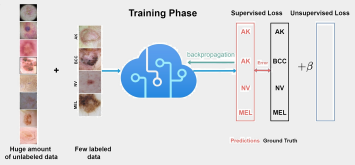
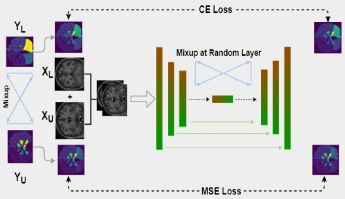
Computer Aided Medical Procedures
Prof. Dr. Nassir Navab

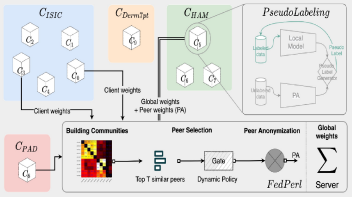


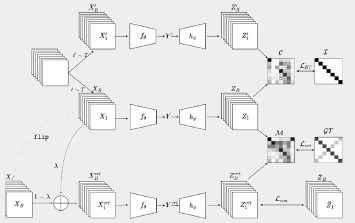
Dissertation

Annotation-Efficient Medical Imaging with Deep Learning

Tariq Mousa Ahmad Bdair







Fakultät für Informatik
Technische Universität München

Technische Universität München
TUM School of Computation, Information and Technology

Annotation-Efficient Medical Imaging with Deep Learning

Tariq Mousa Ahmad Bdair

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

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Hans Michael Gerndt

*Prüfer*innen der Dissertation:*

1. Prof. Dr. Nassir Navab

2. Prof. Dr. Georg Langs

3. Prof. Dr. Shadi Albarqouni

Die Dissertation wurde am 06.10.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 17.04.2023 angenommen.

Tariq Mousa Ahmad Bdair
Annotation-Efficient Medical Imaging with Deep Learning
Dissertation, Version 0.0

Technische Universität München
Fakultät für Informatik
TUM School of Computation, Information and Technology
Boltzmannstraße 3
85748 and Garching bei München

Abstract

Medical imaging is essential for examining the human body, internal organs, disease developments, and many other medical applications. However, dealing with medical images needs decent experience, and it is subject to individuals' differences. Consequently, computer-based approaches have been developed to overcome the above limitations. Among these methods are the deep-learning ones. Although deep learning methods have dominated all traditional ones, one drawback is that they require big annotated data, which is expensive, time-consuming, need experts, and is not always available. Therefore, this thesis addresses the insufficiency of annotated data by proposing annotation-efficient medical imaging in deep learning-based approaches. We tackle the problem from three related perspectives. The first perspective devises a novel data augmentation technique that enriches the model with new data points generated from random linear interpolation of labeled and unlabeled data. Our semi-supervised learning approach overcomes the limitations of previous works by exploring the input and latent spaces in the training process where virtual training signals are generated. This method showed state-of-the-art segmentation results on a couple of public MRI datasets for brain structures, and superior performance on CT scans for COVID-19 infection.

While accessing the labeled and unlabeled data at one site is not always feasible, a potential solution could be leveraging the distributed data in remote locations without breaching privacy or so-called federated learning. Thus, the second perspective proposes a semi-supervised federated learning method that exploits global knowledge and employs peer learning and knowledge sharing inspired by the educational sciences. In a nutshell, our method constructs clients' communities based on their similarities, then encourages similar clients, a.k.a peers, to learn from each other to create precise pseudo labels for the unlabeled data. Furthermore, we propose a peer anonymization technique to enhance privacy and hide clients' identities adhering to the regulations of federated learning. This method was applied to more than 72,000 dermoscopic skin cancer images collected from five public datasets and distributed to ten clients under four scenarios covering most real-life applications. In a set of extensive experiments, our method achieved the best results. Furthermore, we extended our work to include a dynamic learning policy that controls the learning stream between peers, which has demonstrated effectiveness, especially for out-of-distribution clients.

Although the unlabeled data is missing their annotations, they are still wealthy with another type of information hidden in their representations. Thus, the third perspective employs representation learning in a self-supervised learning paradigm. Precisely, without annotations, our strategy to learn better representations is to generate virtual embeddings by mixing the actual data using random percentages. Then we train the model to extract hidden information by decomposing the new embeddings to their original components and regressing the mixing factors. Further, we support our method by proposing a self-consistency between

the augmented and original embeddings, which forces the linearity and enhances the results. Finally, we tested our approach on eight standard and medical data benchmarks. In addition, we compared it with recent works, achieving superior performance in many downstream tasks.

Zusammenfassung

Die medizinische Bildgebung ist unerlässlich für die Untersuchung des menschlichen Körpers, der inneren Organe, der Entwicklung von Krankheiten und vieler anderer medizinischer Anwendungen. Der Umgang mit medizinischen Bildern erfordert jedoch ein hohes Maß an Erfahrung und unterliegt individuellen Unterschieden. Daher wurden computergestützte Ansätze entwickelt, um die oben genannten Einschränkungen zu überwinden. Zu diesen Methoden gehören die Deep-Learning-Methoden. Obwohl die Deep-Learning-Methoden alle traditionellen Methoden dominiert haben, besteht ein Nachteil darin, dass sie große annotierte Daten benötigen, die teuer und zeitaufwändig sind, Experten erfordern und nicht immer verfügbar sind.

Daher befasst sich diese Arbeit mit dem Mangel an annotierten Daten, indem sie eine annotationseffiziente medizinische Bildgebung in Deep-Learning-basierten Ansätzen vorschlägt. Wir gehen das Problem aus drei verwandten Perspektiven an.

In der ersten Perspektive wird eine neuartige Technik zur Datenerweiterung entwickelt, die das Modell mit neuen Datenpunkten anreichert, die durch zufällige lineare Interpolation von gekennzeichneten und nicht gekennzeichneten Daten erzeugt werden. Unser halbüberwachter Lernansatz überwindet die Einschränkungen früherer Arbeiten, indem er die Eingabe- und latenten Räume im Trainingsprozess erforscht, in dem virtuelle Trainingssignale erzeugt werden. Diese Methode hat bei einer Reihe von öffentlichen MRT-Datensätzen für Gehirnstrukturen hervorragende Segmentierungsergebnisse und bei CT-Scans für COVID-19-Infektionen eine überragende Leistung gezeigt.

Die zweite Perspektive schlägt eine halbüberwachte föderierte Lernmethode vor, die globales Wissen ausnutzt und Peer-Learning und Wissensaustausch, inspiriert von den Bildungswissenschaften, einsetzt. Kurz gesagt, unsere Methode konstruiert Client-Gemeinschaften auf der Grundlage ihrer Ähnlichkeiten und ermutigt dann ähnliche Clients, auch Peers genannt, voneinander zu lernen, um präzise Pseudo-Etiketten für die unetikettierten Daten zu erstellen. Darüber hinaus schlagen wir eine Technik zur Anonymisierung der Peers vor, um die Privatsphäre zu verbessern und die Identitäten der Clients zu verbergen, wobei wir uns an die Regeln des föderierten Lernens halten. Diese Methode wurde auf mehr als 72.000 dermatoskopische Hautkrebsbilder angewendet, die aus fünf öffentlichen Datensätzen gesammelt und in vier Szenarien, die die meisten realen Anwendungen abdecken, auf zehn Clients verteilt wurden. In einer Reihe von umfangreichen Experimenten erzielte unsere Methode die besten Ergebnisse. Darüber hinaus haben wir unsere Arbeit um eine dynamische Lernpolitik erweitert, die den Lernstrom zwischen den Peers steuert, was sich als wirksam erwiesen hat, insbesondere für Clients, die nicht an der Verteilung beteiligt sind.

Die dritte Perspektive setzt auf das Lernen von Repräsentationen in einem selbstüberwachten Lernparadigma. Ohne Anmerkungen besteht unsere Strategie zum Erlernen besserer Repräsentationen darin, virtuelle Einbettungen zu generieren, indem wir die tatsächlichen Daten mit Zufallsanteilen mischen. Dann trainieren wir das Modell, um versteckte Informationen zu extrahieren, indem wir die neuen Einbettungen in ihre ursprünglichen Komponenten zerlegen und die Mischungsfaktoren regressieren. Darüber hinaus unterstützen wir unsere Methode, indem wir eine Selbstkonsistenz zwischen den augmentierten und den ursprünglichen Einbettungen vorschlagen, die die Linearität erzwingt und die Ergebnisse verbessert. Schließlich haben wir unseren Ansatz an acht Standard- und medizinischen Daten-Benchmarks getestet. Darüber hinaus haben wir ihn mit neueren Arbeiten verglichen und in vielen nachgelagerten Aufgaben eine bessere Leistung erzielt.

Acknowledgments

I extend my endless gratitude to Almighty ALLAH for blessing me in completing this work. My great thankfulness is to Him for His mercy, blessing, and help. I wish Almighty ALLAH accept this work and grant me this help to thank Him.

First of all, I would thank Prof. Dr. Nassir Navab for his invaluable guidance, concern, enthusiasm, and support. This work has evolved between his hands, has grown up, and has ended under his excellent supervision.

A full thank goes to Dr. Shadi Albarqoni for all the mentoring and support during the doctorate. He inspired me and was my leader and teacher on this journey.

I express my complete gratitude to the dissertation committee, committee chair, Prof. Dr. Michael Gerndt, and Prof. Dr. Georg Langs for reviewing and evaluating this dissertation.

I express my special gratitude and appreciation to my parents, father Mousa, mother Afaf, my wife Sumaya, my children, Abderrahman, Noor, Hamza, and Mays, and my brothers; Hadeel, Moutasem, Abdullah, Ramzi, Aseel, and Gazal. I want to thank all these precious people in my life for their endless support, patience, and prayers. I dedicate my work as a gesture of my indebting to them.

Special thanks to all the CAMPers I met during these years. Also, I am very thankful to all my friends for their encouragement and support.

Lastly, I would like to thank The German Academic Exchange Service DAAD (Deutscher Akademischer Austauschdienst) for enabling me to pursue my Ph.D. This work has not been done without their generous fund during these years.

Contents

1 Introduction

> *In the name of Allah, Most Gracious, Most Merciful Read (Prophet Muhammad) in the Name of your Lord who created (1) created the human from a (blood) clot. (2) Read! Your Lord is the Most Generous, (3) who taught by the pen, (4) taught the human what he did not know. (5)*
>
> — **Holy Quran**
> (Sura 96: AL-ALAQ, Ayah 1-5)

> *And Allah has revealed to you the Book and wisdom and has taught you that which you did not know. And ever has the favor of Allah upon you been great.*
>
> — **Holy Quran**
> (Sura 4: AN-NISA, Ayah 113)

1.1 Motivation

Recent years have witnessed enormous development and improvements in using computer and machine learning algorithms in real-life applications, such as autonomous driving, speech recognition, recommendation systems, health care, finance, military, education, robotics, agriculture, surveillance, etc. All these systems generally work by consuming large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. However, despite all mentioned benefits, these systems have many drawbacks related to finding high-quality training data, which is expensive [240], time-consuming [71], needs experts [58], and is not always available [240]. These limitations are severe in the medical domain. On top of that, medical data can hold sensitive information and have business value limiting it from being publicly available [216].

Consequently, massive efforts are dedicated by researchers to find alternative strategies to compensate for the scarcity of labeled data. While these alternatives can vary significantly, this thesis addresses the insufficiency of annotated data by proposing annotation-efficient deep learning-based approaches for medical imaging. **In this context, the primary motivation of this dissertation is to find methods and techniques and leverage different available resources to provide the deep learning models with new training signals that are not available by the labeled data alone**.

1.2 Problem Statement

In this dissertation, we address the following research questions, which are considered necessary for deep learning problems, in particular when dealing with the **deficiency of annotated medical images**:

- **To what extent does the huge available non-annotated data in training helps build robust deep learning method suitable to the complexity of medical images?**

  Given that the unlabeled data is cheap, easy to find, and available in vast amounts, can we build a semi-supervised learning method [49] that combines labeled and unlabeled data intelligently and efficiently to generate proper and novel training signals without needing more human annotations? Furthermore, can knowledge in the labeled data be transferred effectively to unlabeled data to create accurate pseudo labels, then these unlabeled data with its artificial annotations enhance the models' accuracy? We address these research questions in Chapter 4 of this dissertation.

- **Can we incorporate and seek other available data resources from remote locations to build powerful deep-learning models?**

  It is known that different medical centers have their data on their premises. However, some or most of these data are available without annotations. Nevertheless, these medical institutes are willing to collaborate to model a beneficial clinical usage of their data without breaching privacy. Hence, can we mitigate the scarcity of annotated data by building collaborative methods in which the distributed knowledge is shared instead of isolated without any utilization? Furthermore, can we find an approach to distill the understanding between different clients, given that some lack the annotated data? Thus, creating a global model that exploits this accessible and distributed data is desirable. Fortunately, this perspective can be addressed by the so-called federated learning [193]. We address these concerns in this dissertation's second contribution in Chapter 5.

- **Can we extract the knowledge from unlabeled data to perform the same tasks achieved by the labeled ones and handle labeled data shortage?**

  Representation learning [25] is a set of techniques that allows a model to automatically discover the representations needed for different deep learning tasks, such as classification from raw data. This method replaces manual annotations and enables a machine to learn and use the features to perform a specific task. Can we build useful representations from the unlabeled data transferable to different downstream supervised tasks? Thus, utilizing such an approach could solve the need for labeled data. These research questions have been addressed in Chapter 6 of this dissertation.

1.3 Contributions

To this end, this thesis tackles the above problem from three related contributions that handle the scarcity of labeled data. We summarize our contributions as follows:

Contribution I. We addressed our first research question; to what extent does the huge available non-annotated data in training helps build robust deep learning method suitable to the complexity of medical images? In summary, we successfully incorporated a vast amount of unlabeled data and a few labeled ones to augment the model with new data generated by mixing both data types. This approach is dealt with by a well-known research direction, namely semi-supervised learning [49]. Specifically, our method generates new virtual data points by performing linear interpolation between labeled and unlabeled data. Similarly, the pseudo labels for the new data are generated. These new data are created in the input space and the hidden spaces. Then we train the model by leveraging both data types (*i.e.* , labeled and pseudo-labeled) in a semi-supervised medical image segmentation task, including five public datasets of brain MRI images and two lung infection CT datasets. This contribution is covered in detail in **Chapter 4** of this dissertation.

Contribution II. To answer the second research question, we explore utilizing knowledge sharing distilled in multiple locations without violating privacy issues in the so-called federated learning [193]. In this contribution, peer learning [267] from educational sciences and ensemble averaging from committee machine [268] were utilized to build a semi-supervised federated learning framework [130]. Peer learning enables our method to exchange knowledge between similar clients, while ensemble averaging enables us to create anonymized peer that reduces communication and hides clients' identities. Further, we proposed static and dynamic learning policies to control client learning streams. Our application of this approach is to classify skin lesions in a database of more than 72,000 dermoscopic images distributed to ten clients. This dissertation's details of this contribution are described in **Chapter 5**.

Contribution III. We answer the third research question, Can we mine and extract the knowledge from unlabeled data to perform the same tasks achieved by the labeled ones and handle labeled data shortage? We address this part by employing the representations learning [25] of the unlabeled data before fine-tuning the model with a few annotations in the so-called self-supervised learning methods. In this approach, we build upon current contrastive learning methods [132, 206]. Then, we propose an auxiliary task that composes new data from the original ones, then learn the model to regress the percentage of such composition of the mixed-up images. Further, we propose a self-consistency term that forces the linearity and consistency between composed and original embeddings for better training. Finally, we test our method on eight public datasets, achieving superior accuracy in different downstream tasks. All these details are found in **Chapter 6** of this dissertation.

1.4 Outline

This dissertation is organized as follows:

Chapter 1: ***Introduction.*** In the first part of this dissertation, we introduce the motivation and context of our work, problem description, contributions and solutions to the stated problem, and the outline of this dissertation.

Chapter 2: ***Background.*** The first section introduces medical imaging, its definition, modalities, applications, and challenges. Then, in the second section, we give a brief history of computer-based methods focusing on deep learning.

Chapter 3: ***Learning Paradigms.*** In the third chapter, we introduce the essentials and relevant technical details of deep learning in general, including the common learning paradigms; Supervised, Unsupervised, Semi-supervised, Self-supervised, and Federated Learning paradigms.

Chapter 4: ***Data Augmentation via Random Linear Interpolation in Semi-Supervised Learning***. This chapter presents **Contribution I**. The motivation, contributions, and related works are covered in sections 4.1, 4.2, and 4.3, respectively. The method is described in section 4.4, and the experiments and results are presented in section 4.5 and discussed in section 4.6.

Chapter 5: ***Knowledge Sharing via Static & Dynamic Peer Learning in Semi-Supervised Federated Learning***. This chapter covers the second contribution. The motivation is mentioned in section 5.1, while our contributions and related works are presented in sections 5.2 and 5.3, respectively. The method is presented in section 5.4, while the experiments and results are reported in section 5.5 and discussed in section 5.6.

Chapter 6: ***Representations Learning via Virtual Embeddings and Self-consistency in Self-Supervised Learning***. Our last **Contribution III** is presented in this chapter. The motivation is mentioned in section 6.1, the contributions are presented in 6.2, while related works are in section 6.3. The methodology is presented in chapter 6.4, the experiments and results reported in chapter 6.5, and discussed in chapter 6.6.

Chapter 7: ***Conclusion & Future Works.*** In the last chapter of this dissertation, we summarize our contributions' findings and observations in section 7.1. While we present our vision of the possible future directions in section 7.2.

2 Background

> *Great discoveries are made accidentally less often than the populace likes to think*
>
> **— Wilhelm Conrad Röntgen (1845–1923)**
> (German mechanical engineer and physicist and the first Nobel Prize Winner in Physics in 1901, in recognition of the extraordinary services he has rendered by discovering the X-rays subsequently named after him)

2.1 Medical imaging

2.1.1 Definition

Medical imaging is described as the procedure of capturing the internal body organs and tissues for clinical use, medical intervention, as well as visual monitoring of the function of some organs or tissues [258]. Medical Image plays a fundamental role in the medical field [2] since it provides a tool to examine different diseases [242], quantify human organs [213], therapy planning [202], tumor development monitoring [88, 146], diagnostic aid systems [70], and intra-operative assistance [117].

These wide ranges of applications led to a great interest in the medical image. Globally, the market size of medical imaging is estimated at 20.1 billion USD in 2021, and it is expected to grow by 5.2% in 2028 [251]. Further, the revenue forecast is expected to be USD 28.6 billion in 2028 [251]. At the same time, the high interest in using medical imaging comes from the increasing demand for early-stage diagnosis of chronic disease and rising aging demographics, which is expected to boost the demand for diagnostic imaging across the globe.

2.1.2 Modalities

Medical images include a broad spectrum of technologies such as magnetic resonance imaging (MRI), X-ray radiography, ultrasound, endoscopy, positron emission tomography (PET), computed tomography (CT) scans, optical coherence tomography (OCT), dermoscopy, and others. While covering all these modalities is out of the scope of this thesis, next, we only brief the commonly used ones.

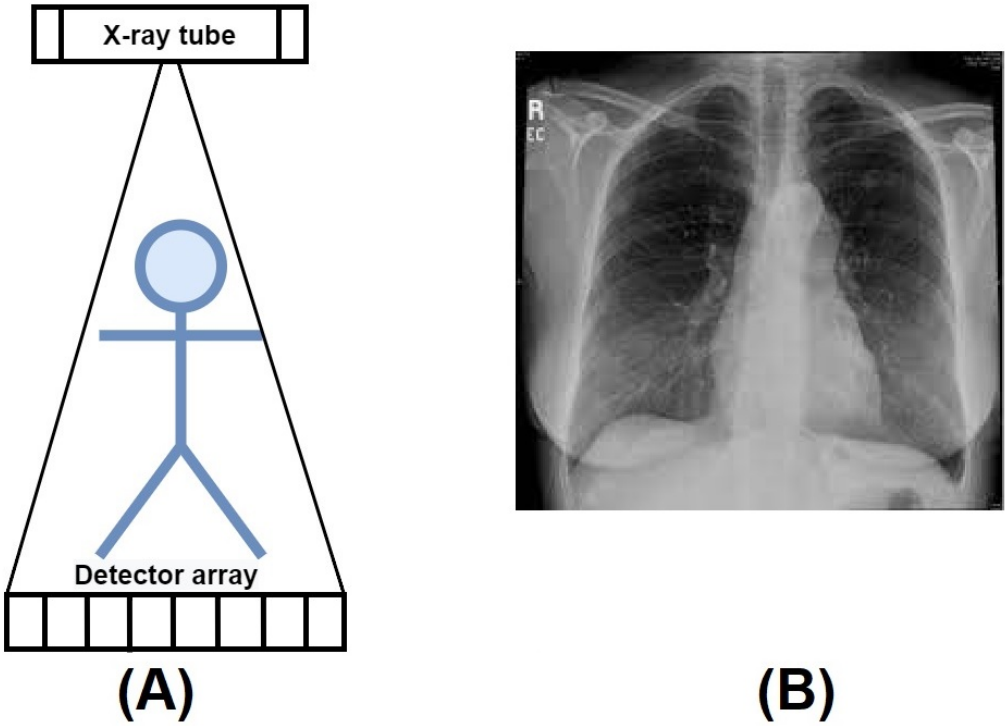


Fig. 2.1. (A) an Illustrative diagram of the working principle of the X-ray machine and (B) a sample X-ray image.

