD	Differentially Private Stochastic Gradient Descent.
DR-TiST	Disentangled Representation for Time Series Translation.
DRIT	Disentangled Representation for Image-to-Image Translation.
DS	Discriminative Score.
FID	Frechet Inception Distance.
GAN	Generative Adversarial Networks.
InfoGAN	Information Maximizing Generative Adversarial Networks.
LSTM	Long Short-term Memory.
MiVo	Mean of incoming Variance of outgoing.
ML	Machine Learning.
MMD	Maximum Mean Discrepancy.
PCA	Principal Components Analysis.
PID	Proportional Integral Derivative.
PS	Predictive Score.
RCGAN	Recurrent Conditional Generative Adversarial Networks.
RDP	Renyi Differential Privacy.
RGAN	Recurrent Generative Adversarial Networks.
RNN	Recurrent Neural Networks.
t-SNE	T-distributed Stochastic Neighbor Embedding.
TimeGAN	Time-series Generative Adversarial Networks.
TRTS	Train on Real Test on Synthetic.
TSTR	Train on Synthetic Test on Real.
UNIT	Unsupervised Image-to-image Translation.
VA	Visual Analytics.
WGAN	Wasserstein Generative Adversarial Networks.
WGAN-GP	Wasserstein Generative Adversarial Networks with Gradient Penalty.

CHAPTER

1

INTRODUCTION

AI became a powerful means to leverage machines and accomplish tasks automatically and efficiently by imitating the decision-making process of humans. Therefore, huge efforts are made to broaden its adoption and develop smart industrial products that meet the global celerity requirements. This technology is playing a central role for businesses and organizations to improve their operating systems and their benefits. By its numerous applications, AI was deeply investigated for different purposes ranging from predictive maintenance, and various applications in the automotive industry to detecting rare diseases in healthcare. By way of example, recent ML applications help to increase customer satisfaction using chatbots or virtual assistants that support customers and process their requests quickly and efficiently. Despite all the recent improvements, AI has not yet reached the peak of its impact.

In spite of its fast and continuously growing adoption, the implementation of AI in real-world situations is still facing key challenges. A major challenge is the lack of data, i.e., the insufficient amount of available data and its poor quality. To rigorously test the developed AI methods, we must have high-quality data which faithfully depicts a full picture of the to-be-automatized situation and its complexity. Industrial partners and ML experts are struggling to (i) define the characteristics of the dataset that properly describe their use case and (ii) collect the required amount of data. Another fundamental issue is the growing privacy concerns. As a matter of fact, external collaborations on improving and expanding the use of ML, are highly restricted, and sharing data remains a sensitive subject. Furthermore, the protection of the data and the ML applications, from cybersecurity attacks, is considered a high priority.

1.1 Motivation

To build widely applicable and trustworthy ML models, a lot of researchers focused on improving the quality of the dataset that is used in the ML application by generating more data. As a matter of fact, a sufficient amount of data and a balanced dataset where each class is equally well represented are essential for a good performance for any ML model. A newly investigated DL architecture involving two neural networks a Generator generating from random noise new data and a Discriminator, a binary classifier, competing against each other called GAN [1] attracted a lot of researchers. This DL technique is based on a simple and intuitive idea: while D tries to maximize its log-likelihood to distinguish between the real and the generated data, G aims at minimizing the log-probability of the generated samples that are recognized as false.

The state-of-the-art GAN models are showing an outstanding performance in generating diverse high-resolution images such as with BigGAN [2] and StyleGAN [3, 4]. Moreover, the evaluation and the assessment of the quality of the obtained data are for this type of data straightforward. By way of example, the Inception Score [5, 6] and FID [7] are commonly used in the evaluation process. These methods rely on a pretrained model for image classification namely the inception model [8] trained on the ImageNet dataset with over 15 million labeled high-resolution images and hence can be easily applied to other image datasets. Unfortunately, such efficient generative models and evaluation techniques are not available for other types of data such as time series making these methods not applicable. Indeed, finding a suitable generative model for time series is challenging and a lot of factors must be considered when designing a good generation framework for this type of data. One main requirement is that the time series generation process preserves the temporal relationship between the data points. Moreover, as for other types of data, an efficient generative model should be able to find a compromise between sample fidelity, i.e., the similarity of the generated data to the real ones, and sample diversity, i.e., reproducing the variation of the real data.

Lately, numerous generative models for time series data have been presented. A first generative model for continuous sequential data was proposed by Mogren [9] namely C-RNN-GAN consisting of a recurrent Generator and Discriminator. A well-known method to improve the performance of GAN is to condition GAN on additional information extracted from the original data [10]. In this context, Esteban et al. proposed a more efficient model, RCGAN [11], i.e., a recurrent GAN augmented with auxiliary conditional information. More recently, Yoon et al. introduced TimeGAN which preserves better the temporal dynamic between the data points and can be used on mixed datasets, i.e., a dataset with static and temporal data. In this work, we will prove that these state-of-the-art frameworks, despite generating realistic time series in simple environments, fail to produce acceptable results preserving the inter- and intra-class diversity of the original dataset on multi-class datasets. Driven by the recent success of conditional GAN [10] and similar to RCGAN, we introduce ClaRe-GAN a *generative class sensitive framework for time series.*

Our approach is based on the simple and intuitive idea: to deal with datasets of high-variability, learning their inter- and intra-class variations is the basis, and hence, our

starting point. We assume that time series from various classes within the same dataset share some common information that should be contained in the generated data. To extract this information, we use class-specific encoders, one encoder per class, and make a shared-latent space assumption enforcing the latent vectors to be mapped to a common latent space. At the same time, a major challenge when using GAN on datasets with multiple classes is not only to preserve the classes but also the diversity within each class. To achieve this, we use an extra *class discriminator*, that is trained by applying a class adversarial loss to enable an efficient comparison between the latent codes of the different encoders and a precise extraction of the class-specific attributes. Thanks to this special setup we ensure that the extracted latent codes are class-specific, and, at the same time, contain the high-level representations of the original dataset. Moreover, our framework uses a collection of loss functions borrowed from image-to-image translation techniques and generative models for images to improve the diversity of the generated time series. We finally prove that a generator conditioned on these representations outperforms the existing models by generating time series of much higher quality, i.e., diverse samples that preserve the classes of the original dataset and their properties very well (high-fidelity).

To enhance the diversity of an existing dataset and to explore never seen conditions, researchers used GAN to tackle another issue namely image-to-image translation. The task consists in learning a mapping function between two visual domains to map the elements of an image to another visual context. Such techniques can improve the quality of a dataset and the robustness of a ML model by testing it in new setups that can be more complex or more specific. It can enrich the original dataset with new data reflecting non-realistic and never seen conditions.

The field of image-to-image translation is also experiencing rapid progress. In fact, a lot of works, that have been proposed, aim to determine a mapping function between two visual domains. Some of them [12] require paired-data during the training process to transform a specific image from a source domain to a target domain. Others achieve image-to-image translation without coupled data. By way of example, CycleGAN [13] involves two discriminators and two generators to perform image-to-image translation and relies on a cycle consistency constraint to ensure cyclic reconstruction of the images. Latest works, such as UNIT [14] propose a more complex structure that involves coupled GAN. Furthermore, DRIT [15] suggests a new structure that captures the attribute and content specific features of the image. In this work, we exploit the recent progress in image-to-image translation to perform time series translation, i.e., to map time series from one domain to another domain for example to map different machines in different operating environments.

To this end, we propose DR-TiST [16, 17] that enables an efficient time series translation. We apply our algorithm to three various use cases where translating time series can be advantageous and relevant, namely translating ventilation systems, human activities, and DC motors. While in the first scenario we transfer the time series behavior of a ventilation system to the environmental conditions of another one, we try in the second scenario to transform human sensor measurements to depict specific human activities. Finally, in the third scenario, we focus on finding the optimal controller for a

non-accessible DC motor. The performance of DR-TiST is compared to CycleGAN-VC [18], a special form of an image-to-image translation algorithm used for voice conversion. We demonstrate that the time series generated by DR-TiST are more realistic than the ones generated by CycleGAN-VC.

In many situations, such as in medical or industrial domains, the AI deployment is limited, and its power is still unknown. The extremely restricted amount of available data and privacy issues [19] bound the collaboration possibilities. For example, it was proven that launched attacks against ML models may reveal sensitive information about the training dataset and infer its membership. In these cases, providing private synthetic data that depict the real data's behavior and rigorously describe the application domain can encourage collaborations between data owners and external partners on improving and expanding the use of AI. Time series generation and translation techniques are key method to enhance the diversity of the original data by either producing more data or mapping the original time series from a source to a target domain with never seen conditions. Combining these methods with privacy will provide an excellent opportunity to obtain more data and guarantee privacy.

To provide data with high-diversity and privacy guarantees, we extend existing methods and develop new ones to enable a time series generation and translation with DP. In general, such approaches will encourage data holders to publish their data and hence provide interesting complex real-world use cases for the research community. At the same time, it will permit them to find new collaboration opportunities and work safely with external partners on improving the use of AI in their application domains. In fact, the synthetic data still depict the general pattern of the real data and have the same reactivity to the ML model. However, they do not expose sensitive information contained in the initially recorded time series such as rare diseases, machine parameters, or times when the machines were on/off... In this work, we equip the developed and existing generation and translation frameworks with a private discriminator relying on DPSGD to enable a time series generation and transformation with privacy guarantees. The developed methods are evaluated in terms of privacy and usefulness on different use cases. We show that data translated and generated with these techniques are private and still valuable. Our approaches constitute a first step in transforming data with privacy guarantees.

Evaluating the generative models for time series remains complicated and is performed in most the cases visually by assessing the quality of the synthetic data. Indeed, this task imposes new criteria that must be considered such as the temporal relationship between the data points. To overcome these issues, researchers suggested different compatible metrics: Esteban, Hyland, and Rätsch [11] proposed two evaluation metrics while presenting RCGAN namely TSTR and TRTS that compute the test accuracy of a ML model trained on a set of synthetic data and tested on a set of real data and vice versa. Yoon et al. used the PS and the DS to evaluate TimeGAN [20]. In this case, a 2-layer LSTM is trained to distinguish between the real and synthetic data or to predict the next timestamp based on the previous part of the time series. Additionally, MMD [21] was used in the training [22] and the evaluation of GAN to measure the similarity between the distribution of the real and generated data. In spite of the significant number of these

proposed metrics, they have never been systematically compared to each other. This prevented us from identifying the most accurate method and from pointing out objective compulsory criteria that represent a reliable metric. Furthermore, it seems difficult to determine their advantages and disadvantages in terms of efficiency and discriminability as well as their ability in detecting different problems that may occur during the training process such as overfitting, mode collapse, or mode dropping.

To tackle these issues, we extensively study these metrics by performing different evaluation tests [23]. Furthermore, we propose a new metric, MiVo [24], that doesn't rely on a ML model and simultaneously takes into account the temporal characteristics of this variety of data by performing a bidirectional check starting from real to synthetic and afterward from synthetic to real. Finally, we will demonstrate that our new metric outperforms the existing ones in all conducted tests and is more efficient in terms of memory and time consumption. As the evaluation of the translated time series remains ambiguous and use case specific, we adapt in this thesis MiVo by applying it in a sophisticated manner to enable a comparison between the target time series and the ones obtained after the translation procedure.

The previously described evaluation metrics give the ML expert a global overview of the GAN model behavior by computing a score per iteration. This enables a fast exploration of the different training iterations and a quick selection of the iteration with the best score and hence the best performance. However, in many situations, it is not enough to trust the computed score and it remains essential to rigorously investigate and explore the generated data. By way of example, it might be helpful to compare the obtained data with the real ones visually by a human judge. At the same time, comparing an important amount of time series and highlighting their similarities and differences is a complex task. In this thesis, we present a VA system [25, 26] to guide the ML expert in this evaluation process for time series data. The developed framework presents a method that makes the real and generated data easily comparable by combining VA (Colorfield, TimeHistograms) with algorithmic methods and enables the ML expert to trust the trained GAN model. The presented approach consists of two views, namely a GAN Iteration View showing similarity metrics between real and generated data over the iterations of the generation process and a Detailed Comparative View equipped with different time series visualizations such as TimeHistograms, to compare the generated data at different iteration steps. Starting from the GAN Iteration View, the user can choose suitable iteration steps for detailed inspection.

1.2 Research Questions and Contributions

To sum up, we consider in this thesis the problem of finding novel and efficient algorithms to achieve time series generation and translation. Particularly, we focus on developing an efficient generative model for time series stemming from multi-class datasets. Moreover, we aim at designing a framework that can map time series across different application domains.

Finding a suitable generative model for time series is a challenging task. In fact, many aspects must be taken into consideration. The first challenge which is valid for all types of data is to find the right trade-off between sample diversity and sample fidelity. In addition to that, it is extremely important that the generated time series preserve the temporal relation between the data points and the dynamism of the real ones. In spite of the important number of recently proposed generative models for time series namely C-RNN-GAN [9], RCGAN [11], and TimeGAN [20], their performance is still limited on datasets with high-variability such as multi-class datasets. To generate synthetic time series of high-quality for multi-class datasets, it is essential to capture the class-specific attributes as well as the class-independent properties:

RQ1G: How can we generate synthetic multi-class time series with the right trade-off between sample diversity and sample fidelity? To tackle this research question, we present ClaRe-GAN [A] and hence contribute as follows:

- Present a novel generative class sensitive framework for time series.
- Compare our method to the state-of-the-art generative models and assess the performance of all the models against three criteria namely diversity, fidelity, and utility of the generated data.
- Test the performance of all models on different publicly available datasets that vary in the length of the time series, the number of classes, and the amount of data per class.

A lot of effort was made to develop efficient methods that achieve image-to-image translation [13, 14, 15, 27], i.e., that map the elements of an image from an initial visual domain to another visual domain. Such techniques can enhance the diversity of an image dataset by synthesizing new data describing non-existing situations. Inspired by them, we consider the problem of developing an efficient algorithm that maps time series from a source to a target domain:

RQ1T: How can we adapt existing image-to-image translation methods to translate time series between different application domains? In this context, we propose in contributions [B, C], DR-TiST, an efficient algorithm for time series translation:

- Introduce a new method to translate time series from an initial domain $\mathcal{A}$ to a target domain $\mathcal{B}$ by adapting an existing image-to-image translation algorithm, namely DRIT [15], to time series data.
- Test DR-TiST on three use cases where we transfer the behavior of a first machine to the environmental domain of a second machine, transform sensor measurements depicting human activities and tune the controller of a non-accessible DC motor.
- Compare the performance of our algorithm to CycleGAN-VC [18] a modified version of CycleGAN [13] designed for voice conversion and consequently able to process sequential data.

Moreover, we consider the problem of generating and translating time series while preserving the privacy of the original dataset. The obtained datasets after the generation and translation should not reveal some sensitive information contained in the real dataset. This research problem can be divided into two questions:

- **RQ2G: How can we generate synthetic time series with both a low privacy loss and a high utility across different domains?** We tackle this question by presenting differentially private generative models for time series [D] as follows:
  - Combine the existing time series generation algorithms with DP and propose DP-ClaRe-GAN, DP-C-RNN-GAN, and DP-TimeGAN.
  - Identify the DP generative algorithms that generate time series with low privacy and a high utility across different domains.
  - Test the performance of all models on different publicly datasets.
- **RQ2T: How can we translate time series between different application domains so that the characteristics of the real dataset are not revealed?** We tackle this question by proposing a differentially private version of CycleGAN-VC and DR-TiST [E] and contributing as follows:
  - Combine the existing time series translation algorithms with DP and propose DP-CycleGAN-VC and DP-DR-TiST.
  - Compare the performance of the developed algorithms in terms of utility and privacy.
  - Test the performance of all models on three different use cases where translating time series is relevant.

Next, we consider the problem of assessing the quality of the generated and translated time series by easing the comparison between the obtained time series and the target ones. Our main goal is to support the ML expert to either accept or reject the ML model by assessing the quality of its output data:

RQ3: How can we compare generated or translated time series with the real or target ones? To this end, we develop a computational approach, MiVo [F], and a visual approach [G, H] that contribute as follows:

- Propose a new metric to evaluate GAN models that achieve time series generation and translation.
- Support GAN expert in the hyper-parameter tuning.
- Propose a VA approach consisting of two views to compare the time series.
- Propose an overview visualization that helps the ML expert to identify interesting iterations of the GAN generation process.

- Present a comparison interface where the time series are visualized in a compact manner and ordered using PCA to facilitate comparison by juxtaposition.
- Compare the visual and computational quality assessment methods by highlighting the advantages of each approach and giving some instructions on how to use each method.

In the last few years, a lot of metrics were proposed to evaluate the performance of the generative models for time series data. While Esteban et al. proposed TSTR and TRTS as they introduced RCGAN [11], the PS and DS scores were defined by Yoon et al. while presenting TimeGAN [20]. Unfortunately, these metrics have not been compared to each other, and their efficiency in discriminating between real and synthetic samples and in detecting common problems that may happen while training GAN models is unclear. This makes it difficult to determine the most accurate and reliable metric:

RQ4: Which metric allows to detect common GAN training problems? In this research question, we focus on comparing MiVo to the state-of-the-art metrics and highlighting the advantages and disadvantages of each metric in terms of detecting GAN training problems and distinguishing between real and synthetic samples. Our main purpose is to give ML experts some recommendations on how to use these metrics:

- Review the existing GAN evaluation metrics for time series data.
- Analyze and compare these methods by performing different evaluation tests.
- Identify the most accurate metric and determine the strengths and weaknesses of each metric.

To conclude, this thesis contributes with new algorithms for time series generation and translation, by extending the existing and newly presented algorithms to be applicable for private setups and finally by developing a computational and visual method to ease the comparison between the obtained time series and the desired behavior. Table 1.1 summarizes the considered research topics, the underlying contributions, and the chapters of this thesis that tackle the different research questions.

Own publications

[A] H. Arnout, J. Bronner, and T. Runkler. ClaRe-GAN: Generation of Class-Specific Time Series. In 2021 IEEE Symposium Series on Computational Intelligence (SSCI), 2021. [28]

[B] H. Arnout, J. Bronner, J. Kehrer, and T. Runkler. DR-TiST: Disentangled Representation for Time Series Translation Across Application Domains. In 2020 International Joint Conference on Neural Networks (IJCNN), 2020. [16]

[C] H. Arnout, J. Bronner, J. Kehrer, and T. Runkler. Translation of Time Series Data from Controlled DC Motors using Disentangled Representation Learning. In 2021 IEEE Symposium Series on Computational Intelligence (SSCI), 2021. [17]

Table 1.1: Summary of the main contributions in this thesis, the corresponding research questions and the chapters of the thesis where each research topic is addressed.

	Time Series Generation	Time Series Translation
Algorithm	**RQ1G:** How can we generate synthetic multi-class time series with the right trade-off between sample diversity and sample fidelity? **Method:** ClaRe-GAN **Chapter:** 3.1 **Reference:** [A]	**RQ1T:** How can we adapt existing image-to-image translation methods to translate time series between different application domains? **Method:** DR-TiST **Chapter:** 4.1 **Reference:** [B, C]
Differentially Private	**RQ2G:** How can we generate synthetic time series with both a low privacy loss and a high utility across different domains? **Method:** • DP-ClaRe-GAN • DP-C-RNN-GAN • DP-TimeGAN **Chapter:** 3.2 **Reference:** [D]	**RQ2T:** How can we translate time series between different application domains so that the characteristics of the real dataset are not revealed? **Method:** • DP-DR-TiST • DP-CycleGAN-VC **Chapter:** 4.2 **Reference:** [E]
Analysis	**RQ3:** How can we compare generated or translated time series with the real or target ones? **Method:** MiVo, VA framework **Chapter:** 5 **Reference:** [F, G, H] **RQ4:** Which metric allows to detect common GAN training problems? **Chapter:** 5 **Reference:** [F]	

[D] H. Arnout, J. Bronner, and T. Runkler. Differentially Private Time Series Generation. In Computational Intelligence and Machine Learning ESANN 2021 Proceedings, pp. 617-622, 2021. [29]

[E] H. Arnout, J. Bronner, and T. Runkler. Privacy-Preserving Time Series Translation. Submitted to Neural Networks, 2022. [30]

[F] H. Arnout, J. Bronner, and T. Runkler. Evaluation of Generative Adversarial Networks for Time Series Data. In 2021 International Joint Conference on Neural Networks (IJCNN), 2021. [24]

[G] H. Arnout, J. Bronner, J. Kehrer, and T. Runkler. Visual Evaluation of Generative Adversarial Networks for Time Series Data. In Human-Centered AI: Trustworthiness of AI Models & Data (HAI) track at AAAI Fall Symposium, 2019. [25]

[H] H. Arnout, J. Bronner, J. Kehrer, and T. Runkler. Evaluierungsrahmen für Zeitreihendaten. European pat. EP3809334A1. Siemens AG. Apr. 21, 2021. [26]

[I] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim. Towards a rigorous evaluation of XAI methods on time series. In 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 4197-4201, 2019. [31]

[J] U. Schlegel, E. Cakmak, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim. Towards visual debugging for multi-target time series classification. In Proceedings of the 25th International Conference on Intelligent User Interfaces, pp. 202-206, 2020. [32]

1.3 Thesis Outline

The content of this thesis is structured as follows. Chapter 2 introduces the necessary theoretical background and the methodologies that will be applied and used throughout the thesis. The core idea of GAN, a well-established method for image synthesis, is described from an algorithmic point of view. Moreover, common time series generation and translation techniques for time series such as TimeGAN, C-RNN-GAN, and RC-GAN are explained in detail. Furthermore, a key method to guarantee privacy while translating and generating data is presented. In this context, the intuition behind DP and its theoretical definition is introduced.

In Chapter 3, a novel algorithm, ClaRe-GAN, to synthesize new time series for an existing dataset of time series stemming from different classes is presented. In this respect, we discuss the performance of this algorithm in finding the right trade-off between sample diversity, i.e., how different, and heterogeneous are they among themselves, and sample fidelity, i.e., their closeness and similarity to the original dataset, and compare it to the state-of-the-art generative models. Aware that the privacy topic and privacy-preserving approaches are growing in importance, we combine the state-of-the-art generative model with DP to ensure a private time series generation.

Similarly, in Chapter 4, a new framework, DR-TiST, to map time series between different application domains is proposed. Its performance is assessed and compared to CycleGAN-VC, a state-of-the-art translation framework for voice conversion purposes. To this end, we present three different use cases where the time series translation topic is important. The privacy topic is also covered in this chapter. We present DP-DR-TiST and DP-CycleGAN-VC that enable a differentially private time series translation by combining the DR-TiST and CycleGAN-VC with DP. The performance of the DP models is assessed in terms of sample utility and sample privacy.

Chapter 5 presents two different analytical approaches, i.e., a visual and a computational one to assess the quality of time series. Both methods are meant to ease the comparison between the obtained time series from the ML model after a generation or translation process and a set of ground truth time series that depict the desired behavior and the requested output. Our main purpose is to support the ML expert or domain expert while evaluating a generative model and to help him to find the best performing ML model.

Finally, Chapter 6 concludes this work by summarizing the main research contributions that were addressed in this thesis and the obtained results. Furthermore, we highlight some potential future research directions that can be further investigated.

CHAPTER

2

PRELIMINARIES

In this chapter, we extensively review the theoretical background of this thesis and the methodologies that will be applied throughout this work. Particularly, we focus on the data generation topic by presenting GAN a newly introduced technique for image synthesis, and by reviewing the state-of-the-art time series generation algorithms, i.e., C-RNN-GAN [9], RCGAN [11], and TimeGAN [20] and comparing the complexity of their architectures. Moreover, we summarize the existing actual metrics meant to evaluate the performance of these frameworks. After that, we introduce numerous data transformation methods achieving image-to-image translation such as CycleGAN [13], DRIT [15, 27], and UNIT [14] and time series translation such as CycleGAN-VC. Finally, the concept of DP and its purpose are rigorously defined. In addition to that, some newly introduced Differentially Private Machine Learning methods are presented.

2.1 Data Generation

2.1.1 Generative Adversarial Networks

GAN, presented in Algorithm 1, is a well-known technique for image generation introduced by Goodfellow [1] consisting of two neural networks namely a Generator $G : \mathcal{Z} \to \mathcal{X}_g$, that starting from random noise $z \in \mathcal{Z}$ sampled from a distribution p_z tries to generate realistic images, and a Discriminator (D), a binary classifier that learns to perfectly distinguish between a set of real images $\mathcal{X}$ and the images generated by G, i.e., $\mathcal{X}_g$. At the same time, G is trying to minimize the log-probability of the generated samples that are recognized as false. Both neural networks are involved in a minimax game with the aim of learning a distribution p_g that approximates p_r, the distribution

of the real samples. While the D is trained to maximize $log\left(D\left(x\right)\right)$, G aims to minimize $log\left(1-D\left(G\left(z\right)\right)\right)$. The minimax loss can be expressed as follows:

$$\mathcal{L}_{GAN}\left(G,D\right)=\mathbb{E}_{x\sim p_r(x)}\left[log\left(D\left(x\right)\right)\right]+\mathbb{E}_{z\sim p_z(z)}\left[log\left(1-D\left(G\left(z\right)\right)\right)\right], \tag{2.1}$$

where $x\in\mathcal{X}$. The GAN's structure is illustrated in Fig. 2.1.

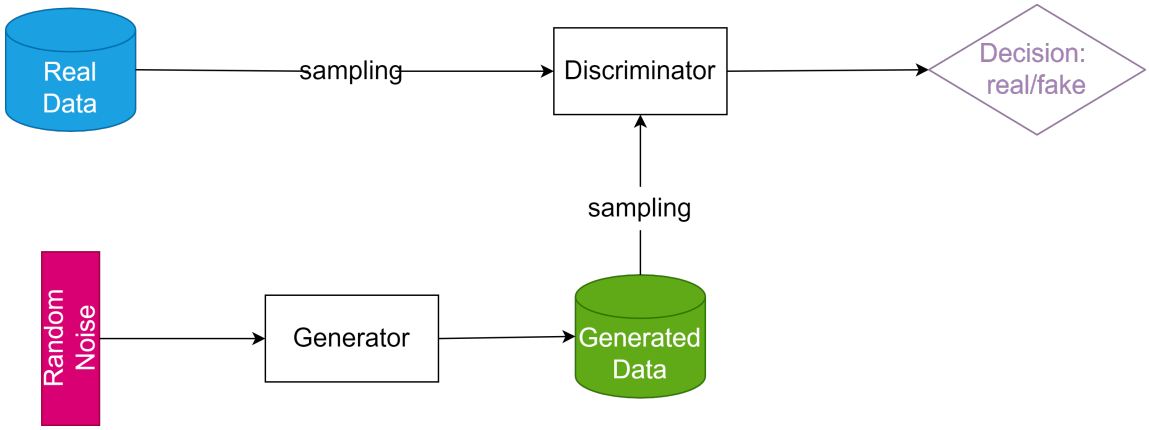

Figure 2.1: Presentation of GAN architecture consisting of two neural networks the Generator and the Discriminator. Starting from noise data are generated by the Generator. Later, the real and the generated data are compared by the Discriminator. While the Generator tries to synthesize data that are as realistic as possible, the Discriminator learns to perfectly distinguish between the real and the generated data. The Generator will over the iterations improve the quality of the data by learning from the decisions made by the Discriminator.

The global optimality and the convergence of GAN were proven from a theoretical point of view in [1]. Global optimality means reaching an optimal Discriminator D for a fixed Generator G. This is achieved when the derivative of D in the minimax loss is equal to zero:

$$D^{*}\left(x\right)=\frac{p_r\left(x\right)}{p_r\left(x\right)+p_g\left(x_g\right)}, \tag{2.2}$$

where $D^{*}\left(x\right)$ denotes the optimal Discriminator. The corresponding minimax loss function can then be computed as follows:

$$\begin{aligned}\mathcal{L}_{GAN}\left(G,D\right)=\mathbb{E}_{x\sim p_r(x)}\left[log\frac{p_r\left(x\right)}{\frac{1}{2}\left[p_r\left(x\right)+p_g\left(x_g\right)\right]}\right]+\\ \mathbb{E}_{x_g\sim p_g(x)}\left[log\frac{p_g\left(x_g\right)}{\frac{1}{2}\left[p_r\left(x\right)+p_g\left(x_g\right)\right]}\right]-2\cdot log\left(2\right).\end{aligned} \tag{2.3}$$

The Kullback-Leibler (KL) divergence of two different distributions p_1 and p_2 is defined as follows:

$$KL\left(p_1\parallel p_2\right)=\mathbb{E}_{x\sim p_1}log\frac{p_1}{p_2}, \tag{2.4}$$

Algorithm 1 GAN [1]

for number of training iterations **do**
  for k steps **do**
    Sample minibatch of m noise samples $\left\{z^1, \ldots, z^m\right\}$ from noise prior $p_g(z)$.
    Sample minibatch of m noise samples $\left\{x^1, \ldots, x^m\right\}$ from noise prior $p_r(x)$.
    Update the disciminator by ascending its stochastic gradient:

$$\nabla_\theta d \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

  end for
  Sample minibatch of m noise samples $\left\{z^1, \ldots, z^m\right\}$ from noise prior $p_g(z)$.
  Update the generator by descending its stochastic gradient:

$$\nabla_\theta d \frac{1}{m} \sum_{i=1}^{m} log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$

end for
The gradient-based updates can use any standard gradient-based learning rule.

and the Jensen-Shannon (JS) divergence denotes:

$$JS\left(p_1 \parallel p_2\right) = \frac{1}{2} KL\left(p_1 \parallel \frac{p_1 + p_2}{2}\right) + \frac{1}{2} KL\left(p_2 \parallel \frac{p_1 + p_2}{2}\right). \tag{2.5}$$

Hence, $\mathcal{L}_{GAN}\left(G, D\right)$ can be expressed as follows:

$$\mathcal{L}_{GAN}\left(G, D\right) = 2 \cdot JS\left(p_r \parallel p_g\right) - 2 \cdot log\left(2\right). \tag{2.6}$$

The rise of GAN enabled to investigate a new revolutionary technique for data generation. In addition to data generation, GAN was used for different purposes ranging from data augmentation [33, 34] to anomaly detection [35]. However, the use of GAN was also coupled with many challenges and numerous training problems:

- Mode dropping happens when some modes are discarded by GAN and dot not appear in p_g. This results in a discrepancy between p_g and p_r.
- Mode collapse occurs when the learned distribution p_g is an average over the different modes of p_r. p_g is hence a collapsed version of all modes in p_r. This also results in a discrepancy between p_g and p_r.
- Overfitting occurs when parts or samples of the training set are ignored by GAN. We ideally expect from p_g to be a uniform distribution over the training samples and to highlight the characteristics and the properties of the training set.
- Vanishing gradients occurs when the Discriminator is very efficient. In this case, the Generator will not be able to follow and learn from its feedback.

2.1.2 GAN Variants

To deal with the previously described training problems and improve the performance of GAN, researchers proposed more sophisticated GAN architectures [36]. cGAN [10] proposes to feed the input of Generator and Discriminator with auxiliary information c. By way of example, starting from random noise z and a specific label c, an image is generated by the Generator. The synthetic images $G(z \mid c)$ are compared to the real images by the Discriminator. In this case, the Discriminator learns to distinguish between $D(x)$ and $D(G(z \mid c))$. In addition to that, other researchers investigated the possibility of improving the efficiency of GAN by revisiting the architecture of the Generator and Discriminator. On the one hand, many works focused on ameliorating the Discriminator performance by using multiresolution discrimination [37, 38] or multiple Discriminators [39, 40, 41]. On the other hand, numerous works proposed new Generator architectures [3, 4] and focused on the form of the input latent space by applying Gaussian Mixture Models [42] or Clustering [43]. Other GAN architectures have later been presented such as DCGAN [44] consisting of a CNN Generator and a Deconvolutional CNN Discriminator or InfoGAN [45] a special form of cGAN that tries to maximize the mutual information between conditional information and the generated data.

Besides the work on the GAN's architecture, many authors focused on improving the performance of GAN by finding the most appropriate loss function. WGAN [46], suggested to use the Wasserstein loss instead of the minimax loss presented in Eq. 2.1 to avoid mode collapse and vanishing gradient problems. In this case, the Wasserstein distance is minimized between the generated and real distributions p_g and p_r:

$$W\left(p_g, p_r\right) = inf_{\gamma \in \Pi(p_g, p_r)} \mathbb{E}_{(x,y)\sim\gamma}\left[\| x - y \|\right], \tag{2.7}$$

where x and y are two data instances, $\Pi\left(p_g, p_r\right)$ is the set of all possible joint distributions between p_g and p_r and $\gamma\left(x, y\right)$ corresponds to one joint distribution.

Later, an extended version of the Wasserstein loss was presented in WGAN-GP [47] that imposes a gradient penalty term on the Discriminator:

$$\mathcal{L}_{WGAN-GP} = \mathbb{E}_{x_g\sim p_g}\left[D\left(x_g\right)\right] - \mathbb{E}_{x\sim p_r}\left[D\left(x\right)\right] + \lambda \mathbb{E}_{\widehat{x}\sim p_{\widehat{x}}}\left[\left(\| \nabla_{\widehat{x}} D\left(\widehat{x}\right) \|_2 - 1\right)^2\right], \tag{2.8}$$

where $p_{\widehat{x}}$ is sampled uniformly between points in p_r and p_g.

Moreover, different forms of regularization have been used to allow for a better convergence [48, 49]. To sum up, a huge effort has been spent in stabilizing the GAN's training and improving its performance and led to an impressive improvement in the quality of the synthesized images by generating images of high-resolution [2, 50].

2.1.3 GAN Architectures for Time Series Data

Let $\mathcal{X}_{ts} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \cup \mathcal{X}_N$ be a set of labeled time series $x_{t=1:T}$ from $N \in \mathbb{N}$ different classes following a distribution p_{rt} where $T \in \mathbb{N}$ is the length of the time series. The subset $S_r = \left\{x^r_{1,1:T}, x^r_{2,1:T}, \ldots, x^r_{k,1:T}\right\}$ is uniformly sampled from p_{rt}. Based on a set of training samples $S^{tr}_r = \left\{x^{tr}_{1,1:T}, x^{tr}_{2,1:T}, \ldots, x^{tr}_{k',1:T}\right\}$ and a validation set $S^{val}_r =$

$\left\{x^{val}_{1,1:T}, x^{val}_{2,1:T}, \ldots, x^{val}_{k'',1:T}\right\}$, GAN aims to learn a distribution p_{gt} that approximates p_{rt} where $k, k', k'' \in \mathbb{N}$. $S_g = \left\{x^{g}_{1,1:T}, x^{g}_{2,1:T}, \ldots, x^{g}_{m,1:T}\right\}$ denotes the set of generated samples drawn from p_{gt} with $m \in \mathbb{N}$.

Driven by the success of GAN on image data, researchers tried to investigate this method for other types of data such as time series. In this context, numerous generative models have been proposed:

C-RNN-GAN As a first attempt, Mogren proposed C-RNN-GAN [9], depicted in Fig. 2.2 (a), equipped with a recurrent Generator and Discriminator. C-RNN-GAN relies on common techniques proposed in [5] to improve the GAN performance. The recurrent Generator takes as input in each cell a random vector and the output of the previous cell. A bidirectional recurrent Discriminator is used to enable to learn the context and the dynamic in both directions. In the original paper, LSTM is used as a recurrent structure. The proposed framework was tested on classical music.

RCGAN Esteban et al. [11] presented RCGAN (Fig. 2.2 (b)) a conditional recurrent GAN. In this case, the input of the Generator and Discriminator are conditioned on auxiliary information c with the aim of improving its performance. This is achieved by concatenating the input of both neural networks at each timestep with c. This ensures that the additional information c cannot be discarded or forgotten during the generation procedure. The authors used LSTM as a recurrent structure for the Generator and Discriminator.

TimeGAN A more sophisticated architecture was presented by Yoon et al. [20] in TimeGAN (Fig. 2.2 (c)) consisting of an embedding and recovery function and a recurrent Generator and Discriminator. The main goal behind TimeGAN was to design a good generative model that preserves the temporal dynamic for mixed datasets with temporal and static data. In addition to the Generator and Discriminator, TimeGAN uses an embedding function $e : \mathcal{S} \times \mathcal{X}_{ts} \rightarrow \mathcal{H}_{\mathcal{S}} \times \mathcal{H}_{\mathcal{X}}$ that maps the vector space of statistic features $\mathcal{S}$ and temporal features $\mathcal{X}_{ts}$ to the latent spaces $\mathcal{H}_{\mathcal{S}}$ and $\mathcal{H}_{\mathcal{X}}$ respectively. Particularly, an embedding network $e_{\mathcal{S}}$ is used to encode the static features and $e_{\mathcal{X}}$ is used to encode the temporal features. In addition to that, a static $r_{\mathcal{S}} : \mathcal{H}_{\mathcal{S}} \times \mathcal{S}$ and temporal $r_{\mathcal{X}} : \mathcal{H}_{\mathcal{X}} \times \mathcal{X}_{ts}$ recovery functions are used to retrieve the original data and to enable a generation of new mixed data.

Through these numerous works, attempts were made to obtain generative models for time series that correctly reproduce the underlying temporal characteristics of a given training dataset. However, we prove in this work that the performance of these models is limited on datasets with high-variability for example containing different classes. In such setups, it is extremely difficult for a generative model to find the right trade-off between sample fidelity, i.e., their similarity to the real time series, and sample diversity. Furthermore, it is essential to preserve the original classes and the variation within each class.

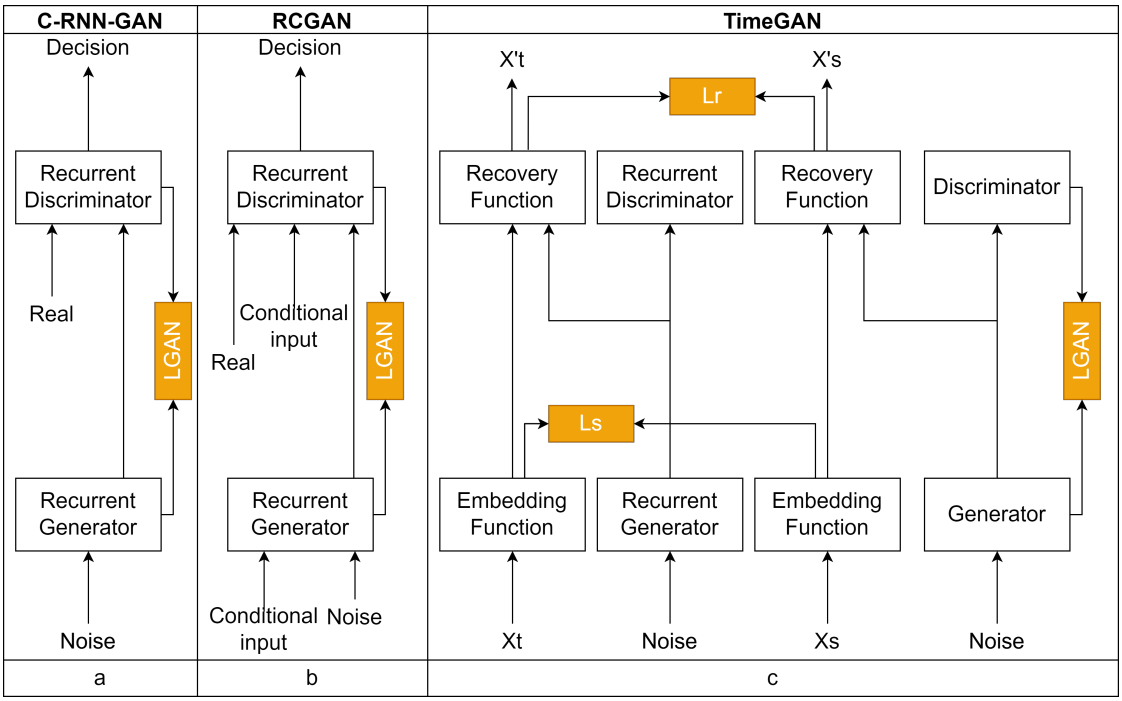

Figure 2.2: Comparison of the architecture of the different generative models for time series data namely TimeGAN, RCGAN and C-RNN-GAN. The used loss functions are depicted in dark blue. Xs, $X^{'}s$ and Xt, $X^{'}t$ denote the static and temporal data respectively.

2.1.4 GAN Metrics for Time Series

For two sets S_r and S'_r randomly sampled from p_{rt} so that $S_r \cap S'_r = \varnothing$ and a set S_g with $|S_r| = |S'_r| = k$ and $|S_g| = m$, a metric $\widehat{\rho} : S_r \times S_g \to \mathbb{R}$ meant to evaluate GAN must fulfill different criteria:

- **Discriminability** An accurate evaluation metric should be able to distinguish between real and synthetic samples in order to compare p_g and p_r. We ideally expect that the distribution of the generated samples p_g reflects the characteristics of the distribution of the real data p_r. If this is not the case, the evaluation metric should be able to spot the problem by discriminating between the real and generated samples. As stated in part 2.1.1, two common problems may occur while training a GAN, that cause a discrepancy between the distributions p_r and p_g namely mode dropping and mode collapse. A reliable metric ρ should be able to detect these phenomena.

- **Detect Overfitting** As stated in [23], overfitting in GAN occurs when parts of the training set are ignored during the generation process. In other words, p_g should ideally be a uniform distribution over the training samples and S_g should represent the properties and the diversity of the training set. A reliable metric should be able to detect this phenomenon.

- **Efficiency** An important property that must be considered when evaluating a metric is its efficiency, i.e., an accurate metric ρ should be efficient in terms of computation and the number of samples. The sample efficiency denotes the ability to compute accurate results with a reasonable number of samples. Moreover, computational efficiency is an important criterion that must be considered when designing new metrics. In practice, the evaluation metric will be used several times during the training or test phase in order to evaluate the behavior of the GAN model.

In the last few years, numerous metrics have been presented to evaluate GAN for time series data. Esteban et al. proposed in their work [11] two evaluation metrics namely TSTR and TRTS. Later, two additional metrics are introduced by Yoon, Jarrett, and van der Schaar [20] namely DS and PS:

- MMD [21] has been widely used to evaluate GAN for different types of data. It compares the distribution of the real data to the distribution of a set of generated samples by computing the squared difference between two sets of samples.

- TRTS [20] denotes training a ML classifier on the real data and reporting the test accuracy when the model is tested on the synthetic ones.

- TSTR [20] denotes training a ML classifier on the synthetic data and reporting the test accuracy when the model is tested on the real ones.

- DS [20] A 2-layer LSTM is trained to discriminate between the real and synthetic samples. During the training process, each real time series is labeled as real and each generated time series is labeled as fake. DS denotes the classification error of a test set consisting of a mix of real and generated samples. This score measures the similarity between both datasets and hence checks whether the generated samples are indistinguishable from the real data, i.e., fulfill the fidelity criterion.

- PS [20] In order to assess the usefulness of the generated time series, a 2-layer LSTM is trained to predict the next coming value for each sequence of the generated time series. The trained model is later tested on the real time series. PS represents the mean absolute error between the predicted and the real values.

- Visual Evaluation Yoon et al. [20] applied t-SNE [51] and PCA [52] on the real and generated time series to enable an efficient comparison of the two distributions in a 2-dimensional space. These visualizations aim to compare the diversity of the real and synthetic samples.

The different methods focus on three important criteria when assessing the quality of the generated data namely *diversity*, *fidelity*, and *usefulness*. *Diversity* means that the generated data should reflect the variation present in the real dataset. While *fidelity* requires that the generated data must be indistinguishable from the real ones, *usefulness* assess the applicability and the utility of the generated data, i.e., the synthetic samples should be as useful as the real data when performing prediction tasks.

It is to be noted that these were not compared to each other. That's why it is difficult to determine the benefits and disadvantages of each approach in detecting the typical GAN training problems described in section 2.1.1.

To evaluate the robustness of GAN metrics to the criteria presented, Xu et al. [23] proposed different tests that can be performed and may reveal many characteristics about a considered GAN metric.

Mixing test The mixing test [23] is a good technique to test the discriminability of each evaluation metric. It consists in computing the score $\rho(S_r, S_g(l))$ between S_r and a mix of real and synthetic samples $S_g(l)$ so that l is the percentage of generated data. At the beginning of the test, the score $\rho(S_r, S_g(l))$ is measured on a set of real data. After that, we progressively augment the ratio l of generated data in $S_g(l)$. The final score is computed on a set of generated data. We ideally expect that the score of ρ increases as the ratio of fake data l varies from 0 to 1 and that the best score is achieved on the real data.

Mode collapse and mode dropping tests As stated in [23], mode collapse and mode dropping can be artificially simulated by finding c clusters in the whole training set S_r^{tr} with k-means and finally, either substituting each cluster present in S_r' by its center to simulate mode collapse or randomly removing it to reproduce mode dropping. The score $\rho(S_r, S_r')$ is then calculated depending on the number of collapsed/ dropped clusters in S_r'. An accurate metric should be sensitive to mode collapse and mode dropping and hence its value should increase with the number of collapsed/ dropped clusters.

Sample efficiency test The sample efficiency test [23] compares the score $\rho(S_r, S_g)$ to the score $\rho(S_r, S'_r)$ for different number of samples n. The sample efficiency denotes the difference between $\rho(S_r, S_g)$ and $\rho(S_r, S'_r)$ as the number of samples n increases. We ideally expect that $\rho(S_r, S_g) - \rho(S_r, S'_r) > 0$ for a reasonable number of samples n. As an additional criterion, we investigate the value of $\rho(S_r, S'_r)$ for different number of samples n and assume that this value will show a stable behavior with a specific number of samples n and converge. This condition should highlight the effect of the number of samples n on the score ρ.

Overfitting test A possible method to evaluate the robustness of a metric ρ in detecting overfitting is presented in [23]. Let S''_r be a mix of samples from S^{tr}_r and another set disjoint from the training and the validation sets. The value $\rho(S''_r, S^{val}_r) - \rho(S''_r, S^{tr}_r)$ called the generalization gap denotes the difference between the score computed to a validation set and the score computed to the training set. We progressively increase the number of training samples of S^{tr}_r present in S''_r. This should reduce the gap between both values to show that S''_r is memorizing more samples from S^{tr}_r.

2.2 Data Translation

2.2.1 Image-to-Image Translation

Several works have been presented to perform image-to-image-translation with the purpose of learning the mapping function between two different visual domains. Isola et al. [53] used in their method, Pix2pix, conditional GAN to translate images to a target domain. In this case, paired-data are required during the training process. This technique was later extended by Wang et al. in CycleGAN [13] which relies on a cycle consistency loss to deal with unpaired images. Liu et al. presented UNIT [14] which are based on coupled GAN that map to a common latent space used as a cycle-consistency constraint. Huang et al. introduced a multimodal version of UNIT, MUNIT [54], that enables a diverse output generation for a source image by decomposing the latent space into a domain-invariant content space and a style space. In contrast to MUNIT [54], BicycleGAN [12] requires paired data during the training process. This technique consists of a Conditional Variational AutoEncoder GAN (cVAE-GAN) and a Conditional Latent Regressor GAN (cLR-GAN) to guarantee a one-to-one mapping between each input image and the corresponding output. DiscoGAN [55] learns to distinguish between the domain and the style of the image by observing the relationships between various visual fields. To enable a multimodal image-to-image translation, DRIT [27, 15] tries to capture the relationship between two different visual domains by dividing each image into content and attribute. In fact, a disentangled representation is applied to the latent space representation. The content consists of the common information between both visual domains whereas the attribute space preserves the domain-specific information. DRIT [15] involves two content encoders, two attribute encoders, two Generators, two domain Discriminators, and a content Discriminator. An input image from domain $\mathcal{A}$ is projected by the attribute encoder and by the content encoder into the domain-specific content space. The content and attribute vectors determined by the encoders are later

used by the Generator to translate images in the domain $\mathcal{A}$. A domain Discriminator learns to distinguish between the real images from domain $\mathcal{A}$ and the images generated by the Generator. Finally, a content Discriminator is involved to differentiate between the learned content representations of both domains. We refer to [15] for a detailed description of the algorithm.

2.2.2 Time Series Translation

In contrast to image-to-image translation, the problem of translating time series was not in-depth investigated. To the best of our knowledge, a unique algorithm was proposed CycleGAN-VC [18] that achieves voice conversion. CycleGAN-VC consists of a modified version of CycleGAN that combines gated CNN with an identity-mapping loss to focus on non-parallel voice conversion. CycleGAN learns a mapping function f between two visual domains $\mathcal{A}$, and $\mathcal{B}$ by means of an adversarial loss and a cycle-consistency loss. While the adversarial loss compares the data obtained after the conversion with the target data, the cycle-consistency loss ensures a rigorous reconstruction of the original data starting from the converted data. With unpaired data, CycleGAN-VC enables an efficient parallel-data-free voice conversion. It relies on gated CNN and an identity-mapping loss to preserve the composition between the input and output during the mapping process in the Generator.

2.3 Differential Privacy

2.3.1 Definition

The notion of DP was rigorously defined by Dwork et al. in their works [56, 57, 58] and became a well-established method for protecting the privacy of individuals in a dataset. It protects the privacy of a dataset by minimizing the influence and the effect of each dataset's instance. Intuitively, the outcome of an algorithm should be insensitive to a small perturbation in the dataset and hence more generalizable. In other words, the result should be independent of the presence or the absence of any individual record. Let's consider two neighboring datasets D and D' and an algorithm $\mathcal{M}$. Two datasets are assumed to be neighboring if they are identical differing in one single instance.

Definition 1 (DP [56, 57, 58]) *According to [57], a randomized algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private if for any subset of outputs S:*

$$P\left[\mathcal{M}(D) \in S\right] \leq e^{\epsilon} P\left[\mathcal{M}(D') \in S\right] + \delta, \tag{2.9}$$

where ϵ and δ control the privacy, P is the randomness of the noise in the algorithm, and $\mathcal{M}(D)$ and $\mathcal{M}(D')$ are the output of the algorithm $\mathcal{M}$ given the dataset D and D'.

It is to be noted that lower ϵ and δ values lead to stricter privacy guarantees.

The post-processing theorem guarantees that any randomized mapping on an (ϵ, δ)-differentially private algorithm has no effect on its privacy:

Theorem 1 (Post-processing [59]) *Given a randomized algorithm $M : D \to R$ that is (ϵ, δ)-differentially private and an arbitrary randomized mapping $f : R \to R'$, $f \circ M : D \to R'$ is (ϵ, δ)-differentially private.*

To achieve DP, the Gaussian mechanism [60] is a commonly used method. It consists in adding Gaussian noise to the output of an algorithm. To estimate the privacy loss for repeated use of differentially private mechanisms, the notion of privacy accountant has been proposed [61]. In this context, Moment Accountant (MA) has been extensively used as it provides a more accurate estimation of privacy loss than the previously proposed methods [57]. Recently, Mironov et al. [60] proved that RDP accountant outperforms MA in estimating the privacy loss for a composite mechanism.

2.3.2 Differentially Private Machine Learning

Recently, a lot of works focused on developing privacy-preserving ML methods. As a first attempt, DPSGD [61] was proposed by Abadi et al. to guarantee privacy for DL models. Later, numerous differentially private GAN approaches have been proposed. DPGAN [62] and dp-GAN [63] achieve DP by means of the differentially stochastic gradient. While DPGAN employs WGAN [46] as an objective function and clips the model weights, dp-GAN opts for the aka improved WGAN [47] and an explicit clipping of the gradients. In contrast to these methods, PATE-GAN [64] enables the generation of data in a privacy-preserving manner by applying the Private Aggregation of Teacher Ensembles (PATE) to GAN. The authors made a modification to the originally proposed PATE method by training the *teacher* Discriminators on disjoint sets of real data. At the same time, the *student* Discriminator is trained using the generated data that are labeled by the teachers, and the Generator loss is computed based on the *student* Discriminator loss. DP-CGAN [65] addresses another problem namely generating labeled private synthetic data. This is achieved by applying DPSGD to cGAN [10]. The proposed method relies also on clipping the Discriminator loss of the real and generated data separately and summing them up together and using the RDP accountant instead of the MA accountant. Finally, DP-FedAvg-GAN [66] applies DP for generative models trained with federated learning and proves that these models are efficient in debugging many image issues. An extensive overview of the existing differentially private GAN approaches was presented by Fan [67]. These models were designed to generate images in a privacy-preserving manner. To the best of our knowledge, a unique differentially private generative model for time series was proposed by Esteban et al. while introducing RCGAN [11].

CHAPTER

3

TIME SERIES GENERATION

In spite of the great success of GAN with images [2, 3, 4], their utility and use on time series data are still limited. This is mainly due to the fact that designing a good generation framework for time series is challenging. One key challenge is the temporal relationship between the data points that must be preserved in the generation process. Furthermore, as for other types of data, an efficient generative model has to find the right trade-off between sample fidelity, i.e., the similarity of the generated data to the real ones, and sample diversity, i.e., reproducing the variation of the real data. This is particularly challenging for datasets with high-variability such as datasets with multiple classes, where one or many classes are misrepresented, i.e., in imbalanced setups or when the difference between the classes is not obvious and outstanding. A possible solution to this problem would be to train a specific GAN for each class. However, this reduces the potential to learn characteristics common to all classes from the full training dataset. For such datasets, it is fundamental to produce samples of high-quality that reflect the inter- and intra-class variation. This means that all the classes of the real dataset must be represented in the synthetic samples and that the diversity within each class should be preserved.

Through numerous works [9, 11, 20], attempts were made to obtain generative models for time series that correctly reproduce the underlying temporal characteristics of a given training dataset. However, we prove in this work that the performance of these models is limited on the datasets with high-variability such as datasets stemming from different classes. To tackle this issue, we present in this section ClaRe-GAN [28] a novel generative model for multi-class time series datasets that outperforms the existing ones. The sophisticated architecture of ClaRe-GAN allows for better learning of the inter- and intra-class variations, i.e., the class-specific properties and the properties that are

common to the different classes. It consists of a recurrent GAN, a class-specific encoder for each class, and a class Discriminator allowing for efficient and reliable extraction of the class properties as well as the general properties of the dataset. This ensures a generation of new diverse time series of high-quality.

The performance of our model is evaluated against different criteria namely *diversity*, *fidelity*, and *usefulness* and compared to the state-of-the-art models. In order to enable a fair comparison, we use the same evaluation metrics as the ones used by Yoon et al. while introducing TimeGAN [20], i.e., DS and PS. While the DS assesses the *fidelity* of the generated time series through a classifier meant to distinguish between the real and the generated datasets, the PS evaluates the *usefulness* of the generated data on prediction tasks. Furthermore, we visualize the real and the synthetic time series in full and reduced dimensions using PCA [52] and t-SNE [51] analysis such as in [20] to allow direct comparison and to assess the *diversity* of the generated time series.

Finally, we cope in this section with the problem of preserving the privacy of the dataset's instances during the generation process. As a matter of fact, privacy issues prevent data owners from improving AI performance as it makes external collaborations binding. An alternative solution would be to generate new private data that can be shared without restriction and that don't contain some sensitive information of the original dataset but still have its same utility. To allow data sharing without confidentiality concerns, we combine the existing GAN models for time series namely TimeGAN [20], ClaRe-GAN [28], and C-RNN-GAN [9] with DP.

We test our approaches on a collection of datasets from the UEA & UCR Time Series Classification Repository [68] which vary in time series' length and number of classes. In this context, we compare the different generation techniques with- and without privacy guarantees and evaluate their performances against each other.

In the conducted experiments, we show that ClaRe-GAN is scalable with the length of the time series and the number of classes and conclude that our framework produces time series of high-quality. It outperforms the existing state-of-the-art methods visually and computationally enabling significant progress in the research area of generative models for time series. Moreover, we prove that DP-ClaRe-GAN, the DP version of ClaRE-GAN, finds the best trade-off between the privacy of the generated data and their utility.

3.1 ClaRe-GAN: new Algorithm for Time Series Generation

In cGAN [10], labeled setups were used to condition the input of the generator leading to a significant improvement in the quality of generated data. In a similar fashion, we exploit the supervised setup to find a good generative model for time series stemming from different classes. Our main goal is to learn the inter- and intra-class variations of the dataset and to exploit this information to improve the quality and the diversity of the generated time series. ClaRe-GAN, illustrated in Fig. 3.1, is composed of class-specific encoders, one encoder per class, a *class Discriminator*, and recurrent Generators and Discriminators. This structure enables efficient extraction of the class-dependent and

class-independent attributes. The class-specific encoders are trained in a parallel manner using the real time series belonging to the different classes and the *class Discriminator*. Our model can therefore be used for any number of classes.

For each class $n \in \{1, \ldots, N\}$, we use a GAN consisting of a Generator G_n and a Discriminator D_n. The Discriminator D_n is a binary classifier that tries to distinguish between the real time series and the ones synthesized by the Generator G_n. The minimax game between both components is used, for each class n, as follows:

$$\mathcal{L}_{GAN}\left(G_n, D_n\right) = \mathbb{E}_{x \sim p_r(x)}\left[log\left(D_n\left(x\right)\right)\right] + \mathbb{E}_{z \sim p_z(z)}\left[log\left(1 - D_n\left(G_n\left(z\right)\right)\right)\right], \tag{3.1}$$

where z is a noise vector sampled from a distribution p_z and $x \in \mathcal{X}_n$.

A class-specific encoder $E_n : \mathcal{X}_n \rightarrow \mathcal{C}$ learns a representation for each time series $x \in \mathcal{X}_n$ belonging to the same class n by mapping them into a vector of predetermined length. This class-specific encoder is used to gather class-specific and class-independent attributes by learning the factor of variation of each real time series:

$$c_x^n = E_n\left(x\right). \tag{3.2}$$

We assume that all time series are mapped to the same latent space $\mathcal{C}$. This is achieved by sharing the weights [14] of the last 2 layers of the encoders. This assumption guarantees a common extraction of the high-level representation of the time series, i.e., we extract the class-independent properties. At the same time, the class-specific encoders are trained adversarially by means of an additional *class-specific Discriminator*. It discriminates between the representation of the different time series to enable efficient extraction of the class-specific features. The extracted latent codes are later concatenated with an input noise vector in a sophisticated manner to generate time series of high-quality. Furthermore, we make no restriction on the architecture of these encoders in our case we used CNNs, but architectures such as fully connected layers or RNN can replace those as well.

Inspired by [27, 15], we impose a class Discriminator D_X^{Cl} to discriminate between the learned class representations of the encoders E_n allowing a more precise extraction of the different class features. The *class Discriminator* is trained by applying a class adversarial loss to improve the quality of the variation learned by the encoders. For a dataset with 2 classes, the class adversarial loss can be expressed as follows:

$$\begin{aligned}\mathcal{L}_{adv}\left(E_1, E_2, D_{\mathcal{X}}^{Cl}\right) = \mathbb{E}_{x_1}[0,5 \cdot \log\left(D_{\mathcal{X}}^{Cl}\left(E_1\left(x_1\right)\right)\right) + \\ 0,5 \cdot \log\left(1 - D_{\mathcal{X}}^{Cl}\left(E_1\left(x_1\right)\right)\right)] + \\ \mathbb{E}_{x_2}[0,5 \cdot \log\left(D_{\mathcal{X}}^{Cl}\left(E_2\left(x_2\right)\right)\right) + \\ 0,5 \cdot \log\left(1 - D_{\mathcal{X}}^{Cl}\left(E_2\left(x_2\right)\right)\right)].\end{aligned} \tag{3.3}$$

As in [3, 4], our generator $G_n : \{\mathcal{C}, \mathcal{Z}\} \rightarrow \mathcal{X}_n$ is equipped with a mapping function $f : \mathbb{R}^m \rightarrow \mathcal{W}$ consisting of 3 fully connected layers. However, our mapping function f is

used on the noise vector z instead of the latent vector c_x^n, i.e., $f(z) = w$ where the noise vector z is sampled randomly from a pre-defined distribution. In our case, we used the Gaussian distribution. The obtained vector w is later concatenated with the latent code c_x^n and fed to a RNN. Like all other generative models designed for time series, we opt for the usage of RNN due to their well-known ability in modeling sequential data. We used in our case LSTM but we make no restriction on the recurrent architecture.

Our Discriminator $D_n : \mathcal{X}_n \to [0, 1]$ minimizes the adapted Discriminator loss function that was originally proposed by Goodfellow when introducing GAN, i.e., Eq. 3.1. We use multi-scale Discriminators [69, 70] originally designed for images. In this case, many Discriminators are used and trained with different image resolutions. In our experiment, we found that this multi-scaling technique, i.e., feeding the same input time series under different levels of compression to a multitude of Discriminators eases the training process and improves the quality of the synthesized time series. In addition to the class adversarial loss, we apply further loss functions that facilitate the training and improve the quality of the generated time series. To improve the diversity of the generated data and

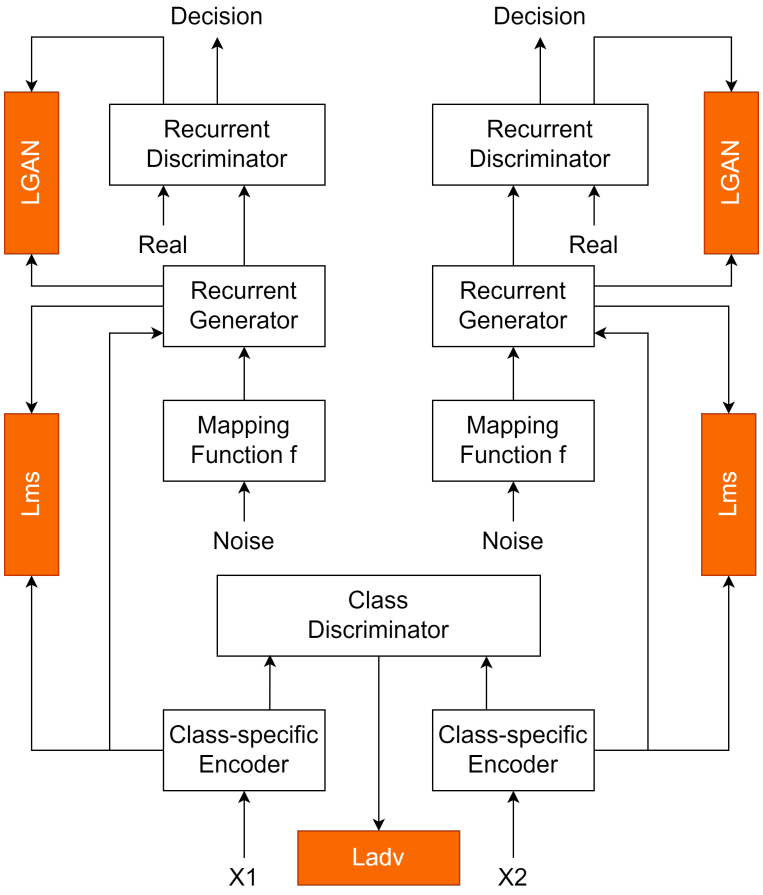

Figure 3.1: Representation of the ClaRe-GAN architecture for a dataset with 2 classes, i.e., $N = 2$. $X1$ and $X2$ are time series from two different classes.

to prevent mode collapse, we apply the mode seeking regularization term [71] that helps to capture the different modes present in the real dataset by maximizing the ratio of the distance between two generated time series $G_n(x, z_1)$ and $G_n(x, z_2)$ given an input time series x, and two latent noise vectors z_1 and z_2,

$$\mathcal{L}_{ms} = \max_{G_n} \left(\frac{d_x\left(G_n(z_1, x), G_n(z_2, x)\right)}{d_z(z_1, z_2)} \right), \tag{3.4}$$

where d_* denotes the mean absolute error.

The full objective function of our framework can then be written as:

$$\lambda_{GAN}\mathcal{L}_{GAN}(G_n, D_n) + \lambda^c_{adv}\mathcal{L}_{adv}\left(E_1, E_2, D^{Cl}_{\mathcal{X}}\right) + \lambda_{ms}\mathcal{L}_{ms} + \lambda_c \parallel c^n_x \parallel^2, \tag{3.5}$$

where $\parallel c^n_x \parallel^2$ is a $L2$ regularization term applied to prevent overfitting and λ_{GAN}, λ^c_{adv} and λ_{ms} are the model parameters. The used loss functions are summarized in Fig. 3.1. In our experiments, we used $\lambda_{GAN} = 1$, $\lambda^c_{adv} = 1$ $\lambda_c = 0.01$ and $\lambda_{ms} = 1e-5$.

3.2 DP*: Privacy-preserving Approaches

The AI evolution is currently compromised by important constraints such as the lack of data and the growing privacy matters. As a matter of fact, external collaborations on improving and expanding the use of AI are highly restricted, and sharing data remains a sensitive subject. In many medical or industrial domains, the lack of data and privacy concerns prevent researchers from improving the efficiency of AI. In these cases, publishing synthetic privacy-preserving data that depict the behavior of the original dataset, could enlarge the scope of AI's applicability. Thereby, it will also preserve their privacy.

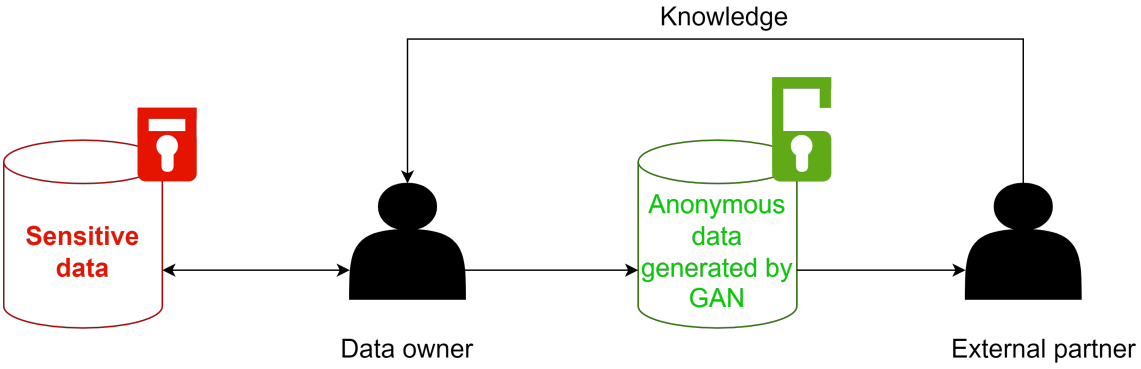

Figure 3.2: Use Case Scenario: Data owners holding sensitive data can use a differentially private version of GAN to generate new anonymous time series. These data can be shared with external partners without confidentiality concerns and both parts can work together safely.

Let's consider a scenario, illustrated in Fig. 3.2, where data owners want to improve the performance of AI in a specific use case by collaborating with some external partners.

For example, they want to find a better performing ML model for some medical data or a model that correctly predicts the state of a machine. In these cases, it will be enough to give the external partner some synthetic data with the same reactivity to any ML model. Thus, it will not reveal rare diseases that can be easily detected in the original dataset or some sensitive information about the machine parameters or its properties e.g., times when the machines were on/off... To this end, the data owner can use a privacy-preserving version of GAN to generate new anonymous data and can share it with the external partner who will not have access to the original ones.

One can assume that the samples generated by GAN differ from the original ones and don't contain their sensitive information. However, there is no guarantee that the Generator by repeatedly sampling from p_{gt} will not reproduce the training dataset or generate time series containing sensitive information. We will extend the existing generative models for time series to eradicate privacy concerns. Concretely, we will extend the GAN algorithms for times series namely -TimeGAN [20], ClaRe-GAN [28], and C-RNN-GAN [9]- with a DP component to pull out the strengtheners and weaknesses of each method. Our approach relies on changing their Discriminators with a private Discriminator that uses DPSGD [61] and on tracking the spent privacy loss using the RDP accounting technique. We evaluate the results visually and computationally and assess the usefulness and the privacy of the data generated by the differentially private models.

We modify the architecture of the previously described generative models, in the following called DP-TimeGAN, DP-C-RNN-GAN and DP-ClaRe-GAN, to generate time series in a privacy-preserving manner. This is achieved by changing their original Discriminators with a private Discriminator equipped with a DPSGD [61]. Two techniques are used to achieve privacy namely clipping gradient and adding random noise. During the training procedure, the per-example gradients of the Discriminator loss are computed for the real and generated data. Afterward, both values are clipped to the minimum value between their L2-norm and a clipping value C. The clipped gradients are summed up and Gaussian noise $N(0, \sigma^2 C^2)$ is added where σ is noise multiplier. Based on the post-processing theorem [59], we guarantee that the use of the private Discriminator in any generative model makes the Generator and all other architecture's components (encoders, etc.) private. The spent privacy loss is computed using the RDP accounting technique [72] as it enables a tighter privacy estimation than the moment accountant technique and easy computation of the privacy budget curve for a composite mechanism.

3.3 Experiments

3.3.1 Datasets Description

We evaluate the performance of our approaches on a collection of publicly available datasets from the UEA & UCR Time Series Classification Repository [68, 73] with time series of different properties namely ItalyPowerDemand, TwoLeadECG [74], Yoga, DistalPhalanxTW [75] and FreezerRegularTrain. The datasets used in our experiments vary in the length of the real time series ranging from 24 to 425, the number of classes present in the real dataset ranging from 2 to 6, the number of time series per dataset, the number

of time series per class: balanced and imbalanced datasets, and the characteristics of the times series such as the class properties or the level of noise. Table 3.1 summarizes the characteristics of the used datasets.

Table 3.1: Summary of the characteristics of the used datasets. The datasets are publicly available in the UEA & UCR Time Series Classification Repository [68] and differ in the length of the time series the number of classes and the ratio of data per class.

Dataset	***Length***	***Number of classes***	***Ratio of data available per class***	***Size***
ItalyPowerDemand	24	2	50%-50%	1096
TwoLeadECG	82	2	50%-50%	1192
FreezerRegularTrain	301	2	50%-50%	2878
Yoga	425	2	50%-50%	3300
DistalPhalanxTW	80	6	34,18%-34,18% -3,48%-5,76% -16,08%- 6,32 %	539

3.3.2 Experimental Setup

The experiments for the ClaRe-GAN algorithm are conducted on t2.large AWS EC2 instances with 8 GiB of system memory and 2 vCPUs. We compare our method to the state-of-the-art generative models for time series data namely TimeGAN [20], RCGAN [11], and C-RNN-GAN [9]. For an equitable comparison between the algorithms, we use the same type and number of layers for the recurrent Generators and Discriminators: 2-layer LSTM with 100 hidden units. For ClaRe-GAN, after a rigorous investigation and hyper-parameter tuning, we used 3 scales for the Discriminator, $\lambda_{GAN} = 1$, $\lambda_{adv}^{c} = 1$, $\lambda_c = 0.01$ and $\lambda_{ms} = 1e^{-5}$.

To enable a fair comparison between the different frameworks, we use as part of our evaluation the evaluation methods proposed in [20] when introducing TimeGAN. It is to be noted that TimeGAN was also compared to the previous existing frameworks namely RCGAN and C-RNN-GAN. We perform a computational and visual evaluation by computing the PS and DS and by visualizing the synthetic and real samples:

- Visual Evaluation: We use the evaluation techniques presented by Yoon et al. in [20] and described in section 2.1.4 to compare the distribution of the real and generated samples. However, we will show in the following that the evaluation of time series similarity in a reduced dimension space alone is not sufficient to ensure high-quality time series. To convince the reader about the generated time series, we opt for an additional visualization method where we plot all the real time series and the generated time series side-by-side to enable a direct comparison.

- Computational Evaluation: The computational evaluation is achieved by computing the DS and PS [20] described in section 2.1.4.

Later, we evaluate the performance of the designed differentially private frameworks visually and computationally. We test the developed differentially private GAN on all the previously described datasets. Their performances are compared to the existing differentially private GAN model for time series data of RCGAN. To guarantee a fair comparison between all the frameworks, we use the same architecture for their Generators and Discriminators a 2-layers LSTM with 100 hidden units and the same number of iterations, i.e., 100. In contrast to the differentially private RCGAN and DP-ClaRe-GAN, the labels are not generated with the data for DP-TimeGAN and DP-C-RNN-GAN. We label the data generated by these frameworks manually by finding the nearest real time series. In all the experiments, we use $C = 0.3$, $\sigma = 0.3$ and $\delta = 10^{-3}$. For each dataset, we compute the spent privacy ϵ and assess the utility of the generated time series. As performed while introducing differentially private RCGAN, the unique differentially private GAN model for time series, we perform the computational evaluation and assess the utility of the datasets by computing the test accuracy of TSTR and TRTS [11]. In both cases, the ML model is used to classify the time series of the datasets. For each framework, we take the iterations with the best performance (best TSTR and TRTS values). This is achieved by computing the TSTR and TRTS accuracy values for 100 time series generated by each framework. As ML model we use Random Forest [76]. In this set of generated time series all the classes of the original dataset are represented with the same number of time series. Our main goal is to find the best performing method, i.e., the method that finds the right balance between privacy and utility of the generated time series.

3.3.3 Results

We compare ClaRe-GAN to the state-of-the-art generative models for time series. The results of the PCA and t-SNE analysis are illustrated in Fig. 3.3 and Fig. 3.4 respectively. We clearly see that C-RNN-GAN shows a limited performance in terms of sample diversity. A better performance was noticed for RCGAN with the ItalyPowerDemand, TwoLeadECG, and Yoga datasets. However, the distribution of the samples generated by RCGAN differs from the distribution of the real samples. Furthermore, the samples generated by TimeGAN are not as diverse as the real dataset for the TwoLeadECG dataset and TimeGAN fails in capturing the distribution of the real samples for the other datasets. The PCA and t-SNE plots show that there is a significant improvement in the diversity of the samples generated by ClaRe-GAN in comparison with the other methods. We clearly see that the distribution of the samples generated by ClaRe-GAN is the closest to the distribution of the real samples and that the performance of ClaRe-GAN scales well with the number of classes and the ratio of data available per class (DistalPhalanxTW dataset) and the length of the time series (Yoga and FreezerRegularTrain datasets). ClaRe-GAN was able to capture the distribution of the real data independently from the dataset and its properties.

To allow a direct comparison of the time series, we visualize the real and generated time series by each model. The results for the DistalPhalanxTW, Yoga, TwoLeadECG, FreezerRegularTrain, and ItalyPowerDemand datasets are depicted in Fig. 3.5, Fig. 3.6, Fig. 3.7, Fig. A.1, Fig. A.2, Fig. A.3 and Fig. A.4. We clearly see that visualizing both datasets in a reduced 2-dimensional space is not enough. In fact, Fig. 3.7 shows that the time series generated by RCGAN are noisy and the main class properties are ignored during the generation process. Moreover, according to Fig. 3.3, TimeGAN is showing a good performance and the difference between the distribution of the generated and the real samples is not as important as for RCGAN. However, we see in Fig. 3.5 that the time series generated by TimeGAN don't reflect the properties of the real dataset and are smoother than the real ones. In contrast to that, ClaRe-GAN captures properly the properties of the real dataset and its class characteristics. While the state-of-the-art methods presented a poor or limited performance, our method was able to learn the inter- and intra-class variations of the original dataset and to reflect these properties in the generated dataset.

The fidelity and usefulness of the generated time series are assessed with DS and PS respectively. The obtained results are summarized in Tables 3.2 and 3.3. They show that the fidelity and usefulness of the time series synthesized by C-RNN-GAN are limited, except for the FreezerRegularTrain where the PS was equal to 0.106. It is to be noted that the time series generated for this dataset are not realistic at all. Better performance is noticed for RCGAN and TimeGAN. For all the datasets, except for FreezerRegularTrain, and independently from their characteristics, ClaRe-GAN achieves the lowest PS and DS which proves that the time series generated by this framework are of high-fidelity and are as useful as the real time series. For example, a great improvement was noticed in terms of DS (half of the best DS achieved by the other methods) for the Yoga dataset. Moreover, for the DistalPhalanxTW dataset, an imbalanced dataset with 6 classes, the ClaRe-GAN's PS is equal to 0,13. While achieving the best DS for the FreezerRegularTrain, the PS of ClaRe-GAN is still high compared to the other methods. In this case, it is important to say that the time series generated by the other methods are not realistic. It is to be noted that ClaRe-GAN is also the most efficient in terms of computation.

Last but not least, we evaluate the differentially private generative methods presented in section 3.2. Fig. 3.8 illustrates the test accuracies values of TRTS and TSTR for the different datasets and different frameworks. The figures show that the best privacy values are achieved by C-RNN-GAN. At the same time, its TSTR and TRTS accuracies are really low. For all the datasets, ClaRe-GAN presents better privacy and TRTS TSTR values than the existing DP algorithm of RCGAN. It is to be noted that ClaRe-GAN and RCGAN generate labeled data. Especially, we noticed a great improvement in terms of privacy for ItalyPowerDemand TwoLeadECG and DistalPhalanxTW datasets. TimeGAN is characterized by high TRTS and TSTR values for higher - but still reasonable - privacy values, by way of example 0.8 and 0.74 for TwoLeadECG dataset. Moreover, it achieves better privacy values for DistalPhalanxTW and ItalyPowerDemand. We have also noticed that the TSTR and TRTS values of RCGAN are around 0.5 which shows a limited utility of the generated data.