X-ray, the most widely used medical imaging modality, was discovered by W.C. Röntgen in 1895. X-rays take the form of ionizing radiation with a usual energy range between 25 keV and 500 keV. A traditional X-ray machine has an X-ray pipe that sends a short pulse of X-rays that travels through the human body. Those X-ray pulses that are not absorbed or scattered reach a large area detector creating an image on a film [195]. An Illustrative diagram of the working principle of the X-ray machine and a sample X-ray image for the chest are shown in Fig. 2.1.

Ultrasound consists of sound waves with high frequencies significantly more than the range of human hearing (>20,000 Hz). Ultrasonic images, also known as sonograms, are created by sending ultrasound pulses into tissue using a probe. The ultrasound pulses echo off tissues with different reflection properties and are returned to the probe, which records and displays them as images. Many different types of images can be created. The most standard is a Brightness or B-mode image, which shows the acoustic impedance of a two-dimensional cross-section of tissue [75, 286]. An Illustrative diagram of the working principle of the ultrasound machine and a sample image of a pregnant woman are shown in Fig. 2.2.

A computed tomography scan, introduced by Hounsfield in 1972, and also known as a **CT** scan, is a non-invasive form of x-rays used to obtain detailed internal images of the body for diagnostic purposes. CT scanners take multiple X-ray measurements from various angles using a row of sensors and a rotating X-ray pipe positioned in the gantry to measure X-ray attenuations by different tissues inside the body. These measurements are then treated on a computer using reconstruction algorithms to produce cross-sectional or tomographic images of a body [285]. An Illustrative diagram of the working principle of the CT machine and a sample CT scan for the lung are shown in Fig. 2.3.

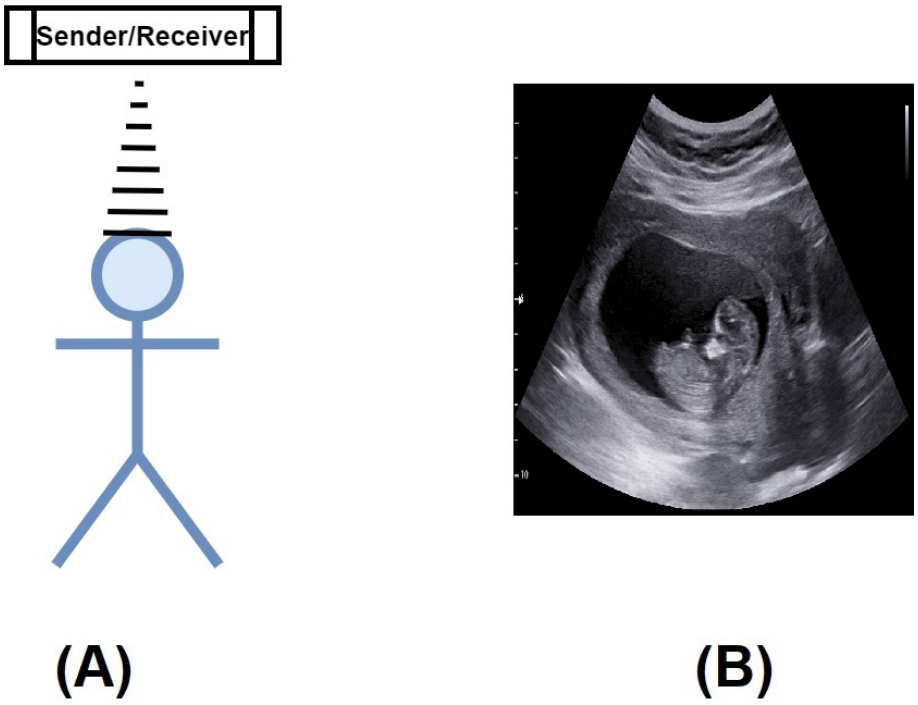


Fig. 2.2. (A) Illustrative diagram of the working principle of the ultrasound machine and (B) a sample ultrasound image for a pregnant woman.

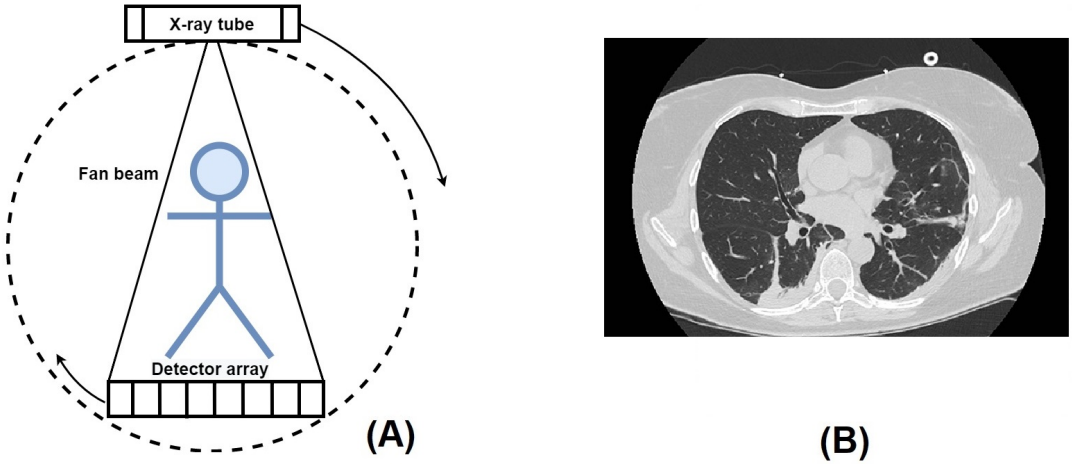


Fig. 2.3. (A) an Illustrative diagram of the working principle of the CT machine and (B) a sample CT scan for the lung.

Magnetic resonance imaging (MRI) uses the property of nuclear magnetic resonance, where powerful magnets are emitted to excite hydrogen nuclei of water molecules in human tissue, producing a detectable signal spatially encoded, resulting in images of the body. Specifically, a Radio Frequency (RF) device sends a pulse to the region of interest in the body to be examined. Once protons absorb the RF pulse, the RF pulse changes its direction to the original magnetic field. When the RF pulse is turned off, the protons align back to the primary magnet and emit radio waves. This radio-frequency emission from the hydrogen atoms in the water of

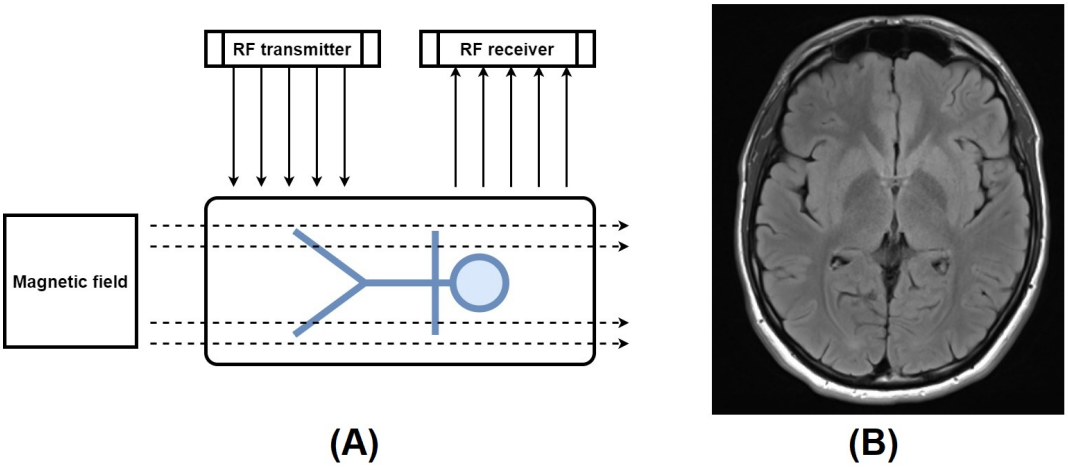


Fig. 2.4. (A) An Illustrative diagram of the working principle of the MRI machine and (B) a sample MRI of the brain.

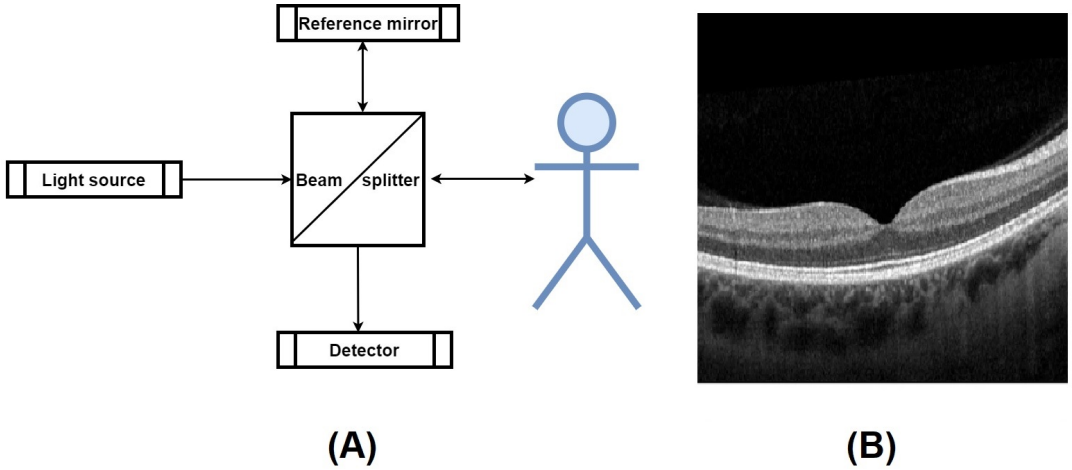


Fig. 2.5. (A) An Illustrative diagram of the working principle of the OCT machine and (B) a sample OCT for the retinal image.

human tissue is caught and rebuilt into an image [195]. An Illustrative diagram of the working principle of the MRI machine and a sample MRI for the brain are shown in Fig. 2.4.

Optical coherence tomography (OCT) is a non-contact non-invasive technique for cross-sectional tissue imaging with a high resolution of 20–5 μm, much higher than other medical imaging modalities like MRI. OCT has similar working principles to ultrasound images. However, it typically uses light in the near-infrared spectral field, with a penetration depth of several hundred microns in tissue. The reflected light is computed with an interferometric set-up to reconstruct the depth profile of the example at the region of interest. Because OCT uses light in the near-infrared, it travels much faster than ultrasound [13]. An Illustrative diagram of the working principle of the OCT machine and a sample OCT for the retinal are shown in Fig. 2.5.

Dermoscopy, also known as epiluminescent microscopy or surface microscopy, is a method that allows the visualization of pigmented cutaneous lesions of the reticular dermis. This

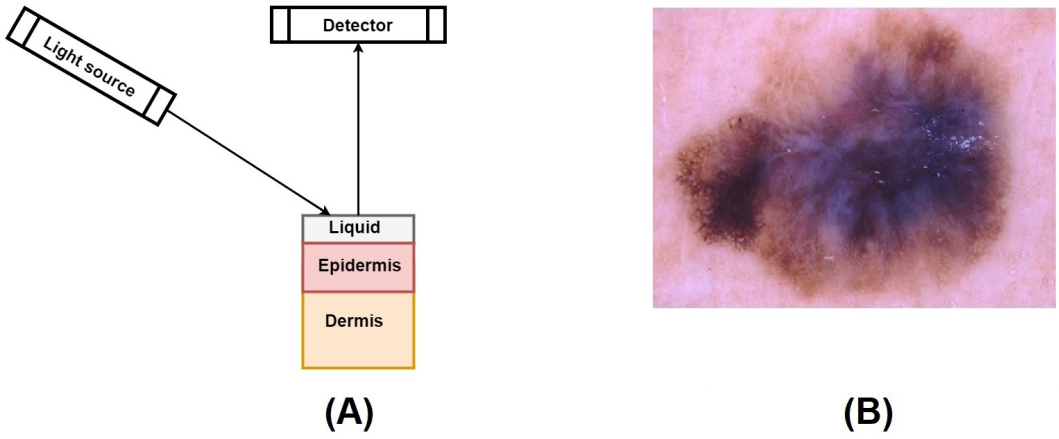


Fig. 2.6. (A) An Illustrative diagram of the working principle of the dermoscopy machine and (B) a sample skin lesion.

technique includes using a device similar to an otoscope but with a specific contact lens. The device generates a beam of light sent on the cutaneous surface at an angle of 20°. Light reflection is removed by putting a fluid at the interface between the epidermis and the device's glass slide. The visualization of the dermoscopic features results from the existence of hemoglobin and melanin in the various skin layers [44]. An Illustrative diagram of the working principle of the dermoscopy machine and a sample skin lesion are shown in Fig. 2.6.

2.1.3 Applications

The success of medical imaging and the advances in computing power enhanced the research in the direction of image-processing solutions for medical pathology. Over the past decades, several applications have been developed and improved to assist radiologists in this new era of medical imaging. The following list includes the most common medical imaging applications developed for clinical and research purposes.

Organ and tissue segmentation: Medical image segmentation involves the extraction of regions of interest (ROIs) or areas of the anatomy required for a particular study. These ROIs include but are not limited to (i) lung [42] and bone [77, 125] in X-rays, (ii) Breast cancer [124] in Ultrasound images, (iii) abdominal organs such as liver, spleen, kidneys, pancreas, gallbladder, and aorta in CT scans [83, 321], (iv) skin lesion in dermoscopic images [287, 305], (v) brain tissue [80, 226], stroke lesions [136] and cardiovascular and heart [211] in MRI, and (vi) retinal layers in OCT [111].

Disease classification and diagnosis: In many cases, medical images are rich with texture, and morphological features, which are increasingly used in healthcare for diagnosing, classification, planning, guiding treatment, and monitoring disease progression [39, 153, 169, 180, 282, 296].

Abnormality Detection: It is the identification of irregular makers or observations that deviate from standard data distribution *i.e.* out-of-distribution, such as a tumor or lesion [22, 23, 237, 330].

Registration: As medical images are increasingly used in the medical health sector, in many of these studies, multiple images are obtained from patients at different times, often with various imaging modalities. The information provided by the various imaging modalities is often complementary. Therefore, possible uses in improving how these images are compared and combined, aka image registration. Medical image registration is used in considerable clinical applications, such as motion tracking, image guidance, dose accumulation, image reconstruction, segmentation, and many other applications [92, 114, 189, 190, 276].

2.1.4 Challenges

Over the last few years, medical images have noticed significant improvements due to their wide range of applications and crucial role in detecting, diagnosing, and treating diseases. In the clinic institutes, human experts such as radiologists and physicians primarily perform medical image interpretation and analysis. Medical image analysis aims to extract information effectively and efficiently for improved clinical diagnosis. In addition, the current advancements in biomedical engineering have made medical image analysis one of the highest research and growth areas. Still, manual analysis of medical images is not easy and involves many challenges.

2.2 Deep Learning

Due to the previous limitations of manual medical data analysis mentioned in section 2.1.4, researchers have started investigating a computer-based approach to achieve accurate, fast, cheap, and consistent analysis. Thus, machine learning, including deep learning methods, are the keystone of today's artificial intelligence (AI) advancement, brings new improvements to clinical practice with medical images [57, 180, 320]. For instance, machine learning has been shown to function on par with medical experts to analyze various diseases from medical images [184]. Further, software applications are starting to be licensed for clinical usage [239, 266]. The following sections briefly describe the automated analysis focusing on deep-learning-based methods.

2.2.1 Brief History of Computer-based Methods

Once the researchers were able to generate and provide medical images into a computer, they built automated-based systems for analysis. One early implementation of the computer-based methods of the 1970s in medicine was by Shortliffe [247], which led to the development of rule-based, expert systems to suggest different antibiotic therapies for patients. In the following attempts to the 1990s, medical image analysis was done with sequential application of low-level pixel processing such as region growing, thresholding, and edge and line detector

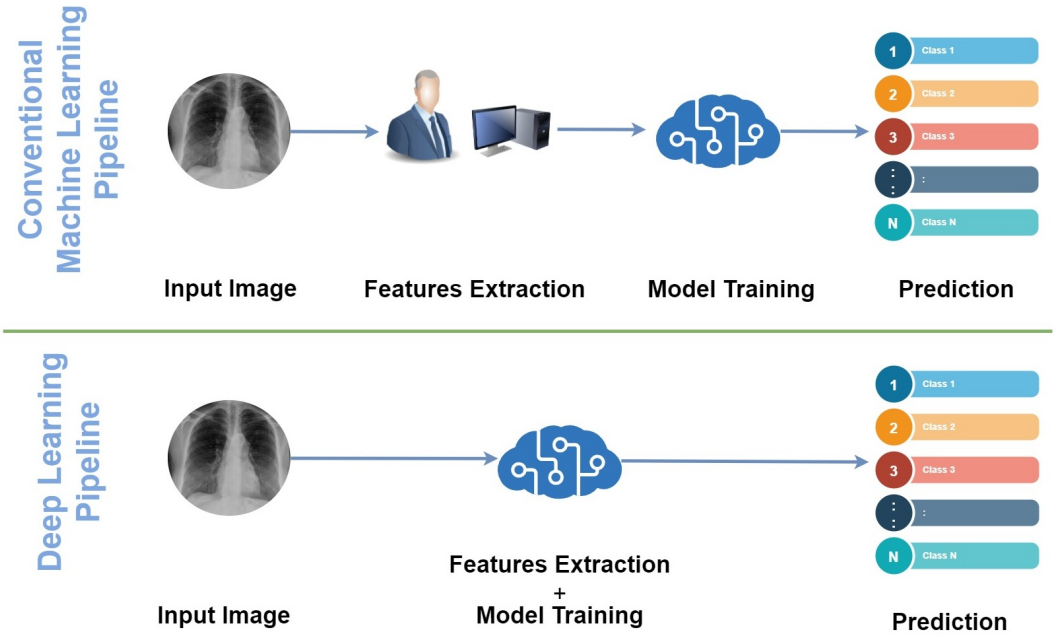


Fig. 2.7. Machine and Deep Learning Methods Pipelines.

filters [212, 230], and mathematical models such as fitting lines, circles, ellipses, and Markov random field [82, 212].

By the end of the 1990s, AI algorithms progressed from heuristics-based techniques to manual feature extraction techniques, where training data is used to design a system in so-called supervised learning techniques. These computer algorithms determine the optimal decision boundary in the high-dimensional feature space. Consequently, extracting discriminant features from medical images is a vital stage in designing such systems. Until that point, this process is still done by human researchers, a.k.a handcrafted features. Examples include active shape models and atlas methods [145]. Because the success of these methods depends on the quality of extracted features, most efforts of the researchers were focused on finding the most discriminating features.

The logical next evolution is to computerize the feature extraction step. This idea lies at the root of many deep learning algorithms; artificial neural networks are composed of many layers that receive and transform input data (e.g., medical images) into outputs or predictions (e.g., disease present/absent) without any manual features extraction steps in the so-called end-to-end learning approach. In Fig. 2.8, we present an illustrative diagram showing the conventional machine learning methods pipeline compared to the deep learning ones. Because deep learning methods relax the need for a manual finding of the most discriminating features, the researchers focus on finding the best architecture that can automatically extract the most valuable and generalizable representations from the data. Thus, we summarize the most successful deep-learning architectures in medical images in the next sections.

2.2.2 Deep Learning Methods

In the last two decades, more and more deep-learning methods have been widely applied in medical imaging. This impressive success of deep learning over other machine learning techniques is attributed to the fact that many traditional machine learning methods rely on handcrafted texture or morphological feature extraction to identify the valid characteristics in the image. However, such designed features need human experts and intensive efforts. Moreover, the manual-designed features are often problem-specified and barely generalizable, i.e., not ensured to perform for other types of images [245]. Deep learning models consist of multiple linear and non-linear processing units organized in a deep architecture to capture high-level features presented in the data such that it can solve intricate problems, *i.e.* computer vision and medical image tasks that were very hard to solve in the past.