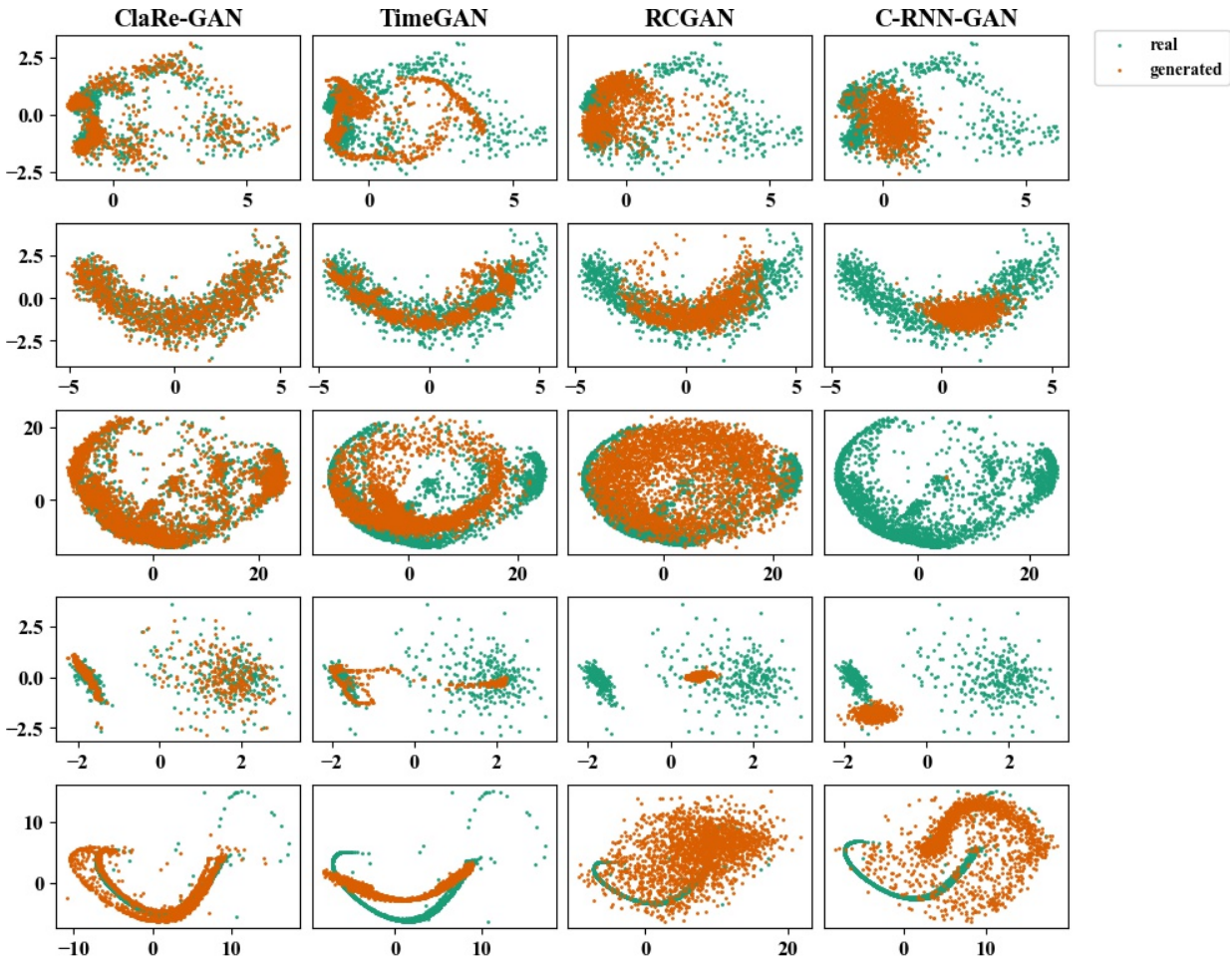

Figure 3.3: Comparison of the real (in green) and generated (in orange) data with PCA. Each row presents the results of a specific dataset (from top to bottom): ItalyPowerDemand, TwoLeadECG, Yoga, DistalPhalanxTW, and FreezerRegularTrain. A good performing GAN should be able to capture the distribution of the real dataset, i.e., we expect a strong similarity between the distribution of the real and the generated data in this 2-dimensional space.

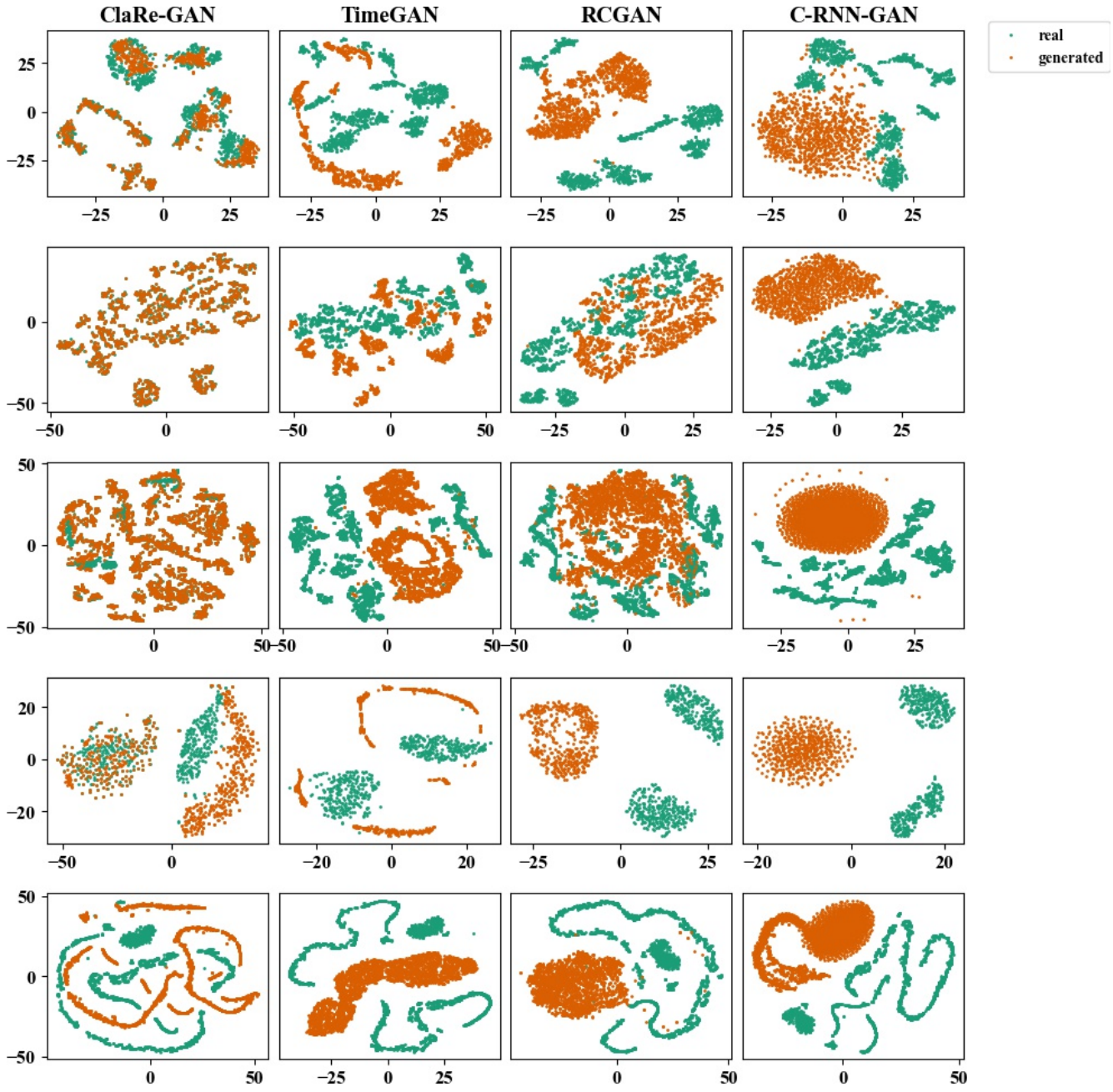

Figure 3.4: Comparison of the real (in green) and generated (in orange) data with t-SNE. Each row presents the results of a specific dataset (from top to bottom): ItalyPowerDemand, TwoLeadECG, Yoga, DistalPhalanxTW, and FreezerRegularTrain. A good performing GAN should be able to capture the distribution of the real dataset, i.e., we expect a strong similarity between the distribution of the real and the generated data in this 2-dimensional space.

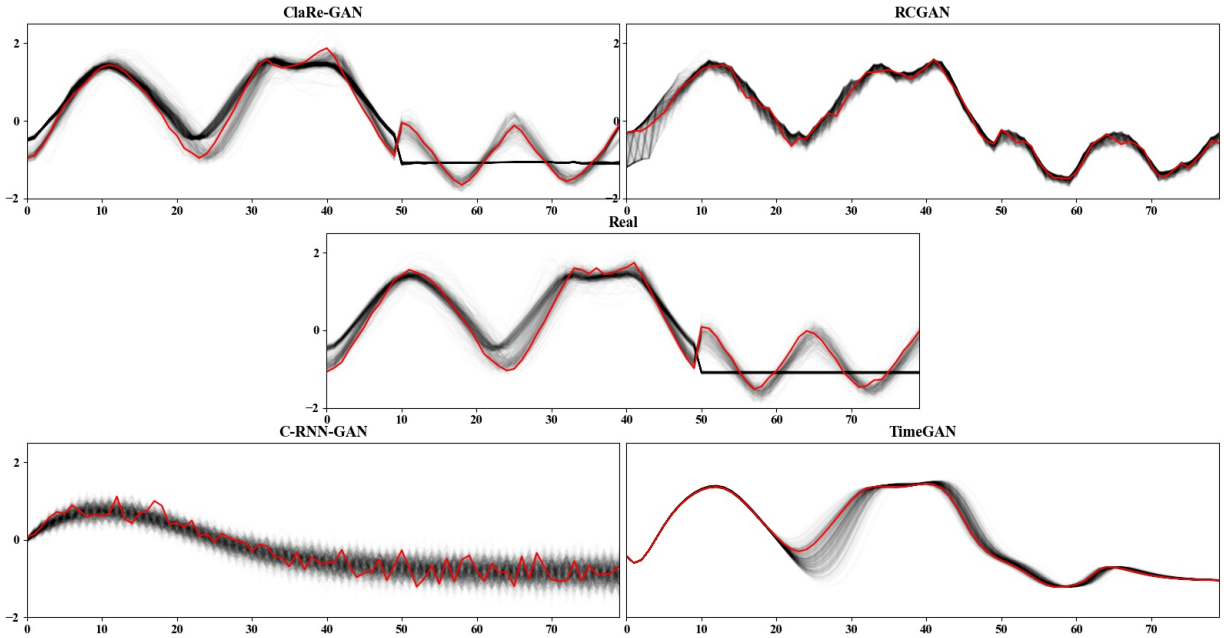

Figure 3.5: Illustration of the real and generated time series by the different frameworks for the DistalPhalanxTW dataset with 6 classes. The time series are depicted in black. An example is highlighted in each subplot in red. ClaRE-GAN was the unique algorithm that generated time series with a plateau for the last 30 timestamps. This class, originally appearing in the real data, was ignored by the other frameworks.

Table 3.2: DS computed on the time series generated by the different frameworks (ClaRe-GAN TimeGAN, RCGAN, and C-RNN-GAN) for the different datasets namely TwoLead-ECG, Yoga, and DistalPhalanxTW. A lower DS denotes a high-fidelity to the real datasets.

Dataset	***ClaRe-GAN***	***TimeGAN***	***RCGAN***	***C-RNN-GAN***
ItalyPowerDemand	0.221	0.4911	0.354	0.4997
TwoLeadECG	0.224	0.3985	0.2633	0.4498
Yoga	0.08	0.2	0.17	0.4998
DistalPhalanxTW	0.4273	0.496	0.447	0.4981
FreezerRegularTrain	0.39	0.41	0.47	0.45

Table 3.3: PS computed on the time series generated by the different frameworks (ClaRe-GAN TimeGAN, RCGAN, and C-RNN-GAN) for the different datasets namely TwoLead-ECG, Yoga and DistalPhalanxTW. A lower PS denotes better usefulness of the generated time series.

Dataset	*ClaRe-GAN*	*TimeGAN*	*RCGAN*	*C-RNN-GAN*
ItalyPowerDemand	0.1	0.112	0.1	0.304
TwoLeadECG	0.117	0.1246	0.127	0.5965
Yoga	0.156	0.157	0.16	0.5349
DistalPhalanxTW	0.1349	0.1749	0.2164	0.4784
FreezerRegularTrain	0.48	0.448	0.05	0.106

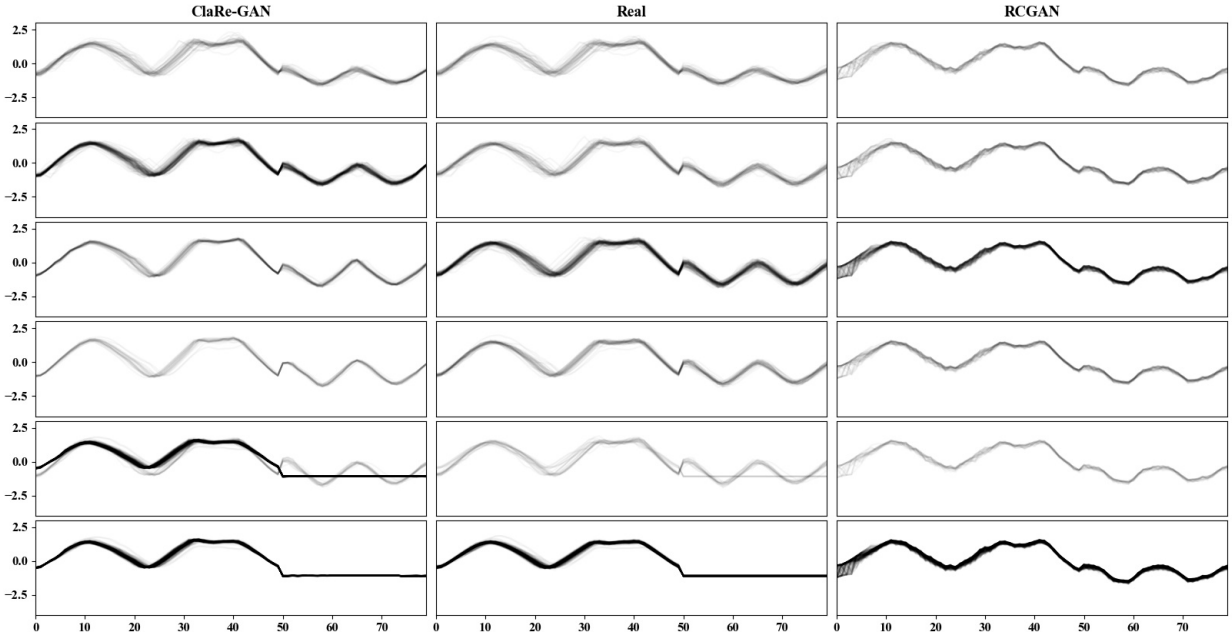

Figure 3.6: Illustration of the classes generated by ClaRe-GAN, the classes of the real dataset, and the classes generated by RCGAN for the DistalPhalanxTW dataset.

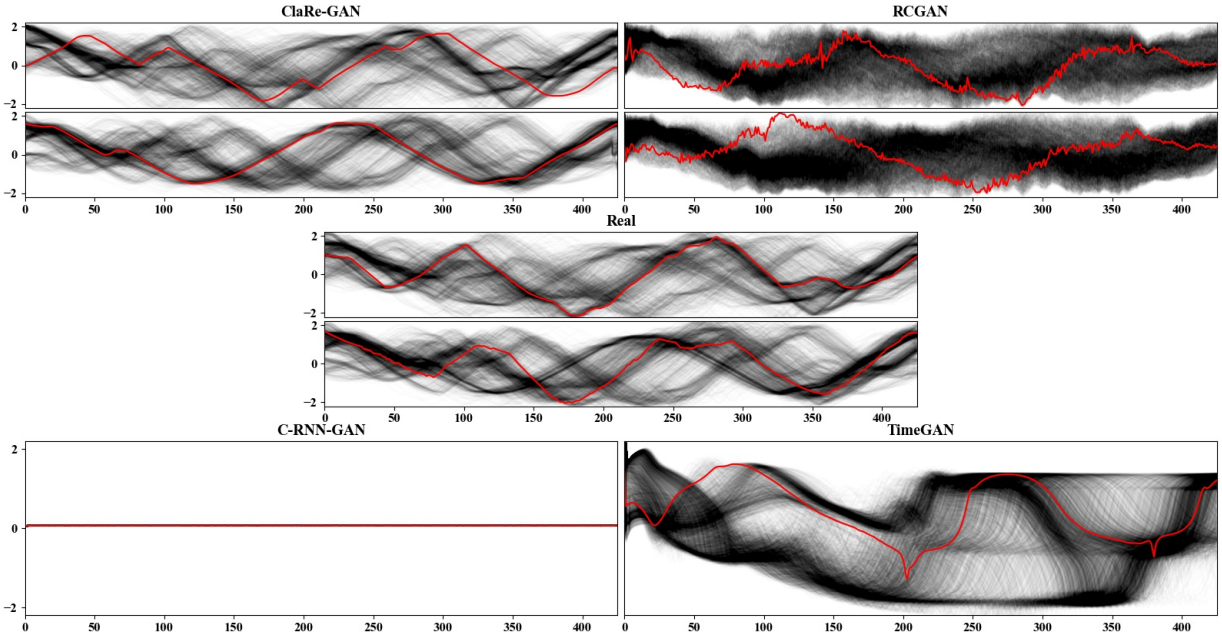

Figure 3.7: Illustration of the real and generated time series by ClaRe-GAN, RCGAN, C-RNN-GAN, and TimeGAN for the Yoga dataset. The time series are depicted in black. The red line presents an example time series for each subplot.

Fig. 3.9 illustrates the time series generated for the TwoLeadECG dataset. TimeGAN and RCGAN generate noisy time series similar to the real dataset. The time series generated by ClaRe-GAN differ from the real ones and are more private. This corresponds to the privacy values presented in Fig. 3.8, i.e., for the TwoLeadECG dataset Clare-GAN $\epsilon = 147.2$ compared to $\epsilon = 287.57$ and $\epsilon = 442.15$ for TimeGAN and RCGAN. C-RNN-GAN generates noise. This explains why C-RNN-GAN achieves the best ϵ values in Fig. 3.8.

In this chapter, we considered the problem of generating time series with and without privacy boundaries. In this context, ClaRe-GAN a novel generative model for time series datasets stemming from different classes is introduced. After that, the state-of-the-art generative models are combined with DP to ensure a private time series generation. In the next chapter, we will focus on another but similar problem namely time series translation, a method that improves the diversity of a dataset by exploring new and non-existing conditions.

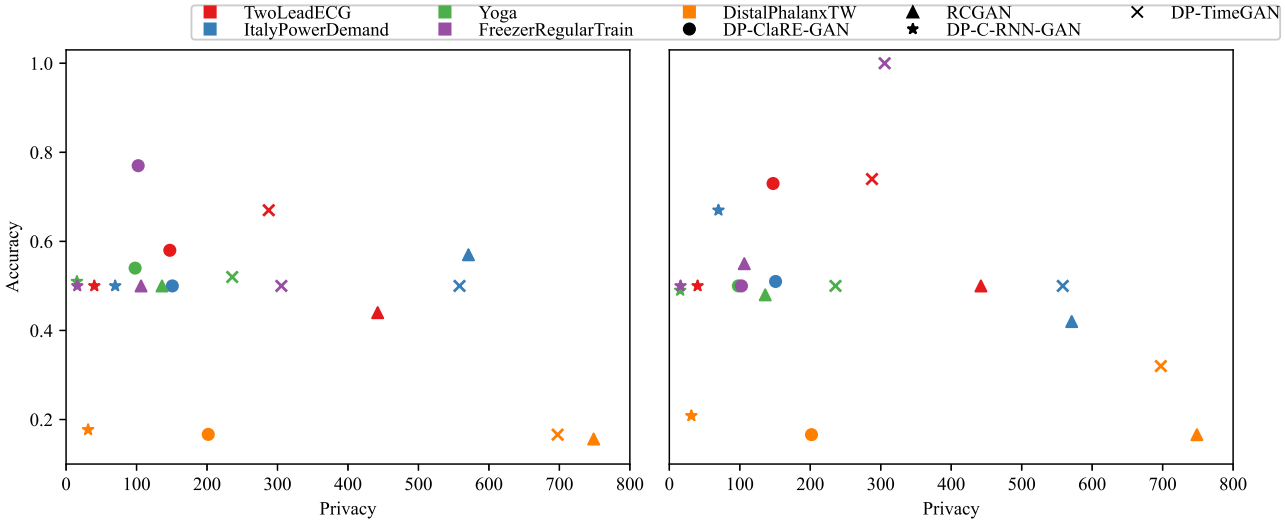

Figure 3.8: Test accuracy values of TSTR and TRTS methods for the differentially private generative models and the different datasets depicted in the left and right subfigure respectively. While a higher accuracy value denotes better usefulness of the generated data, a lower ϵ value denotesa better privacy.

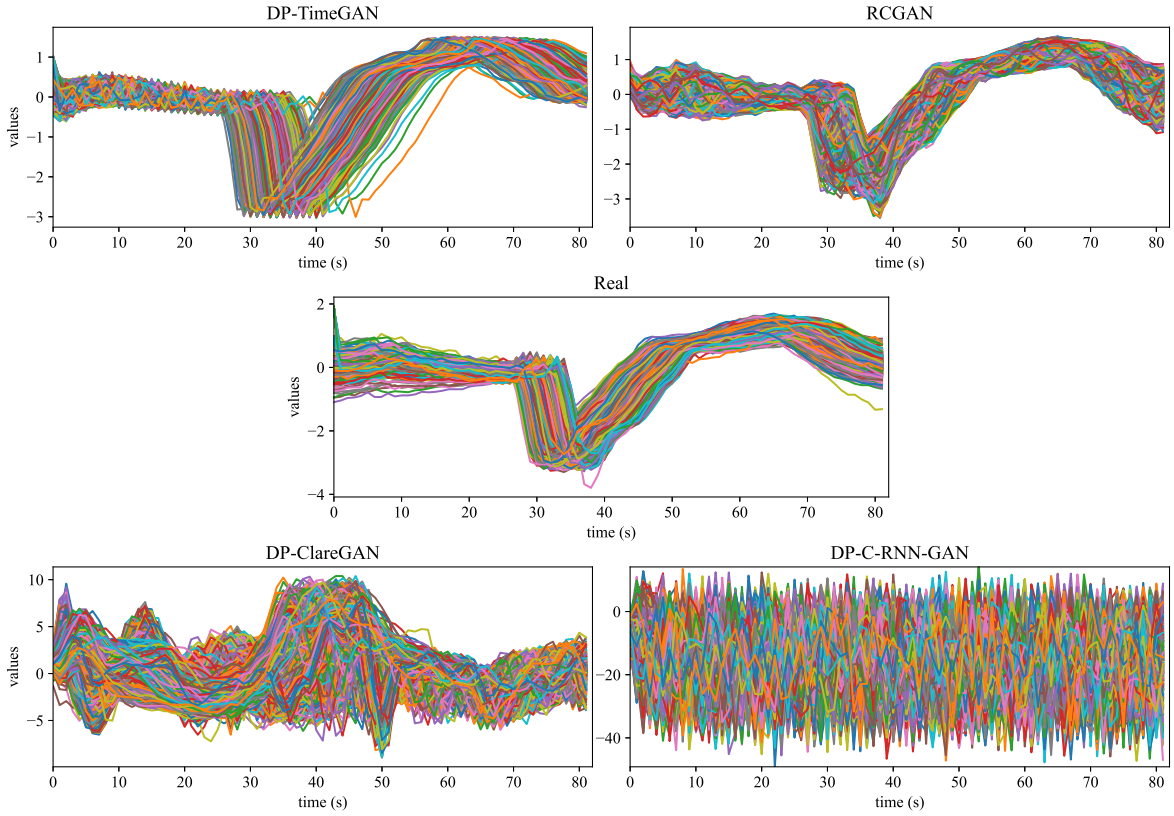

Figure 3.9: Illustration of the real times and the time series generated by the different differentially private models, i.e., DP-TimeGAN, RCGAN, DP-ClaRe-GAN, DP-C-RNN-GAN for the TwoLeadECG dataset.

CHAPTER

4

TIME SERIES TRANSLATION

In the last few years, huge progress has been made to enable image-to-image translation, i.e., to map the components of images from a source visual domain to a target domain. In this chapter, we exploit the recent improvements made in this field to tackle another issue, namely time series translation. Our main purpose is to find suitable techniques to map time series from one domain to another one e.g., to map different machine behaviors to different operating environments.

Two rooms are equipped with two different ventilation systems from different manufacturers. Each ventilation system starts cooling whenever the room temperature raises above a temperature threshold t_{max} and stops when the temperature drops below a minimal temperature t_{min}. The first room is small and consequently cools down and warms up faster. Hence, the corresponding ventilation system is frequently turned on and off. The second room is bigger. In this case, the ventilation system needs more time to cool down and the room's temperature will not warm up rapidly. The ventilation system of this room will represent slower on/off cycles. The performance of these ventilation systems is only comparable if they are running in the same environment and under the same conditions.

Now assume that we want to use time series data from normal operations to build a condition monitoring system. A straightforward approach is to build individual monitoring systems for each specific machine and environment. However, it is often desirable to build only one generalized condition monitoring system that can be applied to any machine in any environment. This approach often leads to better model quality because the model can be based on larger amounts of data from different machines and different operating conditions; and it avoids the effort to manage many individualized models (deployment, maintenance, etc.). However, for building such generalized models, it is

necessary to translate time series data between different machines and different environments. For this purpose, we propose a new method called DR-TiST [16, 17], a modified version of DRIT [15], and compare its performance with CycleGAN-VC.

The performance of DR-TiST is tested on different use case scenarios namely translating ventilation systems, human activities, and DC motors. First, we consider a real-world use case where we transfer the time series behavior of a ventilation system to the environmental conditions of a different ventilation system and introduce new evaluation metrics to evaluate its performance. After that, we focus on transforming sensor data depicting different human activities. Finally, we consider the problem of translating time series of one controlled DC motor to imitate time series from another motor. Our main goal is to test different controllers and find the best performing controller for a motor operating in the field without knowing its mathematical model. By means of DR-TiST, we split the time series of each control system into two representation vectors: a first vector depicting the motor characteristics and its operating mode and a second vector describing the controller effect. We test our method on a scenario where we simulate the behavior of two different controlled DC motors. We map the behavior of a controller of a lab motor to a field motor. For all scenarios, the performance of DR-TiST is compared to CycleGAN-VC [18], a special form of an image-to-image translation algorithm used for voice conversion presented in section 2. We demonstrate that the time series generated by DR-TiST are more realistic than the ones generated by CycleGAN-VC.

In spite of the broad adoption of AI, its impact and evolution are still limited due to the constantly increasing privacy matters. In many domains, such as medical or industrial domains, the AI's potential is still restricted due to privacy concerns making for example external collaborations binding. To allow data sharing without confidentiality concerns, we extend the existing time series translation methods that map time series from a source to a target domain with DP. To this end, we propose DP-DR-TiST and DP-CycleGAN-VC the differentially private versions of DR-TiST and CycleGAN-VC. We assess the performance of these algorithms against two essential criteria, i.e., privacy and utility. In this context, we prove that the translated time series are private and useful. The experiments are conducted on the previously described three translation use cases. The obtained results show that DP-DR-TiST outperforms DP-CycleGAN-VC in finding the right trade-off between sample privacy and sample utility.

4.1 DR-TiST: new Algorithm for Time Series Translation

DR-TiST achieves time series translation thanks to the disentangled representation proposed by DRIT. Each time series is divided into a functional behavior highlighting the properties of the time series and a context describing the environmental setup. For example, a time series depicting the behavior of an engine over time can be divided into a functional behavior depicting its behavior in the on/off states and its operating mode, i.e., times at which it is off or on. Based on the extracted functional behavior and operating mode, it is possible to translate the functional behavior of this time series to other operating modes or to simulate the behavior of other engines in its operating mode.

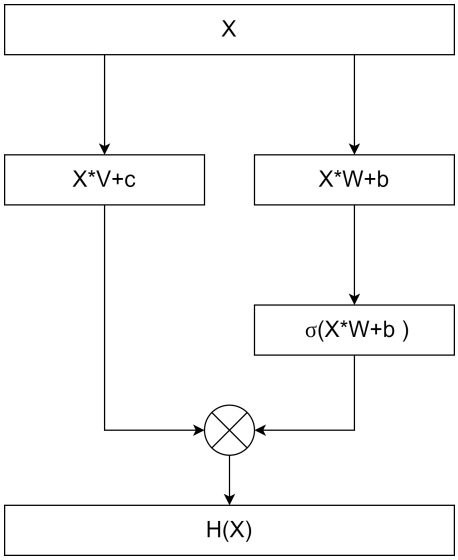

Figure 4.1: Illustration of the gated CNN structure. The output of the layer $H(X)$ for an input X is computed by multiplying element-wise $X \cdot V + c$ and $\sigma(X \cdot W + b)$ where $X \cdot V + c$ and $X \cdot W + b$ are the resulting vectors of the convolution on the input X and the sigmoid function is applied on $X \cdot W + b$.

For two machines M_1 and M_2, DR-TiST achieves representation disentanglement using functional behavior encoders $\{E^F_{M_1}, E^F_{M_2}\}$ context encoders $\{E^C_{M_1}, E^C_{M_2}\}$ Generators $\{G_{M_1}, G_{M_2}\}$ Discriminators $\{D_{M_1}, D_{M_2}\}$ and an adversarial Discriminator D_{adv}. Based on weight-sharing and the adversarial Discriminator D_{adv}, each time series will be mapped to a common latent space $\mathcal{C}$ depicting the functional behaviors and environmental setup spaces $\mathcal{A}_{M_1}$ $\mathcal{A}_{M_2}$. Intuitively, we assume that both machines have different parameters and differ in their environmental setup and hence show different behaviors but share some common physical properties that must be recognized by the algorithm.

Disentanglement and encoding are achieved based on two main techniques: weight-sharing and the machine characteristic Discriminator D_{adv} . To guarantee that $E^F_{M_1}$ and $E^F_{M_2}$ are mapped to the same latent space, in DR-TiST [16], the weight of the last layers of both encoders are shared. At the same time, it is essential to properly encode the engine-specific characteristics of M_1 and M_2 and hence the encoded machine representations must be different. This is achieved by means of the additional Discriminator D_{adv} which discriminates between the machine representations, while $E^F_{M_1}$ and $E^F_{M_2}$ aim to learn representations that are hard to discriminate. The adversarial loss can be hence

computed as follows:

$$L_{adv}(E^F_{M_1}, E^F_{M_2}, D_{adv}) = \mathbb{E}_F \left[\frac{1}{2} log\left(D_{F_1}\right) + \frac{1}{2} log\left(1 - D_{F_1}\right)\right] + \mathbb{E}_F \left[\frac{1}{2} log\left(D_{F_2}\right) + \frac{1}{2} log\left(1 - D_{F_2}\right)\right], \quad (4.1)$$

where $D_{F_1} = D_{adv}(E^F_{M_1}(f_1))$ and $D_{F_2} = D_{adv}(E^F_{M_2}(f_2))$ and f_1 and f_2 are time series of M_1 and M_2 respectively.

The encoders, Generators, and Discriminators of DRIT were originally designed with a two-dimensional CNN in order to process images. This neural network structure is not suitable for time series data due to its sequential structure. To adapt DRIT to time series data we make two major modifications: apply the gated CNN structure and replace the two-dimensional CNN with the one-dimensional CNN that takes the temporal relationship between the data points into consideration. Gated temporal convolutions were originally introduced by Dauphin et al. [77] and achieved state-of-the-art results in language- and speech modeling. In contrast to recurrent networks where the output of a layer is computed with the recurrent function $h_i = v\left(h_{i-1}, w_{i-1}\right)$, gated CNN can be employed in a parallel manner. This allows for faster computation. Gated CNN utilizes a Gated Linear Units (GLUs) as an activation function. The output of a layer $l + 1$, H_{l+1}, is computed based on the output of the layer H_l and the model parameters W, V, b, and c as follows:

$$H_{l+1} = \left(H_l \cdot W + b\right) \otimes \sigma\left(H_l \cdot V + c\right), \quad (4.2)$$

where $\otimes$ is the element-wise product and σ is the sigmoid function. Fig. 4.1 shows the gated CNN structure.

The gated structure is tested on the different components of DRIT. Tests show that the best results are obtained when it is only applied to the Generator. Thus, the proposed DR-TiST structure applies gated CNN on the residual blocks of the Generator. As the last modification, we integrated instance normalization [78], a well-known method for improving the quality of images during the generation process, and replace the deconvolution blocks used in the Generator with a pixel shuffler [79]. The resulting residual blocks used in DR-TiST are illustrated in Fig. 4.2 and the full Generator's architecture is depicted in Fig. 4.3. The Discriminators and the encoders remain unchanged.

4.2 DP*: Privacy-preserving Approaches

In the last few years, the world was witnessing a rapid evolution in the field of AI and its application domains ranging from banking to industrial applications. Indeed, a crucial and urgent need for robust and efficient ML models that are widely applicable, is observed. The recent success in ML and DL is strongly related to the amount of

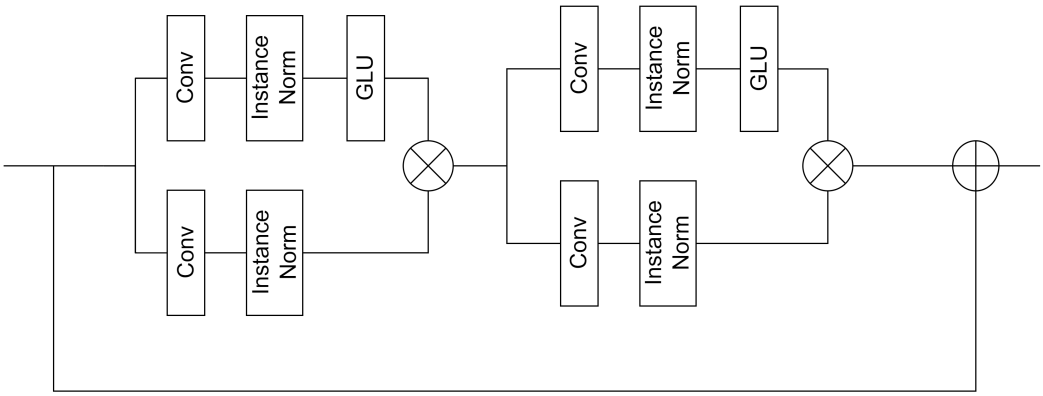

Figure 4.2: Illustration of the residual block used in the Generator of DR-TiST. Conv denotes a convolution, instance norm denotes instance normalization and GLU denotes the activation function.

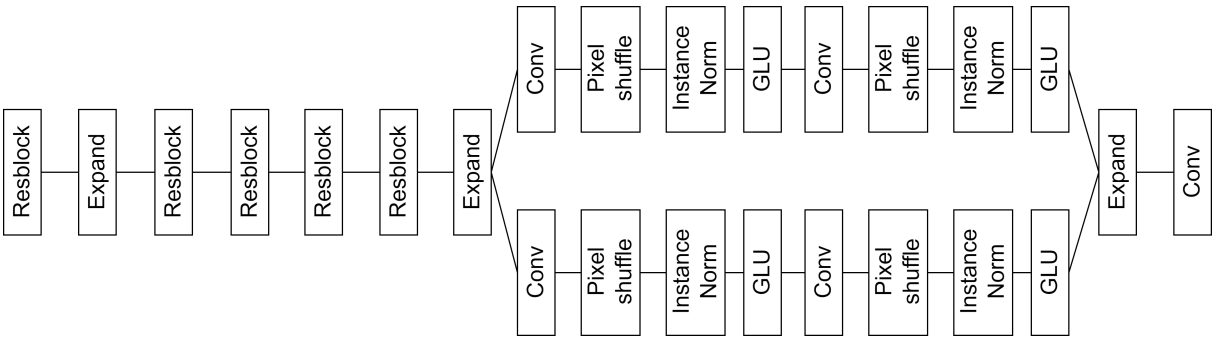

Figure 4.3: Illustration of the Generator's architecture used in DR-TiST.

publicly available data e.g., Imagenet, CIFAR... At the same time, the growing privacy considerations are limiting the applicability of ML. This pushed us to investigate privacy-preserving translation approaches to obtain more data for an existing dataset and simultaneously guarantee privacy. In the following, we describe two situations from the healthcare domain (illustrated in Fig. 4.4) and industrial domain (illustrated in Fig. 4.5) where such methods are of paramount importance.

By way of example, a data owner of individual-level data such as medical data wants to cooperate with an external partner on developing efficient ML algorithms that efficiently detect some illnesses. To protect the privacy of the persons that participated in the study and to generate more data for each class, it is essential to use privacy-preserving AI techniques. The newly generated data can be safely shared with external partners.

A machine manufacturer owns n machines with different mechanical characteristics that are placed in different environments and hence presenting different environmental behaviors. The manufacturer wants to cooperate with external partners on developing robust AI algorithms that can be applied to the different machines independently from their operating environments and their mechanical characteristics. To this end, a dataset depicting the behavior of many machines in different environments is needed and crucial in assessing the robustness of the developed AI methods. Such a dataset can be obtained using time series translation techniques. Starting from a collection of data depicting the behavior of each machine in one environment, it is possible to simulate the behavior of the different machines in all the operating environments. However, each recorded time series from the original dataset contains some sensitive information that should not be shared with the external partners such as how often the machine was turned on and off. Thus, privacy issues prevent the machine manufacturer from sharing the obtained data. To deal with this problem, it is essential to encode for each machine its functional behavior in terms of its mechanical properties and its environmental setup without exposing the sensitive information of each recorded time series. By way of example, we consider an algorithm meant to perform the translation between two machines machine 1 and 2. It should learn that machine 1 was frequently turned on/off whereas machine 2 is characterized by slower on/off cycles without memorizing the exact times at which the machines were turned on/off in each recorded time series.

In this part, we propose an extended version of the time series translation algorithms to transform time series between different environments with privacy guarantees. In general, such approaches will encourage data holder to publish their data and hence provide interesting complex real-world use cases for the research community. At the same time, it will permit them to find new collaboration opportunities and work safely with external partners on improving the use of AI in their application domains. In fact, the new data still depict the general pattern of the real data and have the same reactivity to the ML model. However, they do not expose sensitive information contained in the initially recorded time series such as rare diseases, machine parameters or times when the machines were on/off...

In the following, we call DP-DR-TiST and DP-CycleGAN-VC the extended version of DR-TiST and CycleGAN-VC that achieve time series translation in a privacy-preserving manner. The algorithms perform time series translation by learning the functional be-

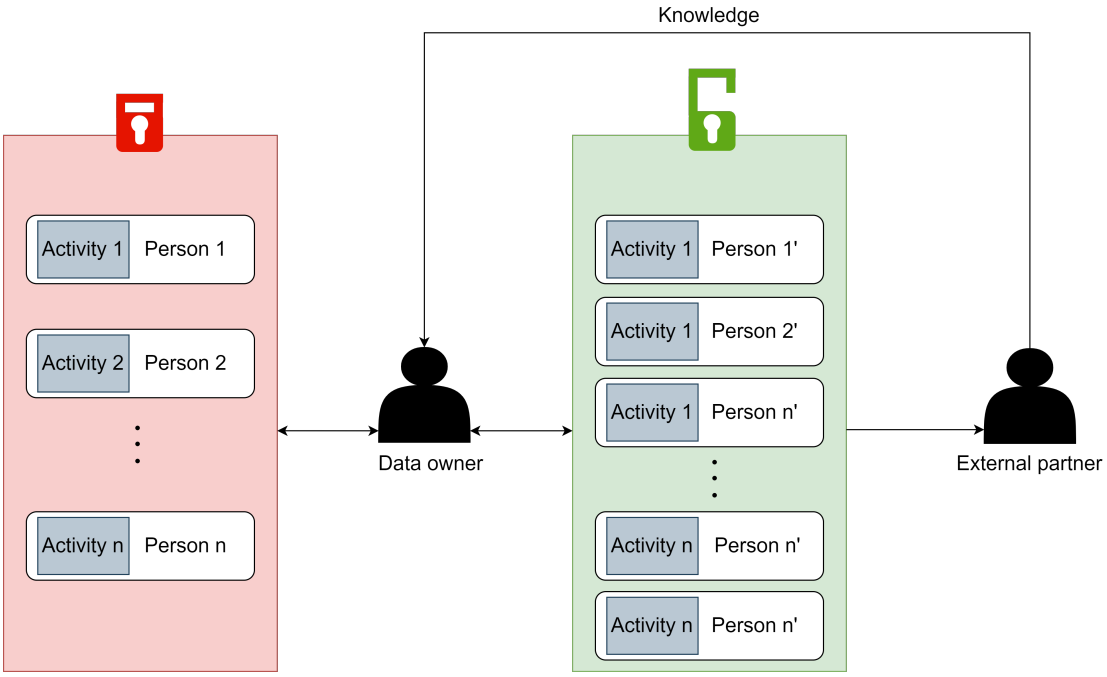

Figure 4.4: Illustration of the private *Human Activities Dataset* Use Case: A data owner holding individual-level sensor measurements of human activities can use the privacy preserving translation techniques proposed in this work, DP-DR-TiST and DP-CycleGAN-VC, to create a new *differentially private* dataset where each activity is mapped to all the persons and anonymized. This generated dataset can be freely shared and can hence be used for external collaborations.

havior and the operating mode of two machines without violating their privacy. In general, the data recorded from both machines may contain some sensitive information related to the machine's characteristics or to its environment. This information should not be shared between both domains, i.e., source and target domain, and should not be considered during the translation task. By way of example, to preserve privacy, it is essential to learn the general functional behavior of a machine instead of learning the time series specific features that are related to special and private conditions and that may be revealed during the recording process.