2.2.3 Basic Building Blocks

Perceptron

Perceptron or artificial neuron is the most fundamental building block of all deep learning models; see Fig. 2.8. Inspired by the human and animal nervous system's biological neurons, a single perceptron consists of a mathematical operation that performs the weighted sum of multiple inputs and computes activation value as output. Mathematically, we can represent the perceptron by a function $f(\mathbf{x})$, where it receives a vector of inputs $\mathbf{x} \in \mathbb{R}^N$. Each of these inputs is multiplied by its corresponding weight. Thus, we need a vector of weights, $\mathbf{w} \in \mathbb{R}^N$. The results are then summed and added to a so-called bias term $b \in \mathbb{R}$ analogously capture the neuron-specific potentials. The resulting mathematical function $f(\mathbf{x})$ simulates neural behavior with a simple linear combination of $\mathbf{w}$, $\mathbf{x}$, and b. Thus, our perceptron can estimate linear functions and solve linearly separable tasks such as linear regression. However, to add more abilities such as binary classification tasks, non-linear and differentiable activation function $\sigma(.)$ is attached to the previous result where the activation function evaluates output against threshold τ, which leads to the neuron's final output:

$$f(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b) = \sigma(w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b) \tag{2.1}$$

Note that bias and weights are learnable parameters. Conventionally, training the parameters is done by solving a least-squares optimization problem based on the input data X with corresponding labels Y. Initially; the weight and bias values are set randomly. Then, as training progress, both parameters are updated closer to the desired values, leading to more accurate output. Both bias and weight parameters have their influence on the training. While weights describe the strength and importance of the input data, bias represents how close or far the predictions are to their actual value. Weight shows how the output will behave regarding the changes in the input. A small weight value represents the input's minor importance, while a more considerable weight value conveys a more significant input value. On the other hand, a slight bias means that the network is making more hypotheses about the shape of the output, whereas a high bias value makes fewer hypotheses about the form of the output.

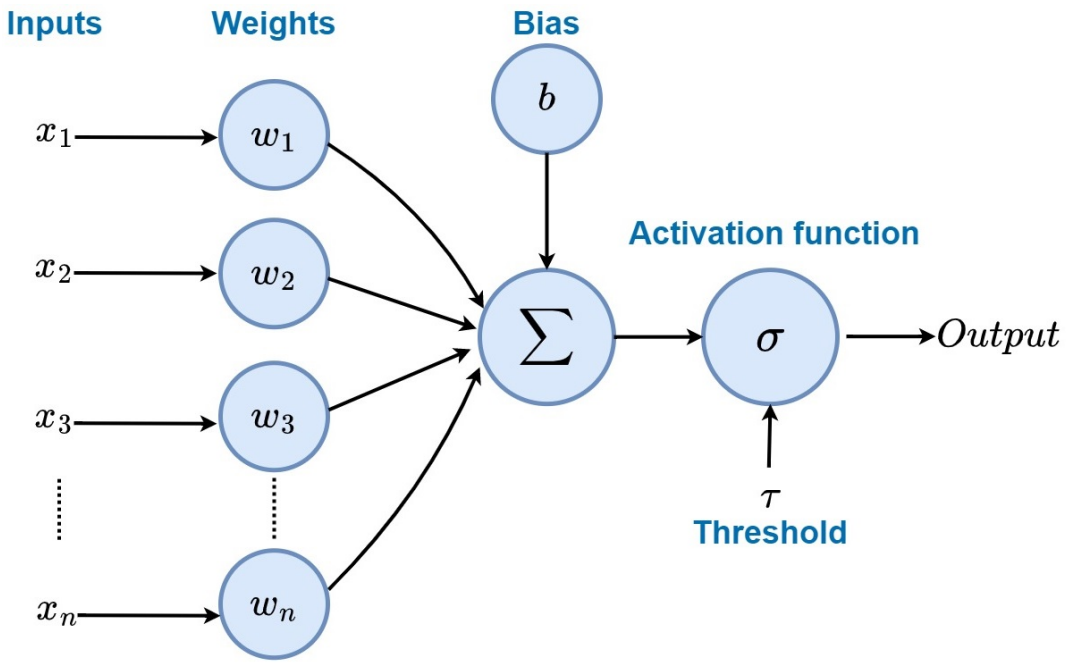


Fig. 2.8. A graphical illustration of the Perceptron.

Artificial Neural Networks

Thus far, we have seen that a single perceptron can approximate simple functions. However, to add more capabilities to estimate more complex and non-linear functions, one can interconnect more than artificial neurons to form so-called artificial neural networks (ANN). Neurons inside the network are arranged in three layers; the input layer, the hidden layer, and the output layer (see Fig 2.9. A). As its names suggest, the neurons in the input layer receive the input data. In contrast, neurons in the output layer produce the predictions or the estimations. On the other hand, the neurons in the hidden layer receive the outputs from the input layer's neurons and feed them to the output layer. Each neuron in a layer has its weight and shares one bias with other neurons in the same layer. Yet, no biases are shared across different layers. Mathematically, Eq. 2.1 can be extended to include all neurons in ANN and is given by:

$$f(\mathbf{x}) = \sigma(\mathbf{w}_o^T \sigma(\mathbf{w}_h^T \mathbf{x} + b_h) + b_o), \tag{2.2}$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}_h$ and $\mathbf{w}_o$ are vectors containing the weights for the hidden and output layers, receptively, and b_h and b_o are the biases for the hidden and output layers, receptively.

The Multilayer Perceptron

While adding more neurons extended the abilities of a single perceptron, the researchers have found that attaching more hidden layers also increases the power of our neural network to solve even more complex tasks, which leads to the invention of the Multilayer Perceptron (MLP), see Fig 2.9. B. The MLP stacks multiple but at least two hidden layers to form a neural network of L layers. Consequently, Eq. 2.2 becomes:

$$f(\mathbf{x}) = \sigma(\mathbf{w}_L^T(\dots \sigma(\mathbf{w}_1^T \sigma(\mathbf{w}_0^T \mathbf{x} + b_0) + b_1) \dots) + b_L) \tag{2.3}$$

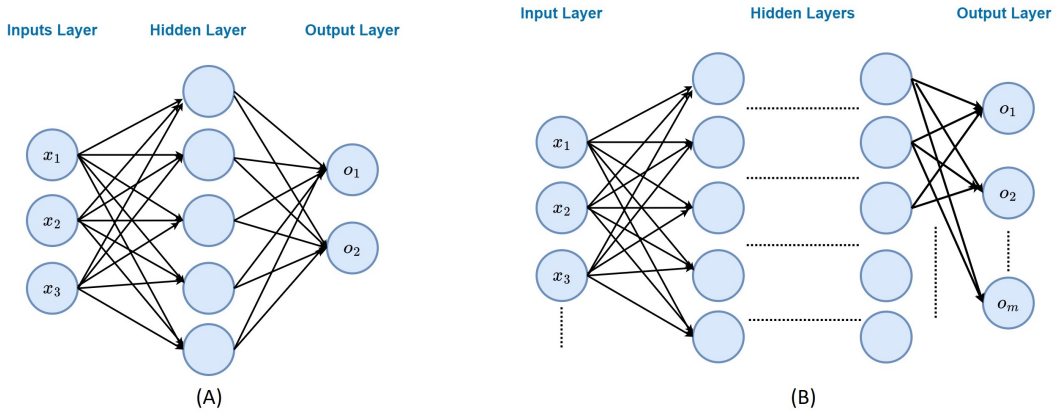


Fig. 2.9. A graphical illustration of the (A) Artificial Neural Networks, and (B) Multilayer Perceptron.

Usually, the layers from 1 to $L-1$ are denoted as hidden layers, while layers 0 and L represent the input and output layers of the MLP. As the computation power has increased over the last two decades, researchers have used more and more hidden layers. Hence, deep neural networks, so-called deep learning, have rapidly evolved to solve real-life applications such as computer vision, speech recognition, natural language processing, medical imaging, and many more.

Activation Functions

We have mentioned that the artificial neuron takes input data or the output from a previous neuron, then performs a linear transformation, *i.e.* , multiplies the input data with weight, then adds results to bias. This operation is followed by an evaluation of the activation function, leading to the neuron's output, which is forwarded to the next layer or neuron. The most straightforward activation function is linear activation, where no transform is applied. A network of only linear activation functions is directly trained but cannot learn complex mapping functions. However, linear activation functions are still used in the output layer for networks that predict a quantity (e.g., regression problems). On the other hand, the goal of the non-activation functions is to add more capacities to the artificial networks such that they can achieve more complex tasks. Note that the activation functions are only used in the output and the hidden layers. Although all hidden layers typically use the same activation function, the output layer uses a different activation function from the hidden layers depending on the required task. Next, we are listing the most commonly used activation function.

Sigmoid function [33]: The sigmoid, or the logistic function, is the most known and widely used activation function for a long time, and it is known for its S-shape curve. This function takes any real value as input and outputs values from 0 to 1. Inputs larger than 1.0 are cropped to the value 1.0, whereas values smaller than 0.0 are cut to 0.0. We can use the sigmoid function to separate the input space into two groups where the threshold is 0.5, which makes it excellent for binary classification problems in the output layer, see Fig. 2.10. (A).

Hyperbolic tangent function [33]: The hyperbolic tangent function, a.k.a tanh, has a similar S-shape to the sigmoid function; see Fig. 2.10. (B). However, the output values are between -1.0 and 1.0. The tanh function is easier to train, has a better predictive performance than sigmoid, and is usually used in the hidden layers.

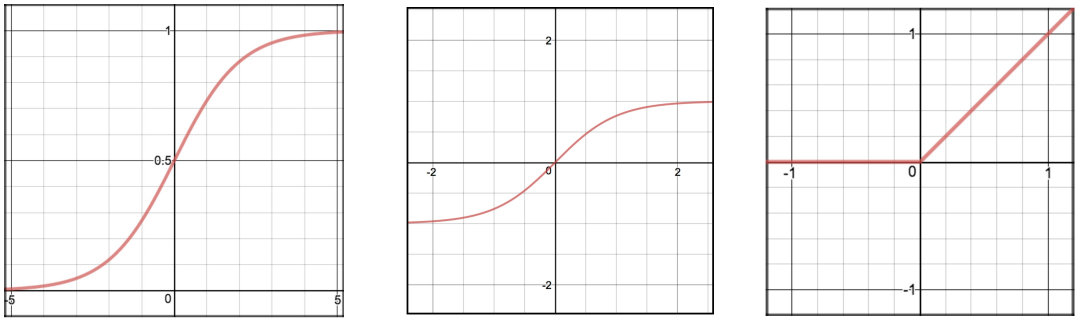


Fig. 2.10. A graphical illustration of the most commonly used activation functions.

Both the sigmoid and tanh functions suffer from a saturate problem. Large values cut to 1.0, while small values cropped to -1 for tanh and 0 for sigmoid. Another drawback is that both functions are less sensitive to changes beyond their mid-point, *i.e.* , 0.5 and 0.0 for sigmoid and tanh, respectively. With such limitations, it becomes challenging for the learning algorithm to continue to adapt the weights to improve the performance of the model or the so-called vanishing gradients problem.

Rectified linear unit [198]: Also known as the ReLU activation function, it is considered the most used function for hidden layers. ReLU performs a simple calculation that returns the input value if it is greater than zero or 0.0 if the input is equal or less than zero. This property enables ReLU to overcome the limitation of sigmoid and tanh activations. Specifically, it is less prone to vanishing gradients problems, see Fig. 2.10.(C).

Softmax function [33, 101]: In contrast to the previous functions, the softmax function, Eq.(2.4), outputs a vector of values that sum to 1.0 that can be interpreted as probabilities of class membership. This feature makes the softmax function the best fit in the output layer and the most popular function for multi-class prediction tasks.

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{2.4}$$

The common property of the previous activation functions is that they are differentiable for a given input value. This property makes the neural networks trainable using a learning algorithm such as the backpropagation algorithm to adjust the model's parameters toward the best results.

Backpropagation

By far, the most used algorithm for training. Backpropagation was proposed in 1986 [228] and is an intelligent and efficient way of calculating the partial derivatives of the loss function with respect to the learnable parameters, *e.g.*, weights, and biases. After the gradients or derivatives calculation, the network's parameters are gradually adjusted toward the optimal

solution that minimizes the loss function using a learning rate step using the so-called gradient descent optimization algorithms [227]. Nowadays, all training methods are based on different variants of gradient descent (*e.g.*, stochastic gradient descent [218], momentum [214], and Adam [147]).

Consider a neural network and training data consisting of input X and their labels Y. Also, consider a loss function $\mathcal{L}$ that measures the difference of the model's output for every training sample from the target value of Y. Our goal is to optimize our loss function by searching in the parameters space using the backpropagation algorithm. Mathematically, backpropagation makes use of the chain rule for computing the derivatives. For instance, given an input example x_i and its label y_i, the algorithm calculates the derivative of the loss with respect to every parameter of the network $f_L(X;\theta)$, where θ is the network parameters, and L is the total number of layers:

$$\frac{\partial}{\partial\theta}\mathcal{L}(f_L(x_i),y_i) = \frac{\partial}{\partial\theta}\mathcal{L}(f_L(f_{L-1}(\dots f_1(x_i))),y_i) \tag{2.5}$$

The above equation can be extended to compute the derivatives of the loss w.r.t. a parameter $\theta_{l,j}$ of the layer l:

$$\frac{\partial}{\partial\theta_{l,j}}\mathcal{L}(f_L(x_i),y_i) = \frac{\partial}{\partial f_L(x_i)}\mathcal{L}(f_L(x_i),y_i).\frac{\partial}{\partial f_{L-1}(x_i)}f_L(x_i).\dots.\frac{\partial}{\partial\theta_{l,j}}f_l(x_i) \tag{2.6}$$

After calculating the derivatives w.r.t. all network parameters via backpropagation, an iterative approach, the so-called gradient-descent, is used to update each parameter in the direction of the opposite derivatives. For example, to update the parameter $\theta_{l,j}$ of the layer l, the following simple updating rule is used:

$$\theta_{l,j} = \theta_{l,j} - \eta\frac{\partial}{\partial\theta_{l,j}}\mathcal{L}(f_L(x_i),y_i), \tag{2.7}$$

where η is a hyperparameter called learning rate that controls the fraction of the gradient used to update the parameters. The learning rate may be the most significant hyperparameter when training our neural network. Selecting too small values may result in a long training process, whereas too large values may lead to an unstable training process.

2.2.4 Common Loss Functions

The loss or cost function measures how well our learning algorithm predicts the data by calculating the difference between the predicted values and actual labels. Thus, the lower the loss function, the better our model is. Of course, no loss function fits all algorithms in machine learning. However, various factors are involved in choosing a loss function for a specific problem, such as the type of machine learning algorithm and the required task. Assume our dataset consists of n training samples, where $X = \{x_1, x_2, \dots, x_n\}$ is the training data, $Y = \{y_1, y_2, \dots, y_n\}$ is the corresponding labels set, *i.e.* ground truth. For each input data x_i, $\hat{y}_i$ represents the predicted label produced by our network such that $\hat{y}_i = f_L(x_i;\theta)$.

Mean Square Error Loss (MSE): Also known as squared L2 norm, mainly used in regression and image reconstruction tasks, it measures an average of squared differences between the ground truth and the predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{2.8}$$

The MSE loss function punishes the model for making errors by squaring them. This feature makes the MSE loss function less robust to outliers. Therefore, using it with a dataset with many outliers is not recommended.

Mean Absolute Error(MAE): Also known as L1 norm, mainly used in regression and image reconstruction tasks, it measures an average of absolute differences between the ground truth and the predicted value. Compared to the MSE loss function, the MAE loss function is more robust to outliers.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{2.9}$$

Binary Cross Entropy Loss (BCE): Used for binary classification and segmentation tasks. It measures the accuracy of a classification model whose output is a probability value between 0 and 1. An essential characteristic is that cross-entropy loss heavily punishes predictions that are wrong but confident.

$$BCE = -(y_i.\log(\hat{y}_i) + (1 - y_i).\log(1 - \hat{y}_i)) \tag{2.10}$$

Cross Entropy Loss (CE): An extended version of the BCE loss used in multi-class classification and segmentation tasks.

$$CE = -\frac{1}{n}\sum_{i=1}^{n} y_i.\log(\hat{y}_i) \tag{2.11}$$

One limitation of the cross-entropy loss is that the calculation is done discretely, without considering whether adjacent pixels are boundaries, mainly when used for segmentation tasks.

Dice Loss: Dice loss originates in or Sørensen–Dice coefficient. It measures the dice coefficient between the predicted and the actual values. In statistics, the Dice coefficient measures the overlap between two sets over their union. For example, if two sets, A and B, overlap perfectly, the dice get their maximum value of 1. In contrast, the dice value equals 0 when there is no overlap. Therefore, Dice loss considers local and global information, making it suitable for segmentation tasks.

$$DSC = 1 - \frac{1}{c}\sum_{i=0}^{c}\frac{\sum_j^n 2y_i^j\hat{y}_i^j + \epsilon}{\sum_j^n y_i^j + \sum_j^n \hat{y}_i^j + \epsilon} \tag{2.12}$$

c is the number of classes, and ϵ is a small number added to avoid division by 0.

Kullback-Leibler divergence Loss: The Kullback–Leibler divergence, also called relative entropy, measures how one probability distribution differs from a second reference probability distribution—mainly used to measure the distance between two continuous distributions.

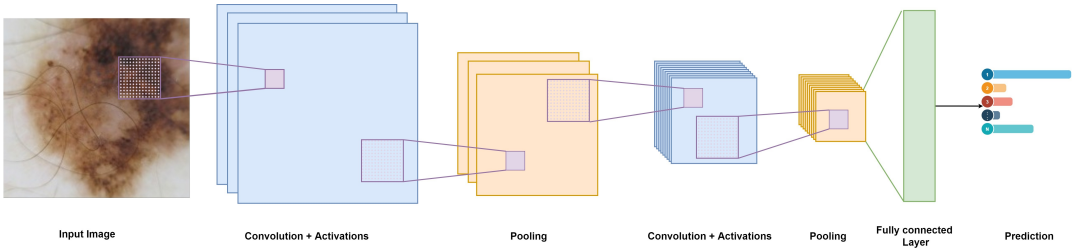


Fig. 2.11. A graphical illustration of the convolutional neural network (CNN).

Hence, suitable for regression tasks. In the simple case, when the Kullback–Leibler divergence equals 0, it indicates that the two distributions have identical quantities of information.

$$KLD(\hat{y}||y) = \sum_{i=1}^{n} \hat{y}_i \log \frac{\hat{y}_i}{y_i} \tag{2.13}$$

2.2.5 Common Deep Architectures

Convolutional Neural Network (CNN) and its variants

We have shown in previous sections that ANNs can be used to achieve different machine-learning tasks. However, at the heart of any successful ANN is the number of parameters used to train the model, which becomes more significant as the data gets complicated. For example, data comes in high-dimensional space in vision tasks such as detecting diseases or tumor segmentation in medical images. In such cases, the number of parameters and computation power needed to train a good ANN model becomes very high. Fortunately, these limitations have been mitigated by proposing the convolutional neural network (CNN) [165].

In a nutshell, A convolutional neural network (CNN) is a fundamental building block in any modern deep-learning architecture for computer vision. CNN uses a mathematical operation called convolution in place of general matrix multiplication. Building CNN architectures has three main layers: Convolutional Layer, Pooling Layer, and Fully-Connected Layer, besides the input and output layers. The Convolution Layer and the Pooling Layer are used as Feature-Extraction components, while the Fully Connected Layer is just simple neural network architecture and is used to perform the prediction task based on the convolutional block's input. At the core of the CNN is the convolution operation, which remarkably decreases the number of parameters used in ANNs, simultaneously captures the spatial information in the images, and learns appropriate representation for the task in an end-to-end fashion. A standard CNN consists of multiple layers stacked at each other and composed of different building blocks; see Fig. 2.11. In the following, we briefly present the most important ones.

Convolutional Layer: A core building block of the CNN and possess the major part of the computational load of the network. The convolution operation performs a dot product between two matrices, the kernel or filter matrix (typically 3x3 matrix), and a smaller part of the input data, a.k.a receptive field, see Fig. 2.12. The filter consists of learnable parameters that are initialized randomly using some initialization algorithms at the beginning of the training, such as Xavier [100]. Then backpropagation algorithm is used to update the parameters

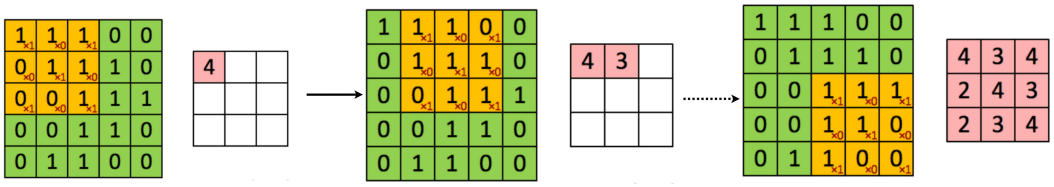


Fig. 2.12. A graphical illustration shows convolutional steps. (Source. https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/)

after each optimization step. During the forward pass, the kernel slides across the height and width of the image, producing the image representations or so-called activation or feature map of that receptive field. Then, the filter is moved with a defined step (called stride) to apply the operation to each filter-sized patch of the input image at the end. Consequently, these representations capture helpful information at each spatial position of the input data. Moreover, the researchers proposed adding zeros pixels around the input's border, called Zero-padding, to preserve the image size and the information at the edge. Usually, multiple convolutional layers are stacked together and then followed by an activation function such as the ReLU function [198].