To guarantee privacy, the architecture of DR-TiST and DP-CycleGAN-VC is modified and both methods are equipped with a private Discriminator that relies on DPSGD. The private Discriminator is based on two main techniques: clipping gradient and adding random noise. During the training, the per-example gradients of the Discriminator loss is computed for the real and synthetic data separately. The gradients are later clipped to the minimal value between their $L2$-norm and a clipping value C and summed up. Finally, Gaussian noise $N(0, \sigma^2 C^2)$ is added to the values, where σ is a noise multiplier. As stated in the post-processing theorem [59], defined in section 2.3, the private Discriminator makes any generative model, independently from its architecture

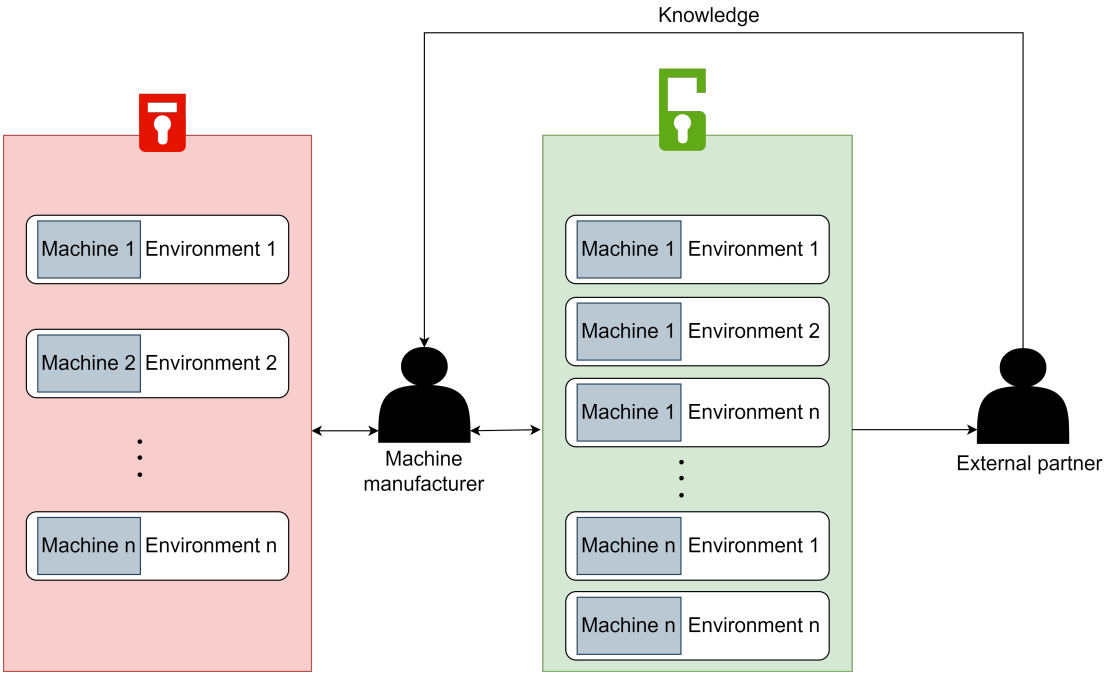

Figure 4.5: Illustration of the private *Ventilation Systems Dataset* Use Case: A Machine manufacturer holding n machines placed in n different environments can use the privacy preserving translation techniques proposed in this work, DP-DR-TiST and DP-CycleGAN-VC, to create a new *differentially private* dataset depicting the behavior of each machine in all the environments. This generated dataset can be freely shared and can hence be used for external collaborations.

complexity, private. To track the privacy loss, we use the RDP accounting technique [72]. This method enables more precise estimation of the privacy loss than MA and guarantees at the same time an easy calculation of the privacy loss for composite mechanisms.

4.3 Experiments

In the following, we consider three different datasets to test the performance of DR-TiST and CycleGAN-VC and their DP variants.

4.3.1 Datasets Description

Ventilation Systems Dataset In this use case, we are taking into consideration two ventilation systems, system 1 and system 2, placed in two different rooms which constitute the environmental setup. The first room, room A, is huge and has a cold environment contrary to the second one, room B, which is warmer and smaller. The variation of the temperature is different from one room to another. In fact, due to the size of room A, its temperature takes time to rise and fall. Hence, its ventilation system will not be

frequently switching on and off. This leads to slow on/off cycles, i.e., times at which the machine switches its operating mode. On the other hand, in room B, the on/off cycles are changing faster because the temperature rises and falls quickly. Generally, condition monitoring systems are designed for individual ventilation systems and operating conditions leading to a very specific solution for exactly one setup. When AI-based condition monitoring systems are designed using only the available data of both machines in their particular environment, very single models will arise. There is a strong interest to design condition monitoring systems, that generalize, and hence can be applied to any machine in any environment. In reality, the amount of data is hard to collect, and it is only feasible if they are available. We work around this problem by generating a more heterogeneous training data set, consisting of "mixed" time series data, where system 1 controls room B and system 2 room A, which are non-realistic conditions.

To the best of our knowledge, this is the first study with the goal of aligning time series. Thus, the evaluation of DR-TiST on a real sensor dataset where the behavior may be unexpected will be complicated. To be able to exactly evaluate the performance of DR-TiST, we design a synthetic dataset where the success or failure of translating the behavior of one system into the operating mode of another system is obvious and quantifiable. To this end, we simulate the behavior of two different engines of two different ventilation systems, engine 1 and engine 2. The engines are turned on and off in different time slots and are operating differently, i.e., engine 1 is frequently turned on and off while engine 2 shows a more stable behavior.

Following discussions with domain experts, we decided to model the machines' behavior using the standard exponential behavior. Hence, the machine's behavior in the on and off states is computed as follows:

$$y_{on}(t) = MRS \cdot \left(1 - \exp\left(-\frac{t - \tau_1}{\tau}\right)\right) + n, \tag{4.3}$$

$$y_{off}(t) = \exp\left(-\frac{t - \tau_0}{\tau}\right) + n, \tag{4.4}$$

where τ and MRS characterize the engines and $n \sim N(0, 0.01)$ is Gaussian noise.

Engine 1, driving the ventilation system in room A, is running with a low Maximal Rotational Speed (MRS) equal to 1. Whereas a more efficient engine 2 is placed in the hotter environment, room B. Its MRS is equal to 1.5. Moreover, we assume that engine 1 is older and therefore slower in reaching the MRS value or the minimal value when it is started or stopped. Thus, it has a larger value of $\tau = 5$, compared to the newer engine 2 with a value of $\tau = 2$. Table 4.1 summarizes the characteristics of engine 1 and 2 used in our experiments. The initially collected data depict the properties of engine 1 and 2 in the conditions of room A, i.e., slow on/off cycles, and room B, i.e., fast on/off cycles respectively. We consider the task of generating the functional behavior of engine 1, characterized by $MRS = 1$ and $\tau = 5$, in the operating mode of engine 2 characterized by fast on/off cycles and vice versa. Fig. 4.7 illustrates the expected time series translation scheme and Fig. 4.6 describes the transformation process.

In our experiments, we evaluate the performance of DR-TiST with three different methods: visual inspection and two additional metrics that rely on ground truth data.

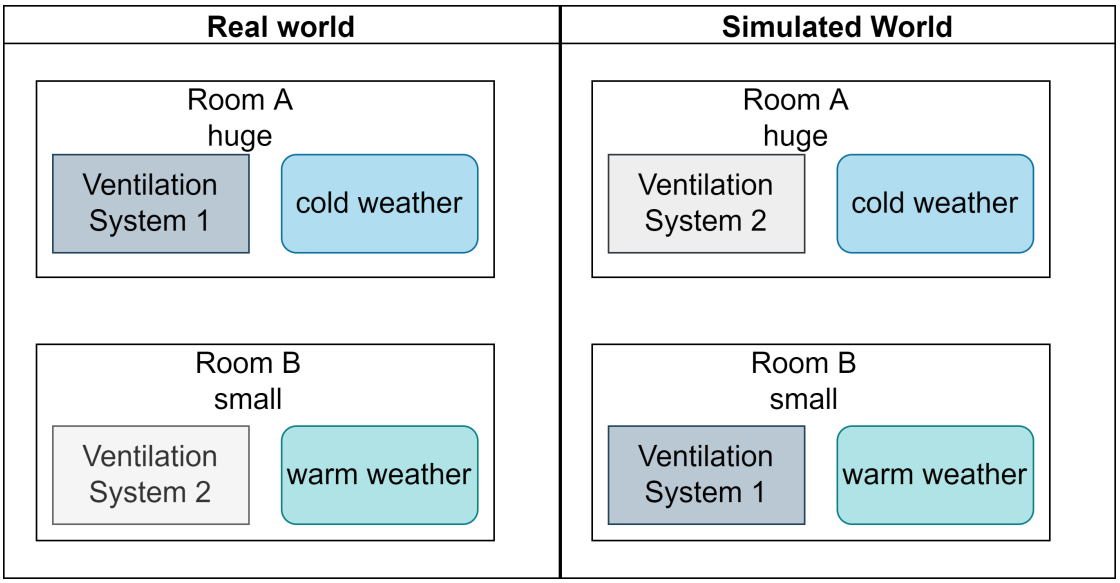

Figure 4.6: Use Case Scenario Ventilation Systems Dataset: given the real-world conditions we use DR-TiST to translate the behavior of ventilation system 1 to room B and ventilation system 2 to room A.

	Engine 1	Engine 2
$MRS[Hz]$	1	1.5
$\tau[min]$	5	2

Table 4.1: Characteristics of machine 1 and machine 2 of ventilation system 1 and 2 in the on and off states.

	μ_1	μ_2
$Test\ 1$	20	20
$Test\ 2$	30	40
$Test\ 3$	35	45
$Test\ 4$	40	50

Table 4.2: Mean of the Bernoulli distribution of the on/off times in the different experiments.

- Visual inspection: The evaluation of image-to-image translation techniques is still an open research problem. Current evaluation methods involve humans and rely on a visual inspection to assess the quality of the generated data [25]. Various works [53, 80, 81] rely on user studies to evaluate the realism of the generated images. Inspired by previous works, as an initial check, we perform a visual investigation to assess how realistic a time series appears to a domain expert, and we visually inspect the quality of the translated time series. Beyond visual inspection, we evaluate the performance of DR-TiST with two additional methods that assess the quality of the generated time series by comparing them to ground truth data.

- Error in on/off time prediction: Our main goal is to simulate the behavior of a first machine with the on/off times of a second machine, i.e., we want to transfer the behavior of a machine to the time domain of a second one. During the simulation of the behavior of each engine, we save its on/off times. This will correspond to the expected on/off time for the other engine. The expected behavior for each machine Y is then computed based on its expected on/off times and on the equations 4.3 and 4.4. The Root Mean Square Error (RMSE) between Y and the time series generated by DR-TiST, namely $\widehat{Y}$ can be calculated as follows:

$$RMSE = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} \left(Y_i - \widehat{Y}_i\right)^2}, \tag{4.5}$$

  where N denotes the number of data points per time series.

  Moreover, we compute the mean of the point-wise difference between the expected and obtained time series Y and $\widehat{Y}$:

$$D = \frac{\sum_{i=1}^{N} \mid Y_i - \widehat{Y}_i \mid}{\sum_{i=1}^{N} \mid Y_i \mid}. \tag{4.6}$$

Human Activities Dataset As a second use case, we considered the task of generating sensor measurements depicting a specific human activity. Based on public motion sensor data [82] describing different activities, the main goal would be to map measurements describing a specific activity to another target activity. The data were collected using a smartphone placed on the waist and from a group of 30 persons such that their ages were ranging from 19 to 48 years. We focus in our experiments on transforming the acceleration measurements in the x-direction consisting of time series with 128 data points. More precisely, we consider the task of transforming three human activities namely walking, sitting, and laying. We focus on their three possible combinations:

- Test 5: starting from sensor measurements depicting the activity laying we generate time series depicting the activity walking and vice versa

- Test 6: starting from sensor measurements depicting the activity sitting we generate time series depicting the activity walking and vice versa.

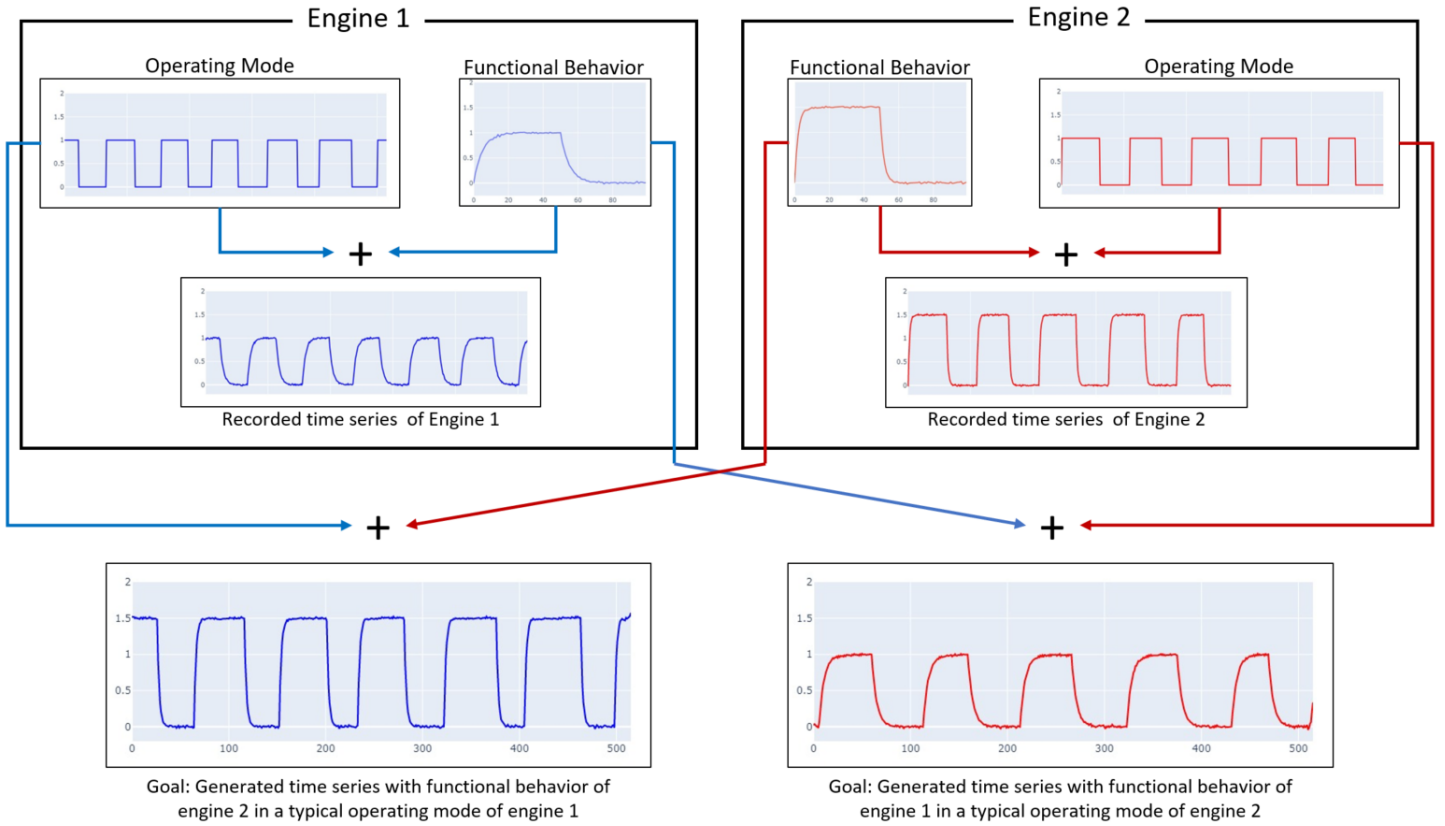

Figure 4.7: Time series translation: The proposed algorithm divides a given time series into operating mode and functional behavior. This learned representation allows to map the functional behavior of engine 1 into the operating mode of engine 2 and vice versa and hence helps to simulate the behavior of different engines in different environmental setups.

- Test 7: starting from sensor measurements depicting the activity laying we generate time series depicting the activity sitting and vice versa.

The performance of DR-TiST and CycleGAN-VC is assessed by comparing the obtained dataset, i.e., after the transformation with time series depicting the target activity. By way of example, for test 5, we compare the obtained data after transforming the time series depicting the laying to walking to a bench of time series depicting the walking activity. This is achieved by computing TRTS and TSTR. For the differentially private frameworks, we want to ensure that the new generated time series are private (by computing the privacy spent loss ϵ) and are still useful by comparing the generated time series to real time series of these classes (by computing TSTR and TRTS).

DC Motors Dataset A system consisting of a DC motor operating in the field, called field motor from here on, and its corresponding speed controller is given. The characteristics of this motor and hence its mathematical model are unknown. We want to test other controllers on this motor in order to find a better performing setup. Unfortunately, performing tests on this motor will be money- and time-consuming and modeling the motor is not possible as its mathematical model and parameters are unknown. At the same time, another motor with other characteristics is available in the laboratory (in the following referred to as the lab motor). Our main purpose is to find the optimal controller for the already operating motor in the field by artificially simulating the impact of different lab controllers on it.

To this end, we apply DR-TiST on time series recorded from both control systems (lab and field systems) to simulate a new control system depicting the effect of the lab controller on the field motor. Our method is equipped with a motor characteristic encoder, used to encode the properties of the field motor, and a controller effect encoder, which encodes the controller characteristics. This enables to split the time series of each control system into motor characteristics and controller effects and to simulate a new control system where the field motor is controlled by the lab controller.

Our scenario consists of DC motors with two different controllers operating in two different environments, i.e., field and lab, containing a DC motor and a controller. While we assume to have full control over the motor in the lab, allowing us to test any controller, the field motor is not accessible. However, a set of measurements from the field motor can be used as a basis to learn its characteristics. To do so, we will use the time series depicting the behavior of both control systems and use DR-TiST [16], presented in section 4.1, to disentangle the recorded time series into motor behavior and controller effect. The learned representations are later used to generate sensor measurements of the field motor when it is controlled by the lab controller: a configuration that was impossible to test in real life.

Such techniques can be used to easily test the performance of different controllers on numerous motors with different characteristics, without accessing or modeling the considered motor. We hence provide the valuable condition in which motor type and characteristics can be completely unknown.

DC motors are widely applied for different industrial purposes and are used to drive many devices ranging from automotive to medical machines. This is in particular due to the fact that the speed of a DC motor is easily controllable [83, 84]. This allows a high motor performance and usability for a wide range of applications [85, 86, 87]. Recently, numerous approaches focused on combining ML and statistical methods with Control Theory. By way of example, a Pythagorean fuzzy correlation-based approach has been proposed in [88] and an event-triggered gradient tracking algorithm for distributed optimization was presented in [89]. In addition to that, an approach that relies on fuzzy coefficient of impulsive intensity [90] has been applied to nonlinear impulsive control system. While some authors focused on the synchronization [91] and stability [92] of memristive neural networks, others considered a model of hybrid impulsive and switching Hopfield neural networks [93]. Inspired by them, we apply in this work a method that was originally designed to translate time series between different application methods to solve a control problem, i.e., find the optimal controller for a non-accessible motor [17].

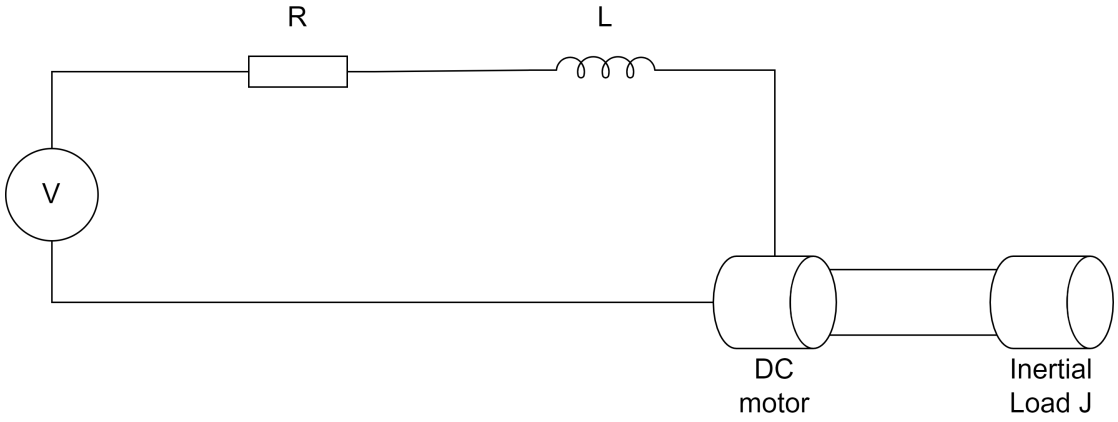

Figure 4.8: Control equivalent circuit of an armature controlled DC motor.

Our use case consists of two speed control systems of DC motors. We use a mathematical model of the DC motors [94] to simulate real-world conditions. A DC motor is a rotary electrical motor that converts direct current electrical energy into mechanical energy [95]. In our example, we will use separately excited motors. We consider a system consisting of the electrical equivalent circuit of an armature controlled DC motor and a rotor. A voltage source V is applied to the motor's armature inducing a rotational speed. The described system is depicted in Fig. 4.8.

The motor torque T and the back Electromotive Force (EMF) e can be computed using:

$$T = K_t i, \tag{4.7}$$

$$e = K_e \dot{\theta}, \tag{4.8}$$

where K_t is the motor torque constant, i is the armature current, $\dot{\theta}$ is the angular velocity (rad/s) and K_e is the electromotive force constant. In the rest of the thesis, we assume that $K_e = K_t = K$.

By applying the Newton's and Kirchhoff's Voltage Law (KVL) to the electrical circuit we get

$$J\ddot{\theta} + b\dot{\theta} = Ki, \tag{4.9}$$

$$V = R \cdot I + L \cdot \frac{di}{dt} + e, \tag{4.10}$$

$$V - K\dot{\theta} = R \cdot I + L \cdot \frac{di}{dt}, \tag{4.11}$$

where J is rotor inertia ($Kg.m^2$), b is viscous friction constant ($0.1 \cdot N.m.s$), V is the armature voltage (V), I is the armature current (A), R is the armature resistance (Ω), L is the armature inductance in (H).

We now apply the Laplace transform to the derived equations

$$s\left(Js + b\right)\theta\left(s\right) = KI\left(s\right), \tag{4.12}$$

$$\left(Ls + R\right)I\left(s\right) = V\left(s\right) - Ks\theta\left(s\right). \tag{4.13}$$

We assume that at $t = 0$, $\dot{\theta} = i = 0$. The state-space model of the DC motor can be computed with

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} = \begin{bmatrix}-\frac{b}{J} & \frac{K}{J}\\ -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{L}\end{bmatrix}V, \tag{4.14}$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix}. \tag{4.15}$$

The transfer function of the DC Motor can be expressed by

$$P\left(s\right) = \frac{\dot{\theta}\left(s\right)}{V\left(s\right)} = \frac{K}{\left(Js + b\right)\left(Ls + R\right) + K^2}. \tag{4.16}$$

In order to control the speed of this motor, the previously described system is related to a controller.

As a next step, we consider the problem of controlling the speed of a nonlinear DC motor [96, 97]. To this end, we consider in addition to the viscous friction, the Coulomb friction:

$$T_c = C_a \cdot sgn(\dot{\theta}), \tag{4.17}$$

where C_a denotes the Coulomb friction coefficient. The Coulomb friction is added to the Eq. 4.12 as follows:

$$J\ddot{\theta} + b\dot{\theta} + C_a \cdot sgn(\dot{\theta}) = Ki. \tag{4.18}$$

The structure of the speed control system for a DC motor is presented in Fig. 4.9.

Even though we do not know the mathematical model of the field motor and since it is not accessible due to its current use, we still looking for a good controller. To this end, we will use another accessible motor in the lab and a recorded dataset from the field motor.

We consider two speed control systems:

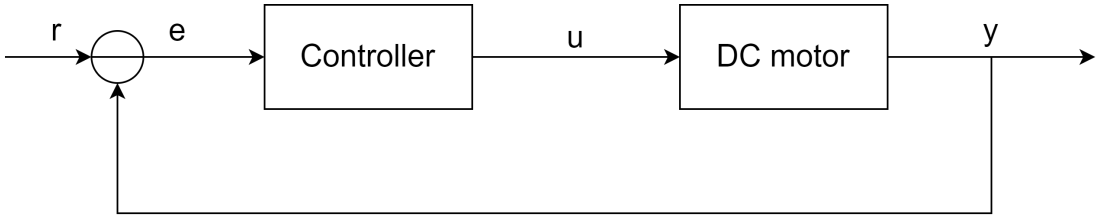

Figure 4.9: Speed control of a DC motor: the controller is used to control the speed of the DC motor by computing the difference error e between the desired speed r and the current output y.

- Field system cannot be modified and is equipped with a field motor and controller where the controller presents a sub-optimal set of parameters
- Lab system is equipped with a lab motor and a modifiable controller enabling us to run different tests.

Our main goal is to find the optimal controller for the field motor by solely performing real-world tests on the lab motor. Starting from real-world conditions, i.e., the running field and lab system, we generate new data where we translate the impact of the lab controller to the field motor, as illustrated in Fig. 4.10. This is achieved by disentangling the recorded time series of both systems into motor behavior and controller effect. This allows us to map the behavior of any motor to any controller and especially to simulate the behavior of the field motor when it is controlled with the lab controller without performing real tests on the field motor. We use DR-TiST to disentangle the effects of the motor behavior and the effects of the controller in the time series. The used architecture, illustrated in Fig. 4.10, presents two encoders a motor characteristic encoder that learns the motor behavior and a controller effect encoder that learns the impact of the controller on the considered system.

The encoded information is later used to generate new time series depicting the behavior of the field motor when it is controlled with the lab controller, by combining the encoding of the field motor behavior with the lab controller effects. Hence, with DR-TiST, it is possible to map a motor behavior to any controller stemming from another system. It is then easy to test the reactivity of a motor to different speed controllers with different characteristics. In order to test and thoroughly evaluate the feasibility of the previously described scenario, we simulate the behavior of two different speed control systems depicting real-world conditions in different situations. The DC motors are simulated based on Eq. 4.14, Eq. 4.16, and Eq. 4.18 for the nonlinear motor, and the used parameters of the considered lab and field motors are summarized in Table. 4.4. The lab and field control systems, as illustrated in Fig. 4.9, consist of a controller and a DC motor. For each system, we simulate 1000 multivariate time series with a duration of 200 seconds consisting of three different signals, namely the output of the system y, the system state u and the reference signal r. The reference signal r is assumed to change every 50 seconds from one stable state to another. The levels of the steady states are

Table 4.3: Parameters of field and lab controllers.

Test	Field controller	Lab controller
8	$(0.05, 0.01, 0.01); C_a = 0$	$(0.3522, 0.2317, 0.0798); C_a = 0$
9	$(0.05, 0.01, 0.01); C_a = 0$	$(0.2, 0.1, 0.05); C_a = 0$
10	$(0.05, 0.01, 0.01); C_a = 0.005$	$(0.3522, 0.2317, 0.0798); C_a = 0.005$
11	$(0.05, 0.01, 0.01); C_a = 0.005$	$(0.2, 0.1, 0.05); C_a = 0.005$

selected randomly between 1 and 1.5. The error signal is computed as follows $e = r - y$.

Table 4.4: Physical properties of the field and lab motors.

	Field motor	Lab motor
$J(kg.m^2)$	0.01	0.015
$Tb(N.m.s)$	0.00003	0.00003
K	0.023	0.01
$R(\Omega)$	1	1
$L(H)$	0.5	0.5

We perform four different tests where we change the controller lab properties and add a nonlinear behavior to the plant by considering the Coulomb friction. In all the experiments, we use a PID-controller. The transfer function of the PID-controller is computed as follows:

$$C(s) = K_p + \frac{K_i}{s} + K_d s. \tag{4.19}$$

The field motor is controlled with a same controller with $K_p = 0.05$, $K_i = 0.01$ and $K_d = 0.01$. While a linear model is used for the DC motor in tests 8 and 9, DC motors of tests 10 and 11 are characterized by a non-linear behavior. The lab controller parameters of test 8 were obtained in [98] with the Ziegler-Nichols method and are hence supposed to be the optimal parameters for the field motor. Table 4.3 summarizes the controller parameters used in the different tests. To be able to evaluate the time series with DR-TiST for a specific reference signal, we simulate time series $y_{gt}(t)$, where gt stands for ground truth, showing the expected behavior, i.e., when the field motor is controlled with the controller lab and compare it to the output $y(t)$ generated by DR-TiST.

Our goal is to show that we can efficiently translate the behavior of a controlled DC motor without performing tests on it. To this end, we compare a simulated expected behavior with the time series generated with DR-TiST by computing the following metrics [99, 100]:

- **Steady State Error** e_{ss}, we compute the difference between the reference signal r and the final achieved value of $y(t)$.
- **Rise Time** tr, the time it takes for the $y(t)$ to increase from 10% to 90% of its final value.
- **Overshoot** can be calculated with the formula: $(m_{val} - f_{val})/f_{val} \times 100$ where m_{val} denotes the maximal value and f_{val} denotes the final value.

Using these metrics, it is possible to compare the expected controller effect to the effect of the controller simulated by DR-TiST in terms of stability and efficiency. In the evaluation, we compute the mean of tr and e_{ss} for 100 times series with a duration of 200 seconds and reference signal changing every 50 seconds and compare those metrics for our generated and ground truth time series.

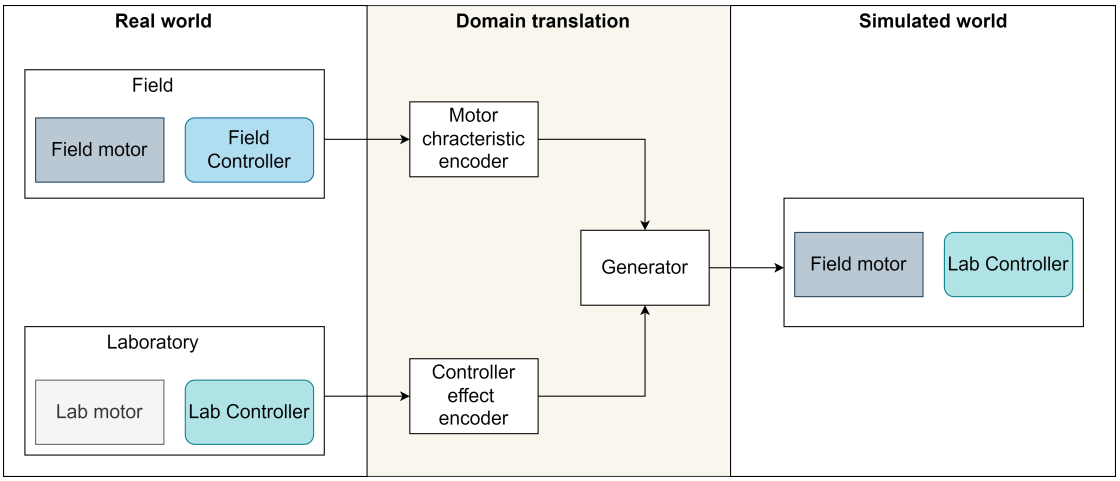

Figure 4.10: Use Case Scenario DC motors Dataset: given the real-world conditions we use DR-TiST to translate the behavior of the field motor and the lab controller. Time series depicting both control systems, i.e., field and lab systems are used.

4.3.2 Experimental Setup

Given initial data depicting the properties and the operating mode of engine 1 and engine 2, our main goal is to make sure that DR-TiST is able to generate new data that depict the behavior of engine 1 with the on/off cycles of engine 2 and vice versa. To this end, we conduct four different tests where we change the distribution of the on/off times for each engine. The mean of the Bernoulli distribution is varied for the different tests. Table 4.2 illustrates the mean of the on/off times for the different tests where μ_1 and μ_2 denote the mean of the Bernoulli distribution for the on/off times of engine 1 and engine 2 respectively. For test 1, the same Bernoulli distribution is used for engine 1 and engine 2, i.e., $\mu_1 = \mu_2 = 20$. In contrast to test 1, μ_1 and μ_2 are in the other tests different. The initial data are computed based on the equations

4.3 and 4.4, the characteristics of the machines presented in Table 4.1, and the on/off cycles sampled from the Bernoulli distribution. In the training phase, we use 1000 time series for each engine with 508 data points. Additionally, in order to assess the impact of the amount of data on the performance of the framework, we repeat test 2 with a different number of data points per time series, namely 208 and 416. We evaluate the performance of DR-TiST by generating 100 time series in the test phase. We expect that DR-TiST is able to generate new time series of engine 1 in the time domain of engine 2 and vice versa. Finally, we compare the performance of DR-TiST to CycleGAN-VC, a modified version of CycleGAN designed for voice conversion purposes. As CycleGAN-VC is computationally expensive, we reduce the length of the data points per time series to 208 and rerun tests 2 and 3.

For the activity dataset, we compute the TRTS and TSTR accuracies for 50 time series that are mapped to the corresponding activity. Our main goal is to compare the obtained time series after the transformation to the target behavior. The TRTS and TSTR accuracies are reported for different classification methods namely Random Forest (RF) [76], Decision Trees (DT) [101], Logistic Regression (LR), Support Vector Machines (SVM) [102] and XGBoost (XGB) [103].

The performance of DP-DR-TiST and DP-CycleGAN-VC is assessed in terms of privacy and accuracy for the different tests depicting the various use cases. First, the frameworks are trained to translate 1000 time series of each class to another target class. During the training, at each iteration, the TSTR and TRTS values are computed for a bench of randomly selected and translated time series. The model of the iteration with the best performance (best TSTR and TRTS values) is selected and will be used in the assessment of privacy and utility. To enable a fair comparison between the frameworks, the same maximal number of iterations is used namely 100 iterations and we use $C = 0.3$, $\sigma = 0.3$ and $\delta = 10^{-3}$. For the evaluation, the behavior of 50 time series of each class is translated to the target class. This means that all the classes of the original dataset are equally well-represented in the evaluation procedure. Finally, the spent privacy loss is computed and the accuracies values TRTS and TSTR are calculated for these translated time series. Our main purpose is to identify the best-performing framework. In other words, the framework that finds the right balance between privacy and utility of the translated time series.

4.3.3 Results

Fig. 4.12 and Fig. 4.13 illustrate the time series generated by the DR-TiST for engine 2 and engine 1 of the ventilation systems dataset respectively. We see in Fig. 4.12 the recorded time series of engine 1. The time series produced by DR-TiST have the functional behavior of engine 2 and the same on/off cycles as the ones of engine 1. Fig. 4.13 demonstrates the originally recorded time series of engine 2, i.e., with a maximal value of 1.5, and the corresponding time series generated by the algorithm with the same on/off times and different functional behavior. For the different examples, we clearly see that the trained model was able to simulate the behavior of engine 1 with the on/off times of engine 2 and vice versa. This corresponds to the desired behavior. Since we

want to show that this is not only an exemplary time series, for each test, we compute our quantifiable metrics on a set of 100 generated time series. Table 4.7 summarizes the results of $RMSE$ and D values respectively when generating time series of engine 1 and engine 2 in the different experiments. The obtained results demonstrate that test 1 has the lowest $RMSE$ and D values, i.e., $D_{eng1} = 0.0652$ and $RMSE_{eng2} = 0.26$. It is to notice that in test 1 $\mu_1 = \mu_2$. In test 2, $RMSE_{eng1}$ and $RMSE_{eng2}$ are higher when DR-TiST is trained with 208 data points instead of 416 data points. Hence, the amount of data has an impact on the results. Test 3 and test 4 are characterized by higher $RMSE$ and D values. By way of example, D_{eng2} is equal to 0.5 and 0.16 for test 3 and test 4 respectively.

Examples of time series generated by CycleGAN-VC and DR-TiST in tests 2 and 3 are presented in Fig. 4.14 and 4.11 respectively. The point-wise differences between the time series generated by CycleGAN-VC and the expected time series are higher than the ones generated with DR-TiST. Moreover, the time series of DR-TiST are more realistic and fit better to the target time domain than the time series generated by CycleGAN-VC. Fig. 4.11 shows that for test 3 the time series of CycleGAN-VC present a completely wrong on/off cycles. Tables 4.5 and 4.6 show the $RMSE$ and D values of DR-TiST compared to CycleGAN-VC. In tests 2 and 3, DR-TiST outperforms CycleGAN-VC. The D and $RMSE$ values of DR-TiST are for both experiments lower than the ones of CycleGAN-VC, i.e., in test 4 $D2_{cyc} = 1.77$ and $D2_{DR} = 0.43$. It is to be noted that DR-TiST is faster and more efficient than CycleGAN-VC in terms of time and computation.

	Engine 2 Generated		Engine 1 Generated	
	$D2_{cyc}$	$D2_{DR}$	$D1_{cyc}$	$D2_{DR}$
$Test\ 2$	0.41	0.36	0.31	0.2
$Test\ 3$	1.77	0.43	1.66	0.35

Table 4.5: D values for generating time series of engine 1 and engine 2 by CycleGAN-VC and DR-TiST for test 2 and 3. D_{cyc} and D_{DR} denote the values of D for the time series of CycleGAN-VC and DR-TiST respectively.

	Engine 2 Generated		Engine 1 Generated	
	$RMSE2_{cyc}$	$RMSE2_{DR}$	$RMSE1_{cyc}$	$RMSE1_{DR}$
$Test\ 2$	0.565	0.51	0.25	0.18
$Test\ 3$	1.37	0.59	0.86	0.28

Table 4.6: RMSE values for generating time series of engine 1 and engine 2 by CycleGAN-VC and DR-TiST for test 2 and 3. $RMSE_{cyc}$ and $RMSE_{DR}$ denote the values of $RMSE$ for the time series of CycleGAN-VC and DR-TiST respectively.

For the human activities dataset, the TRTS and TSTR values are computed for the different datasets and both translation algorithms DR-TiST and CycleGAN-VC. The

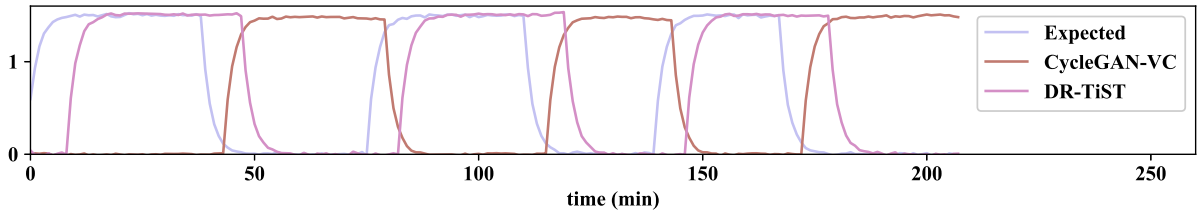

Figure 4.11: Comparison of the time series of engine 2 generated with the operating mode of engine 1 in test 3 where $\mu_1 = 35$ and $\mu_2 = 45$. Time series generated by CycleGAN-VC has a completely different behavior than the expected time series. The time series produced by DR-TiST are more realistic.