Three hyperparameters control the output size of the convolutional layer; depth, stride, and padding. The depth represents the number of filters we would like to use; each filter learns to look at something different in the input, such as various oriented edges or blobs of colors. On the other hand, the stride is responsible for sliding the filter. So, for instance, when the stride is 1, we move the filters one pixel at a time, and when the stride is 2, the filters jump 2 pixels simultaneously as we slide them around. This will produce smaller output volumes spatially. Finally, padding adds Zeros around the borders. Adding padding to a CNN-processed image provides more accurate image analysis and preserves the information at the borders. Formally, the convolutional layer accepts the volume of a shape of $W1 \times H1 \times D1$, *i.e.* , Width $\times$ Height $\times$ Depth, and requires four hyperparameters; the number of filters K, filter size F, the stride S, and the padding size P. Based on that, the output size after the convolution operation takes the shape of $W2 \times H2 \times D2$, where $W2 = (W1 - F + 2P)/S + 1$, $H2 = (H1 - F + 2P)/S + 1$, and $D2 = K$ (the output depth equals the number of filters K). Note that a common set of hyperparameters is F=3, S=1, P=1.

Convolutional neural networks are comprised of multiple convolutional layers. When we feed an image in a convolutional layer, each layer generates several activation functions passed on to the next layer. However, this operation generates a different level of abstraction related to the spatial location at which the kernel matrix focuses in the receptive field. The first layer usually extracts basic features such as horizontal or diagonal edges. Then, this output is passed on to the next layer, which detects more complex features such as corners or combinational edges. Finally, moving deeper into the network can identify more complex features such as objects, faces, etc.; see Fig 2.13.

Pooling Layer: Used between the convolutional layers and performs dimensionality reduction to the feature map. The benefits of this step are two folds. First, it reduces the computational power. Second, it extracts prevalent features, which are rotational and positional invariant, thus maintaining the process of effectively training the model. In practice, there are two

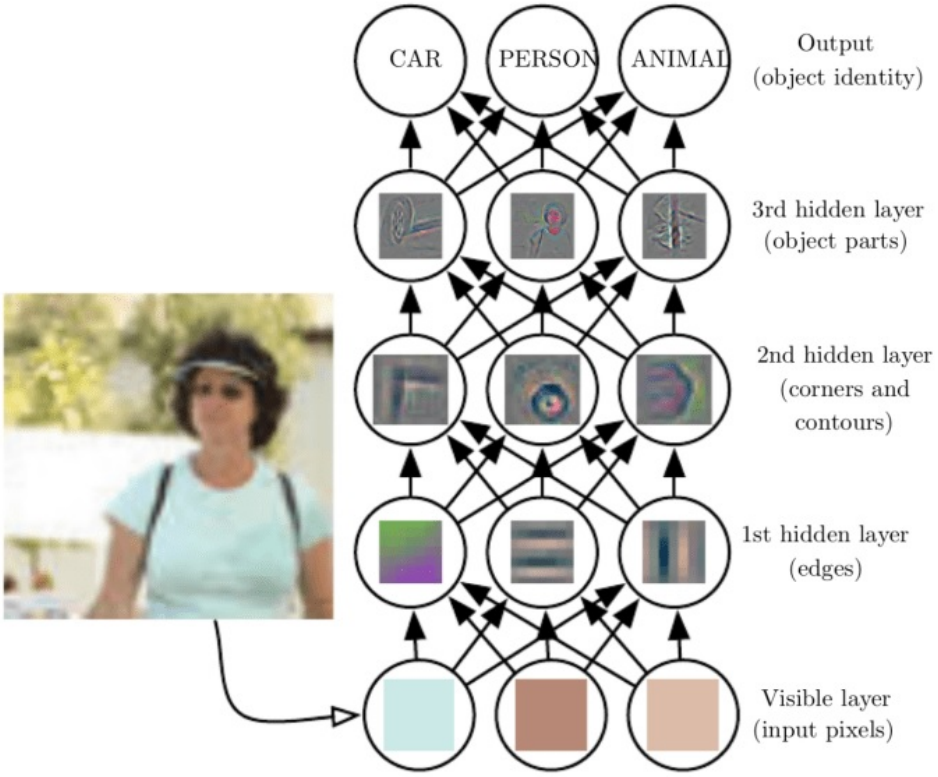


Fig. 2.13. A graphical illustration shows what different convolutional layers learn from the image. (Source. https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/)

types of Pooling: Max Pooling and Average Pooling. Max pooling returns the maximum value within the kernel window, while average pooling returns the average value; see Fig. 2.14. The pooling layer accepts input of a shape of $W1 \times H1 \times D1$, requires two hyperparameters, the spatial size F and the stride S, and produces an output of size $W2 \times H2 \times D2$, where $W2 = (W1 - F)/S + 1$, $H2 = (H1 - F)/S + 1$, and $D2 = D1$. Note that a standard set of hyperparameters is F=2, S=2, and it is not common to pad the input using zero padding.

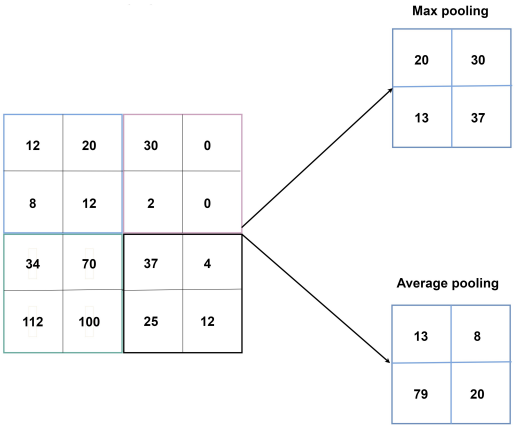


Fig. 2.14. A graphical illustration of the pooling layer, max and average pooling.

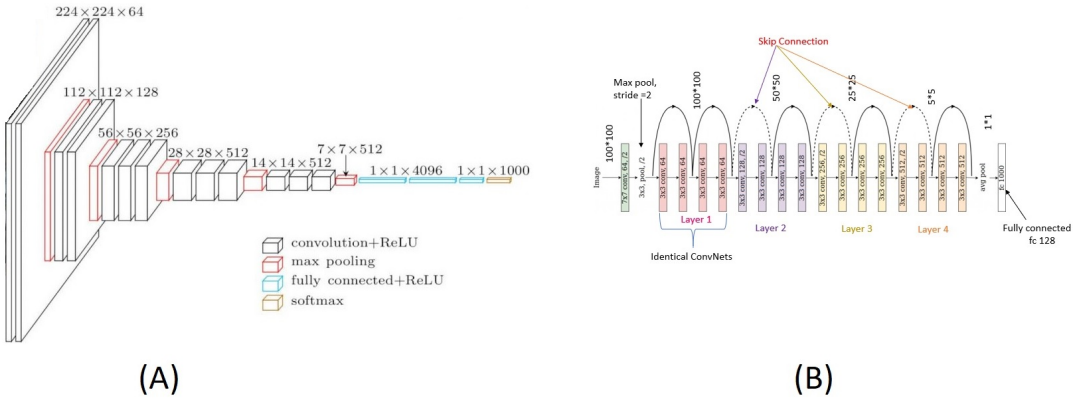


Fig. 2.15. A graphical illustration of (A) VGGNet [250], and (B) ResNet [110].

Fully Connected Layer: After a set of convolutional and pooling layers, we have successfully allowed the model to learn valuable features. The next step is to flatten the final result and feed it to a regular neural network or multilayer perceptron for prediction purposes, also called a fully connected layer. Flatten operation means converting the shape of the output to a column vector. For example, a tensor of shape $D1 \times D2 \times D3$ will be converted to a vector of the size $D \times 1$, where $D = D1 * D2 * D3$. The flattened result is passed to a fully connected layer, and backpropagation is applied to every training iteration. Over a sequence of epochs, the model can differentiate between high and low-level features in images and use them in the prediction.

CNN-based known architectures: Several architectures based on Convolutional Networks have become popular in computer vision. Among these are (i) **LeNet** [164]: one of the earliest convolutional neural networks that promoted deep learning development. Developed by Yann LeCun in 1989 and consists of 5 layers, two convolutional layers, and three fully connected or dense layers. LeNet used to read zip codes and digits. (ii) **AlexNet** [159]: was developed by Alex Krizhevsky in 2012 and was the first work that popularized Convolutional Networks in Computer Vision. AlexNet significantly outperformed the second runner-up, in the ImageNet ILSVRC challenge in 2012, by more than 10.8 percent. The Network shares a very similar architecture to LeNet. Still, it was deeper, bigger, and stacked Convolutional Layers on top of each other. (iii) **VGGNet** [250]: VGG stands for Visual Geometry Group and was developed by Karen Simonyan and Andrew Zisserman from Oxford. VGGNet depends on using a large number of convolutional layers and showed that they are essential of good performance. For instance, VGG16 contains convolutional and fully connected layers in an extremely homogeneous model that only utilizes 3x3 convolutions and 2x2 pooling in the whole network, see Fig. 2.15.(A). A drawback of the VGGNet is that it is expensive to evaluate and uses a huge memory and parameters (140M). (iv) **ResNet** [110]: or Residual Network proposed using the so-called skip connections or shortcuts. Skip-connection means skipping some of the layers in the neural network and passing the output from one layer directly to the following layers. It is used to avoid the vanishing gradients [116] and to mitigate the degradation (accuracy saturation) problem. On the other hand, ResNet does not use fully connected layers, while it heavily depends on batch normalization, see Fig. 2.15. (B).

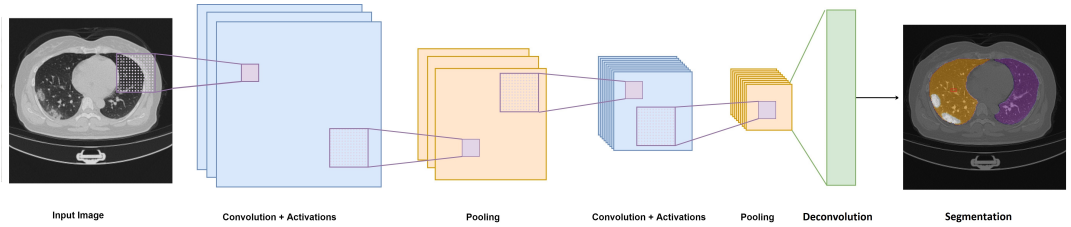


Fig. 2.16. A graphical illustration of the FCNN architecture. FCNN replaces the last classification layer by a deconvolution filter (up-sampling) for dense pixel-wise prediction.

Fully Convolutional Neural Network (FCNN)

We have shown that CNN-based architectures perform classification at the image level, *i.e.* , they assign one label for the whole image. However, we need to label each pixel in the image for medical image segmentation or segmentation tasks in general. Thus, the segmentation task is considered a particular case of the classification task at the pixels level. In this regard, Long et al. [186] proposed one of the first works contributing to the extension of classification CNNs to segmentation tasks. The proposed architecture is trained end-to-end for the semantic segmentation problem and is called a fully convolutional network (FCNN [186]) because it consists only of convolutional layers, see Fig 2.16. The FCNN takes an image of arbitrary size and produces a segmented image of the same size. The authors modified CNN-based architectures (such as AlexNet or VGG16) to have a non-fixed size input and replaced all the fully connected layers with convolutional layers. The author proposed using an up-sampling operation because traditional convolutional layers produce several feature maps with small sizes and dense representations. This operation is called deconvolution and consists of a convolutional layer with a stride equal to 1 and creates an output larger than the input. FCNN considered different up-sampling strategies, which incorporate local and global information in the prediction of the segmentation map. For instance, one strategy considers only the outputs after the last pooling layer; hence, it includes information obtained at a coarse level. While other approaches incorporate the results of previous pooling layers, giving a more refined level of data, which leads to improvements in the prediction.

U-Net [222]: One of the most successful architectures for medical image segmentation. The authors proposed using encoder-decoder architecture while introducing skip-connections between corresponding layers of the two paths; see Fig. 2.17. The encoder or downsampling part has an FCNN-like architecture for extracting features with 3x3 convolutions. Each block at the encoder halves the spatial dimension of the image and duplicates the number of feature channels. The decoder or upsampling part uses a deconvolution operation to reduce the number of feature maps while increasing their height and width, hence, recovering spatial resolution. The kip-connections are used to concatenate the feature maps in the encoding path with their peer in the decoder path to avoid losing pattern information. Eventually, a 1x1 convolution processes the feature maps to generate a segmentation map, thus categorizing each pixel in the input image. This concept has been successfully extended to 3D medical data using the **V-Net** [196] and **3D U-Net** [60].

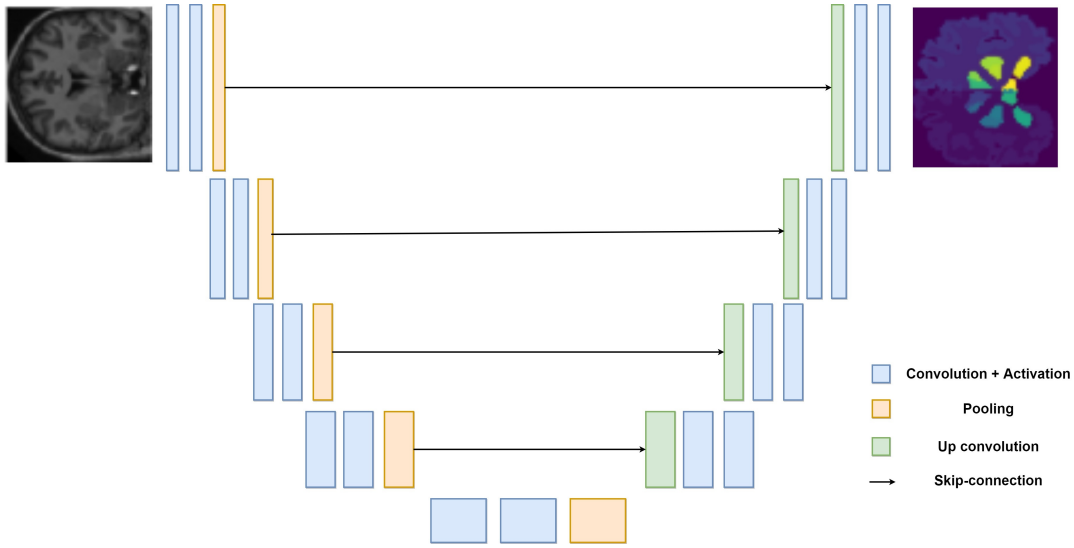


Fig. 2.17. A graphical illustration of the Unet architecture.

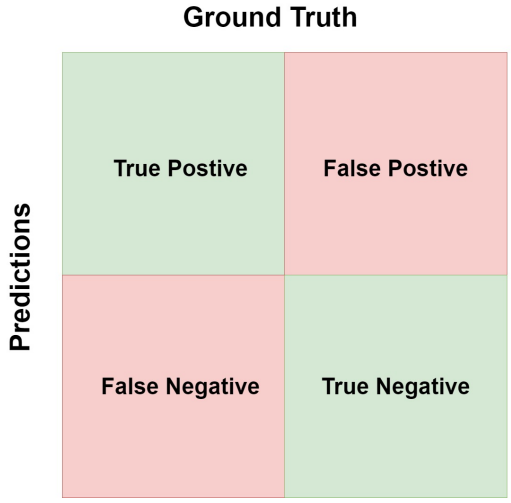


Fig. 2.18. Confusion Matrix.

2.2.6 Common Evaluation Metrics

It is required to comprehensively and intensively validate and standardize machine learning solutions to adopt in real clinical scenarios and greatly benefit from the recent advances in deep learning networks. On the other hand, increasing generalization abilities and ensuring the reproducibility of algorithms is essential, particularly in the case of healthcare procedures where patients' safety is the highest priority. Furthermore, the evolution from theoretical research to clinical practice requires algorithms to be validated on vast patient cohorts, ideally coming from multiple laboratories, to ensure the broad applicability of the methods. To validate computer-based methods, we need to quantify the similarity between the outcomes of the trained model and the ground truth labels produced by expert physicians on a specific task. Several metrics have been used to assess the performance of the trained model. However, most metrics are based on the confusion matrix, Fig 2.18.

A confusion matrix (CM) is a performance measurement matrix for machine learning algorithms. Each row of the matrix contains actual classes while each column contains the predicted classes or vice versa – both options are found in the literature. The entries of the CM are (i) True Positive (TP): the prediction accurately indicates the presence of a condition, *i.e.* , disease. (ii) True Negative (TN): the prediction accurately indicates the absence of a condition. (iii) False Positive (FP): the prediction inaccurately indicates the presence of a condition. (iv) False Negative (FN): the prediction inaccurately indicates the absence of a condition.

Accuracy: It measures the percentage of correct predictions of the model. Accuracy defines the general performance of the model, but it is not considered as a reliable indicator in case of class imbalanced datasets:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

Precision: Also known as positive predictive values (PPV) and measure the percentage of positive predictions that were actually positive. It is more suitable than Accuracy in the case of imbalanced class datasets:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.15}$$

Recall or Sensitivity: Hit rate or true positive rate (TPR). It measures the percentage of actual positives that were predicted correctly as positives. It is more suitable than Accuracy in the case of imbalanced class datasets:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.16}$$

F1-score [253]: Also known as Dice coefficient. It is defined as the harmonic mean of precision and recall. It is more suitable than Accuracy in the case of imbalanced class datasets:

$$\text{F1} = \frac{2 \times (Recall \times Precision)}{Recall + Precision} = \frac{2TP}{2TP + FP + FN} \tag{2.17}$$

Specificity: Also known as selectivity, or actual negative rate measures a model's ability to generate a correct negative result for truly negative cases:

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{2.18}$$

Area Under Receiver Operating Characteristic (AUROC): A receiver operating characteristic (ROC) curve displays how well a model can classify binary outcomes. The AUROC shows the model's ability to discriminate between positive and negative examples, assuming balance data. It is created by plotting the true positive rate (TPR), a.k.a sensitivity or recall, against the false positive rate (FPR) at different classification thresholds. The false-positive rate is calculated as (1 - specificity). AUC measures the entire area under the ROC curve. The higher the area (maximum value is 1), the better the overall performance. See Fig. 2.19. (A).

Area Under Precision-Recall (AUPR) curve: Similar to the AUROC curve, The precision-recall curve is used for evaluating the performance of binary classification algorithms. For example, it is often used when classes are heavily imbalanced, as in many medical datasets. Also, like the AUORC, the AUPR provides a graphical sketch of a classifier's performance across multiple

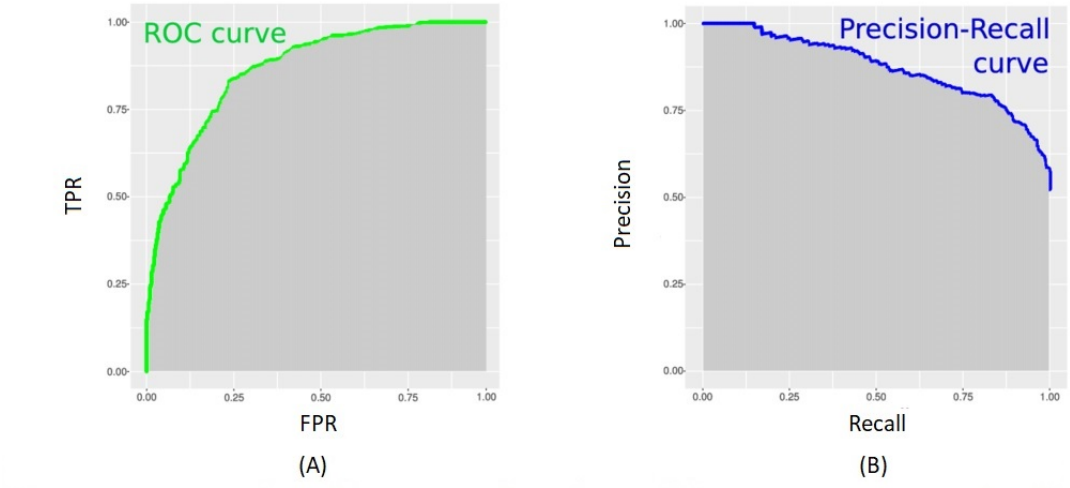


Fig. 2.19. An illustration shows (A) Area Under Receiver Operating Characteristic (AUROC). (B) Area Under Precision-Recall (AUPR) curve.

threshold settings instead of a single value (e.g., accuracy, f-1 score, etc.). See Fig. 2.19. (B).

Hausdorff distance (HD) [219]: The distance metrics are used in the segmentation tasks and measure the deviation between the outer surfaces S and S' of the segmentations Y and Y', such that the distance between a point s on surface S and the surface S' is given by the minimum of the Euclidean distance $d(s, S') = \min_{s' \in S'} \|s - s'\|_2$. Calculating this for all pixels gives the total distance between the surfaces S and S': $d(S, S')$. Now, the largest difference between the surface distances is defined as the Hausdorff distance (HD) and calculated as:

$$\text{HD} = \max[d(S, S'), d(S', S)] \tag{2.19}$$

The Mean Surface Distance (MSD) [32]: This metric measures the average variation between the surfaces, *i.e.* , the segmentation and the GT, and is given as:

$$\text{MSD} = \frac{1}{n_s + n_{s'}} \left(\sum_{s=1}^{n_s} d(s, S') + \sum_{s'=1}^{n_{s'}} d(s', S) \right), \tag{2.20}$$

Where n_s and $n_{s'}$ are the number of pixels for the surfaces S and S' respectively.

Risk-Coverage (RC) curve [94]: RC curve plots the risk as a function of the coverage and is used to investigate the uncertainty evaluation and model confidence. The coverage denotes the percentage of the input processed by the model without rejection, while the risk denotes the level of risk of the model's prediction [94]. For a selective model, the mode abstains from the prediction of input sample x if the prediction confidence of that sample is below a specific threshold, e.g., 0.5. The higher coverage with lower risk, the better the model is. Because the Risk-Coverage curve is not famous in medical imaging, we provide a detailed definition next.

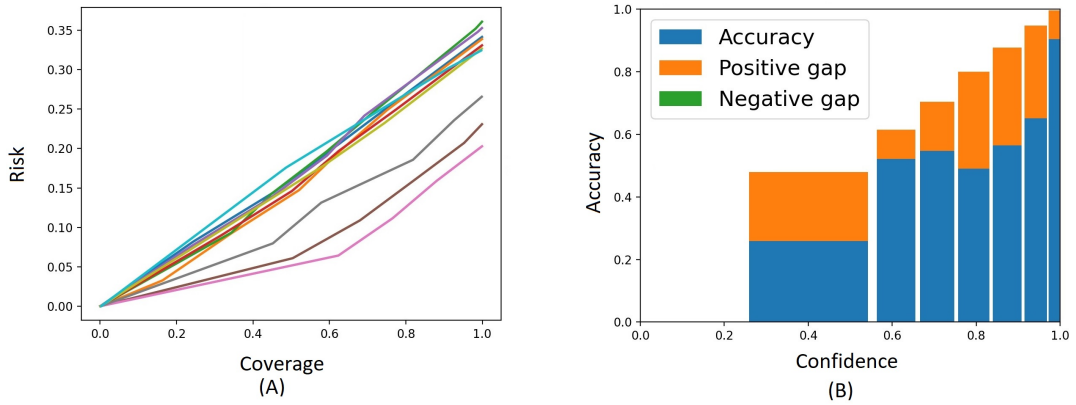


Fig. 2.20. An illustration shows (A) the Risk-Coverage curve. Each line represents a different classifier. (B) Reliability Diagram.

Consider a multi-class classification task. Let $\mathcal{X}$ be a set of input images with $\mathcal{Y}$ labels. Now, let $P(X,Y)$ is a distribution over $\mathcal{X} \times \mathcal{Y}$. f is classifier $f : \mathcal{X} \to \mathcal{Y}$, and the *true risk* of f w.r.t. P is $R(f|P) \triangleq E_{P(X,Y)}[\ell(f(x), y)]$, where $\ell : Y \times Y \to \mathbb{R}^+$ is a loss function. Given $D_m = \{(x_i, y_i)\}_{1=1}^m \subseteq (\mathcal{X} \times \mathcal{Y})$ is a sampled i.i.d labeled data from $P(X,Y)$. The *empirical risk* of the classifier f is defined as:

$$\hat{r}(f|D_m) \triangleq \frac{1}{m} \sum_{i=1}^{m} \ell(f(x_i), y_i). \tag{2.21}$$

A *selective classifier* is a pair (f, g), where f is a classifier, and $g : \mathcal{X} \to \{0, 1\}$ is a *selection function*, which represents a binary qualifier for f as follows:

$$(f, g)(x) \triangleq \begin{cases} f(x), & \text{if g(x) = 1;} \\ reject, & \text{if g(x) = 0.} \end{cases} \tag{2.22}$$

Thus, the selective classifier abstains from prediction at x **iff** $g(x) = 0$. The performance of the selective classifier is quantified using its *coverage* and *risk*.

Definition 1 (coverage) *The coverage of a selective classifier (f, g) is the mean value of the selection function $g(X)$ taken over the underlying distribution P,* i.e. *, is the probability mass of the non-rejected region in $\mathcal{X}$,*

$$\Phi(f, g) \triangleq E[g(X)] \tag{2.23}$$

Definition 2 (risk) *The risk of a selective classifier (f, g) is defined as the average loss on the accepted samples,*

$$R(f, g) \triangleq \frac{E[\ell(f(X), Y).g(X)]}{\Phi(f, g)} \tag{2.24}$$

The entire performance of such a classifier can be specified by its risk-coverage curve, which defines the risk as a function of coverage. An illustration figure is shown in Fig. 2.20. (A).

Reliability Diagram (RD) [106]: On the other hand, the reliability diagram plots the accuracy as a function of confidence such that in the ideal case *i.e.* , a perfect calibrated model, the RD will plot the identity function. For instance, suppose that we have 1000 samples, each with 0.85 confidence; we expect that 850 samples should be correctly classified. RD divides the predictions into bins of confidence, *i.e.* , $B_v; v \in \{1, ..., V\}$, where V is the total number of bins. Then, the average accuracy and the confidence for each bin B_v are calculated as Eq.(2.25) and Eq.(2.26), respectively.

$$acc(B_v) = \frac{1}{|B_v|} \sum_{i \in B_v} \mathbb{I}(\hat{y}_i = y_i) \tag{2.25}$$

$$conf(B_v) = \frac{1}{|B_v|} \sum_{i \in B_v} \hat{p}_i \tag{2.26}$$

Where $\hat{y}_i$, y_i, and $\hat{p}_i$ are the prediction, ground truth, and the confidence for sample i, respectively. The difference (gap) between the accuracy and the confidence can be positive when the confidence is higher than the accuracy and negative when the accuracy is higher than the confidence. These gaps are shown in the RD using different colors; see Fig. 2.20. (B). For a perfect calibrated model, $acc(B_v) = conf(B_v)$ for all $v \in \{1, ..., V\}$. However, achieving a perfect calibrated model is impossible [106].

Expected and Maximum Calibration Errors [78]: Denoted as ECE and MCE, respectively. Based on the previous definitions, ECE and MCE are calculated, where ECE is defined as the difference in the weighted average of the bins' accuracy and confidence. At the same time, MCE represents the maximum difference; see Eq.2.27 and Eq.2.28, respectively.

$$\text{ECE} = \sum_{v=1}^{V} \frac{|B_v|}{s} \Big| acc(B_v) - conf(B_v) \Big| \tag{2.27}$$

$$\text{MCE} = \max_{v \in \{1,...,V\}} \Big| acc(B_v) - conf(B_v) \Big| \tag{2.28}$$

s is the number of samples in bin B_v. For a perfectly calibrated model, ECE and MCE both equal 0.

To calculate the reliability diagrams and calibration errors, we adopted, in this thesis, an adaptive binning strategy [78] that depends on fixable intervals in the calculations. This strategy is more accurate than using fixed intervals [78]. Practically, we can realize the intervals used from the figure itself. For example, the width of the bars in Fig. 2.20. (B) represents the ranges used to calculate ECE and MCE.

2.2.7 Overfitting

Deep Learning models have made significant improvements in different tasks. That is attributed to the deep network architectures consuming massive amounts of data. One of the biggest challenges deep learning models face is the generalization ability. Generalizability is defined as how well the model evaluates data never seen before (testing data). Models with

poor generalizability are called overfitted, performing very well on the training data while not performing well on the testing data. Overfitting occurs when the model estimates the training data perfectly, not generalizing to testing or unseen data.

There are different strategies to reduce overfitting. Some of the techniques focus on the model's architecture itself. This contains series of improvements of more complicated architectures from AlexNet [159] to VGG-16 [250], ResNet [110], Inception-V3 [259], DenseNet [122], and recently vision transformers (ViT) [81]. Other strategies focus on data augmentation; see section 2.2.8. While the last strategies investigated the functional solutions such as batch normalization [126], dropout regularization [257], and transfer learning [241, 284] that will be described in the following paragraphs.

Batch Normalization [126]: When training deep neural networks, one serious problem is that the data distribution for each internal layer may change due to randomness in the parameter initialization and the randomness in the batch data after each epoch or mini-batch. These slight variations cause the learning algorithm to chase a moving target. This phenomenon, called internal covariate shift [126], yields an unstable and slower training process. Therefore, batch normalization [126] is used by bringing all the information into a similar distribution and reducing the variability in the data. Batch normalization is implemented by subtracting the batch mean and dividing it by the batch standard deviation, see Eq.2.29. Note that the learned statistics (mean and standard deviation) are used in the inference time instead of the actual statistics.

$$\hat{\mathbf{x}} = \gamma \frac{\mathbf{x} - \mathrm{E}[\mathbf{x}]}{\sqrt{\mathrm{Var}[\mathbf{x}]}} + \delta \tag{2.29}$$

where $\mathbf{x}$ represents the input batch for a particular layer, $\hat{\mathbf{x}}$ is the normalized batch, γ and δ are learnable parameters, E[.] is the batch mean, and Var[.] is the batch standard deviation.

Dropout [257]: Dropout is a regularization method that reduces overfitting by removing a set of neurons or hidden layers and their connections during the training step. One way to interpret how dropout helps mitigate the overfitting problem is to consider that a particular neuron should learn how to produce a meaningful feature under a randomly chosen sample of the other units, reducing complex co-adaptations. Dropout is used only during training and turned off at testing time.

Transfer Learning [241, 284]: Transfer Learning works by teaching a model on an extensive dataset such as ImageNet [73], then using the model's weights as the initial weights in a new task. Usually, the weights in convolutional layers are reused while other network weights, including fully-connected layers, are replaced, then finetuned using the new dataset. This approach is practical since many image datasets share low-level spatial characteristics better learned with big data.

2.2.8 Data Augmentation

Data Augmentation is a technique that can artificially extend the size of a training set by creating altered data from the existing one. That includes making minor transformations to data to generate new data points. Augmentation addresses overfitting from the heart of

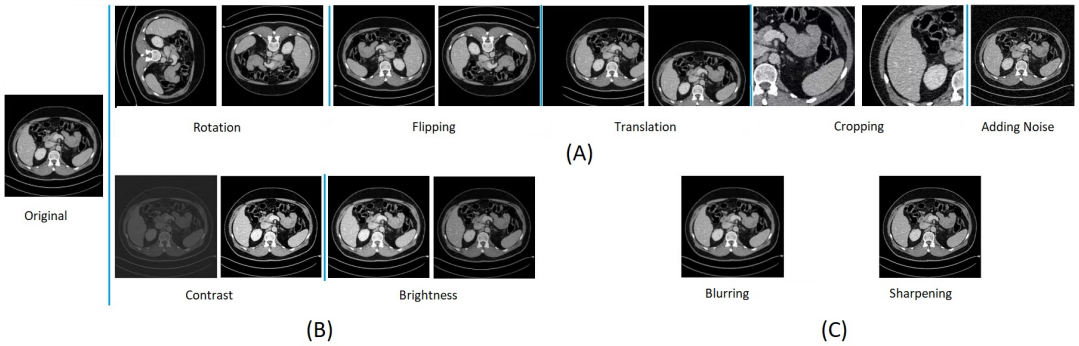


Fig. 2.21. A sample of different data augmentation approaches. (A) geometric transformations. (B) color transformations. (C) Kernel filters.

the problem; the training dataset itself. Data augmentation can generally be divided into geometric transformations, color transformations, and kernel filters [246]. Surveying all these methods is out of the scope of this thesis. However, we will brief on the most common ones in the next paragraphs.

Geometric transformations: These methods enclose using image processing functions on the input image. Geometric transformations include (i) Flipping: horizontally or vertically flipping the given image randomly with a given probability. (ii) Rotation: rotate the image randomly with a specific angle. (iii) Resizing: changing the size of the input image to a given ratio. (iv) Cropping: randomly select a part or location of the image, then resize it to its original size. That includes center cropping, *i.e.* , selecting the center part of the image, or random cropping, *i.e.* , random location of the image. (v) Translation: repositioning the image along the X or Y directions (or both). (vi) Scaling: zoom in or out of the image along the X or Y directions. (vii) Noise injection: injecting a matrix of random values usually drawn from a Gaussian distribution. A common example is salt and pepper noise, presented as random black and white pixels spread through the image. A sample of geometric transformations is shown in Fig. 2.21. (A).

Color transformations: Image data consists of 3 piled matrices of size height×width, representing pixel values for a particular RGB color. Lighting variations are amongst the most repeated challenges to image recognition problems. Therefore, building a model that invariant these color space changes, also known as photometric transformations, is of high importance. Therefore, building a model invariant to these color changes, also known as photometric transformations, is highly important. The most common color augmentations are (i) Brightness: changing the image's brightness. The resultant image becomes darker or lighter compared to the original one. (ii) Contrast is the degree of divergence between an image's darkest and brightest parts. A sample of color transformations is shown in Fig. 2.21. (B).

Kernel filters: Kernel filters include sharpening and blurring the images. These filters apply a sliding widow of $n \times n$ matrix across the image. Intuitively, blurring images could lead to higher antagonism to movement blur during testing. Besides, sharpening images could encapsulate more details about objects of interest. A sample of kernel transformations is shown in Fig. 2.21. (C).

3 Learning Paradigms

> *The superiority of the learned man over the devout worshipper is like that of the full moon to the rest of the stars (i.e., in brightness). The learned are the heirs of the Prophets who bequeath neither dinar nor dirham but only that of knowledge; and he who acquires it, has in fact acquired an abundant portion.*
>
> **— Prophet Muhammad ibn Abdullah (PBUH) (570–632 CE)**
> (Ranked No.1 in "The 100: A Ranking of the Most Influential Persons in History" book by Michael H. Hart, 1972)

Machine learning (ML) involves the methods and algorithms that enable a computer program to learn from data. The standard learning paradigm is the fully-supervised learning. In this learning scheme, data comes in input data and their corresponding labels. The tasks attended by this learning scheme look to find a mapping between the input values and desired outputs. However, in real-life scenarios, especially in medical data, finding such pairs is rare and expensive, and labeling unlabeled data is time-consuming and needs expertise in the domain. Although the literature provides different learning strategies to overcome these limitations, this thesis will focus on three learning paradigms: semi-supervised learning in the standard setting, semi-supervised learning in the federated setting, and self-supervised learning. Before we dive into the details, we give an overview of fully and unsupervised learning in the next sections, as they are the basic learning schema in many medical imaging tasks.

3.1 Supervised Learning

Supervised learning is a machine learning algorithm where the training dataset consists of inputs paired with their correct outputs known as ground truths [154]. We aim to optimize the model's parameters by minimizing a previously defined cost function utilizing the labeled training dataset. After the training phase, the trained model can predict outcomes to new unseen data; see Fig. 3.1.

In the supervised learning paradigm, we are given a labeled dataset $\mathcal{S}_L = \{\mathcal{X}, \mathcal{Y}\}$ consisting of N training pairs (x_i, y_i), where $\mathcal{X} = (x_1, \ldots, x_N)$ and $\mathcal{Y} = (y_1, \ldots, y_N)$, such that x_i represents the input image and y_i is the ground truth or the label for each sample $i \in \{1, \ldots, N\}$. Our objective is to learn a function $f_\theta(x) = \hat{y}$, parameterized by θ, that minimizes the error on the training examples. Here, θ groups all the learnable parameters (e.g., weights

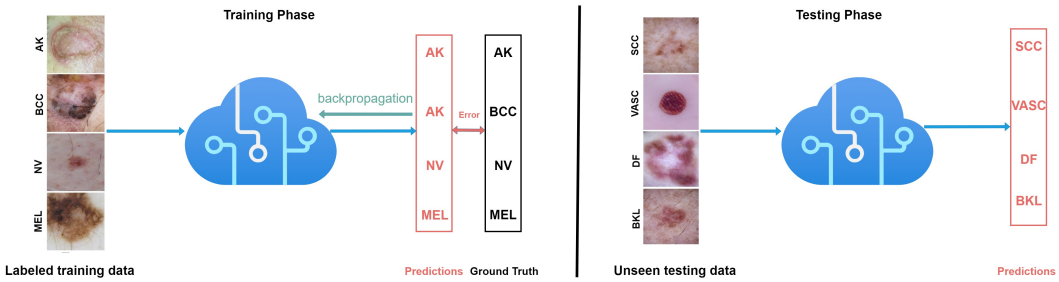


Fig. 3.1. An illustration figure shows the supervised learning paradigm at the training and testing phases.

and biases), and $\hat{y}$ is the predicted outcome. The desired function f can be approximated by finding the optimal set of parameters θ^* that optimize the loss functions $\mathcal{L}$:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f(x_i), y_i) \tag{3.1}$$

The loss function, a differentiable function, comes in different forms depending on the required task. For instance, Mean Squared Error loss (MSE) is the default choice loss in the case of regression tasks, while the Cross-Entropy (CE) loss is widely used for classification tasks. A list of the common loss functions is presented in section 2.2.4.

3.1.1 The Limitations of Supervised Learning

Recently, considerable improvements in supervised deep learning methods have been achieved in different medical image problems [64, 95, 105, 168, 187, 226, 308], and show their applicability to a wide range of datasets without requiring a human expert [127]. Furthermore, deep learning has shown a human-level performance when dealing with cancer classification [87, 269]. Yet, large amounts of labeled training datasets are needed to achieve such success. Unfortunately, annotating the training data comes with many limitations. Next, we listed some of them.

Needs experience: Manual annotation requires highly experienced physicians. Because medical image analysis aims to aid radiologists and clinicians in making the diagnostic and treatment process more efficient, it requires excellent knowledge to extract the disease's most correlated and relevant underlying features from the image [58].

Costly and time-consuming: Manual annotation, such as segmentation, is not only demanding for expert graders but also time-consuming and tedious for clinical use. For example, the average time for manual brain segmentation for one case achieved by an expert radiologist is 14.5 minutes [71]. On the other hand, whole-body computed tomography (WBCT) has been utilized extensively in human trauma medicine to diagnose injuries in severely traumatized patients. Nevertheless, the average cost to analyze each case is 1200 USD [240]. These problems become more burdensome when dealing with large-scale datasets or multi-center trials [10, 245].

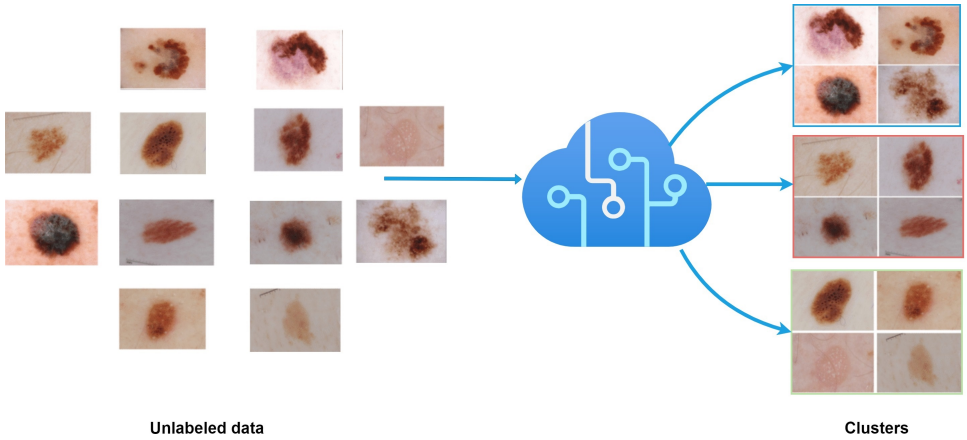


Fig. 3.2. An illustration figure shows the unsupervised learning paradigm. Unsupervised learning finds the underlying structure of the data, then clusters it into groups based on the similarities.

Inter or intra-observer variability: The manual analysis is prone to errors due to different experiences and backgrounds between evaluators and yields so-called inter-observer variability [120]. Also, due to significant variations in pathology and potential fatigue of human experts, the same radiologist might evaluate the same case differently, a.k.a. intra-observer variability [245].

3.2 Unsupervised Learning

In unsupervised learning, on the other hand, no specific labels for the input data are provided; see Fig. 3.2. Let $\mathcal{S}_U = \{\mathcal{X}\}$ a set of unlabeled data, such that $\mathcal{X} = (x_1, \ldots, x_N)$ represents an N data examples, where $x_i \in \mathcal{X}$ for all $i \in [N] := \{1, \ldots, N\}$. Typically, it is assumed that the points are drawn independently and identically distributed (i.i.d) from a common distribution on $\mathcal{X}$. In unsupervised learning, the model attempts to deduce the underlying structure from the inputs or estimate a density likely to have generated $\mathcal{X}$. However, there are also different forms of unsupervised learning, such as clustering the data into groups, outlier detection, and dimensionality reduction. The main characteristic of unsupervised learning is that it trains the network without having a "supervisor" continually correcting the model. While the supervised learning approaches give better performances, in many cases, labeled data are complex, expensive, or impossible to obtain, making using unsupervised learning-based methods essential.