	D_{eng1}	D_{eng2}	$RMSE_{eng2}$	$RMSE_{eng1}$
$Test\ 1$	0.066	0.1693	0.053	0.266
$Test\ 2_{lg208}$	0.202	0.36	0.18	0.51
$Test\ 2_{lg416}$	0.11	0.21	0.1105	0.347
$Test\ 3$	0.309	0.5	0.66	0.26
$Test\ 4$	0.13	0.164	0.122	0.29

Table 4.7: Computed D and $RMSE$ values for the different tests of the Ventilation Systems Dataset. D_{eng1} and D_{eng2}, $RMSE_{eng2}$ and $RMSE_{eng1}$ denote the computed D and $RMSE$ values during the test phase when generating time of engine 1 and engine 2 respectively.

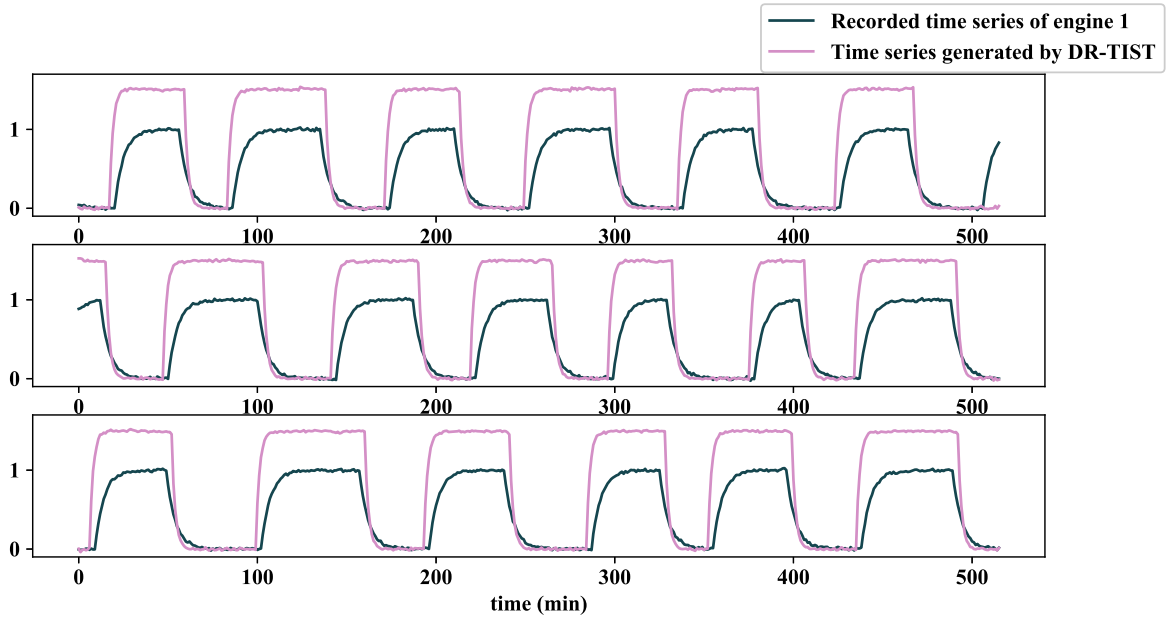

Figure 4.12: Examples of time series of engine 2 in operating mode of engine 1 generated in *test* 4 where $\mu_1 = 40$ and $\mu_2 = 50$. DR-TiST was able to map the functional behavior of engine 2, characterized by a higher amplitude, in the time domain of engine 1.

obtained results are presented in Table 4.8. We notice that for almost all the tests and all the classification algorithms, the TRTS and TSTR values of the DR-TiST algorithm are higher than the ones of the CycleGAN-VC. By way of example, for test 6 the mean of TRTS and TSTR values for the DR-TiST dataset is equal to 0.8949 and 0.6917 respectively, while it is equal to 0.5646 and 0.5163 for the CycleGAN-VC algorithm. This shows that the data obtained by DR-TiST fits better to the target data and hence can better represent the target activity. In general, it is to be noted that the TSTR values are lower than the TRTS values. We conclude that for this dataset the utility and hence the quality of the time series generated by DR-TiST is better than the ones generated by CycleGAN-VC.

For the conducted tests of the DC motors dataset, we visualize the distribution of the rise times tr, the steady-state errors e_{ss} and overshoot values of the ground truth, and the time series generated by DR-TiST in Fig. 4.15, Fig. 4.16, and Fig. 4.17 respectively. Moreover, Table. 4.9, Table. 4.10 and Table. 4.11 present the mean of the rise times, overshoot and steady-state errors of the ground truth and the time series generated by DR-TiST and CycleGAN-VC. We clearly see that for the linear DC motor, i.e., tests 8 and 9, the distributions of the different metrics of the ground truth and the time series generated by DR-TiST are similar. These results nicely show that DR-TiST captured the controller characteristics and that the time series generated by DR-TiST depicts the same reactivity as the expected behavior in $y_{gt}(t)$. Moreover, the mean of tr, e_{ss}, and

	Human Activities					
	Test 5		Test 6		Test 7	
	DR-TiST	Cyc-VC	DR-TiST	Cyc-VC	DR-TiST	Cyc-VC
RF	**0,9949**	0.101	**1**	0.5000	**0.8888**	0.5000
DT	**0.9242**	0.0404	**0.9797**	0.5353	**0.6818**	0.6363
LR	**0.5050**	0.2575	0.5050	**0.7272**	**0.3939**	0.0252
SVM	**1**	0.0101	**1**	0.5000	**0.707**	0.5000
XGB	**0.9899**	0.409	**0.9899**	0.5606	**0.8636**	0.5000
Mean	**0.8828**	0.1636	**0.8949**	0.5646	**0.8603**	0.4323

(a)

	Human Activities					
	Test 5		Test 6		Test 7	
	DR-TiST	Cyc-VC	DR-TiST	Cyc-VC	DR-TiST	Cyc-VC
RF	**0.5714**	0.0816	**0.6224**	0.500	**0.6632**	0.5000
DT	**0.6326**	0.1428	**0.6632**	0.5408	**0.5714**	0.5000
LR	**0.5306**	0.4081	**0.5510**	0.5000	**0.5918**	0.4897
SVM	**0.6938**	0.3061	**0.9897**	0.5000	**0.6224**	0.5000
XGB	**0.5612**	0.1428	**0.6326**	0.5408	0.5000	0.5000
Mean	**0.5979**	0.2162	**0.6917**	0.5163	**0.5897**	0.4979

(b)

Table 4.8: Test TRTS (a) and TSTR (b) accuracies values for the Human Activities dataset, i.e., test 5, test 6 and test 7 computed with different ML models (RF, DT, LR ...). The TRTS values are obtained by training the ML models on real data and testing them with synthetic data obtained with DR-TiST, and CycleGAN-VC (denoted Cyc-VC). Meanwhile, the TSTR values are obtained by training the ML models with synthetic data for obtained with DR-TiST, and CycleGAN-VC, and testing on real data. The test accuracies correspond to the TRTS/ TSTR values.

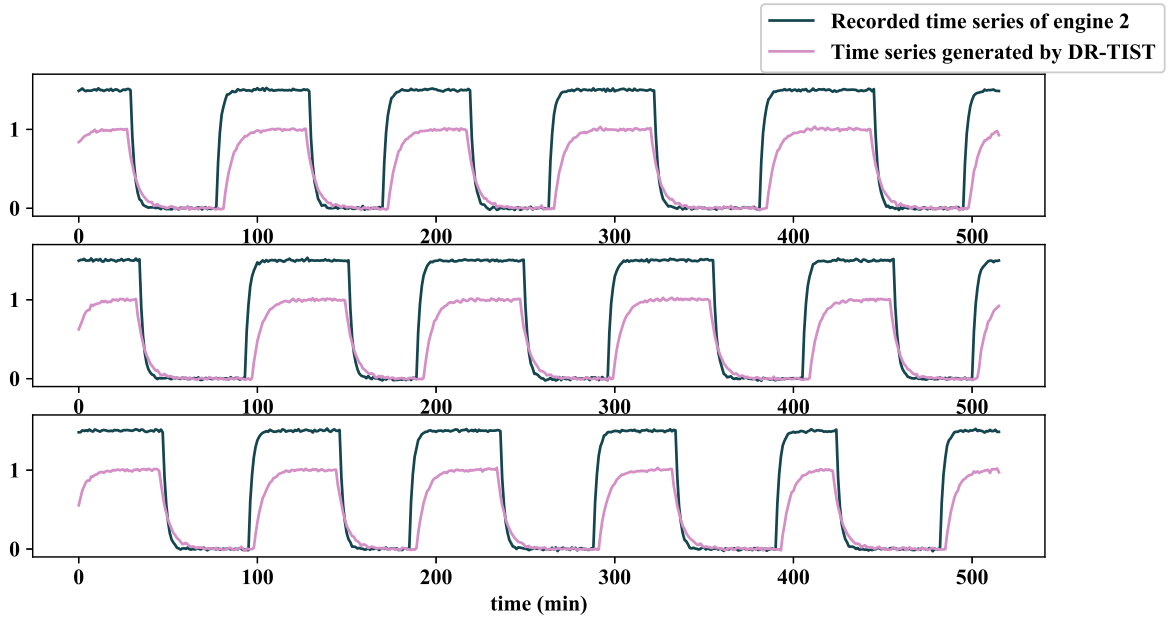

Figure 4.13: Examples of time series of engine 1 in operating mode of engine 2 generated in *test* 4 where $\mu_1 = 40$ and $\mu_2 = 50$. DR-TiST was able to map the functional behavior of engine 1, characterized by a lower amplitude, in the time domain of engine 2.

overshoot values for time series generated by DR-TiST are closer to the ground truth values than the values computed for the time series generated by CycleGAN-VC. This shows that also for these tests DR-TiST achieved the best results and was able to better capture and produce the desired reactivity.

Table 4.9: Mean of rise times in the ground truth tr_{gt}, and generated time series by DR-TiST $tr_{gen-dr-tist}$ and CycleGAN-VC $tr_{gen-cyc}$.

Test	tr_{gt}	$tr_{gen-dr-tist}$	$tr_{gen-cyc}$
8	1.35 ± 0.5	1.7 ± 1	2.95 ± 3.46662
9	1.6 ± 0.9	2.2 ± 1.4	8 ± 13.643
10	3.3 ± 5	3 ± 3.8	10.67 ± 13.85
11	4.35 ± 7.8	6.2 ± 11.7	9.03 ± 15.01146

Fig. 4.18 illustrates the output of a control system generated by DR-TiST and the expected output for the same noisy reference signal. The figure shows that DR-TiST was able to depict the right expected reactivity and hence predict the right output signal.

While the characteristics of the form of the signal, can very successfully be predicted by DR-TiST for tests 8 and 9, our experiments have shown that it is more difficult

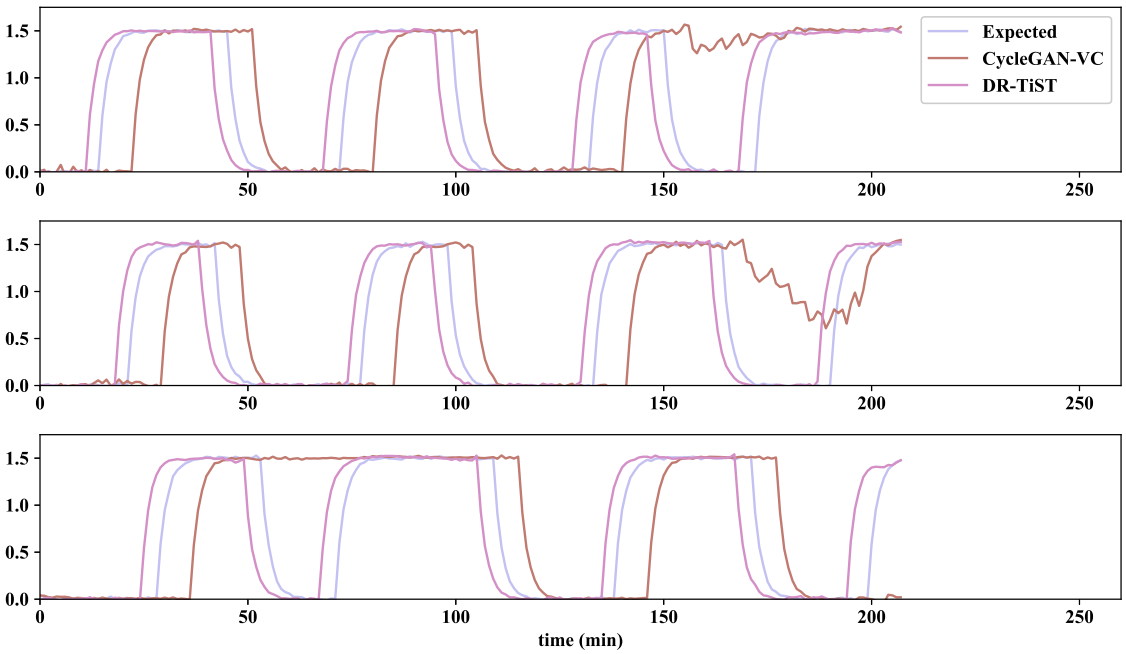

Figure 4.14: Comparison of time series of engine 2 generated with the operating mode of engine 1 in *test* 2 where $\mu_1 = 30$ and $\mu_2 = 40$. The time series generated by CycleGAN-VC and DR-TiST are compared to the expected behavior.

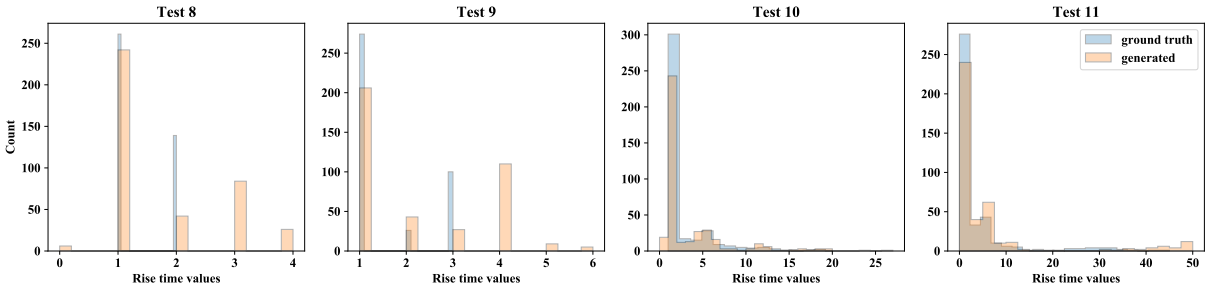

Figure 4.15: Distribution of the rise times for 100 outputs generated with DR-TiST and expected outputs computed as ground truth.

Table 4.10: Mean of overshoot in the ground truth $overshoot_{gt}$, and generated time series by DR-TiST $overshoot_{genDR-TiST}$ and and CycleGAN-VC $overshoot_{genCyc}$.

Test	$overshoot_{gt}$	$overshoot_{genDR-TiST}$	$overshoot_{genCyc}$
8	29.2 ± 24.8	22.75 ± 24.5	919.8991 ± 941.06205
9	21.5 ± 23.6	23 ± 24.8	58.8971 ± 54.0861
10	4.6 ± 3.8	3 ± 1.7	8.49 ± 27.73
11	4.6 ± 3.1	9.4 ± 11.9	1.087152 ± 56.303663

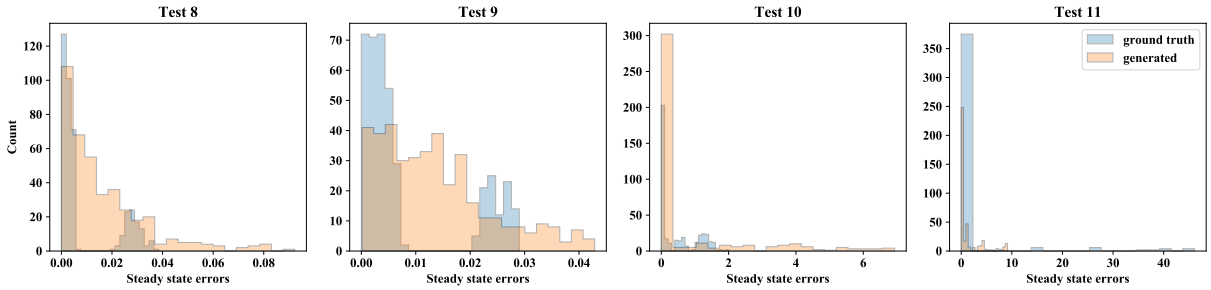

Figure 4.16: Distribution of the steady state errors for 100 outputs generated with DR-TiST and expected outputs computed as ground truth.

to correctly predict the right behavior for the nonlinear model used in tests 10 and 11. Unfortunately, in our experiments, we noticed that it is difficult for the algorithm to predict never seen output levels. We conclude that DR-TiST is really strong in predicting the dynamic of a controller but cannot perfectly extract the part of the motor characteristics, that result in different output levels in $y(t)$. It is to be noted that for all the tests namely the linear and non-linear ones DR-TiST outperformed CycleGAN-VC in terms of rise times, steady-state errors, and overshoot values.

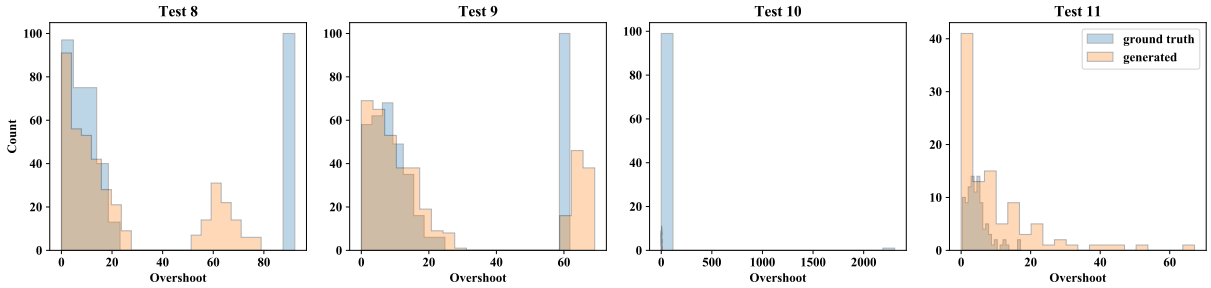

Figure 4.17: Distribution of the overshoot values for 100 outputs generated with DR-TiST and expected outputs computed as ground truth.

Table 4.11: Mean of steady state errors in the ground truth ess_{gt} and generated time series by DR-TiST $ess_{genDR-TiST}$ and CycleGAN-VC ess_{genCyc}.

Test	ess_{gt}	$ess_{genDR-TiST}$	ess_{genCyc}
8	0.009 ± 0.01	0.01 ± 0.016	0.89683 ± 0.755372
9	0.01 ± 0.009	0.01 ± 0.01	0.65748 ± 0.44229
10	0.46 ± 0.57	0.8 ± 1.6	0.95 ± 1.6881
11	2.4 ± 8.1	1.3 ± 2.3	3.2453 ± 1.2556908

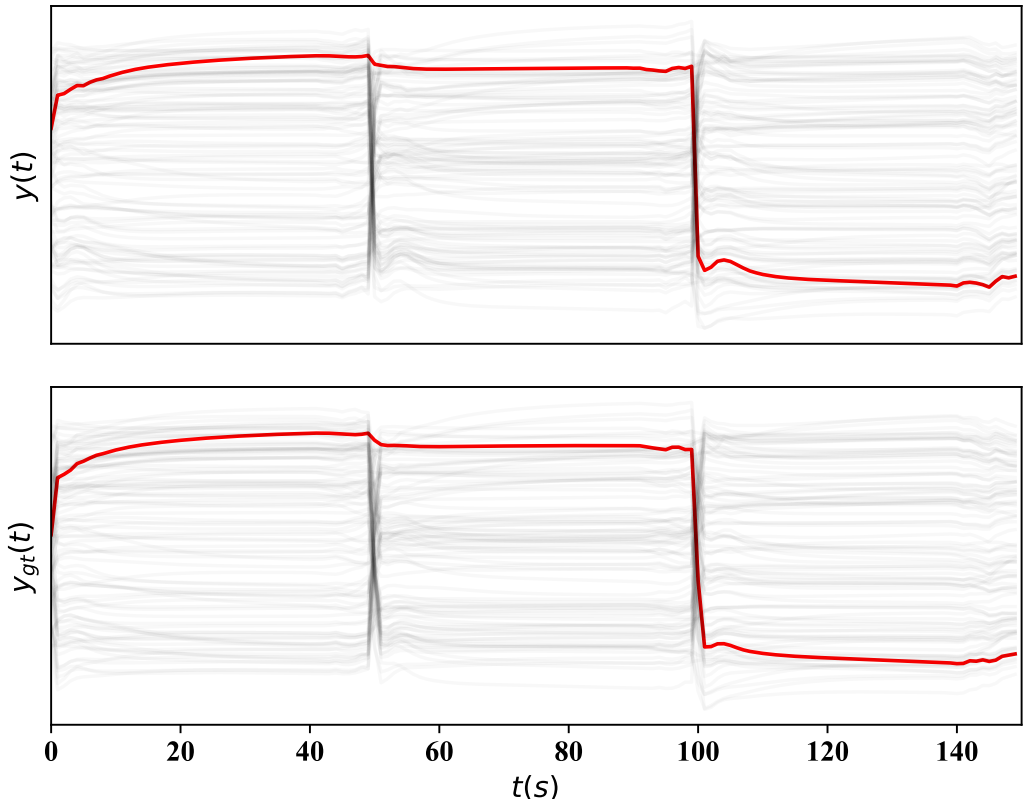

Figure 4.18: The output of a control system generated by DR-TiST (illustrated in the upper part) is compared to the expected output behavior (illustrated in the lower part) obtained by simulating the field motor when it is controlled with the controller lab. Each expected output $y_{gt}(t)$ and generated output $y(t)$ is obtained for the same noisy reference signal r. An example of generated and corresponding expected time series is highlighted in red. The remaining expected and generated outputs are depicted in gray.

Finally, we compare the performance of DP-DR-TiST and DP-CycleGAN-VC in terms of privacy and utility of translated time series for the previously described use cases. Tables 4.13, 4.14, and 4.15 depict the obtained test accuracies for TRTS and TSTR for the ventilation systems, human activities and DC motors use cases respectively. We clearly see that the best average TSTR values were achieved by DP-DR-TiST for almost all the tests. Moreover, for the ventilation systems and human activities use cases, DP-DR-TiST outperformed DP-CycleGAN-VC in terms of TRTS values. By way of example, for the ventilation systems and human activities tests, we noticed that the TRTS accuracies averages of DP-CycleGAN-VC are around 0.5 and that its TSTR accuracies vary between 0.5 and 0.6754, achieved for test 5. On the other hand, TRTS average accuracies of DP-DR-TiST vary between 0.6151 and 0.9181 and its TSTR accuracies are between 0.5767 and 0.8938. To conclude, in almost all the cases and independently of the used ML model, DP-DR-TiST outperformed DP-CycleGAN-VC in terms of utility.

In contrast to the utility, DP-CycleGAN-VC - except for test 3 - outperforms DP-DR-TiST in terms of privacy. By way of example, DP-CycleGAN-VC achieves a privacy loss equal to 76.12 and 54.44 in test 2 and test 6 respectively, while DP-DR-TiST privacy losses are equal to 128.38 and 90.14. The reached privacy losses ϵ are summarized in Table 4.12.

We conclude that DP-CycleGAN-VC is able to achieve the best privacy values, but the generated time series are not as useful as expected. Hence, this method generates private data but degrades drastically the quality of the time series in the translation process. Meanwhile, DP-DR-TiST finds a better balance between privacy and utility. It generates private data with a high utility.

We investigated in this chapter new private and non-private approaches to map time series from a source domain with initial conditions to new and never seen conditions. Next, we consider the problem of analyzing the generated and translated time series and comparing them with the target ones.

Use Case	Test	DP-CycleGAN-VC	DP-DR-TiST
Ventilation Systems	Test 2	**76.12**	128.38
	Test 3	86.32	**35.32**
Human Activities	Test 5	**55.72**	122.01
	Test 6	**54.44**	90.14
DC Motors	Test 8	**30.01**	122.01
	Test 11	**96.52**	122.015

Table 4.12: Obtained Privacy loss ϵ for the different use cases for $C = 0.3$, $\sigma = 0.3$ and $\delta = 10^{-3}$.

	Ventilation Systems			
	Test 2		Test 3	
	DR-TiST	CycleGAN-VC	DR-TiST	CycleGAN-VC
RF	**0.6212**	0.3989	**0.6919**	0.4393
DT	**0.6161**	0.4848	**0.6919**	0.4393
LR	**0.6010**	0.5000	**0.7272**	0.5000
SVM	**0.6161**	0.5	**0.6919**	0.5000
XGB	**0.6212**	0.4797	**0.6919**	0.4797
Mean	**0.6151**	0.4726	**0.6989**	0.4716

(a)

	Ventilation Systems			
	Test 2		Test 3	
	DR-TiST	CycleGAN-VC	DR-TiST	CycleGAN-VC
RF	**0.5757**	0.5000	**0.6919**	0.5000
DT	**0.7020**	0.5000	**0.6161**	0.5000
LR	**0.5858**	0.5000	**0.6010**	0.5000
SVM	0.5000	0.5000	0.5000	0.5000
XGB	**0.5202**	0.5000	**0.6818**	0.5303
Mean	**0.5767**	0.5000	**0.5878**	0.5060

(b)

Table 4.13: Test TRTS (a) and TSTR (b) accuracies values in test 2 and test 3 computed with different ML models (RF, DT, LR ...). The TRTS values are obtained by training the ML models on real data and testing them with synthetic data obtained with DP-DR-TiST, denoted DR-TiST, and DP-CycleGAN-VC, denoted CycleGAN-VC. Meanwhile, the TSTR values are obtained by training the ML models with synthetic data for obtained with DP-DR-TiST, denoted DR-TiST, and DP-CycleGAN-VC, denoted CycleGAN-VC, and testing on real data. The test accuracies correspond to the TRTS/ TSTR values. We use $C = 0.3$, $\sigma = 0.3$ and $\delta = 10^{-3}$.

	Human Activities			
	Test 5		Test 6	
	DR-TiST	CycleGAN-VC	DR-TiST	CycleGAN-VC
RF	**1**	0.5000	**0.7525**	0.5000
DT	**0.9646**	0.5757	**0.8535**	0.5050
LR	**0.7525**	0.6262	**0.5353**	0.4646
SVM	**0.9242**	0.5000	**0.6565**	0.5000
XGB	**0.9494**	0.5353	**0.8383**	0.5707
Mean	**0.9181**	0.5474	**0.7272**	0.5080

(a)

	Human Activities			
	Test 5		Test 6	
	DR-TiST	CycleGAN-VC	DR-TiST	CycleGAN-VC
RF	**1**	0.7653	**0.8775**	0.6938
DT	**0.8571**	0.7040	**0.8061**	0.6836
LR	**0.6836**	0.6326	**0.6938**	0.5000
SVM	**1**	0.5306	0.5000	**0.5714**
XGB	**0.9285**	0.7448	**0.8265**	0.6530
Mean	**0.8938**	0.6754	**0.7296**	0.6203

(b)

Table 4.14: Test TRTS (a) and TSTR (b) accuracies values for test 5 and test 6 computed with different ML models (RF, DT, LR ...). The TRTS values are obtained by training the ML models on real data and testing them with synthetic data obtained with DP-DR-TiST, denoted DR-TiST, and DP-CycleGAN-VC, denoted CycleGAN-VC. Meanwhile, the TSTR values are obtained by training the ML models with synthetic data for obtained with DP-DR-TiST, denoted DR-TiST, and DP-CycleGAN-VC, denoted CycleGAN-VC, and testing on real data. The test accuracies correspond to the TRTS/ TSTR values. We use $C = 0.3$, $\sigma = 0.3$ and $\delta = 10^{-3}$.

	DC Motors			
	Test 8		Test 11	
	DR-TiST	CycleGAN-VC	DR-TiST	CycleGAN-VC
RF	0.5000	**0.6969**	0.5606	**0.7424**
DT	0.48	**0.5000**	0.5000	**0.6060**
LR	0.6110	**0.808**	**0.49499**	0.4242
SVM	0.5000	0.5000	**0.5000**	0.4696
XGB	0.5200	**0.6666**	0.5606	**0.8282**
Mean	0.5222	**0.6343**	0.5232	**0.6140**

(a)

	DC Motors			
	Test 8		Test 11	
	DR-TiST	CycleGAN-VC	DR-TiST	CycleGAN-VC
RF	**0.5510**	0.5000	**0.9183**	0.5000
DT	**0.5612**	0.5000	0.3469	**0.5000**
LR	0.5000	0.5000	**0.6632**	0.5510
SVM	**0.5306**	0.5000	0.5000	**0.5510**
XGB	**0.6836**	0.5000	**0.7244**	0.3571
Mean	**0.5652**	0.5142	**0.5977**	0.4918

(b)

Table 4.15: Test TRTS (a) and TSTR (b) accuracies values for test 8 and test 11 computed with different ML models (RF, DT, LR ...). The TRTS values are obtained by training the ML models on real data and testing them with synthetic data obtained with DP-DR-TiST, denoted DR-TiST, and DP-CycleGAN-VC, denoted CycleGAN-VC. Meanwhile, the TSTR values are obtained by training the ML models with synthetic data for obtained with DP-DR-TiST, denoted DR-TiST, and DP-CycleGAN-VC, denoted CycleGAN-VC, and testing on real data. The test accuracies correspond to the TRTS/ TSTR values. We use $C = 0.3$, $\sigma = 0.3$ and $\delta = 10^{-3}$.

CHAPTER

5

TIME SERIES ANALYTICS

Time-dependent information arises in many fields ranging from meteorology, and medicine to stock markets. The analysis of such data is a central goal in VA, statistics, or ML and many related approaches exist [31, 32, 104, 105]. In particular, for ML methods a sufficient amount of training data and a balanced dataset where each data class is equally well represented are crucial for a good performance. In reality, ML experts often face situations where these criteria are not satisfied. In these cases, generating new data provides a possible solution.

This has pushed researchers to investigate new methods for data generation. In this context, GAN [1] are showing an outstanding performance. However, to trust a ML model e.g., a classifier trained on generated data, it is necessary to assess how realistic these data are and hence the performance of the generation process. Most efforts and best results have been shown for image generation, where the quality of the generated data can be easily assessed with the human eye.

Recently, a lot of effort is made by researchers to discover suitable metrics that evaluate the performance of GAN and substitute a human judge. The Discriminator and Generator losses, for example, cannot be considered as a measure of GAN performance and this ML approach lacks an objective function that defines an appropriate end of iterations with suitable data quality. Various evaluation methods have been described in [106] such as Parzen window or MMD. As proven in [106], the use of these methods has various disadvantages. Other methods e.g., inception score is designed only for images and cannot be easily applied to time series data. This pushed researchers to investigate new metrics for time series. By way of example, TSTR and TRTS were proposed by Esteban et al. [11] while introducing RCGAN.

In spite of their important number, the existing GAN metrics for time series have not been compared to each other and their advantages and disadvantages remain ambiguous. In particular, for an ML expert, it is essential to know which metric can detect which training problem. In this section, we study the state-of-the-art GAN metrics that were described in section 2.1.4 to determine the strengths and weaknesses of each approach. Our main purpose is to guide the ML expert by showing which metric can detect a specific training problem and recommending a possible combination of metrics that can be used together. This is achieved by performing different tests presented in section 2.1.4. Moreover, we propose a new metric MiVo and show that it outperforms the existing ones in detecting many training problems efficiently and simultaneously. Later, MiVo is applied in a different and sophisticated fashion to support the ML expert in the time series translation task.

Such metrics can be helpful while training the models. However, to fully trust the considered GAN model it is essential to visualize the time series. At the same time, finding common characteristics and differences between real and generated data may be a complicated and exhaustive task. In fact, comparing generated datasets to real datasets is challenging, especially for time series data. To deal with this problem, we propose a visual approach to assess the quality of the generated data and to effectively compare them to the real ones. The presented framework can be used by a human judge while training GAN models.

We strongly believe that the computational and visual methods have their own advantages and disadvantages and allow for a different manner of exploration. While a metric might be essential to get a quick overview over the whole iterations, the visual method will allow the ML expert to discover more insights about the data. Our main goal is to ease the quality assessment task for ML experts by presenting a VA method that enables a rigorous and detailed investigation and a mathematical metric that keeps track of the performance of GAN over the iterations in a fast and compact manner. The best evaluation setup would be to start by using the mathematical metric to select some interesting iteration candidates. After that, the selected iterations should be further analyzed with the VA framework.

5.1 Metric for Time Series

To evaluate the performance of a GAN model, it is crucial for an ML expert to have a trackable metric over the iterations. In spite of the significant number of the recently proposed metrics 2.1.4, they have not been compared to each other against key criteria such as efficiency, discriminability, and their ability in detecting common training problems such as overfitting, mode collapse, or mode dropping. Hence, it is impossible to identify the most efficient and reliable evaluation metric and to determine the advantages and disadvantages of each method. To deal with these issues and to rigorously inspect these metrics, we perform different evaluation tests proposed in 2.1.4. Furthermore, we propose a new metric, MiVo, that performs a bidirectional check starting from real to synthetic and afterward from synthetic to real to enable a precise and model-agnostic

evaluation. Finally, we will demonstrate through the conducted tests that our new metric allows for simultaneous detection of numerous training problems and is more efficient in terms of memory and time consumption.

The matrix $\mathbf{D} = (d_{ij})$ with $1 \leq i \leq n$ and $1 \leq j \leq m$ denotes the distances between the time series of the sets S_r and S_g:

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \dots & d_{mn} \end{bmatrix}. \tag{5.1}$$

We define the vectors D_1 of Incoming Nearest Neighbors Distances (INND) and D_2 Outgoing Nearest Neighbors Distances (ONND) as follows:

$$\begin{aligned} D_1 = [&min(d_{11}, d_{12}, \dots, d_{1n}), min(d_{21}, d_{22}, \dots, \\ &d_{2n}), \dots, min(d_{m1}, d_{m2}, \dots, d_{mn})], \end{aligned} \tag{5.2}$$

is the set of minimal distances over the rows of $\mathbf{D}$, i.e., the minimal distances for all generated time series of S_g. A good GAN model will be able to generate realistic data, i.e., we ideally expect that each generated time series achieves a low minimal distance to at least one real time series
and

$$\begin{aligned} D_2 = [&min(d_{11}, d_{21}, \dots, d_{m1}), min(d_{12}, d_{22}, \dots, \\ &d_{m2}), \dots, min(d_{1n}, d_{2n}, \dots, d_{mn})], \end{aligned} \tag{5.3}$$

is the set of the minimal distances over the columns of $\mathbf{D}$, i.e., the minimal distances for all real time series of S_r. A good GAN model should generate diverse samples, i.e., for each real time series we should be able to find a generated partner. Hence, the variance of D_2 should be low.

Training GAN models aim to generate new samples that are realistic and that reflect the diversity of the original dataset. To this end, a good metric should consider these criteria and detect common GAN training problems such as mode collapse. We design a metric, MiVo that focuses on the incoming and outgoing minimal distances and ensures at the same time that the synthetic time series are realistic (a low mean in S_g) and diverse (a low variance in S_r). We propose a metric that combines D_1 and D_2 as follows:

$$\rho = \mu_1 + \sigma_2^2, \tag{5.4}$$

where μ_1 denotes the mean of D_1 and σ_2^2 denotes the variance of D_2. There exists a wide variety of methods to measure the similarity between two time series [107]. To obtain the distance matrix $\mathbf{D}$, we recommend choosing the distance measure that fits better to the considered time series and their characteristics. In all the cases, the novelty in MiVo relies on the bidirectional check performed between the real and generated time series. In the rest of this work and for sake of simplicity, we use $m = n$ and for our data, we use the Euclidean Distance (ED) as a distance measure.

To use MiVo to assess the quality of translated time series, we compute the **D** between a set S_t of translated time series and a set S_{gt} of time series depicting the target behavior. The calculation of MiVo based on **D** remains unchanged.

In this chapter, we will prove that MiVo outperforms the existing GAN metrics described in section 2.1.4. In spite of its major advantages and its good performance, in many situations, it is essential to have a VA system that allows a rigorous investigation and exploration of the generated data and their comparison to the real data in an efficient manner.

5.2 Visual Analytics for Time Series

We propose a human-centered VA approach to support a ML or domain expert in the quality assessment task of generated time series data. The developed framework presents a method that makes the real and generated data easily comparable by combining VA (Colorfield, TimeHistograms) with algorithmic methods and enables the ML expert to trust the trained GAN model. The VA framework consists of two views namely, a GAN Iteration View and a Detailed Comparative View, that support ML and domain experts to assess the quality of time series data generated with GAN by providing:

- An overview visualization that helps the analyst identify interesting iterations of the GAN generation process.
- A comparison interface where the time series are visualized in a compact manner and ordered using PCA [52] to facilitate comparison by juxtaposition.

5.2.1 Design of Evaluation Framework

In this section, we describe our VA framework, which supports ML experts in generating time series data based on a given number of real data. For the sake of simplicity, we consider univariate time series of equal length. In this context, we present a workflow that can be used to evaluate the performance of GAN and assists the ML expert with the data generation. This VA framework is the result of an ongoing collaboration between ML and VA experts. The feedback of practitioners that use GAN to generate time series helped us in the framework's design. Our approach addresses some of the main issues with GAN training that are frequently encountered by ML experts. The VA system fulfills the following design goals:

Goal 1 Find iterations where appropriate behavior is achieved, i.e., the iterations showing a sufficient quality of the generated data. Check if the number of iterations is sufficient or if a higher number of iterations is needed.

Goal 2 Compare the performance of different GAN models with different sets of parameters and support the ML expert in the decision-making process to either trust or reject the current GAN model. Hence, the ML expert should be able to identify which GAN model and subsequently, which set of parameters is better.

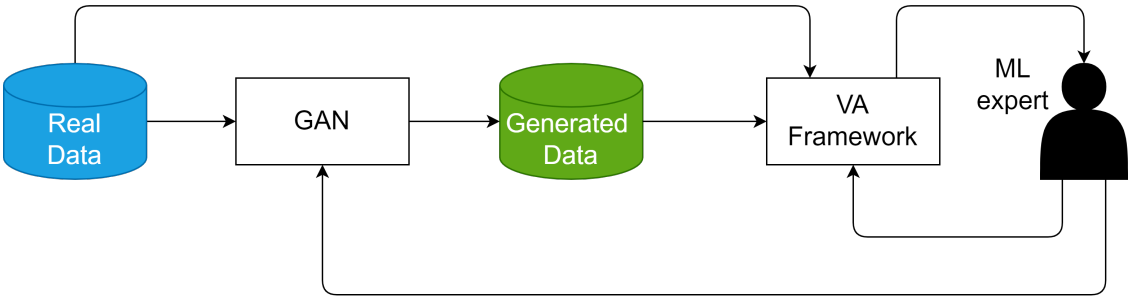

Figure 5.1: A visual evaluation workflow of GAN for time series data: The real and the generated data are integrated in the VA framework. The ML expert may interact with the VA framework to get more insight into the data and their properties. After a rigorous exploration of the data, he or she can decide to terminate the training process if the desired behavior is achieved. Otherwise, he or she has to run the GAN model with different parameters.

Goal 3 Present an adequate method to visually evaluate the quality of the generated data, i.e., detect if the data are noisy or show different behavior compared to the real time series data. ML experts should be able to decide whether the time series generated by a GAN algorithm are realistic.

Goal 4 Detect common GAN training problems such as non-convergence or mode collapse. Mode collapse is an important issue that a ML expert may encounter during training. In this case, the generator collapses to one mode and is not able to produce diverse samples. Almost all proposed GAN models [1, 44] suffer from this issue. Our purpose is to offer the user the possibility to easily identify this phenomenon. Once the problem is detected, the ML expert can use existing techniques [5] to improve the performance of the considered GAN model.