3.3 Semi-Supervised Learning

Despite the recent deep learning-based methods that have achieved state-of-the-art performance in the medical domain, one major drawback of this approach is the necessity for a huge amount of annotated data often unavailable in medical images. These limitations, see section 3.1.1, motivates the researchers to go beyond traditional supervised learning by including other types of data that might be available and cheaper to obtain, *i.e.* , the

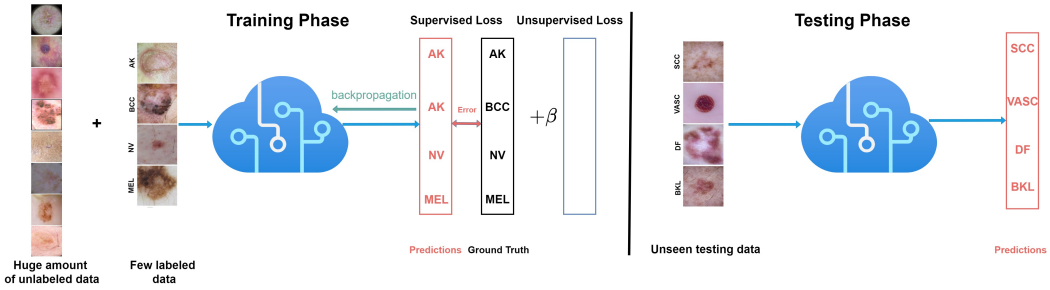


Fig. 3.3. An illustration figure shows the semi-supervised learning paradigm at the training and testing phases. The unsupervised loss takes different forms depending on the used category. β is a hyperparameter that controls the contribution of the unsupervised loss. For example, β could be Zero for some types, e.g., the pseudo labeling method.

unlabeled data. Fortunately, the semi-supervised learning (SSL) [49, 326, 327] framework provides the solution by utilizing a huge amount of unlabeled data along with a few annotated ones in intelligent and efficient ways. Thus, SSL methods have proved their benefits to real cases which fit the nature of medical data, where the scarcity of labeled data is the main characteristic.

The first idea about using unlabeled data in classification is self-learning, self-training, self-labeling, self-learning, or decision-directed learning[49]. This wrapper algorithm starts by training on the labeled data only. Then, some unlabeled points are labeled according to the current decision function. Next, the supervised method is retrained using its predictions as extra labeled data. This idea has been used in the literature for some time (e.g., Scudder (1965) [238]; Fralick (1967) [90]; Agrawala (1970) [3]).

3.3.1 Problem Definition

In semi-supervised learning, we are given a set of labeled $\mathcal{S}_L = \{\mathcal{X}_L, \mathcal{Y}_L\}$ and unlabeled data $\mathcal{S}_U = \{\mathcal{X}_U\}$, where $\{\mathcal{X}_L, \mathcal{X}_U\} = \{x_1, \ldots, x_L, x_{L+1}, \ldots, x_{L+U}\}$ are input images, $x \in \mathbb{R}^{H \times W}$, where H and W are the height and the width of the input image, respectively, $\mathcal{Y}_L = \{y_1, \ldots, y_L\}$ are the ground truth labels, where $y \in \mathbb{R}^{H \times W \times C}$ for the segmentation tasks and $y \in \mathbb{R}^C$ for the classification tasks, and C represents the number of classes. Usually $L << U$.

Our goal is to build a model $f_\theta(x)$ that takes input image x_i and outputs its prediction $\hat{y}_i$. To leverage both labeled and unlabeled data in the SSL paradigm, the objective function takes the form

$$\mathcal{L}_{Total} = \mathcal{L}_{Supervised} + \beta \mathcal{L}_{Unsupervised}, \tag{3.2}$$

Where $\mathcal{L}_{Supervised}$ denotes the supervised loss and trained using labeled data $\mathcal{S}_L$, $\mathcal{L}_{Unsupervised}$ denotes the unsupervised loss and trained on the unlabeled data $\mathcal{S}_U$, and β is a weighing factor that controls the contribution of the unsupervised loss. The weighing factor β can be set as a fixed value between 0 and 1, or in many cases, its importance changes dynamically by using a predefined formula. However, since the model at the early stages of the training phase produces not accurate predictions, a common approach for using β is to start with a small

value (e.g., 0) and then increase it gradually as we proceed in training using exponentiation moving average (EMA).

The unsupervised loss can have different forms depending on the employed SSL approaches; see section 3.3.3 for more details. For example, in the consistency-regularization approach (section 3.3.3), the goal of the unsupervised loss is to minimize the distance between the feature representations of the input data point x and its perturbed version $\hat{x}$, such that $\mathcal{L}_{Unsupervised} = d(f_\theta(x), f_\theta(\hat{x}))$, and $d(\cdot, \cdot)$ is a distance metric. An illustration diagram shows the semi-supervised learning paradigm at the training and testing phases is presented in Fig. 3.3

3.3.2 Assumptions

Semi-supervised learning aims to leverage the unlabeled data in the learning process so that the trained model performs better than the labeled data. Thus, a natural concern emerges: is semi-supervised learning noteworthy? More specifically, compared to a supervised algorithm that uses only labeled data, can we expect that adding the unlabeled data to the training will generate a more accurate prediction? One could assume that the knowledge attained on the marginal data distribution $p(x)$ from utilizing the unlabeled data has to hold valuable information in the inference of the posterior distribution $p(y|x)$. Otherwise, semi-supervised learning will not induce a gain over supervised learning [49]. Hence, for SSL methods to work, certain assumptions have to hold in SSL algorithms [49, 57, 272, 301, 326, 327], see Fig. 3.4.

The smoothness assumption states that if two points $\{x_1, x_2\}$ located in the same high-density region are close, so should be their corresponding labels/outputs $\{y_1, y_2\}$. For example, in Fig 3.4.(B), assume that a labeled data point $x_1 \in \mathcal{X}_L$ and an unlabelled data point $x_2 \in \mathcal{X}_U$ exist, such that x_1 is close to x_2. Then, based on the smoothness assumption, we can predict that x_2 to have the same label as x_1 since proximity—and thereby the label—is transitively propagated through x_1.

The cluster assumption implies that the data tend to form discrete clusters, and the points in the same cluster are more likely to share the same label. Therefore, cluster assumption is considered a particular case from smoothness assumption.

The low-density separation assumption suggests that the good decision boundary line should lie in a low-density region to avoid cutting the same cluster into two different areas, see Fig. 3.4.(B). The smoothness assumption is not violated if we place the decision boundary in this low-density region since it only interests pairs of similar data points. On the other hand, many data points can be expected for high-density areas. Thus, placing the decision boundary in a high-density area violates the smoothness assumption since the predicted labels would be dissimilar for similar data points.

The manifold assumption means that the high-dimensional data could lie or be transferred (roughly) to a low-dimensional manifold. Thus, data points on the same low-dimensional manifold should have the same label; see Fig. 3.4. (C). The manifold assumption helps to avoid

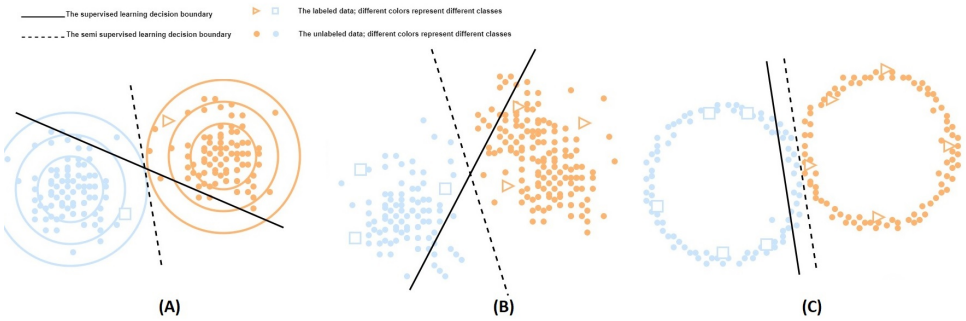


Fig. 3.4. Semi-supervised Learning. (A) The unlabeled data shapes the data distribution and helps to produce a more accurate decision boundary. (B) Smoothness assumption: the points close to each other are likely to belong to the same class. Clustering assumption: the data tend to form clusters, two classes in this figure. Low-density separation assumption (also shown in (B)): a good decision boundary line should lie in a low-density region to avoid cutting the same cluster into two different areas. (C) Manifold assumption: the high dimensional data can be reduced to more low dimensional data.

the well-known problem of many statistical methods and learning algorithms, the so-called curse of dimensionality [49]. This problem is related to the attribute that volume increases exponentially with the number of dimensions; hence, the number of examples required for statistical tasks to estimate the densities is growing exponentially. This problem has significant effects on generative approaches that are based on density estimates in input space. On the other hand, in discriminative methods, a related issue of high dimensions is that pairwise distances tend to become more similar and, thus, less expressive. Nevertheless, with the help of manifold assumption, *i.e.* , the learning algorithm can avoid the curse of dimensionality if the data lie on a low-dimensional manifold.

As mentioned earlier, these assumptions are the foundation of many, if not most, semi-supervised learning algorithms, which generally depend on one or more being satisfied, either explicitly or implicitly [272].

3.3.3 Taxonomy & Categories

Over the past years, several semi-supervised classification algorithms have been proposed. However, these methods vary in the semi-supervised learning assumptions they are based on, how they employ unlabelled data, and how they connect to supervised algorithms.

Transductive vs Inductive: Regarding the availability of the (unlabeled) testing data in the training process, semi-supervised learning can be classified into two settings: the transductive and the inductive learning setting [49]. Transductive learning considers that the unlabeled data, in the training process, are precisely the data to be predicted, and the goal is to generalize over these unlabeled samples. On the other hand, inductive learning supposes that the learned method will still apply to new unseen [301]. Thus, the goal is to output a function defined on the entire space [49]. Inductive methods aim to construct a model that can generate predictions for any object in the input space; hence, they can be used to predict the label of previously unseen data points. Unlabelled data in this setting is used when training this model, but the predictions for new, previously unseen samples are separate once training has been

done. The graph-based methods (sec. 3.3.3) are transductive, while most other SSL methods are inductive.

In general, the recent SSL methods can be grouped into four main categories; (i) Self-Supervision & Entropy Minimization, (ii) Perturbation & Consistency Regularization, (iii) Graph-based methods, and (iv) Generative Methods. Next, we briefly introduce these methods focusing on SSL works on medical images.

Self-Supervision & Entropy Minimization

Among the oldest and most widely known algorithms in the SSL [49, 326, 327]. These methods force the decision boundary to pass through low-density regions to minimize the entropy of the predictions [103]. One way to achieve this in the SSL setting is to generate pseudo labels. Practically, it consists of two alternating steps: (i) training and (ii) pseudo-labeling. First, one or more supervised-based learners are trained on the labeled data. Then, the most confident predictions are used as pseudo-labeled for the next iteration. Finally, this process is repeated using labeled and pseudo-labeled data until convergence. Note that the algorithm here is a purely supervised learning algorithm, unaware of the distinction between initially labeled and pseudo-labeled data; hence β in Eq.(3.15) becomes 0. This approach is also a.k.a wrapper method because the pseudo-labels are generated using a wrapper procedure.

Depending on the number of supervised learners and how the data is being used, these methods can be further divided into (i) self-training: the most basic form where one supervised learner is re-trained on its own most confident predictions, see Fig.3.5. (A). (ii) Co-training: two or more learners re-trained on each other's pseudo labels. Each learner is trained on a different view of the data, *i.e.* , on a subset of the training data, see Fig.3.5. (B). (iii) Ensembling: consisting of either boosting (sequential training)or bagging (parallel training) methods usually builds a stronger learner by ensembling multiple weak learners. In boosting, an individual learner is trained on both labeled data and the pseudo labels from the previous learner [272]. In bagging methods, each weak learner is given a distinct view of the data, uniformly sampled with replacements from the original training data. The final result is aggregated outputs from all individual weak learners [326], see Fig.3.5. (C).

Perturbation & Consistency Regularization

These methods followed the smoothness assumption and are based on the fact that when we realistically perturb or augment the same data point, the predictions for that data point should be similar. In contrast to the pseudo labeling methods, which rely on intermediate steps or supervised base learners, these methods directly optimize an objective function with components for labeled and unlabeled samples [272]. The most typical architecture for these methods is the Teacher-Student model [301]. The student is trained as usual, while the teacher's goal is to generate target predictions for the student.

There are different strategies to generate target predictions; the first one is to apply augmentation or a small amount of noise to the input data while performing dropout regularization inside the network. Thus, we train the model to predict the same output for different perturbations or augmentations of the input data. The so-called Π-Model [231] is a popular method

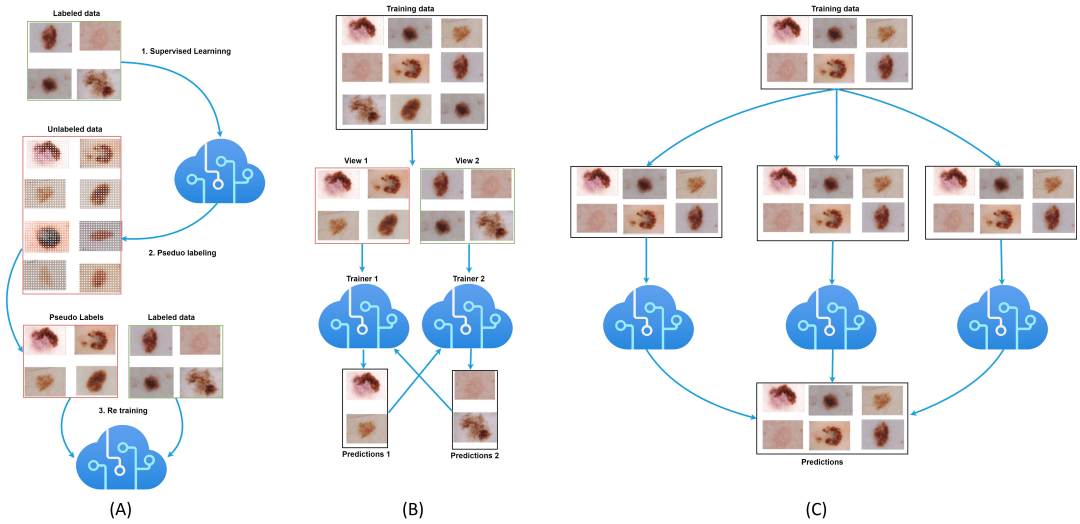


Fig. 3.5. An illustration diagram showing (A): Self-training, (B) Co-training, and (C) Ensembling methods (bagging in this example). These methods are examples of the Self-Supervision & Entropy Minimization approach in semi-supervised learning.

that applies this strategy. Formally, the unsupervised loss appeared in Eq.(3.15) takes the form of:

$$\mathcal{L}_{Unsupervised} = d(f_\theta(x), f_\theta(\tilde{x})), \tag{3.3}$$

where $d(.)$ is any distance function such as $L2$ Norm or KL-divergence, $\tilde{x}$ is the perturbed version of x, and β could be a fixed or dynamic value. However, a main problem with the Π-Model is that it depends on two evaluations of the network under different stochastic conditions, which leads to unstable predictions during the training phase, see Fig.3.6. (A).

Temporal Ensembling [160], overcame this limitation and produced more stable targets using a modified version of the generalization term in the Π-Model by leveraging the Exponential Moving Average (EMA) of past epochs predictions. Specifically, the ensembled outputs are updated with the network outputs z_i after each training epoch, *i.e.* , $Z_i \leftarrow Z_i + (1-\alpha)z_i$, where α is a momentum term. Consequently, the unsupervised loss becomes:

$$\mathcal{L}_{Unsupervised} = d(f_\theta(x), \hat{f}_\theta(\tilde{x})), \tag{3.4}$$

where $\hat{f}_\theta(.)$ is calculated using exponential moving average of $f_\theta(.)$ for the previous predictions, see Fig.3.6.(B).

Mean-Teacher [263] generated more precise targets, hence, the model, by averaging model weights over training steps instead of the model predictions. Specifically, the teacher model's parameters $\hat{\theta}$ were updated using the EMA of the student model's parameters θ. See Fig.3.6. (C).

$$\mathcal{L}_{Unsupervised} = d(f_\theta(x), f_{\hat{\theta}}(\tilde{x})), \tag{3.5}$$

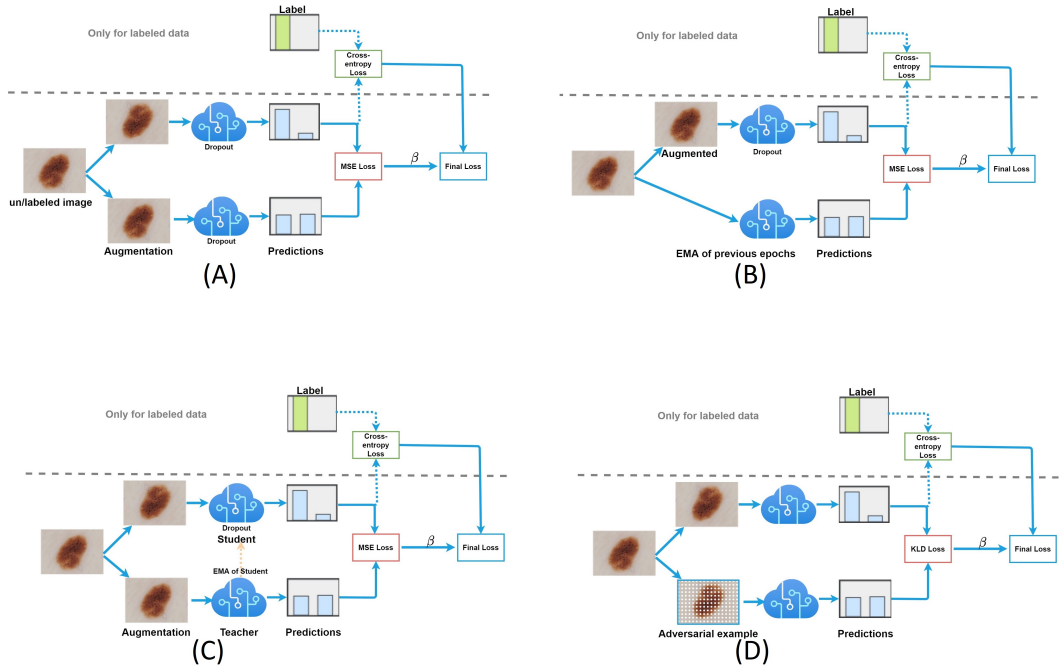


Fig. 3.6. An illustration figure of common Consistency Regularization methods. (A): Π-Model: minimizes the discrepancies of prediction of the same input under two stochastic augmentations, (B) Temporal Ensembling: leverages the Exponential Moving Average (EMA) of past epochs to produce more stable predictions, (C) Mean Teacher: the teacher wights are the EMA of the student wights and (D) Virtual Adversarial Training: instead of stochastic augmentation, virtual adversarial training (VAT) generates a perturbated version of the input image in the adversarial direction.

Instead of using random perturbations as the previous methods, Virtual adversarial training (VAT) [197], Fig.3.6. (D), depends on a tiny perturbation in the adversarial direction r_{adv}, in which the unsupervised loss is trained to minimize the distance function:

$$\mathcal{L}_{Unsupervised} = d(f_\theta(x), f_\theta(x + r_{adv})) \tag{3.6}$$

Graph-based methods

These methods use a graph structure to represent the data, with a node for each labeled and unlabeled example. The edges in the graph represent the connectivity between the nodes. The weights between two nodes are calculated based on an adjacency or similarity matrix $\mathcal{W}$. Where $\mathcal{W}_{i,j}$ is the similarity between samples $x_i, x_j \in \mathcal{X}_L \cup \mathcal{X}_U$. The similarity can be the Euclidean distance, the dot product, or any distance matrix [139]. Based on how they work, the graph methods are argued to be built on the manifold assumption [49]. As mentioned earlier, graph-based methods are transductive, *i.e.* , they directly optimized on the unlabeled data without separation between the training and testing phases. Thus, graphs can propagate the labels from the labeled data to the unlabeled ones based on connectivity and similarity.

Most graph methods utilize the graph Laplacian to represent the data. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ contains the nodes of the graph, *i.e.* , holds the data points (labeled and

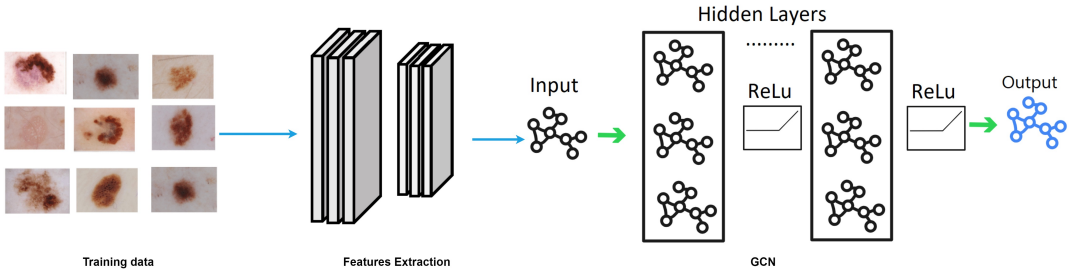


Fig. 3.7. An illustration diagram showing the Graph Convolutional Network (GCN). GCN is an example of the Graph-based approach in semi-supervised learning.

unlabeled), and $\mathcal{E}$ contains the edges in the graph. Now, the weighted adjacency matrix $\mathcal{W}$ is given by

$$\mathcal{W}_{ij} := \begin{cases} w(e) & \text{if } e = (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases} \tag{3.7}$$

where $w(e)$ is the weight of the edge between nodes i and j. Note that a missing edge corresponds to zero weight. Now, the diagonal matrix $\mathcal{D}$ (called the degree matrix of of $\mathcal{G}$) is defined as $\mathcal{D}_{ii} = \sum_j \mathcal{W}_{ij}$. Finally, we can define the normalized and unnormalized graph Laplacian as 3.8 and 3.9, respectively.

$$\Delta' = \mathcal{I} - \mathcal{D}^{-\frac{1}{2}} \mathcal{W} \mathcal{D}^{-\frac{1}{2}}, \tag{3.8}$$

$$\Delta = \mathcal{D} - \mathcal{W}, \tag{3.9}$$

where $\mathcal{I}$ is the identity matrix.

In semi-supervised learning methods, Graph Convolutional Networks (GCNs)[150] is a common graph-based method. GCNs generalize the traditional convolution neural networks (CNN) to the graph domain. In this work, the authors encode the graph structure directly using a neural network model and train on a supervised loss for all nodes with labels. The label information is smoothed over the graph via explicit graph-based regularization [150]. Thus, the unsupervised loss appeared in Eq.(3.15) takes the form of:

$$\mathcal{L}_{Unsupervised} = \sum_{x_i, x_j \in \mathcal{X}} \mathcal{W}_{i,j} \parallel f(x_i) - f(x_j) \parallel^2 = f(\mathcal{X})^T \Delta f(\mathcal{X}) \tag{3.10}$$

Where $\mathcal{X} = \{\mathcal{X}_L \cup \mathcal{X}_U\}$. The above formula assumes that connected nodes in the graph will likely share the same label. Furthermore, utilizing the graph's adjacency matrix lets the model propagate information from the labeled data, allowing it to learn representations of nodes both with and without labels. An illustration of GCN is shown in Fig 3.7.

Generative Methods

The methods mentioned above are all discriminative: they aim to estimate a function that can classify data points, *i.e.* , directly infer the labels [49]. The discriminative methods handle the classification problem without explicitly modeling any data-generating distributions. In

contrast, generative models try to estimate the true density distribution that generated the data [49]. Gaussian mixtures models [192], variational autoencoders (VAE) [148], and Generative adversarial network (GAN) [102] are examples of these methods.

Mixture models are based on the assumption that data are generated from a mixture of K Gaussian distributions [301]. Thus, when the prior $p(y)$ is available and a conditional distribution $p(x|y)$ is correct, data can be assumed to be generated from the mixed models [301]. Each component $j = 1, \ldots, K$ consists of three parameters; a weight π_j (where $\sum_{j=1}^{K} \phi_j = 1$), mean vector μ_j, and covariance matrix Σ_j. We can use, for example, expectation-maximization (EM) [72] to infer these parameters. Thus, when the above conditions are met, and the generative model is correct, the connection between the distribution of unlabeled data and the category labels can be established by assigning to an unlabelled data point $x_i \in \mathcal{X}_U$, the class c that maximizes $\hat{p}(x_i|y_i = c)p(y_i = c)$. In the Gaussian mixture models, $p(y_i = c) = \pi_c$.

VAE [148] is proposed by Kingma and Welling (2013). This architecture is a model that considers each data point x as being generated from a vector of latent variables z. The traditional autoencoders model has a highly complex distribution $p(z)$, which is not easy to use for sampling. In contrast, VAEs constrain $p(z)$ as a simple distribution, such as a standard multivariate Gaussian distribution; hence, the sampling process is straightforward. VAE consists of encoder and decoder networks. At training time, the encoder estimates the parameters of a distribution $p(z|x)$ (e.g., mean and variance) based on the input data point. Then, the latent vector z can be sampled from this distribution and passed through the decoder to reconstruct x. The decoder and encoder are trained together to minimize a combined loss consisting of (i) the Kullback-Leibler divergence between the posterior distribution $p(z|x)$ and some simple prior distribution $p(z)$ and (ii) the reconstruction loss. Kingma et al. [149] proposed two steps to use VAEs in semi-supervised learning. First, the authors train VAE to extract useful latent representations from the unlabelled and labeled data as an unsupervised preprocessing step. Then, the VAE is augmented with the latent representation of the label vector y_i, which encodes the labels for the labeled data and is treated as an additional latent variable for unlabelled data. The architecture is supported by a classification network used to infer the label predictions to implement the last point [149]. See Fig.3.8. (A).

On the other hand, the generative adversarial network [102] consists of two networks; a generator network G and a discriminator network D. The generator task is to generate a fake data point from a noise vector z sampled from some distribution $p(z)$, while the discriminator network aims to distinguish the real from fake data. Both networks are trained alternatively to optimize a single objective function. Specifically, the discriminator's goal is to minimize the objective function, whereas the generator's goal is to maximize it. Thus, the generator and the discriminator networks are playing a min-max game with the value function $V(D, G)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p(x)}[\log(D(x))] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))] \tag{3.11}$$

Because GANs are trained in an unsupervised way, they can be used to handle the SSL. There are many ways to use GANs in SSL settings. However, a popular approach is proposed by the so-called Semi-supervised GAN (SGAN) [204, 232]. Both works independently extended GANs to the semi-supervised setting by using $|Y| + 1$ outputs, where outputs $1, \ldots, |Y|$ correspond to the individual classes, and output $|Y| + 1$ is used to denote fake data points. The loss function

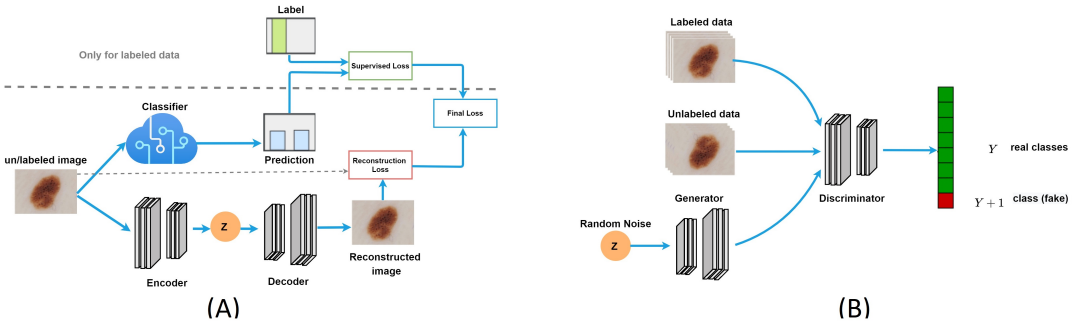


Fig. 3.8. An illustration diagram showing (A) the Semi-supervised VAE and (B) the Semi-supervised GAN. These methods are examples of Generative Methods in semi-supervised learning.

is modified to have the cross-entropy loss of the prediction given the true label for the labeled data points. Further, when an unlabelled data point is given, the discriminators predict the data point is not fake by calculating $\sum_{c}^{|Y|} D_c(x)$ for data point x, where $D_c(x)$ is the value of output c of the discriminator. One interpretation of how GAN enhances semi-supervised learning results was provided by [68], in which the authors showed that the fake examples generated from GAN are located in low-density regions that guide the classifier to find a better decision boundary. The Semi-supervised GAN is illustrated in Fig.3.8. (B).

3.3.4 Realistic Evaluation of Semi-Supervised Learning Methods

Evaluating and comparing machine learning algorithms requires many steps that have a crucial impact on the relative performance of different algorithms. For instance, supervised learning includes hyperparameters fine-tuning, selecting data sets, and partitioning those data sets into training, validation, and test sets. In semi-supervised learning, we should consider more factors. For example, how to select the labeled and unlabeled data and their amount in training. Additionally, how to choose our supervised baselines that needed to evaluate the advantages of adding more unlabelled data. Recently, Oliver *et al.* [205] recommended a set of guidelines for the realistic evaluation of semi-supervised learning algorithms. Next, we will summarize these recommendations.

A shared implementation: To have a fair comparison with different SSL methods, the researchers should use a unified underlying architecture. Oliver *et al.* [205] have noticed that various implementation details (parameter initialization, data preprocessing, data augmentation, regularization, etc.) result in variability in performance. Also, there are discrepancies in the training procedure (optimizer, number of training steps, learning rate decay schedule, etc.). These discrepancies prevent direct comparison between approaches.

High-Quality supervised baseline: SSL aims to obtain better performance using the combination of labeled and unlabeled datasets than what would be received with the labeled data alone. Thus, a fully-supervised baseline trained only on the labeled data is vital to show the advantages of adding more unlabelled data. Further, this baseline should be reported at its best performance.

Comparison to transfer learning: Transfer learning is a commonly used approach to deal with the scarcity of labeled data. Thus, when the source dataset is available, one should consider it as a baseline to compare with it for any successful SSL method.

Class distribution mismatch: When evaluating a semi-supervised learning algorithm, it is suggested to consider the class mismatch between the labeled and unlabeled datasets. While this setting has been neglected by the researchers, in real-life scenarios [205], the unlabeled dataset is unlikely to contain all classes in the labeled data or vice versa. Thus, studying the effect of differing class distributions between labeled and unlabeled data is recommended.

Amount of labeled and unlabeled data: We employ labeled and unlabeled data in training a semi-supervised learning method. A common practice in the literature is to study the effect of changing the size of labeled data [205]. Still, while it is of the same importance, it is less common to systematically vary the amount of unlabeled data. First, it shows to what extent adding more unlabeled data is beneficial. Moreover, it simulates a more realistic case when unlabeled data is relatively small, as in the medical domain [205].

Validation dataset: the validation data is used to finetune the hyperparameters, and the researchers select it in the supervised learning is larger than the training set [205]. However, in a semi-supervised setting and because of the scarcity of annotated data, this large validation set would instead be used as the training set. While selecting validation sets larger than the training could lead to noisier results [205], smaller validation sets constrain the ability to choose models [49]. However, Oliver et al. [205] have found that the trade-off between the two settings can be achieved when the validation dataset is 10% of the training dataset.

3.3.5 Semi-Supervised Learning in the Medical Imaging

The deep learning methods need a large amount of labeled data to achieve the best results, which is expensive and hard to find. Fortunately, the power of semi-supervised learning comes from the efficient utilization of the large amount of unlabeled data, which are cheap and easy to find. Such approaches are fundamental in the medical field, where the problem of labeled data is crucial. In the following paragraphs, we are briefing the most recent works in semi-supervised learning, with a focus on deep-learning-based methods.

Self-Supervision & Entropy Minimization: As mentioned earlier, these methods force the decision boundary to pass through low-density regions to minimize the entropy of the predictions. One way to achieve this in the SSL setting is to generate pseudo labels for the unlabeled data using a model trained on the labeled data. Next, the training process is repeated using labeled and pseudo-labeled data [103]. This approach has been employed by Bai *et al.*[18] for cardiac image segmentation, where the pseudo labels were additionally fine-tuned using the conditional random field (CRF) method [155]. Close to pseudo labeling is Co-Training [34], where confident predictions from separate models, trained using different data views, are utilized to enhance the training. Xia *et al.*[290] employed Co-Training by enforcing multi-view consistency of the unlabeled data for the pancreas and multi-organ segmentation. PLAT [16] exploited an adaptive threshold that avoids noisy signals and generates more accurate pseudo labels to detect the cells in microscopic and stained histology

images. Finally, [177] proposed a pseudo labeling approach, namely self-loop uncertainty, that exploited a self-supervised learning sub-task that solves Jigsaw puzzles to mine the information from the unlabeled data to help the training. While the FCN-based network is optimized to solve Jigsaw puzzles, it produces different segmentation predictions (corresponding to each stage). Then, these predictions are averaged and used as uncertainty estimation yielded by ensembling multiple models to improve the segmentation accuracy in stained tissue and skin lesion images.

Perturbation & Consistency Regularization: These methods train the model to predict the same output for different perturbations or augmentations of the input data. Mean-Teacher [263], one of the most successful methods of consistency regularization, has been employed by Cui *et al.* [66] for brain lesion segmentation. They introduce a segmentation consistency loss to minimize the discrepancy between the outputs of unlabeled data under different perturbations. A similar approach was utilized by Bortsova *et al.* [36] for Chest X-ray image segmentation. Yu *et al.* [304] included the uncertainty information to enable the student model to learn from the reliable targets for left atrium segmentation. Li *et al.* [173] utilized transformation-consistent to enhance the regularization on the pixel level. Interestingly, skin lesions, optical disks, and liver segmentation were demonstrated by their approach. UDC-Net [176] forced the so-called Dual consistency between the predictions of unlabeled images on one side and the predictions of its transformed version and auxiliary decoders on the other side. Further, the consistency is guided by uncertainty measures and applied for COVID-19 lesion segmentation in the CT scans. Moreover, UATS [194] used the consistency between the current prediction of unlabeled images and its ensemble predictions from previous epochs for prostate segmentation. Yet, Wang *et al.* [283], in addition to the consistency between different augmentations of the unlabeled images, forced consistency between the input images and their adversarial direction to classify breast cancer in ultrasound images and ophthalmic disease in the OCT scans.

Graph-based methods: Graph methods represent labeled and unlabeled data in a graph structure, where the nodes represent the data points, and the edges represent the connectivity. The weights represent the distance between the nodes. Graphs can be used to propagate the labels from the labeled data to the unlabeled ones based on connectivity and similarity. Baur *et al.* [21] introduced this concept as a regularization term to the main objective function for MS Lesion Segmentation. The term is based on the Laplacian graphs and attempts to minimize the distance between similar unlabeled and labeled data points in the hidden space. Ganaye *et al.* [93] took advantage of the invariant nature of the brain structure to build an adjacency graph of the brain structures, acting as a constraint to refine the predicted segmentation of the unlabeled data. Graph Convolutional Networks (GCNs)-based approaches have been proposed to handle the unstructured format of some medical data. For instance, GKD [97] distilled the knowledge from the teacher to a student model. The teacher graph injects the available information into soft pseudo-labels. Then, the pseudo-labels train a student graph for Autism spectrum disorder or Alzheimer's disease prediction. RA-GCN [98], on the other hand, addressed the imbalanced class distribution in the medical data by representing each class by a graph-based neural network responsible for the weighting of class samples. The whole architecture, then, is trained in an adversarial manner such that the classifier adapts itself with attention to rare cases.

Generative Models: These models have been extensively used in the past few years to estimate the density distribution of the data using the concept of Generative Adversarial Network (GAN) [102]. Specifically, two networks were used in the training process: the generator and the discriminator networks. The generator aims to produce fake data of a parallel high quality to the original data, while the discriminator intends to distinguish between the fake and the original data. This idea has been utilized by Zhang *et al.*[313] for gland image segmentation by encouraging the discriminator to distinguish between the segmentation results of unlabeled and labeled images while boosting the segmenter (generator) to produce results fooling the discriminator. Nie *et al.*[201] utilized an attention-based approach, based on the confidence map from the confidence network (discriminator), to include the unlabeled data in the adversarial training for pelvic organ segmentation. Chen *et al.*[52] encouraged the model to learn discriminative features for segmentation from unlabeled images, using an autoencoder trained to synthetic segmentation labels to segment tumor and white matter hyperintensities in the brain. SCLLD [7] proposed GAN-based architecture consisting of two training phases to detect COVID-19 infection. First, the weights of the generator and discriminator networks are initialized using the unlabeled data, then fine-tuned by exploiting the labeled ones. VTGAN [140] proposed a semi-supervised GAN-based method to synthesize retinal vascular structure angiograms from fundus images while detecting healthy and abnormal retinas. Transformer-based [81, 274] discriminators take the original and generated images and then produce feature maps for disease classification.

3.4 Federated Learning

The advances in machine learning (ML) and deep learning (DL) methods have led to notable success in medical images and other fields. However, modern DL models require millions of parameters that must be learned from adequate large, curated data sets to reach reliable performance while being secure and equitable and generalizing well to unseen data. When dealing with the medical field, accessing patient data is very hard to obtain because this information is very sensitive, while its usage is strictly regulated [215, 273]. Further, collecting, curating, and maintaining a high-quality medical data is time-consuming, exhausting, expensive, and may have considerable business value, preventing it from being freely shared [216]. On top of that, accessing the labeled and unlabeled data at one site is not always feasible. Thus, a potential solution could be leveraging the distributed data in remote locations without breaching privacy, which is the focus of the so-called federated learning [137, 170, 181, 193, 300].

The are many advantages of federated over the traditional centralized machine learning approaches. First, traditional methods require gathering the distributed data from different devices or institutions to a single location or a server with shared data storage, which may not be feasible and bring serious data privacy and security leake [181]. Second, FL typically utilizes different security techniques to ensure data privacy or security, while the centralized approach pays little awareness to this security issue [323]. Third, FL exploits distributed computing power and resources in multiple regions or organizations, allowing different servers to share the load and preventing any single location from being the bottleneck in the training [181].

3.4.1 Problem Definition and Learning Paradigm

The term federated learning was proposed by Google [193]. The main idea is to build machine-learning models based on datasets distributed across multiple devices, such as mobile devices, while preventing data leakage. FL enables training collaboratively, *i.e.* , in the form of an aggregated model, without transferring local data outside the firewalls of the organizations or devices in which they are located. Rather, the training process happens locally at each participating client (e.g., medical institute, mobile device, or organization). A primary aspect of FL is that only model weights (e.g., parameters, gradients) are shared. Recently, it has been shown that models trained by FL can achieve accuracy levels equivalent to centralized models and superior to single-institutional models [172, 244].

Problem Definition

In this thesis, we consider a problem with medical imaging. Given M clients $\mathcal{C}_m$ who have access to their own local dataset $\mathcal{D}_m \in \mathbb{R}^{H \times W \times N_m}$, where H and W are the height and the width of the input images, and N_m is the total number of images. $\mathcal{D}_m$ consists of labeled $\mathcal{S}_L = \{\mathcal{X}_L, \mathcal{Y}_L\}$, where $\mathcal{X}_L = \{x_1, \ldots, x_L\}$ are input images; $x_i \in \mathbb{R}^{H \times W}$, and $\mathcal{Y}_L = \{y_1, \ldots, y_L\}$; $y_i \in \mathbb{R}^C$ are the corresponding categorical labels for C classes. Given query image x_q, our objective is to train a global model $f(\cdot)$ to predict the corresponding label $\tilde{y_q}$ for x_q, where the local data is leveraged in training in a privacy-preserved fashion.

Objective Function

Let $\mathcal{L}$ denote a global loss function obtained via a weighted combination of M local losses $\{\mathcal{L}_m\}_{m=1}^{M}$, computed from private data $\mathcal{D}_m$, which is located at the individual involved clients and never shared among them. Thus, the objective is to optimize the following overall loss:

$$\min_{\theta} \mathcal{L}(\mathcal{D}; \theta) \quad \text{with} \quad \mathcal{L}(\mathcal{D}; \theta) = \sum_{m=1}^{M} w_m \, \mathcal{L}_m(\mathcal{D}_m; \theta), \tag{3.12}$$

where $w_m = {}^{N_m}/_{\sum_{i=1}^{M} N_i}$ is the respective weight coefficient for each client, and θ is the model parameters.

In the training process, the Stochastic Gradient Descent (SGD) algorithm [218, 331] is generally applied to minimize the loss function [181]:

$$f_{k+1}(x) \leftarrow f_k(x) - \eta_k \Delta f_k(x), \tag{3.13}$$

where $f_k(x)$, $\Delta f_k(x)$, $f_{k+1}(x)$, and η_k represent the learned model, the gradient, the updated model, and the learning rate in the k^{th} iteration, respectively. The learning rate can be dynamically adapted using a local adaptive optimizer (e.g., Adam [147]). In practice, each client typically receives and trains a global aggregated model by executing a few rounds of optimization locally and before sharing weights directly or via a parameter server. The actual process for aggregating parameters depends on the network topology.

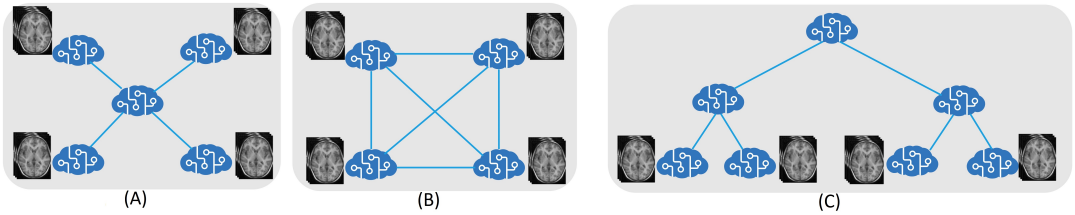


Fig. 3.9. An illustration diagram showing the Federated Learning Topologies (A) Centralized topology, (B) Decentralized topology, and (C) Hierarchical topology.