We design our framework based on these criteria. The ML expert starts the process by generating time series with GAN. The VA framework is then used to check whether the GAN model and the generated data fulfill the desired requirements. If this is the case, the ML expert has succeeded to generate realistic data and can stop the generation process. Otherwise, he or she will have to rerun the GAN model with different parameters and repeat the investigations in the framework. It should be noted that an online evaluation is also possible, i.e., the framework can be used during the training process. As the training process can take up to several days, our approach may help to save valuable time by making sure during the training that the GAN model is going in the right direction or restart the training process if unexpected behavior is detected. The proposed workflow is depicted in Fig. 5.1.

5.2.2 Evaluation Framework Description

Our proposed approach is characterized by two views: a GAN Iteration View that gives the user a general impression about the behavior of GAN over the iterations of the

generation process and a Detailed Comparative View equipped with TimeHistograms [108], Colorfields [109], and line plots to further investigate particular time series selected by the user. The TimeHistogram displays the time-dependent distribution of all time series at a certain iteration. At the same time, the Colorfield visualization allows further investigation and exploration of a multitude of generated time series at a certain iteration and compares them to the real time series. Moreover, a direct comparison between specific time series is made possible using the line plots visualization. To get more insights into the properties of the data, a measure of similarity and a dimensionality reduction technique are used:

1. Similarity measures such as ED or Dynamic Time Warping (DTW) [110, 111] are used as the pairwise distance between two time series.
2. PCA [52] is used to arrange similar time series close to each other and facilitate comparison.

The view consists of two components, namely the incoming and outgoing nearest neighbor distances (see Fig. 5.2a). The user can choose between ED and DTW as a distance measure. The evolution of the INND throughout the iterations is shown in this case. We repeat the same procedure calculating the minima of each real time series to all generated time series over the iterations. This corresponds to the ONND. A PCA is applied to the real time series data to transform the data points of each time series into uncorrelated components. The real data are then sorted based on the first principal component. To make both the real and generated data comparable, the same transformation is applied to the generated data. A heatmap visualization is used to depict the incoming and outgoing minimal distances. The intensity of the color of each pixel highlights the value of the minimal distance. A dark pixel represents a high distance value, while a brighter pixel denotes a lower distance value. The nearest neighbor distances give an overview of the overall performance of GAN over the iterations and allow for different types of investigations:

- Are the time series becoming more realistic with the iterations, i.e., do the INND/ ONND become smaller?
- Are INND / ONND reaching a stable behavior and indicating nearly constant values?
- Is the variation in the real data representative for the generated data i.e., are all types of generated time series equally similar to the real data (INND) and are all real time series equally well represented by the different generated time series, or do generated time series correspond to a limited number of real ones (ONND)?

The user can interactively select interesting iterations in the GAN Iteration View and get more insights about the selected iterations in the other view. This will permit him to identify the iteration with the best behavior.

This view is equipped with TimeHistograms [108] depicting the distribution of the real and the generated data, a Colorfield visualization as a compact representation of

the corresponding time series and a Selected Samples View allowing comparison by superposition (see Fig. 5.2b). In the Colorfield and TimeHistogram Views, the real data are shown on the left and selected iterations of the generated data are shown on the right. This setup enables a comparison by juxtaposition between the real and the generated data as well as between different iterations of the generation process. The user can investigate different iterations at the same time. Both real and generated data are automatically sorted for each iteration step based on the first principal component. The TimeHistograms enable a time-dependent investigation of the distribution of the data and a comparison between the distribution of the real and the generated data. This visualization represents a possibility to check if the model is working properly and capturing the distribution of the real data. The Colorfield visualization is used to depict the time series and enables a rigorous exploration of the generated time series and their properties. Each heatmap represents all the data of a specific iteration where each row corresponds to a time series. This visualization permits the user to compare a high number of time series in an efficient manner. Additionally, a rigorous investigation of some selected time series is made possible with the Selected Samples View equipped with two visualizations. To give the ML expert more insights into the real data, the first plot depicts their median $med(r)$ and the amount of data falling in the $68th$, $95th$, and $99.7th$ percentile denoted with $68prct$, $95prct$, and $99prct$ respectively. The user may add interesting, real, or generated, time series to the plot to investigate their properties and compare them by superposition. Each generated and real time series is denoted with g_iter_id and r_id respectively where $iter$ is the number of the iteration at which the time series was generated and id is the index of the time series sorted with PCA. The second plot highlights the absolute value of the element-wise difference between the selected time series r/g_i and the median of the set of real data $med(r)$. This feature provides additional information about the selected data by directly comparing their behavior to a reference value, namely the median of the real data. Our main concern is to enable an exploration of the behavior of the ML model over the iterations and an investigation of the similarity between the real and the generated data. Hence, the presented human-centered approach gives the opportunity to build a relationship of trust between the ML expert and the AI algorithm.

5.2.3 Use Case

To demonstrate the utility of the developed framework, our ML expert tested the proposed method on a GAN model [9] to generate data based on the real TwoLeadECG dataset [74] presented in section 3.3.1. To reduce the training time, only 30 time points from the real time series are considered. The ML expert used one class in his experiments. In our case, the performance of GAN is evaluated for two different parameter configurations, namely model 1 and model 2. The corresponding results are depicted in Fig. 5.2 and 5.3.

The GAN Iteration View (Figs. 5.2a and 5.3a) depicts the variation of INND and ONND depending on the iterations. The first iterations are characterized by high INND and ONND. As the number of iterations increases, an improvement in terms of INND

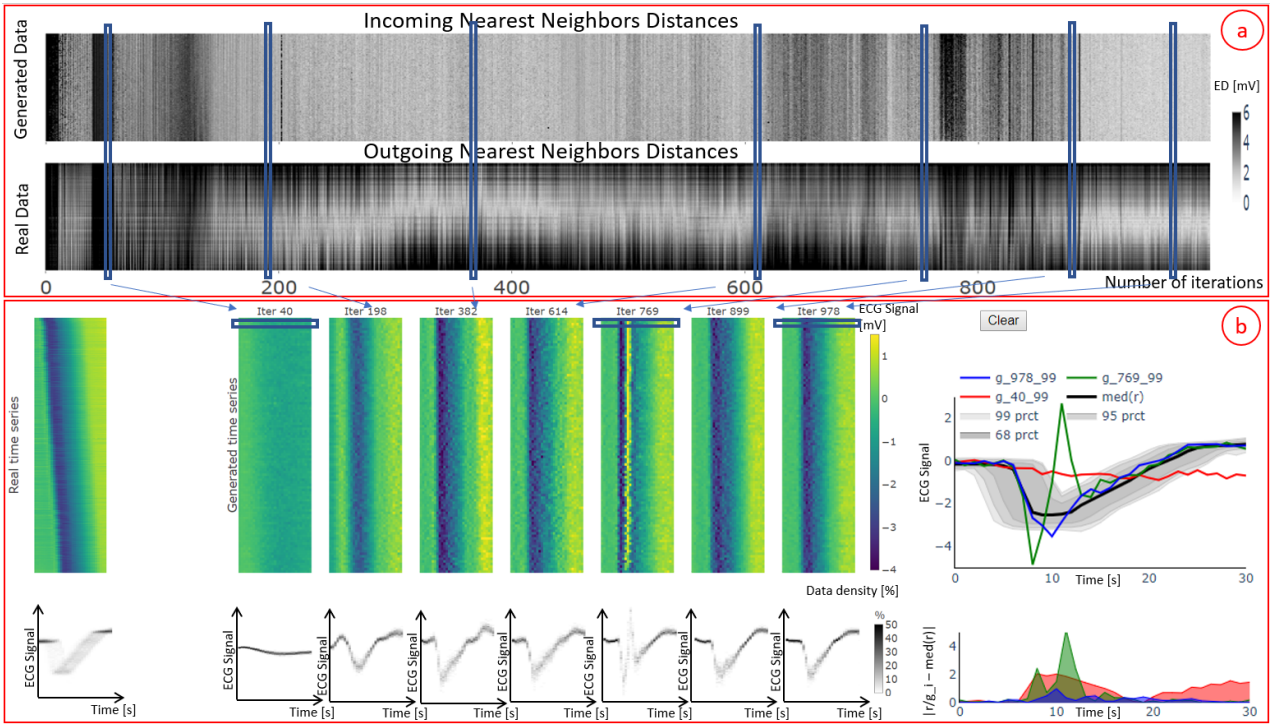

Figure 5.2: Results of a first GAN model generating time series. The computed incoming and outgoing minimal distances namely INND and ONND are integrated in the GAN Iteration View (a). Selected columns in the GAN Iteration View, denoted with blue rectangles, are depicted in the Detailed Comparative View (b).

can be seen. Hence, the generated data are progressively reaching similar values to the original data and the performance of the ML algorithm is increasing with a growing number of iterations. However, the ED values corresponding to some iterations in model 1 sharply increase. Model 2 is showing a more stable behavior. In fact, after approximately 300 iterations, the INND are almost constant. ONND in Fig. 5.2a show that the values of the ED at the top and bottom of the view are still high. As the real time series are sorted with PCA, our expert concludes that the real time series with an important shift are characterized by a high outgoing minimal distance. Hence, the time series produced by the first model are similar to a specific type of the real time series namely the time series that are in the middle. He hypothesizes that this GAN model was not able to reproduce the shift present in the real data and is collapsing to one mode. In contrast to model 1, ONND illustrated in Fig. 5.3a depict a low outgoing ED for all real data. ONND helped the ML expert to verify that the generated data are diverse and do not correspond to a specific type of time series but to almost all real ones.

Afterward, the ML expert selects some interesting columns in the GAN Iteration View and continues his investigation in the other view. For both scenarios, the user selected an iteration at the beginning of the training process, certain columns with low EDs in the middle, a few columns characterized by high INND and ONND in model 1 and 2 and some columns showing a stable behavior within the last hundred iterations. Initially, the time-dependent distribution of the generated data was completely different from the real data and noise was generated. An improvement in the performance is noticeable after approximately 200 iterations. In general, the time-dependent distribution and the quality of the generated data are becoming more realistic over the iterations. An enhancement in the results is observed between the iterations 382, 614 and 899 for model 1 and the iterations 386 and 669 for model 2. To rigorously inspect the behavior of model 1, the user selected some time series generated at different iterations. In the Selected Samples View, he noticed that at iteration 764 the generated data present a strange peak and at iteration 40 noise is generated. Hence, the VA framework helped the ML expert to detect if the data are noisy or have different behavior from the real data (Goal 3).

To conclude, GAN was not able to generate realistic time series in the first iterations at all and is learning the properties and features of the real time series over time. However, the data quality can decrease drastically after one iteration, i.e., iterations 769 and 480 in the first and second scenario respectively. The ML expert confirms that this is expected behavior with neural networks because their performance is not monotonic. An inspection of the last hundred iterations allows the ML expert to find an iteration with the best result (Goal 1). This corresponds to iteration 978 for model 1 and iteration 926 for model 2. In both cases, the generated data are smooth and realistic. However, the TimeHistogram of the data generated with model 1 is still different from the TimeHistogram of the real data. Moreover, the Colorfield View demonstrates that the samples are not as diverse as in the real data. A rigorous investigation of these time series in the Selected Samples View shows that all the generated data are falling in the 68th percentile of the real data and are too close to the median, i.e., their difference to the median of the real data is low. This confirms the hypothesis of the ML expert

when he observed the GAN Iteration View. Thus, the user was able to easily detect the mode collapse phenomenon, one of the hardest training problems for GAN (Goal 4). In order to avoid this problem, the ML expert used in model 2 a normally distributed noise instead of the uniformly distributed noise and applied a technique introduced in [5] namely mini-batch discrimination. In contrast to model 1, we clearly see that model 2 is reproducing the distribution of the real data much better. For further exploration, the ML expert selected different time series from the Colorfield View and visualize them in the Selected Samples View. He noticed that model 2 is reproducing the shift present in the real data. This model is generating time series that are moved to the right and to the left and are characterized, at some data points, with a high difference from the median. As the last step, our expert used the Selected Samples View to directly compare the generated and real data. Fig. 5.4 shows a real and a generated time series selected by the ML expert. We clearly see that the behavior of the generated time series is similar to the behavior of the real data. Hence, the second GAN model presents a more realistic behavior and was able at iteration 926 to generate time series that are rare in the real dataset. The ML expert concludes that model 2 is achieving the desired behavior. Hence, the proposed framework helped the ML experts to find a trustworthy GAN model with a set of parameters producing the best results (Goal 2).

Finally, the ML expert said that it was helpful to see the evolution of the behavior of GAN over the iterations and how the similarity between the real and the generated data improved with the number of iterations. He was able to assess the quality of the generated data and find a reliable GAN model achieving trustworthy results.

In this part, we proposed a visual approach to evaluate and optimize GAN models generating time series data. The proposed method is based on two visualization techniques namely Colorfield and TimeHistogram as well as a distance measure. The distance measure is used in a sophisticated manner to compute the incoming and outgoing nearest neighbor distances. The VA system supports ML experts in the evaluation process. The utility of the developed framework is demonstrated with a real-world use case where the ED is used as a distance measure. In this case, a ML expert evaluated the performance of two different GAN models in generating time series based on existing real ones. He was able to detect that the first GAN model generates samples that are not diverse.

5.3 Experiments

In this section, we focus on testing the existing GAN evaluation metrics for time series data namely: MMD, TSTR, TRTS, PS, DS, and MiVo on the datasets described in section 3.3.1. Our main goal is to evaluate their performance against the different evaluation criteria described in section 2.1.4. We perform the tests of section 2.1.4 in order to identify the metric that allows for efficient detection of common GAN training problems and a robust discriminability between real and synthetic time series. We set for the test of mixing, mode dropping and collapse, and overfitting, $n = 100$ and $c = 30$. For the efficiency test, the absolute number of samples used for evaluation ranges from 10 to 1000 samples in steps of 50 samples more each step. We repeat each test 10 times

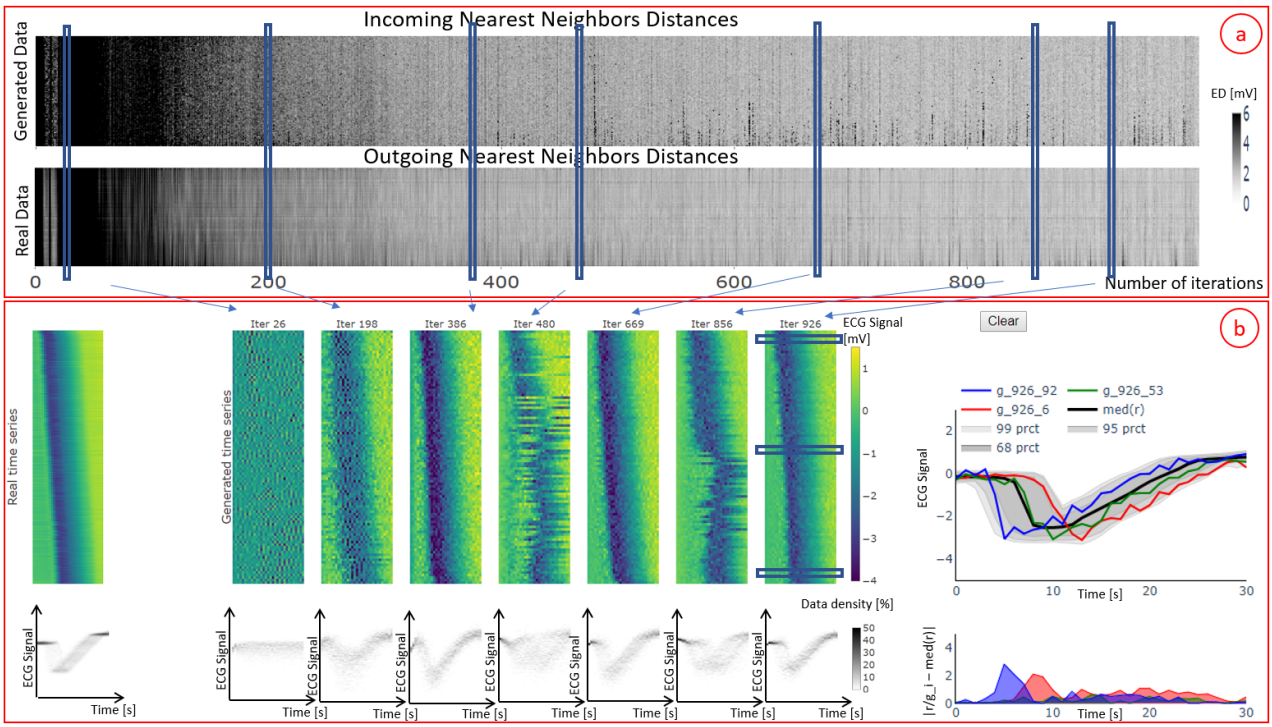

Figure 5.3: Results of a second GAN model obtained by tuning the parameters of model 1 depicted in Fig. 5.2. In comparison to model 1, this model is showing a more stable and smooth behavior in terms of INND and ONND. The Colorfields, depicted in the Detailed Comparative View (b), indicate that the last iteration is reproducing the shift present in the real data and its TimeHistogram is similar to the TimeHistogram of the real data.

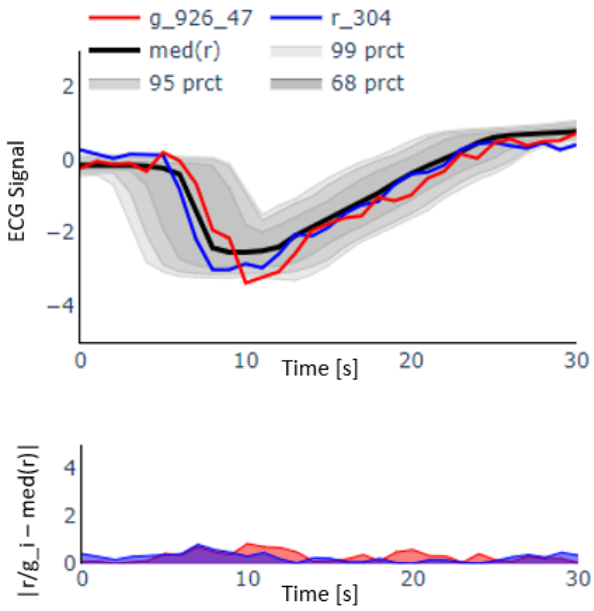

Figure 5.4: Illustration of the Selected Samples View with the median of the real data $med(r)$, $68th$, $95th$ and $99th$ percentile denoted with $68prct$, $95prct$ and $99prct$ respectively, time series g_926_47 generated at iteration 926 by model 2 and a real time series r_304. The absolute value of the element-wise differences of g_926_47 and r_304 to the median $med(r)$ are denoted in red and blue respectively. The time series g_926_47 is falling in the $98th$ percentile of the real data and g_926_47 and r_304 are showing a similar behavior.

and compute the mean over the 10 experiments. In all the experiments, Random Forest [76] is used as the machine learning algorithm for TSTR and TRTS. Since the range of scores of each metric are very different in scale, we normalize the scores achieved by each metric between its minimum and maximum to a range between 0 and 1 to make them directly comparable. TSTR and TRTS compute the test accuracy. To avoid confusion and to make them comparable to the other metrics, we illustrate in all the figures $1 - \rho$ for the metrics TSTR and TRTS. All experiments are conducted on the t2.large AWS EC2 instances with 8 GiB of system memory and 2 vCPUs.

In the next step, we evaluate the results of the same GAN model in two different manners: computationally using MiVo and visually using the VA framework. Our main goal is to compare both approaches and to show the advantages and disadvantages of each method. To this end, the outcome of 1000 iterations is depicted and assessed. For the different considered datasets, we focus on the output of the RCGAN model.

Finally, the MiVo metric is applied in a special fashion to assess the quality of each translation use case presented in chapter 4. We compute the MiVo values between the obtained data after the translation and time series depicting the desired result. A lower MiVo value highlights a better similarity between the obtained and target time series and better performance of the translation process.

We perform the tests described in 2.1.4 to assess the existing time series generation evaluation metrics and to compare them to MiVo.

Mixing A good evaluation metric should be sensitive to the mixing test, i.e., its curve should steadily increase with the ratio of fake samples. A comparison of the different metrics is shown in Fig. 5.5 and Fig. A.5. For the different datasets, the score MiVo increases progressively with the ratio of fake samples in $S_g(l)$ as expected for a metric sensitive to mixing. While DS shows the expected behavior for all the datasets except the ItalyPowerDemand, PS achieves good performance on the TwoLeadECG and FreezerRegularTrain datasets. It is to be noted that MiVo was more sensitive to the ratio of fake samples and showed a more stable behavior than MMD, whereas TRTS and TSTR showed the most unstable behaviors indicating little sensitivity to identify fake samples. To sum up, the best performance is achieved by MiVo closely followed by MMD.

Mode collapse A good evaluation metric should be sensitive to the mode collapse test, i.e., its curve should increase with the ratio of collapsed clusters. The results of the mode collapse test are depicted in Fig. 5.6 and Fig. A.6. PS - except for TwoLeadECG and DistalPhalanxTW -, MMD -except for DistalPhalanxTW - and MiVo show the expected increase of score with the number of collapsed clusters. It is to be note that MMD barely reacts to the effect of mode collapse before 85% of the modes have collapsed. Large variations are present in DS and TRTS. The TRTS score showed the best behavior on the FreezerRegularTrain and DistalPhalanxTW datasets was almost constant for the TwoLeadECG dataset and a fluctuating behavior for the Yoga dataset. The performance of the TSTR shows the expected behavior on the FreezerRegularTrain and DistalPhalanxTW datasets but a worse behavior on the TwoLeadECG dataset characterized by a low reactivity to a ratio of collapsed clusters higher than 0.7 and an unstable

behavior on the Yoga dataset. For the mode collapse test, MiVo and PS show the best performance.

Mode dropping A good evaluation metric should be sensitive to the mode dropping test, i.e., its curve should increase with the ratio of dropped clusters. The results of the simulated mode dropping test are depicted in Fig. 5.7 and Fig. A.7. MiVo and DS are the most sensitive to this phenomenon. However, MiVo showed higher reactivity to the ratio of dropped clusters, especially for the Yoga and FreezerRegularTrain datasets. MMD - except for the yoga and DistalPhalanxTW datasets - was nearly insensitive to the ratio of dropped clusters smaller than 85%. PS, TSTR, and TRTS show noisy behavior. For this test, MiVo outperforms the remaining methods and shows the best behavior closely followed by DS.

Overfitting A good evaluation metric should be able to detect overfitting by increasing its score as the overlapping fraction increases. The effect of artificially simulating overfitting on the different metrics is shown in Fig. 5.8 and Fig. A.8. TRTS, TSTR, and MiVo were able to efficiently detect the overfitting phenomenon, i.e., they are characterized by a growing value for a growing overlapping fraction. On the other side, MMD, PS, and DS have trouble to detect overfitting. They are showing a noisy and unspecific behavior. The best performance is achieved by TRTS and TSTR closely followed by MiVo.

Efficiency The efficiency test is composed of a sample efficiency test and a computational efficiency test.

Sample Efficiency: With reasonable number of samples, it should be easier for an evaluation metric ρ to distinguish between S_g and S_r, than between S_r and S_r'. Fig. 5.9 and Fig. A.9 highlight the values $\rho(S_r, S_r')$ over the number of samples (the upper plot) and the difference $\rho(S_g, S_r) - \rho(S_r, S_r')$ in the second plot. While MMD, closely followed by DS and MiVo stabilizes rather quickly to small scores after a small number of samples is reached, TRTS and TSTR continuously decrease their score when more samples are provided, showing their dependence on a larger sample set due to the need of training data. Although the PS scores are almost constant for a sample size higher than 500 for the TwoLeadECG and FreezerRegularTrain dataset, their values were unstable for the Yoga dataset. The value $\rho(S_g, S_r) - \rho(S_r, S_r')$ is greater than 0 for TSTR, TRTS, MiVo and DS independently of the number of used samples. At the same time, it is difficult for PS and MMD to distinguish between $\rho(S_g, S_r)$ and $\rho(S_r, S_r')$.

Computational Efficiency: The most computationally efficient methods were MMD and MiVo as expected since in this case no ML model is trained. PS and DS were the most inefficient in terms of computation time due to the 2-LSTM layer that needs to be trained. TRTS and TSTR are characterized by faster computation. It is to be noted that we used a RF to train TRTS and TSTR. A slower computation is expected if neural networks are used instead.

The results of the conducted tests are summarized in Table. 5.1:

MMD MMD succeeded in the mixing test. However, it is difficult for MMD to discriminate between real and generated data and between test and training samples (failure in sample efficiency tests and failure in the overfitting test). Also, the

reactivity of MMD to the mode collapse/dropping tests was slow. These results are related to the nature of MMD. MMD compares the distribution of the real and generated samples to each other and hence doesn't consider the properties of the observed time series.

TSTR & TRTS The conducted tests confirm the assumption made by Esteban, Hyland, and Rätsch [11] when introducing TSTR and TRTS: TSTR is more efficient than TRTS, i.e., training a classifier on a set of generated samples and testing it on a set of real may reveal more information about the generated samples. For example, a set of generated samples with a collapsed mode can be easily classified by a ML model trained on the real data. As TSTR and TRTS rely on a ML model, we expect that the number of samples (efficiency) may play an important role in the problem detection and that the use of neural networks may improve the performance of these methods.

PS This metric tries to predict the next coming timestamp for a time series based on the previous data points and hence focuses on the quality of the generated time series (success in the mixing test). Unfortunately, this metric fails in detecting mode dropping or overfitting. It can easily perform forecasting tasks on time series with dropped modes or on time series that correspond perfectly to specific parts of the training dataset (overfitting). We strongly recommend using this metric to detect some undesirable features that may be present in the generated time series such as noise and to use additionally DS, MiVo or TSTR to detect the common GAN training problems.

DS The main goal behind DS is to discriminate between a set of real and generated samples. This explains why DS was successful in discriminating tests such as mixing and mode dropping. Our tests prove that DS is not successful in the overfitting test. We hypothesize that DS fails to recognize that some parts of the training dataset are completely ignored when the generated time series corresponds perfectly to some parts of the training dataset. A major drawback of DS and PS is the computational inefficiency as it requires training a 2-layer LSTM.

MiVo The tests show that the bidirectional check proposed in MiVo can reveal a lot of properties about the generated samples: it can detect if modes are dropped or averaged if overfitting occurs and can help to discriminate between the real and the generated samples. It is to be noted that the behavior of MiVo was monotonic in almost all tests, which shows a high sensitivity to the most commonly occurring training problems and is computationally very efficient due to the simple construction of the method.

We have previously proven that MiVo is the most efficient and accurate metric for the time series generation task. In the following, we apply MiVo to the same five datasets and evaluate the results over the iterations. Fig. 5.10, 5.11, and 5.12 show the results for the TwoLeadECG, Yoga, and FreezerRegularTrain respectively. For most of the datasets, we clearly see that the highest MiVo values were noticed for the first iterations and that

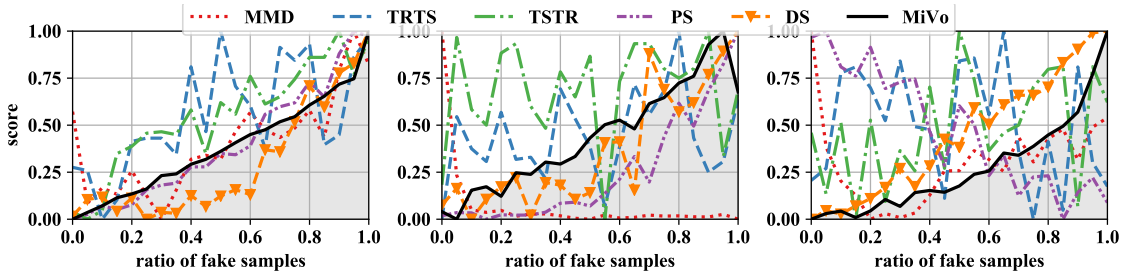

Figure 5.5: Results of mixing test ratio l of fake samples in $S_g(l)$ is augmented progressively for the TwoLeadECG (plot on the left), FreezerRegularTrain (plot in the middle) and Yoga (plot on the right) datasets. The score of the different metrics is computed. We expect that the score of each metric increases as the ratio l increases, i.e., a reliable metric should be able to discriminate between real and generated samples its best score should be achieved on real samples.

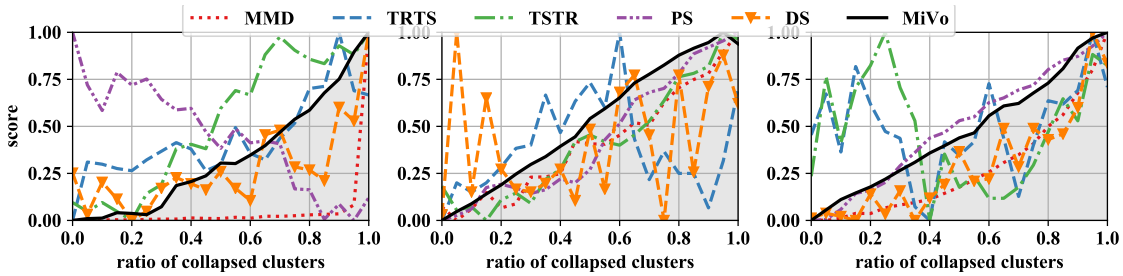

Figure 5.6: Results of mode collapse test for the TwoLeadECG (plot on the left), FreezerRegularTrain (plot in the middle) and Yoga (plot on the right) datasets: the metric score is computed depending on the number of collapsed clusters. A good evaluation metric should be able to detect mode collapse, i.e., its score should increase with the ratio of collapsed clusters.

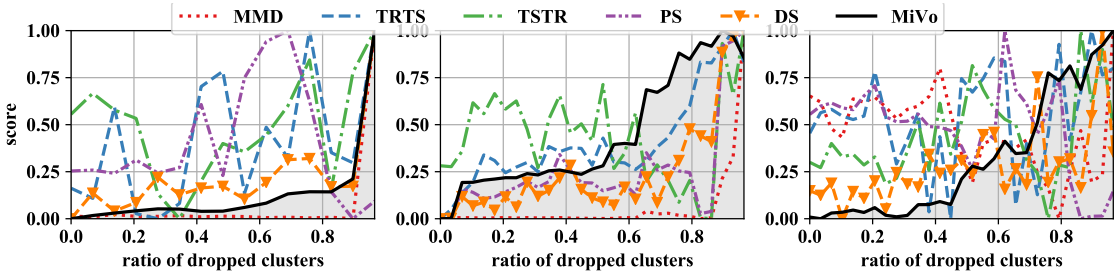

Figure 5.7: Results of mode dropping test for the TwoLeadECG (plot on the left), FreezerRegularTrain (plot in the middle) and Yoga (plot on the right) datasets: the metric score is computed depending on the number of dropped clusters. A good evaluation metric should be able to detect mode dropping, i.e., its score should increase with the ratio of dropped clusters.

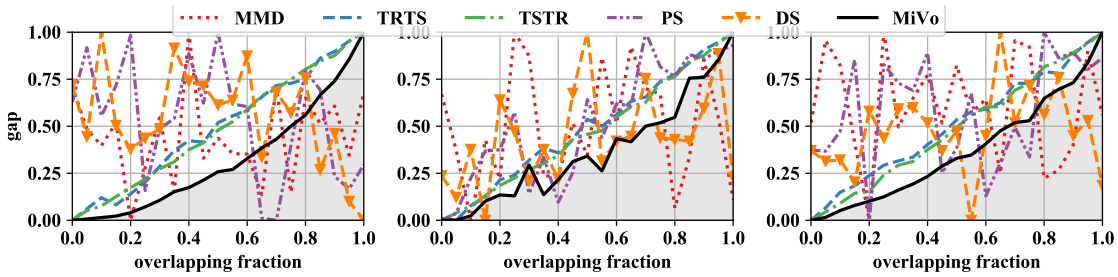

Figure 5.8: Results of the overfitting test for the TwoLeadECG (plot on the left), FreezerRegularTrain (plot in the middle) and Yoga (plot on the right) datasets: the metric score is computed depending on the overlapping fraction between a set S_r'' of real samples and a set S_r^{tr}. An accurate metric should increase its score as more samples of S_r'' are becoming similar to the training set in order to highlight the overfitting phenomenon.

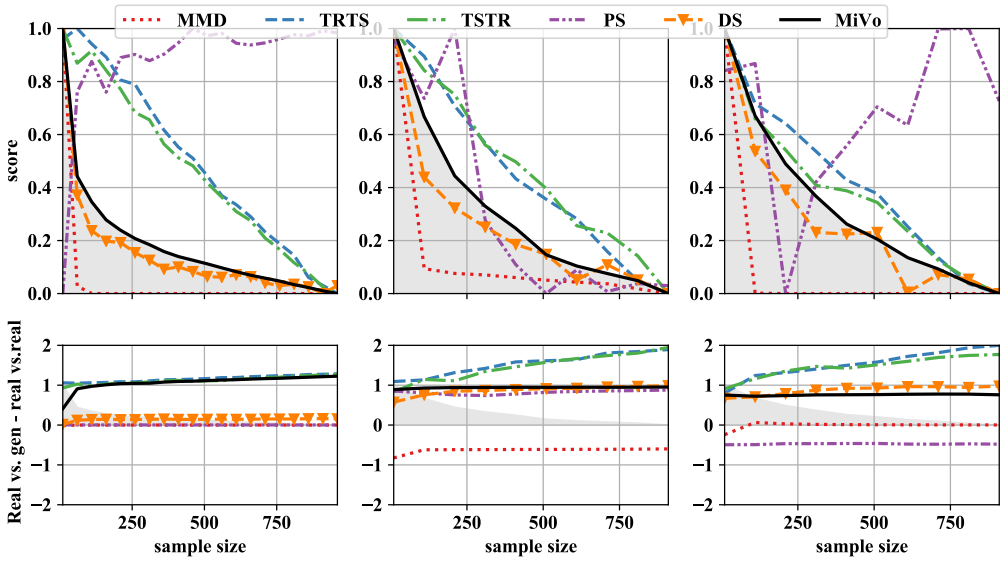

Figure 5.9: Results of the efficiency test for the TwoLeadECG (plot on the left), FreezerRegularTrain (plot in the middle) and Yoga (plot on the right) datasets: the values of $\rho(S_r, S_r')$ are computed with different number of samples (upper plot). The second plot depicts the value of $\rho(S_g, S_r) - \rho(S_r', S_r)$. This difference should be positive. For a reasonable number of samples an accurate metric should score $\rho(S_g, S_r)$ higher than $\rho(S_r', S_r)$.

Table 5.1: Comparison of the different GAN metrics

Tests	Metrics						Figure	Expected behavior
	MMD	TRTS	TSTR	PS	DS	MiVo		
Mixing	+	- -	-	-	-	++	5.5	scores increase with ratio of fake samples
Mode Collapse	-	-	+	+	-	++	5.6	scores increase with ratio of collapsed clusters
Mode Dropping	-	- -	- -	-	+	++	5.7	scores increase with ratio of dropped clusters
Overfitting	-	++	++	-	-	++	5.8	scores increase with an increasing overlapping fraction
Sample Efficiency	+	+	+	-	+	++	5.9	scores stable with reasonable number of samples and $\rho(S_g, S_r) - \rho(S'_r, S_r)$ positive
Computational Efficiency	++	+	+	-	-	+		a high computational efficiency

the registered values are getting lower over the iterations. This shows that there is an improvement in the quality of the generated data over the iterations, i.e., the generated time series are getting closer to the real ones and for each real one, we can find a similar generated. It is to be noted that training and stabilizing GAN models can be complicated and tedious. Moreover, GAN models are known to have a non-monotonic behavior. Indeed, there is no guarantee that the performance of GAN will be stable over the iterations and have the expected behavior, i.e., bad performance at the first iterations, a better one over the iterations, and a stable and best performance for the last iterations. Hence, in many situations, the training procedure and the performance of the GAN model may be unstable. In such cases, such as for the Yoga and FreezerRegularTrain, MiVo can help to identify some iteration candidates with the best performance.

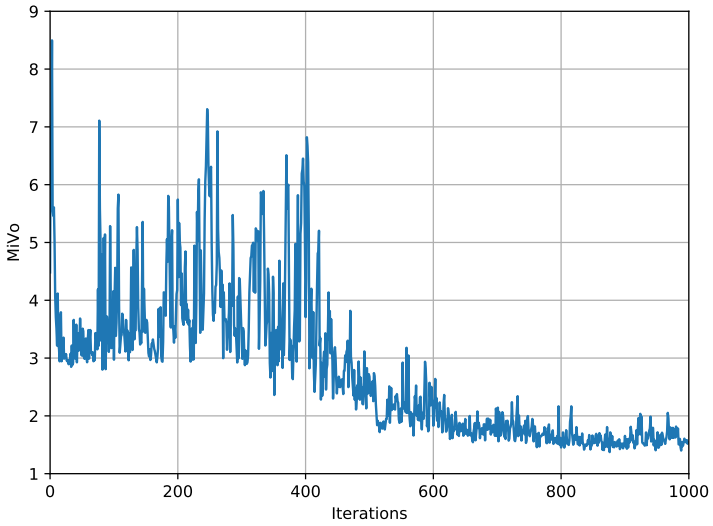

Figure 5.10: MiVo metric computed for the TwoLeadECG dataset over the training iterations. A lower MiVo value denotes a better GAN performance.

In the following, we use the VA framework to visually evaluate the quality of the generated time series for the different datasets. The results of the VA framework are visualized in Figs. 5.13, 5.14, and 5.15. For the TwoLeadECG, the INND and ONND are getting lower over the iterations. This shows that each generated time series is corresponding to at least a real one (high-fidelity) and that for each real time series there is a similar generated time series (high-diversity). In the Detailed Comparative View, we notice an evolution in the quality of the generated data after every 200 iterations. By way of example, the best behavior was noticed at iteration 800 for the TwoLeadECG. This fits perfectly with the output of the MiVo metric illustrated in Fig. 5.10. For Yoga and FreezerRegularTrain, the MiVo metric showed an unstable behavior. This was confirmed in the GAN Iteration View of the VA framework where the INND and ONND values were unstable. We conclude that during the training the model was diverging at some iterations. Furthermore, the sudden peaks in the MiVo values of the Yoga dataset

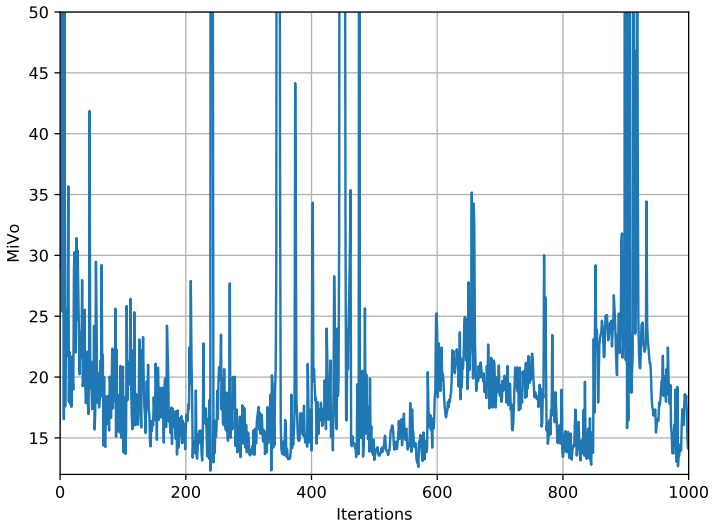

Figure 5.11: MiVo metric computed for the Yoga dataset over the training iterations. A lower MiVo value denotes a better GAN performance.

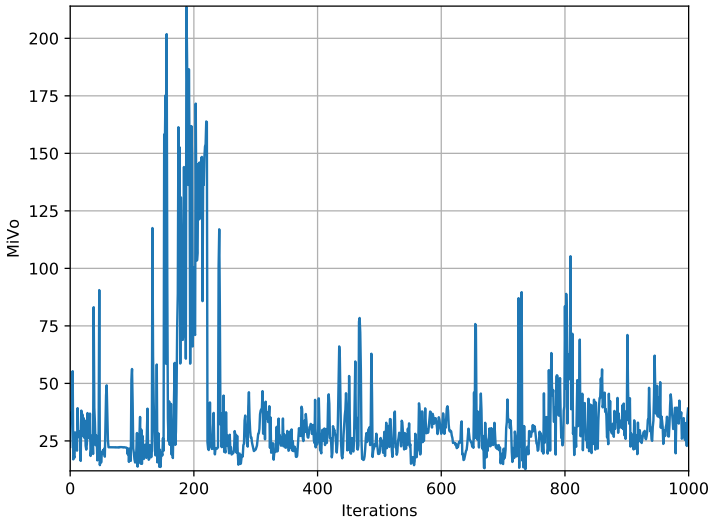

Figure 5.12: MiVo metric computed for the FreezerRegularTrain dataset over the training iterations. A lower MiVo value denotes a better GAN performance.

were also noticed in the GAN Iteration View. At the same time, the ONND values were lower than the INND. It was also interesting to see that even for the iterations characterized by the lowest MiVo values the generated data differ from the real ones. The FreezerRegularTrain dataset is characterized by high INND values. In the ONND View, we see a bright part corresponding to specific time series, i.e., the model is getting closer to a specific part of the real dataset and ignoring the rest of the data. A further investigation in the Detailed Comparative View shows outliers in the time series. Hence, the generated data differ from the real ones. We clearly see that for this dataset it was difficult for the model to achieve the expected behavior.

To conclude, the MiVo metric is really helpful to get a global overview of the GAN performance and to select some iterations with reasonable and low MiVo values but as next step, the selected iterations must be further investigated with the VA framework. We refer to Appendix A.2 for further figures highlighting the MiVo values and the output of the VA framework for the ItalyPowerDemand and DistalPhalanxTW.

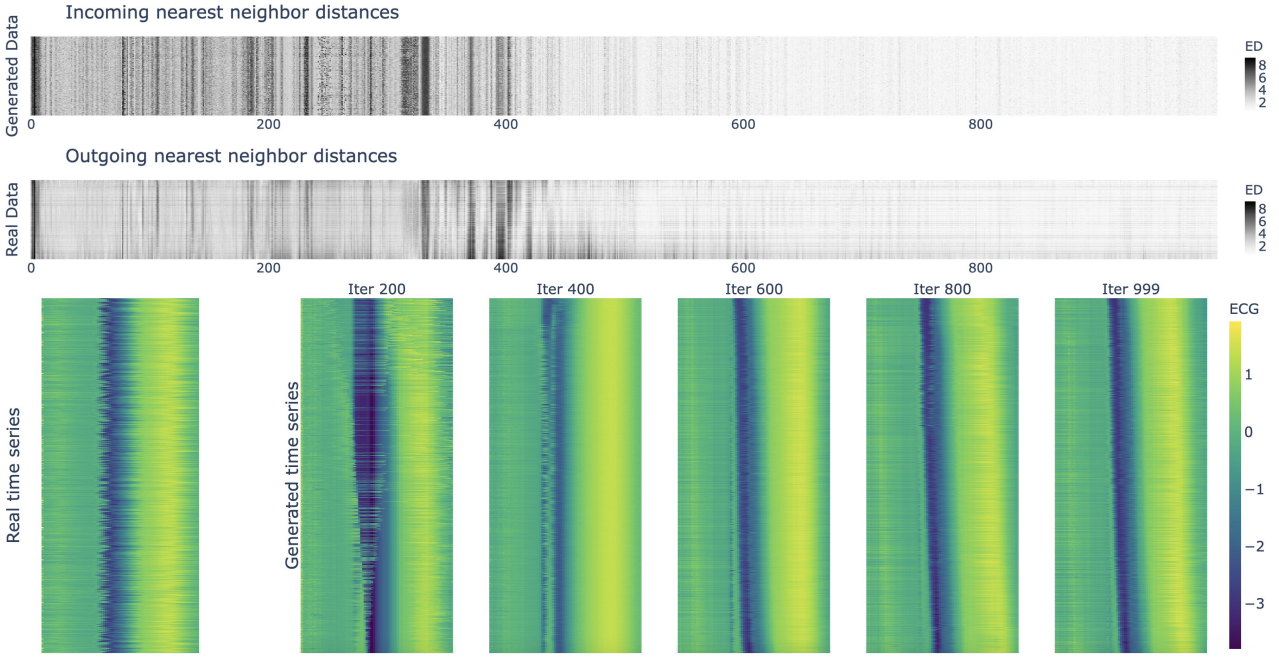

Figure 5.13: Results of the TwoLeadECG dataset illustrated in the VA framework

After that, we use MiVo in a sophisticated manner to evaluate the translated data and to find the iteration with the best performance. The obtained MiVo values for test 3, 5 and 8 are depicted in Fig. 5.16, Fig. 5.17, Fig. 5.18, Fig. 5.19, and Fig. 5.20. For test 3 (for both translation directions) and for test 8 (for the target direction), we notice, an improvement in the computed MiVo values, i.e., the MiVo values are at the beginning high and are getting lower over the iterations. This proves that DR-TiST is capturing the characteristics of both classes over the iterations and improving the quality of the

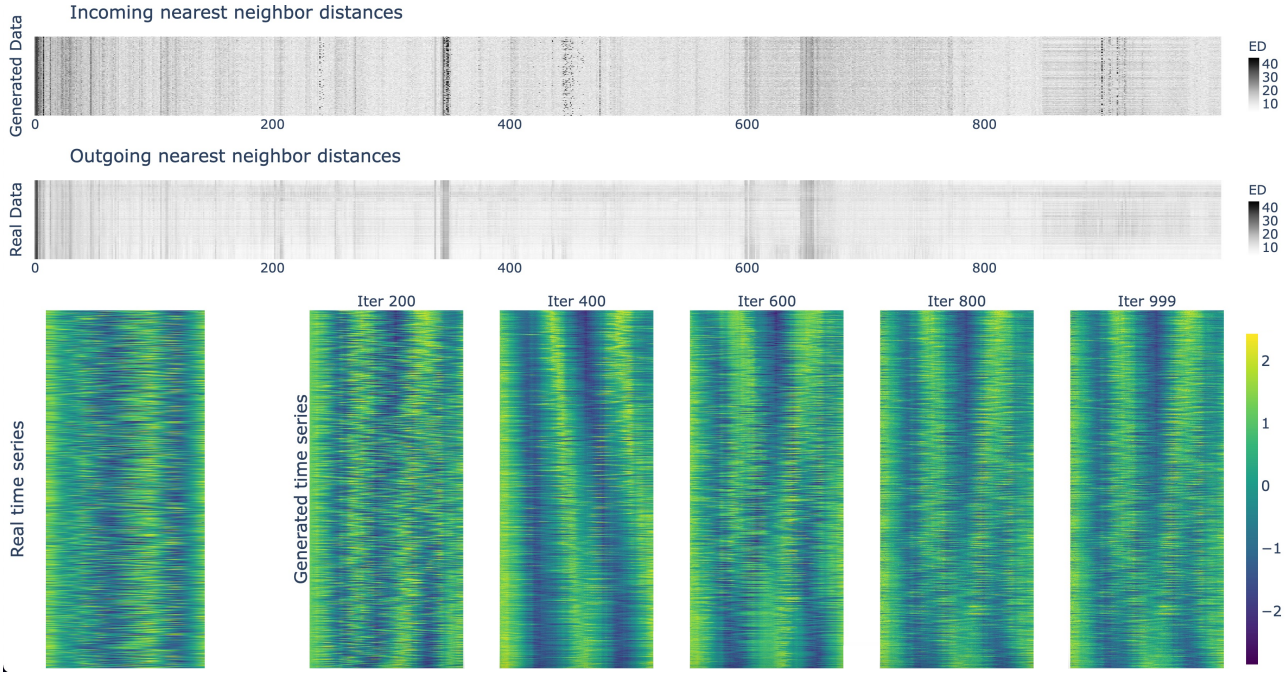

Figure 5.14: Results of the Yoga dataset illustrated in the VA framework

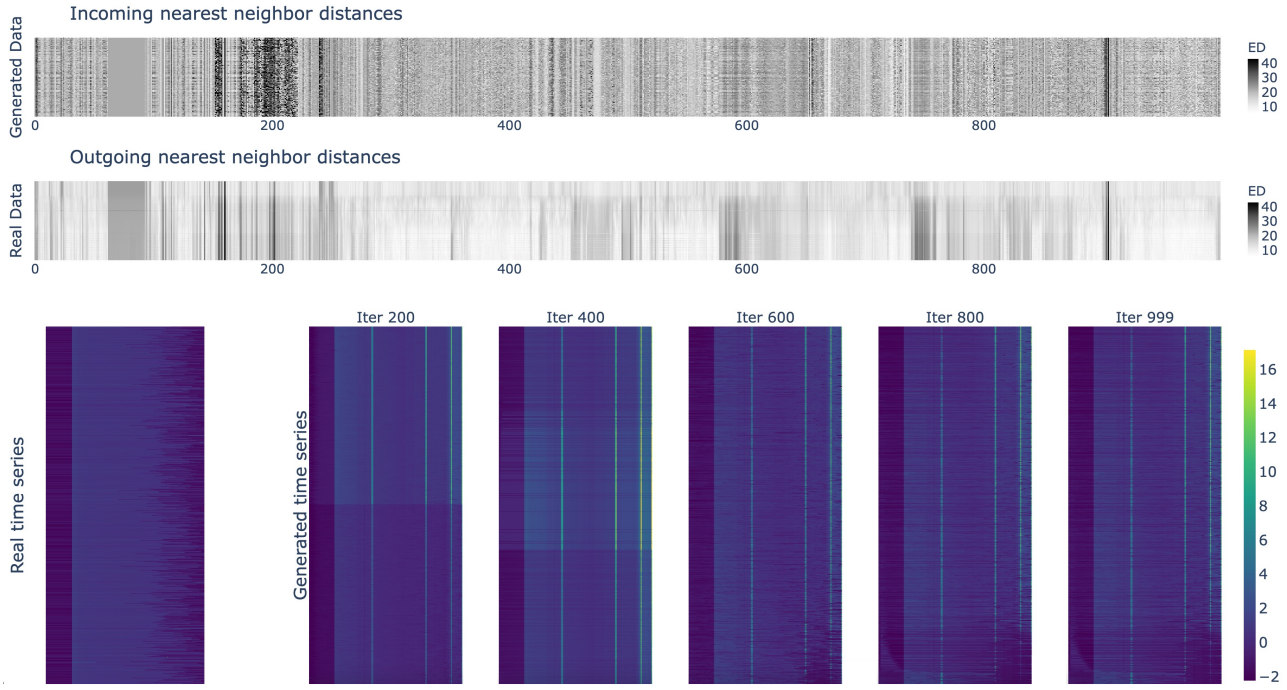

Figure 5.15: Results of the FreezerRegularTrain dataset illustrated in the VA framework

transformed time series so that these are getting closer to the expected behavior. In contrast to that, for test 5, the MiVo values were noisy and unstable. This proves that for DR-TiST, it was difficult to correctly map the time series to the target domain. In such cases, it is important to select iterations with the lowest MiVo values for both transformation directions and to further investigate the obtained and target time series visually. The MiVo computation results of the remaining translation tests of chapter 4 are illustrated in Appendix A.2.

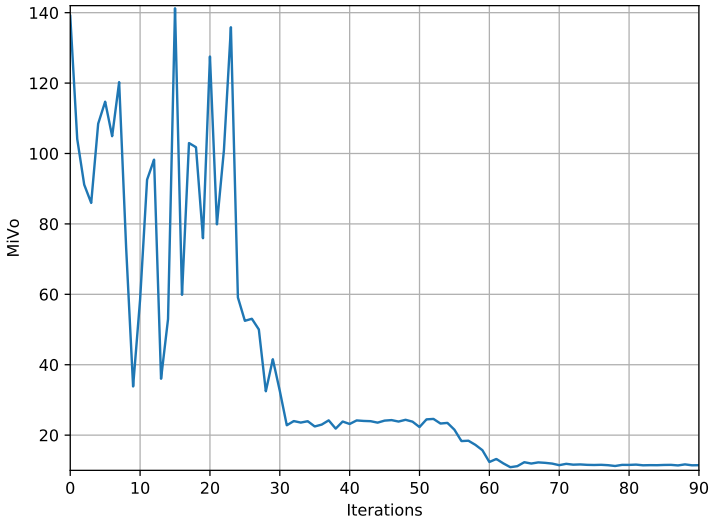

Figure 5.16: MiVo metric computed between target time series depicting the behavior of engine 1 in test 3 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of engine 1 obtained after the transformation and the expected behavior.

In this chapter, we introduced two analytical approaches a computational method, MiVo, and a visual framework to assess the quality of generated data. By means of numerous tests, we tried to compare the performance of MiVo to the existing GAN metrics in detecting common training problems. The conducted tests show that TSTR can be successfully used to detect common GAN training problems such as mode collapse/dropping and overfitting. PS can be considered as a similarity measure between the real and generated data. However, DS and PS are computationally inefficient. MMD was not really sensitive to the conducted tests as it doesn't consider the characteristics of time series. MiVo showed the best performance in all the conducted tests, except for mode dropping where DS showed slightly higher reactivity to the number of dropped modes. Furthermore, a human-centered VA approach is presented to enable in-depth investigation of the generated data and to guide ML experts to either trust or reject a used GAN model. MiVo and the VA approach were used to evaluate the time series generation task computationally and visually with the purpose of highlighting the advantages and disadvantages of each method. The presented approaches constitute a starting point to

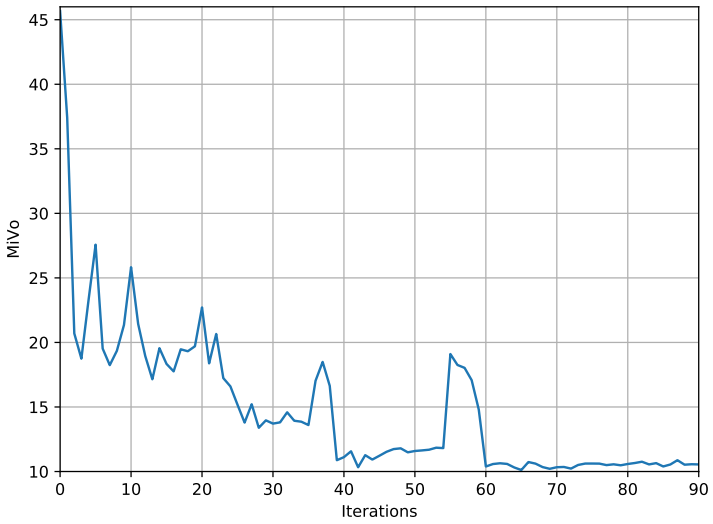

Figure 5.17: MiVo metric computed between target time series depicting the behavior of engine 2 in test 3 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of engine 2 obtained after the transformation and the expected behavior.

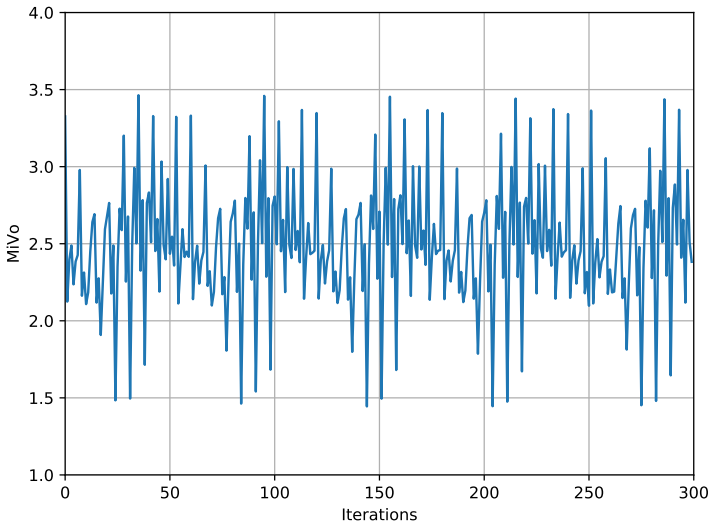

Figure 5.18: MiVo metric computed between target time series depicting the laying activity in test 5 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of laying obtained after the transformation and the expected behavior.

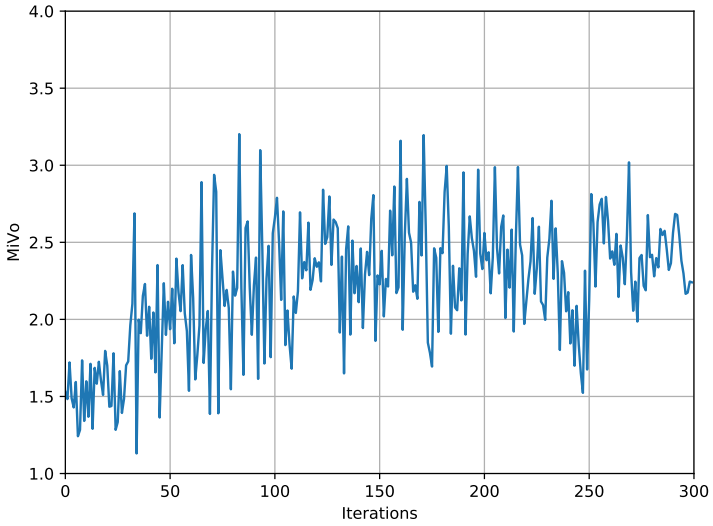

Figure 5.19: MiVo metric computed between target time series depicting the walking activity in test 5 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of walking obtained after the transformation and the expected behavior.

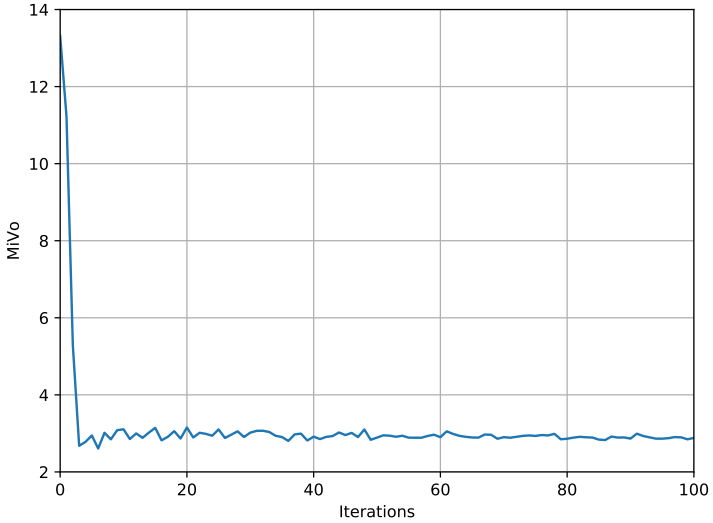

Figure 5.20: MiVo metric computed between time series depicting the target behavior for the DC motor dataset in test 9 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the time series obtained after the transformation and the expected behavior.

guide a human to decide if data generated by a GAN algorithm can be used to build reliable and trustworthy AI models. We believe that this topic will gain importance in the future since more AI algorithms will rely on generated data.

CHAPTER

6

CONCLUSIONS AND FUTURE WORK

The major aim of this thesis was to propose new generative approaches to synthesize more data for an existing dataset or to simulate never seen conditions by mapping time series between different application domains. To this end, we investigated the efficiency and the applicability of GAN on time series data. Motivated by its extensive use and impressive success in numerous computer vision applications, we tried to explore this technique for this special type of data. First, we considered the problem of generating new time series for multi-class datasets and defined the requirements and challenges that are related to this task. Therefore, we presented ClaRe-GAN a sophisticated GAN architecture that uses special encoders and an additional discriminator to learn both the inter- and intra-class properties. In the second part of the thesis, we proposed a novel framework, DR-TiST, for the challenging task of translating time series, i.e., mapping time series from a source to a target domain. Its feasibility and efficiency were investigated for three different use cases where the time series translation topic is important. Moreover, we focused on enabling a privacy-preserving time series generation and translation by combining the existing and new approaches with DP. Finally, we presented two analytical approaches, computational and visual, that ease the comparison between the obtained time series after the generation or translation and the desired behavior.

Chapter 2 provided a thorough literature review and introduced the theoretical background of the thesis. A well-known technique for data generation namely GAN was rigorously defined and the common training problems were determined. Some state-of-the-art GAN models and evaluation methods for time series data were presented. In addition to that, the topic of data translation was deeply investigated by reviewing the corresponding state-of-the-art image-to-image translation methods. Finally, the main

technique that was used in the thesis to guarantee privacy is analyzed. The core idea of DP and the intuition behind it are meticulously explained and the existing differentially private ML approaches are briefly introduced.

Chapter 3 focused on the time series generation topic by tackling two major issues namely synthesizing time series for an existing dataset in a standard as well as a private context. First, we proposed a new GAN architecture specially designed to deal with time series stemming from multi-class datasets. Aware that for datasets with high-variability the performance of the state-of-the-art models is still limited, we presented ClaRe-GAN that improved the quality of the generated data drastically. The new generative model for time series with class information efficiently simulates novel times series by capturing both the class association and variability of the training dataset. Our approach relied on *class conditional encoders* and a *class discriminator* to extract simultaneously class-specific and class-independent features. We compared our model to different state-of-the-art generative models for time series and proved that it extracts effectively the inter- and intra-class properties leading to a significant improvement in the quality of the synthesized time series even in challenging setups such as imbalanced datasets.

Finally, we presented methods to generate time series in a private manner by combining the existing GAN frameworks for time series and ClaRe-GAN with DP. We have shown that the developed frameworks achieve the desired behavior and that DP-TimeGAN and DP-ClaRe-GAN outperform the existing differentially private RCGAN in terms of privacy and usefulness of the generated time series across different domains. Our experiments also show that DP-C-RNN-GAN achieves the best privacy values. However, it decreases drastically the quality of the generated data.

In a similar fashion, we considered in Chapter 4 the same issues while translating time series between different application domains. In this context, we proposed DR-TiST a new algorithm that accomplishes time series translation by decomposing them into a functional behavior and an operating mode. The presented framework applies the gated CNN structure on DRIT an algorithm originally designed for image-to-image translation purposes. The utility of the proposed method is tested on three use cases: to map time series that depict the behavior of many machines between different environmental setups, to generate human activities, and to find an optimal controller for a non-accessible DC motor. As a first use case, we considered a real-world use case where we transfer the behavior of a ventilation system operating in a small room with slow on/off cycles to another environment characterized by faster on/off cycles. The second use case targets transforming sensor measurements that depict a specific human activity to depict another target activity. Finally, the last use case investigated the possibility of simulating the effect of a controller on a DC motor without neither knowing its mathematical model nor performing real-life tests on this motor. Results show that for the different use cases, DR-TiST outperforms CycleGAN-VC both in quality of the generated time series as well as in runtime.

Last but not least, we proposed two methods, DP-CycleGAN-VC and DP-DR-TiST, to translate time series across different domains with privacy guarantees. We showed that the proposed methods can be used for the previously described three different use cases. Our approach relied on DP a well-known method to protect the training dataset

against membership attacks. In almost all tests, DP-DR-TiST achieved a high utility for a reasonable privacy loss while DP-CycleGAN-VC guarantees higher privacy at the expense of the utility. In the future, we recommend practitioners to use DP-DR-TiST for translation purposes as it finds the right trade-off between privacy and utility.

Chapter 5 addressed the problem of assessing the quality of the generated and translated time series by introducing a computational and a visual evaluation method. Our main purpose was to give ML experts guidance to trust or reject a generative model. Therefore, we tested the existing GAN metrics and compared them to each other. We presented a compiled summary of the different existing metrics, their applicability, and their advantages and disadvantages. In addition to that, a new metric, MiVo, is proposed. The novel method relies on the nearest neighbor distances of real and generated data and enables to evaluate simultaneously the quality and the diversity of generated time series by finding a real pair for each generated (quality) and for each real a generated pair (diversity). We proved in our tests that MiVo enables a simultaneous detection of many training problems. After that, we applied MiVo in a special manner to compare the time series obtained after the translation with the desired behavior.

A metric can give the ML expert a global and efficient overview over the iteration. In spite of that, in many situations, a visual inspection is needed to further investigate the time series. To this end, we proposed a VA framework that relies on two visualization techniques Colorfield and TimeHistogram as well as a distance measure to support ML experts in the evaluation process. In general, we recommend ML expert to start the evaluation process with a metric or a combination of metrics that may reveal the common training difficulties and problems in order to select some interesting iterations. As a next step, the VA framework can be used to investigate the generated time series in-depth and to compare them to the real ones in an efficient manner.

In spite of the major contributions of the undertaken work, some challenges remained unsolved. In the following, we highlight some potential future work directions and some research questions that should be considered in the future to broaden the use of the, in this thesis, developed generation and translation techniques.

Throughout this thesis we showed that synthesizing time series that fulfill some specific criteria is a tedious task that must be rigorously investigated. To map time series between different application domains or produce more data for an existing dataset, multiple challenges must be addressed. In spite of that, we presented in this work efficient generation and translation algorithms for time series that are applicable in standard as well as private setups. We also proved that these algorithms outperform the state-of-the-art models. Moreover, we enabled a rigorous evaluation of the developed and existing algorithms using suitable analytical approaches. However, some further challenges must be addressed in the future to make the synthesis task wider practicable:

- How can we adapt the ClaRe-GAN algorithm to be applicable for datasets stemming from a single class?

- How can we perform a multi-domain time series translation?

- How can we protect the privacy of the original dataset that is used for time series generation and translation against more serious attacks than membership attacks such as model inversion and data reconstruction attacks?
- How can we assess the quality of the generated and translated data for multivariate time series? How can we extend MiVo and the VA approach presented in chapter 5 to be suitable for multivariate time series?
- Which VA approach can enable an efficient evaluation of time series translation techniques?

We presented ClaRe-GAN which thanks to its special architecture improved the quality of the generated data drastically. Its outstanding performance is related with the fact that each class is treated separately by encoding its characteristics. This allows for efficient extraction of the class-specific as well as class-independent properties. Unfortunately, the ClaRe-GAN architecture and precisely the *class discriminator* are only applicable for multi-class datasets. In the future, it is crucial to investigate new GAN architectures that generate data of high-quality and that are applicable for data stemming from one class. A possible solution to this problem would be to re-adapt the architecture of ClaRe-GAN to make it usable for such datasets.

We tackled the problem of mapping time series from a source domain $\mathcal{A}$ to a target domain $\mathcal{B}$ by presenting DR-TiST. We used the disentangled representation to split each time series into a functional behavior highlighting the properties of the time series and a context describing the environmental setup. Based on the extracted information, it is possible to map any functional behavior to any other operating mode and simulate never seen conditions. In practice, it may be desirable to map the time series to a nonexistent environmental setup that can be obtained by combining different environmental setups or to investigate new functional behavior stemming from a collection of existing functional behaviors. To deal with these issues, multi-domain translation techniques are of paramount importance. Hence, it is helpful to adapt and extend the architecture of DR-TiST for multi-domain translation purposes.

All the private approaches that we presented in this work namely DP-C-RNN-GAN, DP-TimeGAN, DP-ClaRe-GAN, DP-CycleGAN-VC, and DP-DR-TiST rely on DP a well-known method to protect the training dataset. DP protects the privacy of a dataset by bounding the impact of each dataset's instance. This technique can be successfully used to prevent membership attacks which aim at identifying whether a specific data point belongs to the training dataset. Unfortunately, this method cannot protect the privacy of the dataset against more serious attacks such as model inversion attacks or data reconstruction attacks where there is a risk of retrieving the whole training dataset. In the future, we intend to consider these privacy issues while generating and translating time series.

MiVo and the VA framework are two analytical approaches presented in Chapter 5 with the aim of assessing the quality of the generated data. These methods are only applicable on univariate time series. To enable a rigorous evaluation for multivariate time series the views of the VA system should be re-adapted and extended to enable a

comparison for the different attributes of multivariate time series. Moreover, the MiVo metric was only designed for univariate time series. It is essential to review this metric and the computed INND and ONND to make it appropriate for multivariate time series.

In our experiments, we have noticed that the combination of a computational metric, i.e., MiVo with a visual metric, i.e., our VA Approach enables a rigorous investigation of the generated time series. By way of example, it was helpful to select some iterations with MiVo as it provides a fast and global overview over the iterations and to further investigate them and the corresponding time series with the VA framework. Unfortunately, the views of our VA framework and the concept were uniquely designed to ease the comparison between real and generated time series. Due to the enormous advantages of a visual inspection, it is important to design new VA frameworks that consider the translation task and allow for an efficient and rigorous evaluation of the mapped time series. It is to be noted that assessing the quality of generated data is less complex than evaluating the translated time series. In fact, finding the similarities and differences between the real and the generated data is easier than identifying the properties of the time series that should be preserved, removed, or added while translating time series across different application domains.

BIBLIOGRAPHY

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[2] J. Donahue and K. Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, pages 10542–10552, 2019.

[3] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[4] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[5] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[6] S. Barratt and R. Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[7] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[9] O. Mogren. C-RNN-GAN: A continuous recurrent neural network with adversarial training. In *Constructive Machine Learning Workshop (CML) at NIPS 2016*, 2016.

[10] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[11] C. Esteban, S. L. Hyland, and G. Rätsch. Real-valued (medical) time series generation with recurrent conditional GANs. *arXiv preprint arXiv:1706.02633*, 2017.

[12] J. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A Efros, O. Wang, and E. Shechtman. Multimodal image-to-image translation by enforcing Bi-Cycle consistency. In *Advances in neural information processing systems*, pages 465–476, 2017.

[13] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

[14] M. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017.

[15] H. Lee, H. Tseng, Q. Mao, J. Huang, Y. Lu, M. Singh, and M. Yang. Drit++: Diverse image-to-image translation via disentangled representations. *International Journal of Computer Vision*, 128(10):2402–2417, 2020.

[16] H. Arnout, J. Bronner, J. Kehrer, and T. Runkler. DR-TiST: Disentangled representation for time series translation across application domains. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020.

[17] H. Arnout, J. Bronner, J. Kehrer, and T. Runkler. Translation of time series data from controlled dc motors using disentangled representation learning. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021.

[18] T. Kaneko and H. Kameoka. CycleGAN-VC: Non-parallel voice conversion using cycle-consistent adversarial networks. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2100–2104. IEEE, 2018.

[19] J. Curzon, T. A. Kosa, R. Akalu, and K. El-Khatib. Privacy and artificial intelligence. *IEEE Transactions on Artificial Intelligence*, 2(2):96–108, 2021.

[20] J. Yoon, D. Jarrett, and M. van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 5508–5518, 2019.

[21] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19:513–520, 2006.

[22] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727. PMLR, 2015.

[23] Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018.

[24] H. Arnout, J. Bronner, and T. Runkler. Evaluation of generative adversarial networks for time series data. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[25] H. Arnout, J. Kehrer, J. Bronner, and T. Runkler. Visual evaluation of generative adversarial networks for time series data. In *Human-Centered AI: Trustworthiness of AI Models & Data (HAI) track at AAAI Fall Symposium*, 2019.

[26] H. Arnout, J. Bronner, J. Kehrer, and T. Runkler. Evaluierungsrahmen für zeitreihendaten. European pat. EP3809334A1. Siemens AG, April 2021.

[27] H. Lee, H. Tseng, J. Huang, M. Singh, and M. Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 35–51, 2018.

[28] H. Arnout, J. Bronner, and T. Runkler. ClaRe-GAN: Generation of class-specific time series. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021.

[29] H. Arnout, J. Bronner, and T. Runkler. Differentially private time series generation. In *Computational Intelligence and Machine Learning ESANN 2021 proceedings*, pages 617–622, 2021.

[30] H. Arnout and T. Runkler. Privacy-preserving time series translation. *Submitted to Neural Networks*, 2022.

[31] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim. Towards a rigorous evaluation of xai methods on time series. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4197–4201. IEEE, 2019.

[32] U. Schlegel, E. Cakmak, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim. Towards visual debugging for multi-target time series classification. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 202–206, 2020.

[33] A. Antoniou, A. Storkey, and H. Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[34] F. H. K. S. Tanaka and C. Aranha. Data augmentation using GANs. *arXiv preprint arXiv:1904.09135*, 2019.

[35] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi. A survey on GANs for anomaly detection. *arXiv preprint arXiv:1906.11632*, 2019.

[36] A. Jabbar, X. Li, and O. Bourahla. A survey on generative adversarial networks: Variants, applications, and training. *ACM Computing Surveys (CSUR)*, 54(8):1–49, 2021.

[37] R. Sharma, S. Barratt, S. Ermon, and V. Pande. Improved training with curriculum GANs. *arXiv preprint arXiv:1807.09295*, 2018.

[38] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[39] I. Durugkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

[40] T. Doan, J. Monteiro, I. Albuquerque, B. Mazoure, A. Durand, J. Pineau, and R. D. Hjelm. On-line adaptative curriculum learning for GANs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3470–3477, 2019.

[41] G. Mordido, H. Yang, and C. Meinel. Dropout-GAN: Learning from a dynamic ensemble of discriminators. *arXiv preprint arXiv:1807.11346*, 2018.

[42] M. Ben-Yosef and D. Weinshall. Gaussian mixture generative adversarial networks for diverse datasets, and the unsupervised clustering of images. *arXiv preprint arXiv:1808.10356*, 2018.

[43] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[44] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[45] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2180–2188, 2016.

[46] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017.

[47] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.

[48] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[49] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann. Stabilizing training of generative adversarial networks through regularization. *arXiv preprint arXiv:1705.09367*, 2017.

[50] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[51] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11):2579—2605, 2008.

[52] F. B. Bryant and P. R. Yarnold. Principal-components analysis and exploratory and confirmatory factor analysis. 1995.

[53] P. Isola, J. Zhu, T. Zhou, and A. A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[54] X. Huang, M. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[55] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning*, pages 1857–1865. PMLR, 2017.

[56] C. Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

[57] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[58] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[59] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[60] C. Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[61] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

[62] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

[63] X. Zhang, S. Ji, and T. Wang. Differentially private releasing via deep generative model (technical report). *arXiv preprint arXiv:1801.01594*, 2018.

[64] J. Jordon, J. Yoon, and M. Van Der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2018.

[65] R. Torkzadehmahani, P. Kairouz, and B. Paten. DP-CGAN: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[66] S. Augenstein, H B. McMahan, D. Ramage, S. Ramaswamy, P. Kairouz, M. Chen, R. Mathews, et al. Generative models for effective ML on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019.

[67] L. Fan. A survey of differentially private generative adversarial networks. In *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*, 2020.

[68] A. Bagnall, J. Lines, W. Vickers, and E. Keogh. The UEA & UCR time series classification repository. *URL http://www. timeseriesclassification. com*, 2018.

[69] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[70] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. Stack-GAN++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018.

[71] Q. Mao, H. Lee, H. Tseng, S. Ma, and M. Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1429–1437, 2019.

[72] I. Mironov. Rényi differential privacy. In *IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[73] A. Bagnall, J. Linesand, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, 31(3):606–660, 2017.

[74] A. L. Goldberger, , L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[75] L. M. Davis. *Predictive modelling of bone ageing*. PhD thesis, University of East Anglia, 2013.

[76] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[77] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

[78] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[79] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[80] R. Zhang, J. Zhu, P. Isola, X. Geng, A. S Lin, T. Yu, and A. A Efros. Real-time user-guided image colorization with learned deep priors. *arXiv preprint arXiv:1705.02999*, 2017.

[81] R. Zhang, P. Isola, and A. A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[82] K. Aggarwal, S. Joty, L. Fernandez-Luque, and J. Srivastava. Adversarial unsupervised representation learning for activity time-series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 834–841, 2019.

[83] S. Ekinci, D. Izci, and B. Hekimoğlu. PID speed control of DC motor using harris hawks optimization algorithm. In *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pages 1–6. IEEE, 2020.

[84] S. J. Hammoodi, K. S. Flayyih, and A. R. Hamad. Design and implementation speed control system of DC motor based on PID control and matlab simulink. *International Journal of Power Electronics and Drive Systems*, 11(1):127, 2020.

[85] W. P. Aung. Analysis on modeling and simulink of DC motor and its driving system used for wheeled mobile robot. *World Academy of Science, Engineering and Technology*, 32:299–306, 2007.

[86] Y. Ma, Y. Liu, and C. Wang. Design of parameters self-tuning fuzzy PID control for DC motor. In *2010 The 2nd International Conference on Industrial Mechatronics and Automation*, volume 2, pages 345–348. IEEE, 2010.

[87] T. Sands. Control of DC motors to guide unmanned underwater vehicles. *Applied Sciences*, 11(5):2144, 2021.

[88] P. Ejegwa, S. Wen, Y. Feng, W. Zhang, and N. Tang. Novel pythagorean fuzzy correlation measures via pythagorean fuzzy deviation, variance and covariance with applications to pattern recognition and career placement. *IEEE Transactions on Fuzzy Systems*, 2021.

[89] Y. Feng, W. Zhang, J. Xiong, H. Li, and L. Rutkowski. Event-triggering interaction scheme for discrete-time decentralized optimization with nonuniform step sizes. *IEEE Transactions on Cybernetics*, pages 1–10, 2020.

[90] B. O. Onasanya, S. Wen, Y. Feng, W. Zhang, and J. Xiong. Fuzzy coefficient of impulsive intensity in a nonlinear impulsive control system. *Neural Processing Letters*, pages 1–19, 2021.

[91] Y. Feng, X. Yang, Q. Song, and J. Cao. Synchronization of memristive neural networks with mixed delays via quantized intermittent control. *Applied Mathematics and Computation*, 339:874–887, 2018.

[92] J. Qi, C. Li, and T. Huang. Stability of inertial BAM neural network with time-varying delay via impulsive control. *Neurocomputing*, 161:162–167, 2015.

[93] C. Li, G. Feng, and T. Huang. On hybrid impulsive and switching neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(6):1549–1560, 2008.

[94] R. Kumar and V. Girdhar. High performance fuzzy adaptive control for DC motor. *Global Journal of Research In Engineering*, 2013.

[95] A. R. Hambley, N. Kumar, and A. R. Kulkarni. *Electrical engineering: principles and applications*. Pearson Prentice Hall Upper Saddle River, NJ, 2008.

[96] T. Kara and I. Eker. Nonlinear modeling and identification of a DC motor for bidirectional operation with real time experiments. *Energy Conversion and Management*, 45(7-8):1087–1106, 2004.

[97] A. Woźniak. A simple model of drive with friction for control system simulation. In *International Conference on Computational Science*, pages 897–906. Springer, 2003.

[98] M. M. Sabir and J. A. Khan. Optimal design of PID controller for the speed control of DC motor by using metaheuristic techniques. *Advances in artificial neural systems*, 2014.

[99] P. J. Antsaklis and A. N. Michel. *Linear systems*. Springer Science & Business Media, 2006.

[100] G. F. Franklin, J. D. Powell, M. L. Workman, et al. *Digital control of dynamic systems*, volume 3. Addison-wesley Reading, MA, 1998.

[101] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[102] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[103] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[104] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visual methods for analyzing time-oriented data. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):47–60, Jan 2008.

[105] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[106] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, page arXiv:1511.01844, Apr 2016.

[107] J. Serra and J. L. Arcos. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*, 67:305–314, 2014.

[108] R. Kosara, F. Bendix, and H. Hauser. Time Histograms for Large, Time-dependent Data. In *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*, VISSYM'04, pages 45–54, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[109] A. Gogolou, T. Tsandilas, T. Palpanas, and A. Bezerianos. Comparing similarity perception in time series visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):523–533, Jan 2019.

[110] J. Kruskal and M. Liberman. The symmetric time-warping problem: From continuous to discrete. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, 01 1983.

[111] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

APPENDIX

A

APPENDIX: ADDITIONAL FIGURES

A.1 Time Series Generation

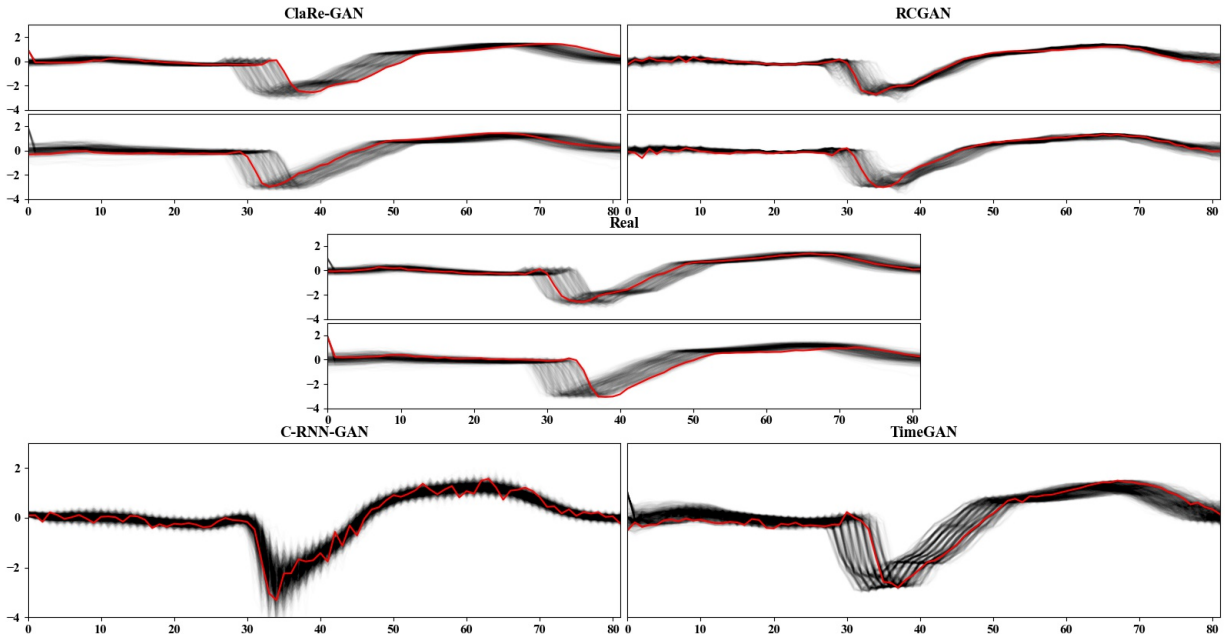

Figure A.1: Illustration of the real and generated time series by ClaRe-GAN, RCGAN, C-RNN-GAN and TimeGAN for the TwoLeadECG dataset. The time series are depicted in black. The red line presents an example time series for each subplot. For the conditional GANs, ClaRe-GAN and RCGAN, and the real dataset we visualize the time series of each class separately.

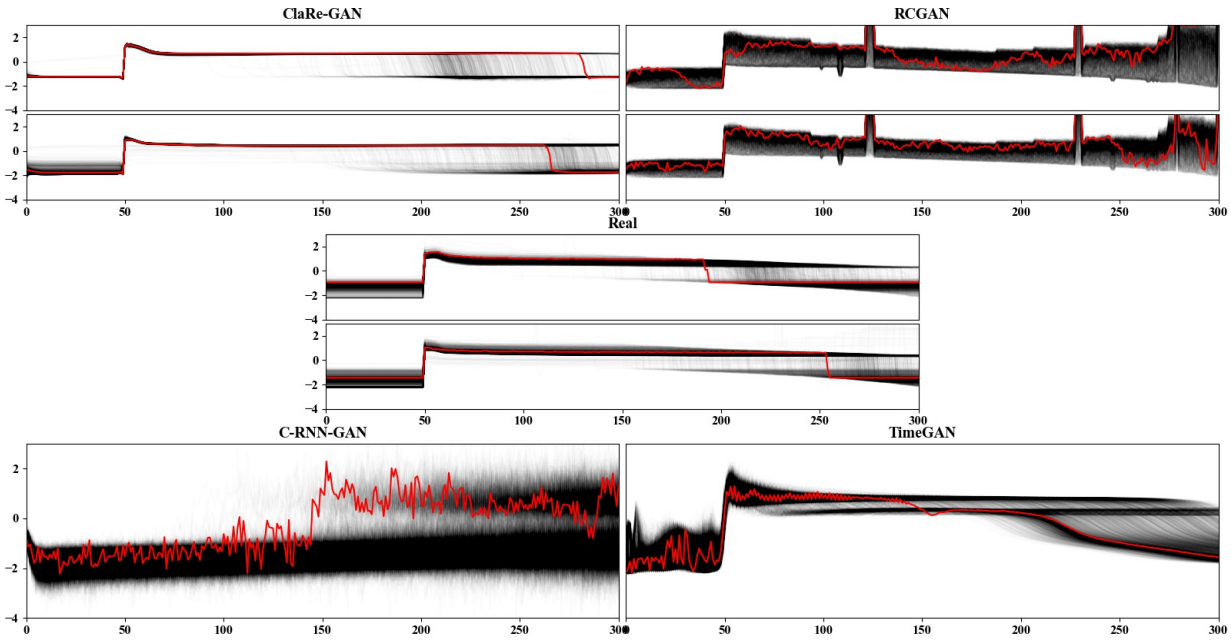

Figure A.2: Illustration of the real and generated time series by ClaRe-GAN, RCGAN, C-RNN-GAN and TimeGAN for the FreezerRegularTrain dataset. The time series are depicted in black. The red line presents an example time series for each subplot. For the conditional GANs, ClaRe-GAN and RCGAN, and the real dataset we visualize the time series of each class separately.

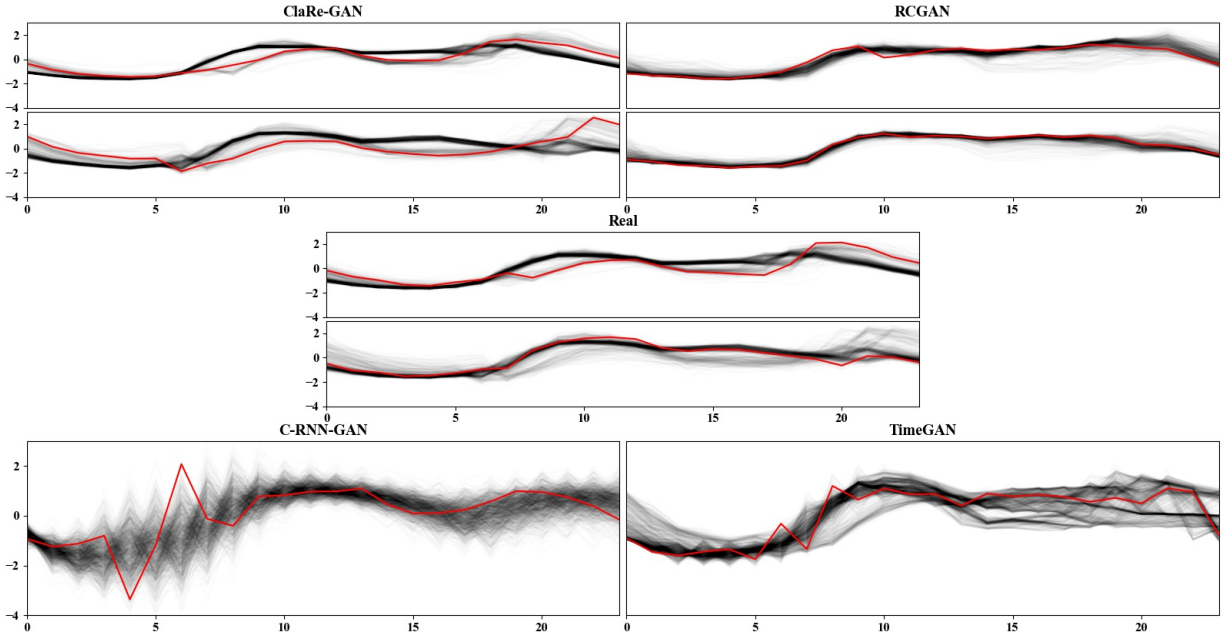

Figure A.3: Illustration of the real and generated time series by ClaRe-GAN, RCGAN, C-RNN-GAN and TimeGAN for the ItalyPowerDemand dataset. The time series are depicted in black. The red line presents an example time series for each subplot. For the conditional GANs, ClaRe-GAN and RCGAN, and the real dataset we visualize the time series of each class separately.

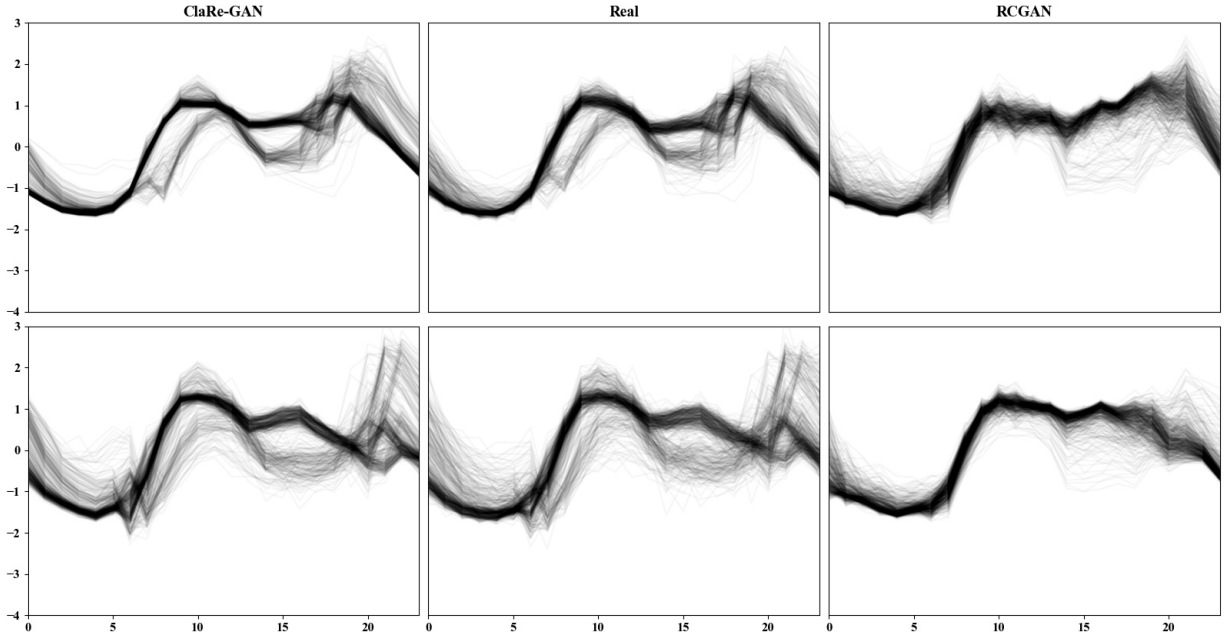

Figure A.4: Illustration of the classes generated by ClaRe-GAN, the classes of the real dataset and the classes generated by RCGAN for the ItalyPowerDemand dataset.

A.2 Time Series Analytics

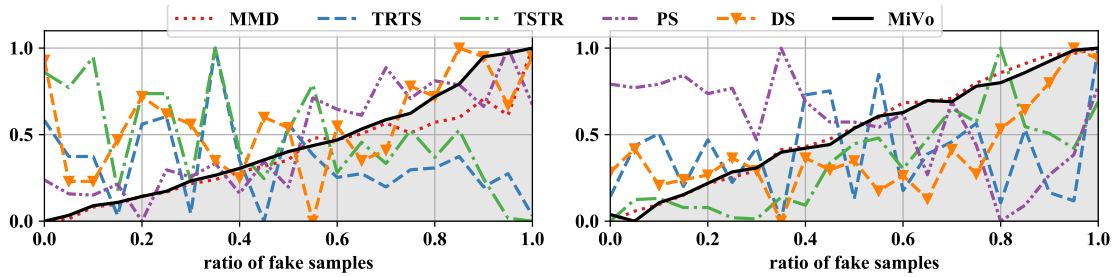

Figure A.5: Results of mixing test ratio l of fake samples in $S_g(l)$ is augmented progressively for the ItalyPowerDemand (plot on the left) and DistalPhalanxTW (plot on the right) datasets. The score of the different metrics is computed. We expect that the score of each metric increases as the ratio l increases, i.e., a reliable metric should be able to discriminate between real and generated samples its best score should be achieved on real samples.

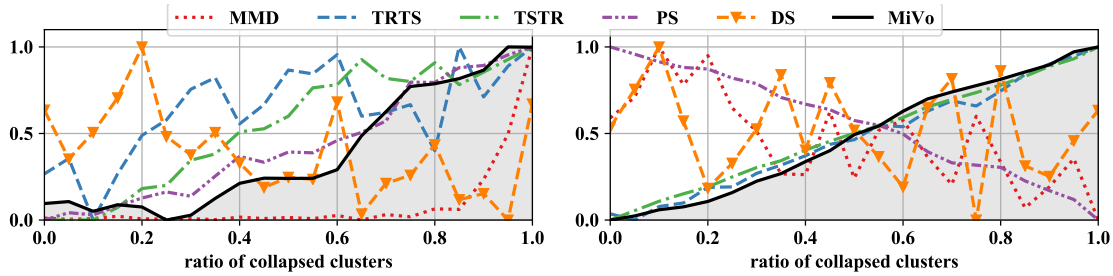

Figure A.6: Results of mode collapse test for the ItalyPowerDemand (plot on the left) and DistalPhalanxTW (plot on the right) datasets: the metrics score is computed depending on the number of collapsed clusters. A good evaluation metric should be able to detect mode collapse, i.e., its score should increase with the ratio of collapsed clusters.

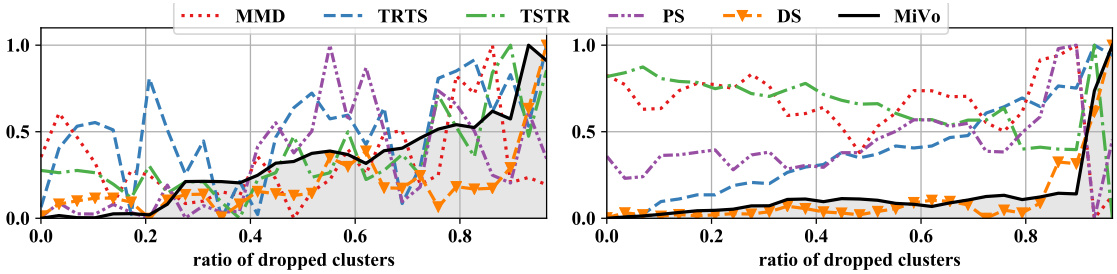

Figure A.7: Results of mode dropping test for the ItalyPowerDemand (plot on the left) and DistalPhalanxTW (plot on the right) datasets: the metrics score is computed depending on the number of dropped clusters. A good evaluation metric should be able to detect mode dropping, i.e., its score should increase with the ratio of dropped clusters.

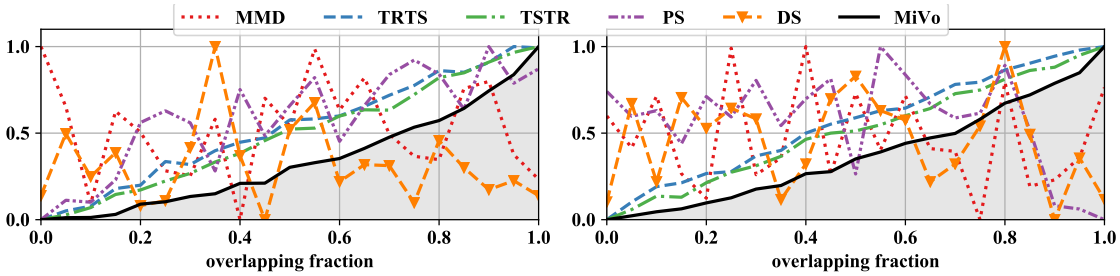

Figure A.8: Results of the overfitting test for the ItalyPowerDemand (plot on the left) and DistalPhalanxTW (plot on the right) datasets: the metrics score is computed depending on the overlapping fraction between a set S_r'' of real samples and a set S_r^{tr}. An accurate metric should increase its score as more samples of S_r'' are becoming similar to the training set in order to highlight the overfitting phenomenon.

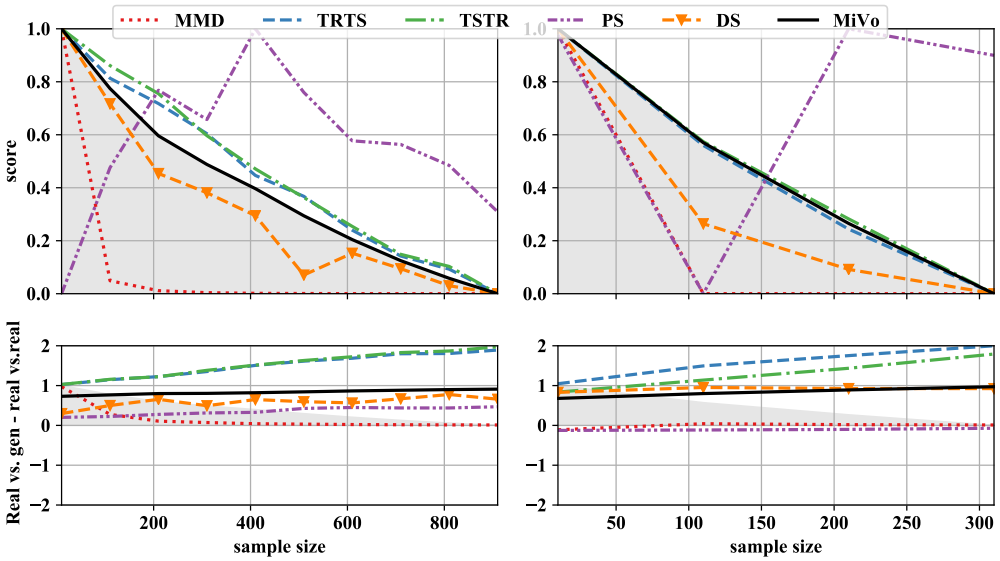

Figure A.9: Results of the efficiency test for the ItalyPowerDemand (plot on the left) and DistalPhalanxTW (plot on the right) datasets: the values of $\rho(S_r, S'_r)$ are computed with different number of samples (upper plot). The second plot depicts the value of $\rho(S_g, S_r) - \rho(S'_r, S_r)$. This difference should be positive. For a reasonable number of samples an accurate metric should score $\rho(S_g, S_r)$ higher than $\rho(S'_r, S_r)$.

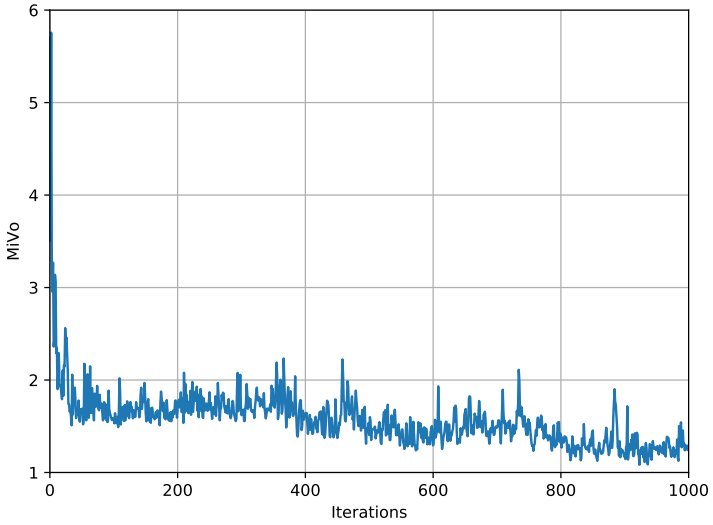

Figure A.10: MiVo metric computed for the ItalyPowerDemand dataset over the training iterations. A lower MiVo value denotes a better GAN performance.

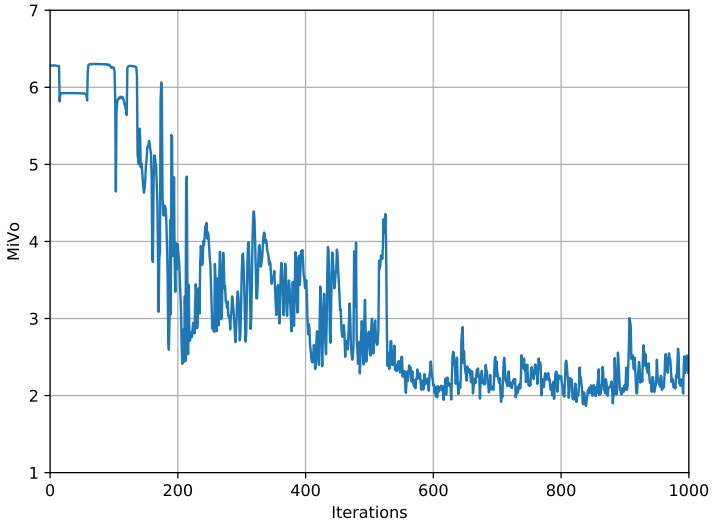

Figure A.11: MiVo metric computed for the DistalPhalanxTW dataset over the training iterations. A lower MiVo value denotes a better GAN performance.

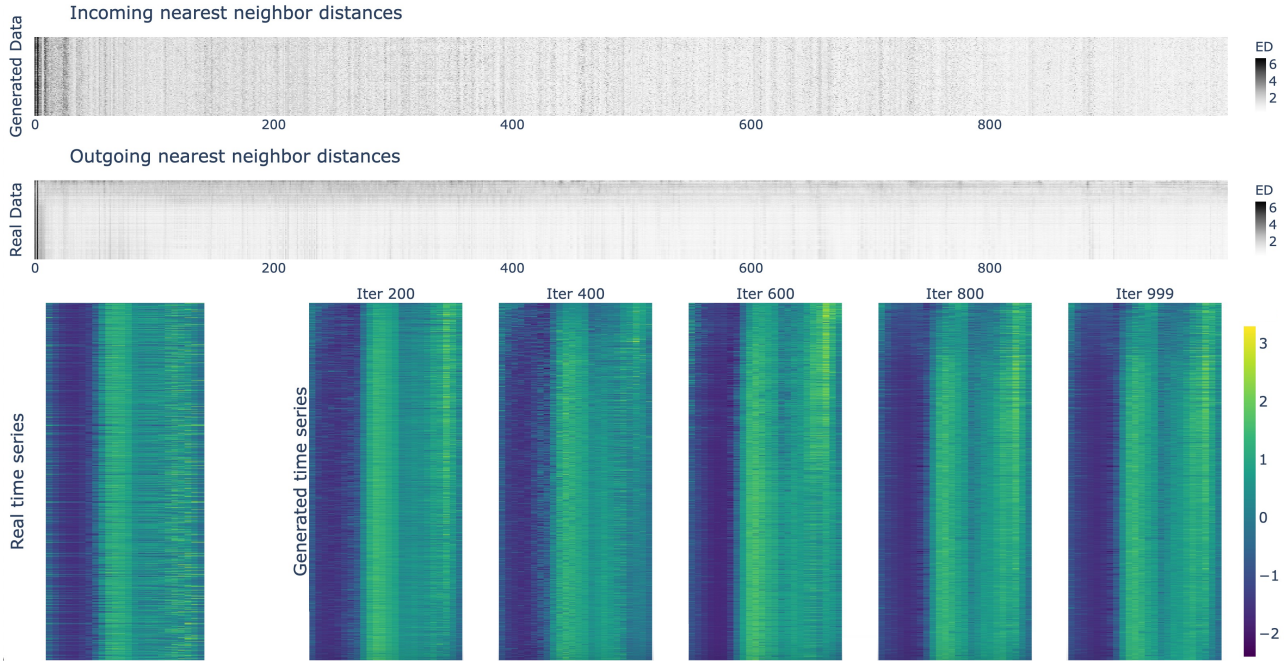

Figure A.12: Results of the ItalyPowerDemand dataset illustrated in the VA framework presented in chapter 5

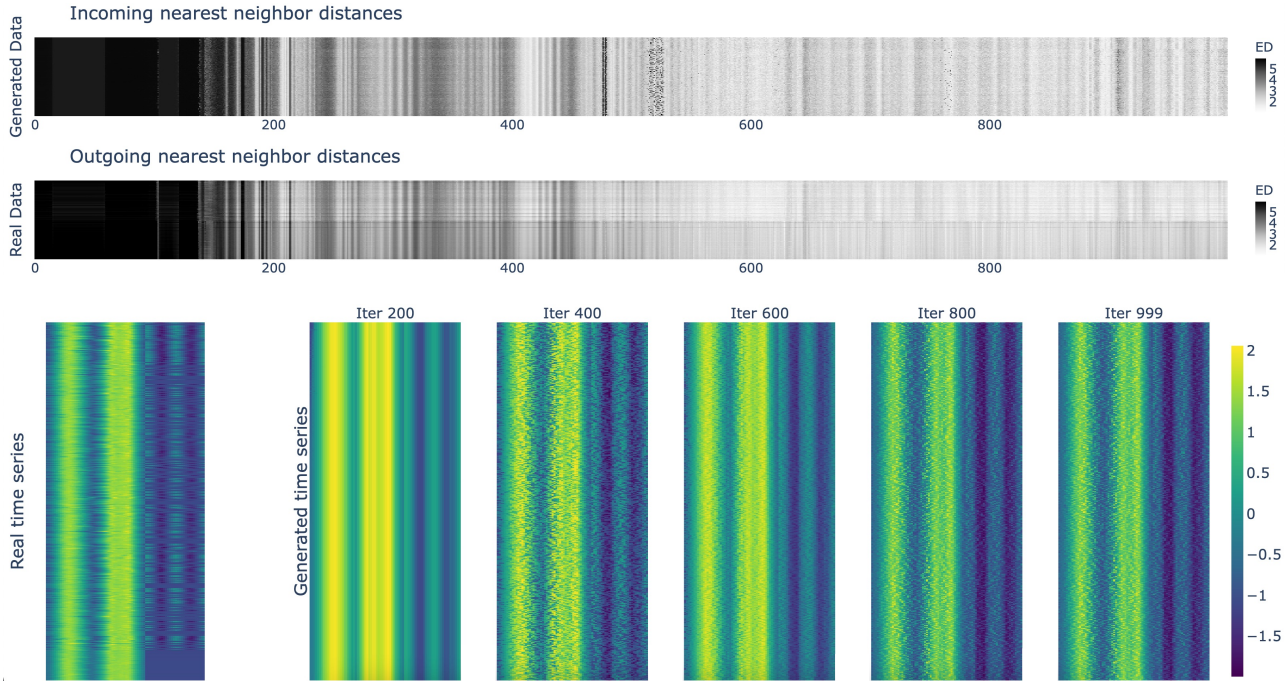

Figure A.13: Results of the DistalPhalanxTW dataset illustrated in the VA framework presented in chapter 5

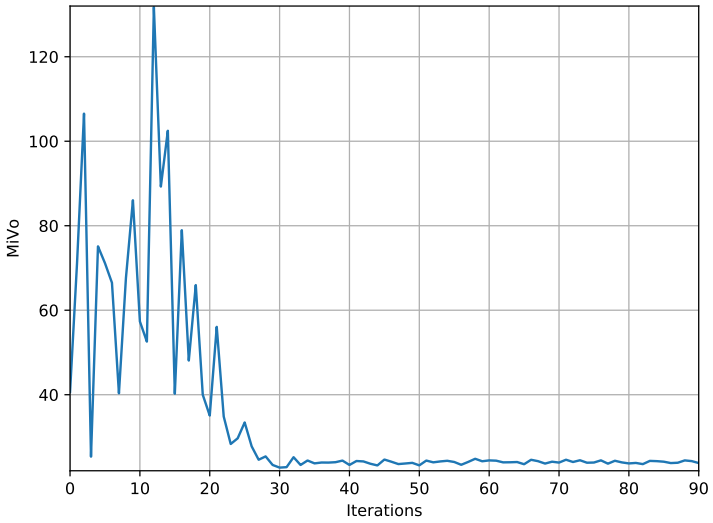

Figure A.14: MiVo metric computed between target time series depicting the behavior of engine 1 in test 2 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of engine 1 obtained after the transformation and the expected behavior.

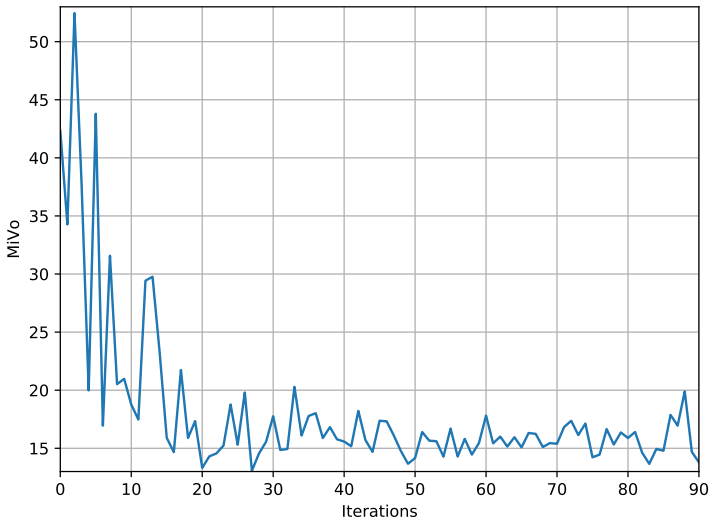

Figure A.15: MiVo metric computed between target time series depicting the behavior of engine 2 in test 2 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of engine 2 obtained after the transformation and the expected behavior.

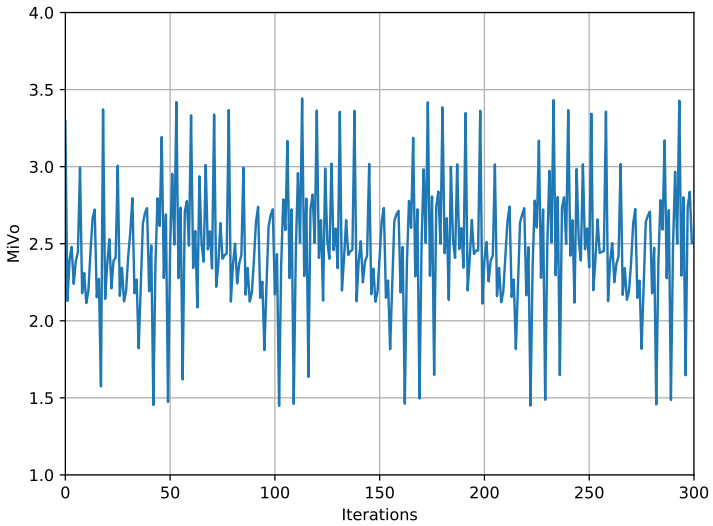

Figure A.16: MiVo metric computed between target time series depicting the sitting activity in test 6 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of sitting obtained after the transformation and the expected behavior.

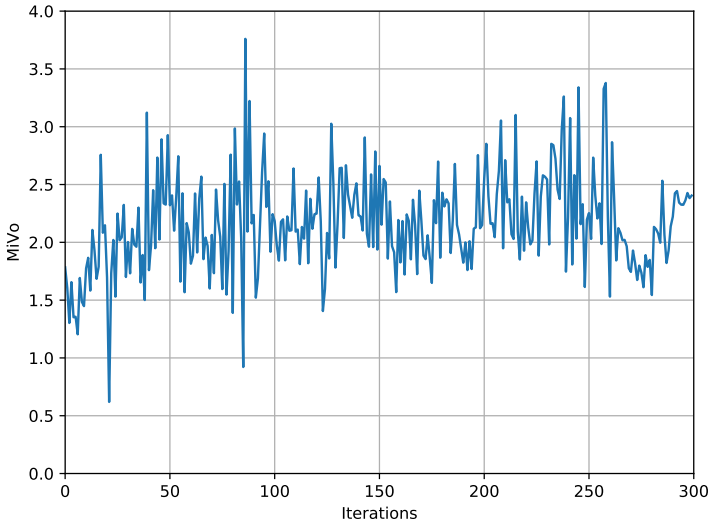

Figure A.17: MiVo metric computed between target time series depicting the walking activity in test 6 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of walking obtained after the transformation and the expected behavior.

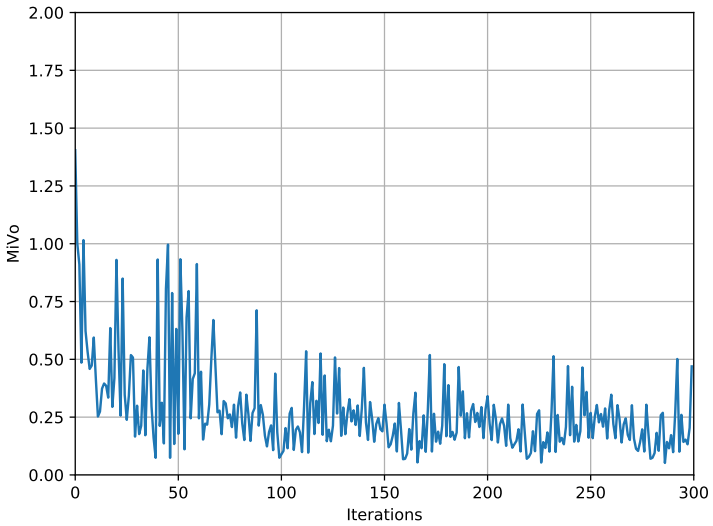

Figure A.18: MiVo metric computed between target time series depicting the laying activity in test 7 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of laying obtained after the transformation and the expected behavior.

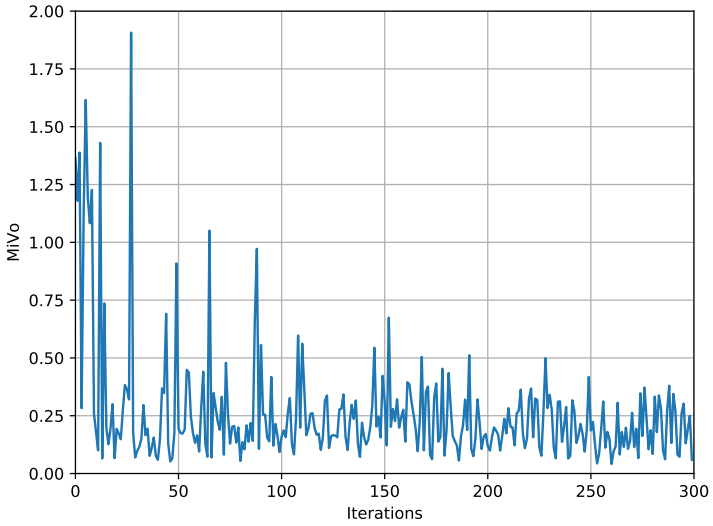

Figure A.19: MiVo metric computed between target time series depicting the sitting activity in test 7 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the representation of sitting obtained after the transformation and the expected behavior.

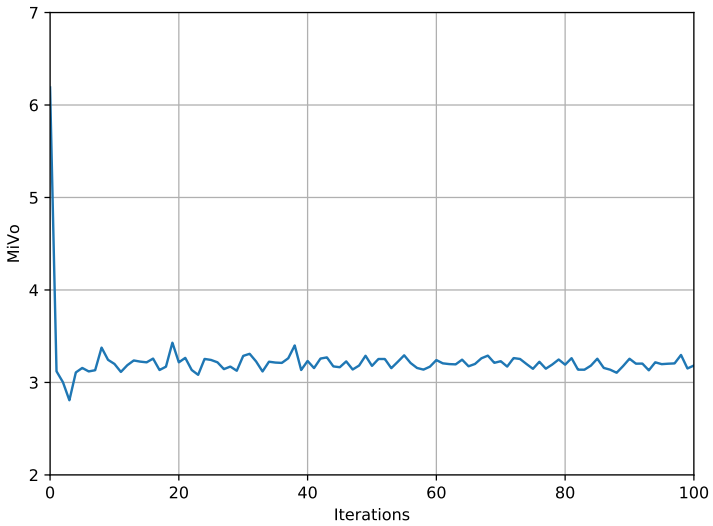

Figure A.20: MiVo metric computed between time series depicting the target behavior for the DC motor dataset in test 8 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the time series obtained after the transformation and the expected behavior.

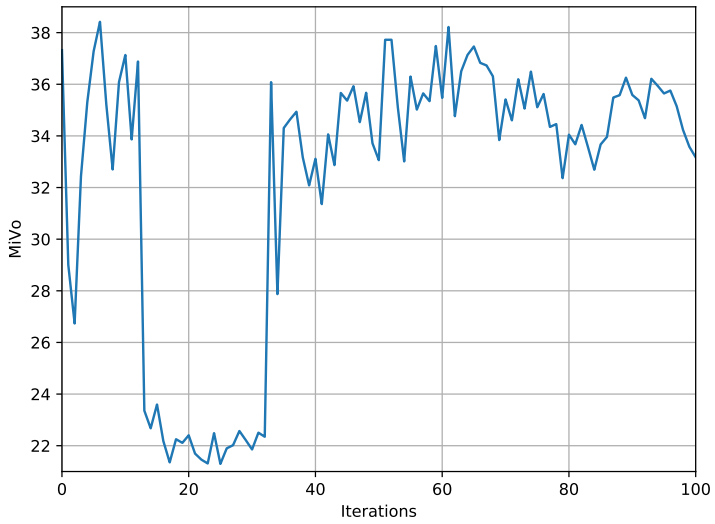

Figure A.21: MiVo metric computed between time series depicting the target behavior for the DC motor dataset in test 10 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the time series obtained after the transformation and the expected behavior.

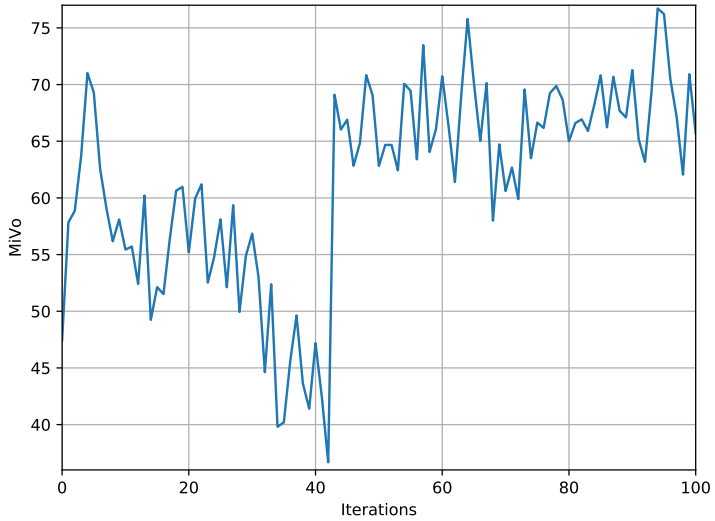

Figure A.22: MiVo metric computed between time series depicting the target behavior for the DC motor dataset in test 11 and the time series obtained after the translation. A lower MiVo value denotes a better similarity between the time series obtained after the transformation and the expected behavior.

LIST OF FIGURES

LIST OF TABLES