Federated Learning Topologies

The FL has different topologies, see Fig. 3.9, but the objective remains the same, *i.e.* , to integrate knowledge learned from distributed data. The most common topologies are (i) Centralized topology: by far the most used topology, consists of a centralized server that coordinates the training iterations and collects, aggregates, and distributes the models to and from the clients, see Fig.3.9. (A). (ii) Decentralized topology: each of the training clients is connected to one or more peers, while the aggregation happens on each node in parallel; see Fig.3.9. (B). (iii) Hierarchical topology: the network is composed of several sub-networks built from a mix of peer-to-peer architecture and centralized server, see Fig.3.9. (C).

Aggregation Algorithms

The actual process for aggregating parameters depends on the network topology. The aggregation algorithms can be either centralized, decentralized, or hierarchical. The centralized aggregation algorithms typically depend on a centralized aggregation server that coordinates and organizes the execution of distributed computing resources. In contrast, hierarchical aggregation algorithms rely on multiple servers for model aggregation. The decentralized aggregation algorithms make each participant equally perform the calculation based on a predefined protocol without relying on a centralized server. In this thesis, we will focus centralized algorithm, which is the most related to our work. In a centralized aggregation approach, a single server collects and averages model weights or gradients sent from multiple computing resources or clients. Then, the server updates the global model using a centralized aggregation algorithm, and then the updated global model is transferred to each selected client for the following computation round.

Many centralized aggregation algorithms have been proposed in the last few years [28, 119, 170, 175, 193, 281]. FedAvg is introduced as the assembly method in the implementation of an FL system by Google. First, a centralized server collects the updates of models from randomly selected users. Then, a global model is aggregated using a weighted sum of each participating client. After that, the new global model is shared with other randomly selected users, and the training process is continued until convergence. While FedAvg is a straightforward approach, other methods are proposed to address additional problems. However, they could be considered an adaptation and a modification to the widely used algorithm; FedAvg [193], and they proposed to solve different problems in the federated learning such as data heterogeneity [170, 281], Non-independent-identically-distributed data (non-IIDness) [175], highly imbalance clients [119], or data heterogeneity in the medical images [28].

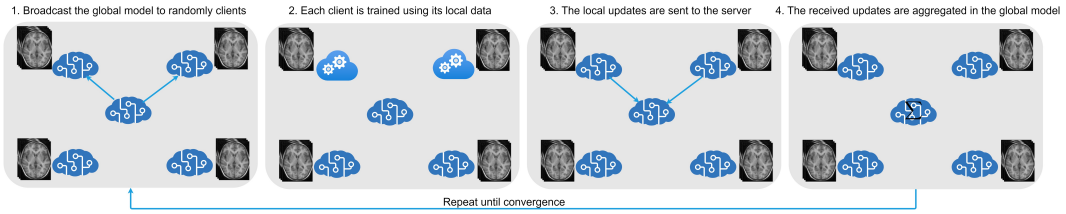


Fig. 3.10. An illustration diagram showing the training process in the FedAvg algorithm.

Training Process

The training process of federated learning, in the FedAvg algorithm, starts at the server by broadcasting initial weights of global model parameters to a random set of participating clients, who share the same model architecture with the global model. Each client, afterward, trains locally on its local data before sending back the updated model parameters to the server. Once all clients send their updates, the server aggregates them using an aggregation algorithm (e.g., FedAvg) to update the global model weights. Next, the updated global model is broadcasted to a new random set of clients before a new round of local training processes starts. Eventually, the previous steps are repeated until the global model converged. During the training, only model weights are shared while data is kept locally. An illustration diagram showing the training process in FedAvg algorithm in Fig.3.10

3.4.2 Characteristics of the Federated Learning

Despite its benefits, FL still has features and considerations when applied in real-life, especially when dealing with medical data. In the following, we mention some of the most popular aspects.

Data heterogeneity & non-IIDness: Applying FL requires coordinating between multiple parties, each with its own dataset. However, this process is not straightforward due to the diversity of data for many reasons causing this variousness; for example, different imaging modalities (e.g., CT, MRI, etc.), data acquisition differences, device manufacturer differences, or local demographics infomration [216]. Therefore, various algorithms, such as FedProx [170], FebBN [175], and FedMA [281], have been proposed to solve this issue. Still, inhomogeneous data distribution represents a challenge in applying many FL algorithms in medical images, especially since many of these algorithms assume IID data across the participants [200, 216, 243].

Privacy, security, & data sharing: Privacy is one of the essential properties of federated learning. Thus, any successful FL algorithm should provide meaningful privacy guarantees, model security, and prevent indirect data leakage [300]. A prominent line of work to handle privacy in federated learning is differential privacy [84]. Differential privacy works by adding noise to the data or using generalization methods to conceal specific sensitive details until the third party cannot distinguish the client, thus, making the data impossible to retrieve and protecting user privacy [300]. In [96], the authors introduced a differential privacy approach to federated learning to protect client-side data by hiding the client's contributions during training. Other approaches include data encryption to protect clients. For instance,

homomorphic encryption [217] and Secure Multiparty Computation (SMC) [35] are adopted to improve privacy through parameter exchange under the encryption mechanism during machine learning [1]. Bonawitz *et al.* [35] introduced a secure aggregation protocol to protect individual model updates. The central server cannot see any local updates but can still observe the aggregated results at each round. However, because these methods work by introducing some noises or encrypting data, this yields trade-offs between accuracy and privacy [170].

Communication efficiency: Federated learning workflow requires extensive exchanging of data (e.g., clients' updates, global model weights). Consequently, it is necessary to develop bandwidth-efficient FL methods to deal with this situation. Compression algorithms such as sparsification and quantization can remarkably decrease the size of data communicated at each round [170]. For example, the ternary compression framework [235] is proposed to compress uplink and downlink communications between the central server and each FL client. Moreover, several works have provided practical strategies in federated settings, such as forcing the updating models to be sparse and low rank [151], performing quantization with structured random rotations [151], and using lossy compression and dropout to reduce server-to-device communication [41].

3.4.3 Semi-Supervised Federated Learning

Thus far, we have discussed different aspects of federated learning. In the problem definition of federated learning, we assume that all clients have a dataset of fully labeled data, see section 3.4.1, while in many realistic settings, data obtained at the client side often comes without labeling due to many reasons such as high labeling cost, or the requirement of expert knowledge. Consequently, it is natural to face a case where a client has only unlabeled data or is partly labeled. The previous setting leads us to a new practical federated learning problem; the so-called Semi-Supervised Federated Learning (SSFL) [130].

Problem Definition

The definition of the federated learning problem that we gave in the section 3.4.1 will be slightly modified to include the unlabeled dataset as follows. Here we assume a direct setting where each client has access to locally labeled and unlabeled data. Given M clients $\mathcal{C}_m$ who have access to their own local dataset $\mathcal{D}_m \in \mathbb{R}^{H\times W\times N_m}$, where H and W are the height and the width of the input images, and N_m is the total number of images. $\mathcal{D}_m$ consists of labeled $\mathcal{S}_L = \{\mathcal{X}_L, \mathcal{Y}_L\}$ and unlabeled data $\mathcal{S}_U = \{\mathcal{X}_U\}$, where $\mathcal{X}_L = \{x_1, \ldots, x_L, x_{L+1}, \ldots, x_{L+U}\}$ are input images; $x_i \in \mathbb{R}^{H\times W}$, and $\mathcal{Y}_L = \{y_1, \ldots, y_L\}$; $y_i \in \mathbb{R}^C$ are the corresponding categorical labels for C classes. Given query image x_q, our objective is to train a global model $f(\cdot)$ to predict the corresponding label $\tilde{y_q}$ for x_q, where the labeled and unlabeled data are leveraged in the training in a privacy-preserved fashion.

Objective Function

Accordingly, the overall objective is to minimize the weighted sum of M local semi-supervised losses. Let $\mathcal{L}$ denote a global loss function obtained via a weighted combination of M local losses $\{\mathcal{L}_m\}_{m=1}^{M}$, computed from private data $\mathcal{D}_m$, which is located at the individual involved

clients and never shared among them. Thus, the objective is to optimize the following overall loss:

$$\min_{\theta} \mathcal{L}(\mathcal{D};\theta) \quad \text{with} \quad \mathcal{L}(\mathcal{D};\theta) = \sum_{m=1}^{M} w_m \, \mathcal{L}_{SSL_m}(\mathcal{D}_m;\theta), \tag{3.14}$$

where $w_m = {}^{N_m}/_{\sum_{i=1}^{M} N_i}$ is the respective weight coefficient for each client, and θ is the model parameters.

Locally, each client optimizes the loss function using any semi-supervised approach that appeared in section 3.3.3. At the same time, the loss function takes the general form as in Eq.(3.15) with a slight modification to adapt to the federated learning setting.

$$\mathcal{L}_{SSL_m}(\mathcal{D}_m;\theta) = \mathcal{L}_{m_{Supervised}} + \beta_m \mathcal{L}_{m_{Unsupervised}}. \tag{3.15}$$

The term Semi-Supervised Federated Learning (SSFL) is relatively new and was introduced by Jeong *et al.* [130]. In their novel work, called FedMatch [130], the authors proposed an approach that utilized the so-called inter-client consistency loss that forces the consistency between similar clients, in addition to decomposing the parameters of the model during weights exchanging with the server. Since then, several contributions have been made to benchmark the SSFL to the community [108, 316]. Some works targeted challenges in semi-supervised federated learning, such as communication efficiency [76, 128], diversity of unlabeled data [315]. While other groups improved the accuracy by exploiting knowledge distillation [234], multi-view training [141], data augmentation techniques [179], graph-based method [279], or adversarial training [332]. Yet, other another line of researchers proposed novel SSFL methods for other domains such as human activity recognition [30, 317], transportation systems [328], or medical imaging [182, 297]. While surveying all these methods is out of the scope of this thesis, we will brief, in the following section, on the FL works in medical images with a focus on semi-supervised methods, which are closely related to our work.

3.4.4 Federated Learning in the Medical Imaging

We have mentioned that the key properties of federated learning are data privacy, Non-IIDness, and communication efficiency, which goes in line with the nature of the medical setting (section 3.4.2). Consequently, federated learning has been investigated by several works in the medical domain [4, 9, 69, 199, 200, 216, 244].

Supervised Federated Learning in Medical Imaging

Li *et al.* [172] have investigated the feasibility of applying federated learning methods for brain tumor segmentation on the BraTS dataset [19] while exploiting differential-privacy techniques to protect the patient data. SiloBN [8] was proposed to train federated learning models robust to inter-center data variability by calculating local-statistic in BN layers. IDA [303] targeted the data heterogeneity in medical imaging. The authors proposed a novel averaging method that is adaptive to meta-information. IDA is based on the inverse distance of each client parameter to the average model of all clients. This allows the global model to reject

or weigh less the clients who may poison other clients yielding an approach that can handle unbalanced and non-iid data in skin cancer classification. [174] have shown the applicability of federated learning for training a multi-site fMRI classification utilization, for the first time, the domain adaptation techniques in a privacy-preserving scheme. FedNorm [28] has been proposed for liver segmentation targeting high data heterogeneity due to multi-modal imaging (e.g., CT and MRI). At the same time, other methods have paved the way for implementing federated models in real-world applications for SARS-COV-2 prediction [89], breast density classification [225], and whole prostate segmentation [233].

Unsupervised Federated Learning in Medical Imaging

FedDis [27] involved disentangled representation learning in an unsupervised scheme for brain anomaly detection and segmentation. In their work, the authors provided a solution to statistical heterogeneity caused by multi-institute data acquisition for MRI brain images. The idea is to separate the model parameters into two spaces; global parameters (the shape of the brain), which will be shared with other clients, and local parameters (the appearance), which will be kept for each client. A GAN-based framework has been proposed in [291] for brain neuroimaging synthesizing. The so-called FedMed-GAN [291] improved the generated image of the standard GAN architecture (centralized training) thanks to federated learning, which effectively incorporated the distributed knowledge from cross-modality data. Curriculum learning also has been employed with unsupervised domain adaptation for the first time in [131] for breast cancer classification in federated learning. The proposed method prioritized the training samples that were most prone to be forgotten after the deployment of the global model. The authors have shown that presenting the training samples in the order suggested by their proposed method is beneficial and boosts the domain alignment between domain pairs.

Semi-Supervised Federated Learning in Medical Imaging

Yang *et al.*[297], among the firsts who introduced semi-supervised learning to federated settings (SSFL), has shown the applicability of SSFL for COVID-19 pathology segmentation. The proposed method has straightforwardly applied a semi-supervised learning method locally in the federated setting globally. At first, a local model is trained in a fully supervised fashion using the labeled data. Then, the trained model is used to produce predictions for unlabeled data, where the predictions with high confidence are used to generate pseudo labels. Next, the pseudo labels are attached to the labeled data before a new training process starts. On the other hand, FedAvg was employed at the server to organize the training between different clients. Another recent work, [182], has proposed the so-called FedIRM for skin lesion classification. In their work, the authors have suggested distilling the knowledge from labeled clients to unlabeled ones by building a disease relation matrix extracted from the labeled clients and providing it to the unlabeled ones to guide the pseudo-labeling process. Liang *et al.*[178] targeted the federated semi-supervised learning with Non-IID local clients suffering from inconsistent reliability among labeled and unlabeled clients. In the so-called Random Sampling Consensus Federated learning (RSCFed), rather than a direct aggregation of local models, RSCFed divides the clients into sub-consensus models randomly and then aggregates the sub-consensus models into the global model. In addition, the authors created a distance-reweighted aggregation strategy that improves the robustness of models. RSCFed has shown its usability in skin lesion classification in dermatoscopic images. Jiang *et al.* [178]

have addressed the class imbalance in the semi-supervised federated learning problem. They utilized a dynamic bank learning scheme consisting of dynamic bank construction to distill various class proportions for each local client and the sub-bank classification to impose the local model to learn different class proportions. The method was evaluated on two public medical datasets, including intracranial hemorrhage diagnosis CT slices and skin lesion diagnosis in dermoscopy images. FedCy [142] was proposed for the first time, in semi-supervised federated learning, to surgical videos by exploiting temporal patterns in the labeled data, which guide unsupervised training toward learning task-specific features for phase recognition.

3.5 Self-Supervised Learning

In the previous sections, we discussed how fully-supervised, semi-supervised, and federated learning have significantly improved machine learning systems and tackled a vast range of real-life problems in medical images by utilizing fully, partially, or distributed annotated data sets. However, the labeling process (fully, partially, or distributed) is often long, expensive, and error-prone. Thus, it represents a bottleneck in further advancements in deep learning. Self-Supervised Learning (SelfSL) is one procedure that can learn complex patterns from unlabeled data. In addition, SelfSL allows machine learning models to work more efficiently when deployed due to their ability to train themselves, thus requiring less training time [248]. Although the unlabeled data is missing their annotations, they are still wealthy with another type of information hidden in their representations. This section will discuss what we have researched in the third perspective by employing representation learning in a self-supervised learning paradigm. While this learning paradigm originated from natural language processing applications, this thesis will focus on the literature from a computer vision and medical imaging perspective.

3.5.1 What is the Self-Supervised Learning

Self-supervised learning is a subset of unsupervised learning methods which has gained more and more popularity in recent years [54, 132]. It seeks supervised feature learning, where the supervision tasks are generated from the unlabeled data by leveraging its structure [86, 183, 185]. This approach enables access to many training instances where supervision is available from the data itself [129, 293]. Therefore, self-supervised learning is an excellent option to explore unlabelled images to improve models' performance in cases where only limited annotations are available [5, 288]. Self-supervised learning pipeline consists of two tasks; see Fig.(3.11). The first task, namely the pretext task, aims to enable the learning of semantic features by generating self-supervised signals from a set of unlabeled data without the need for human annotations [51]. Then, the learned features from the first step are used for subsequent tasks or downstream tasks, e.g., classification or segmentation, where the amount of the annotated data is limited. From the unsupervised learning viewpoint, the self-supervised learning approach relaxes the need for manually annotated data. However, from the supervised perspective, the model is trained with labels generated from the data itself [248]. Note that in the pretext task where the self-supervised learning happens, a model is learned in a supervised way using the unlabeled data by creating labels from the data to enable the model to learn the proper representation. While in the downstream task,

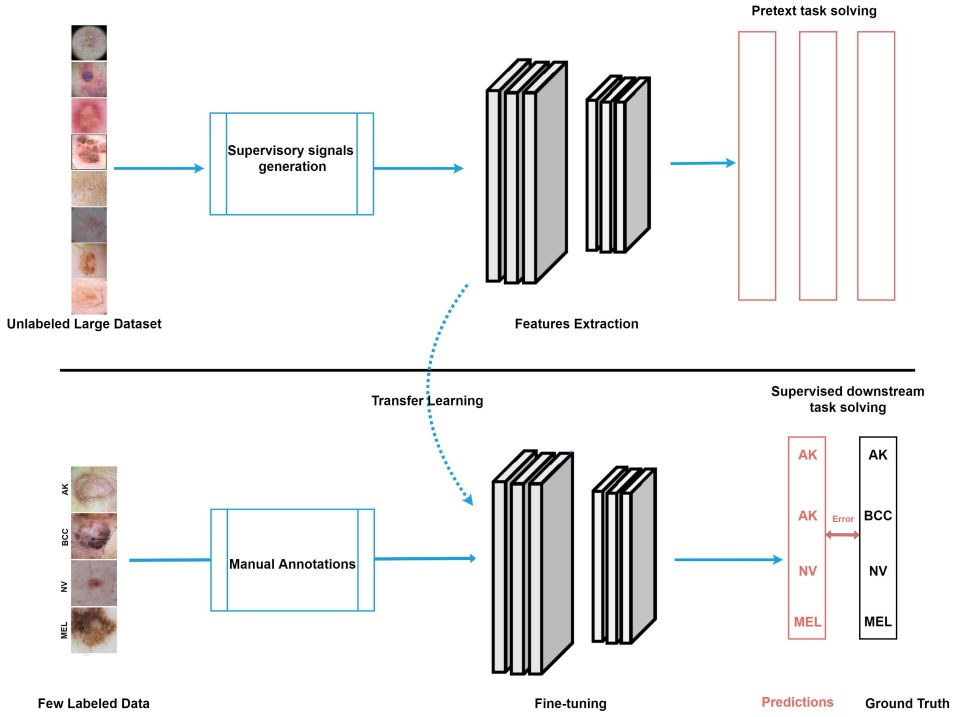


Fig. 3.11. An illustration figure shows the Self-Supervised Learning paradigm. Self-supervised learning pipeline consists of two tasks. The first task, the pretext task (top), aims to enable the learning of semantic features by generating self-supervised signals from a set of unlabeled data. The learned features from the first step are used for subsequent tasks (bottom) or downstream tasks, e.g., classification or segmentation.

the learned representations from the previous task are transmitted as initial weights to the downstream task to perform its intended goal by fine-tuning the labeled data [5, 86, 129, 132, 183, 248, 288, 293].

Pretext tasks

Pretext tasks are central to the self-supervised learning approach and act as its backbone [248]. Pretext tasks are pre-designed tasks for networks to solve, and proper representations are learned by minimizing objective functions of pretext tasks [132]. While the supervision signal for pretext tasks is generated from the data itself based on its structure. The pretext task can be shared among different downstream tasks. The pretext tasks can be predictive, generative, or contrastive [132]. Examples of predictive tasks are relative position prediction [79], Jigsaw puzzle [203], rotation prediction [99], and predict the applied transformation to the image [6]. Generative tasks include image denoising [271], image inpainting [210], image colorization [311], image reconstruction [312]. Nonetheless, contrastive-based tasks are to contrast similar (positive) and dissimilar (negative) pairs. Positive examples are generated by applying a set of random augmentations to an input image, resulting in two transformed views of the same image, while negative examples are any other images different from the altered views [5, 183, 248, 288].

Downstream Tasks

On the other hand, the downstream task may differ according to the researchers' needs and targets. Accordingly, the downstream tasks are computer-specific tasks such as computer vision

applications, medical imaging applications, or other types that can evaluate the quality of features learned by self-supervised learning. High-quality labels like human-annotated labels are needed to solve the downstream tasks. However, in some applications, the downstream task can include data without using human-annotated labels, e.g., semi-supervised learning. These applications can significantly benefit from the pre-trained models when training data are scarce.

3.5.2 Problem Definition

Self-Supervised Learning is considered a particular case of Unsupervised Learning, as both schemes learn without labels. However, Self-Supervised learning approaches rely on pretext tasks that exploit knowledge about the data modality used for training. The self-supervised training considers two steps; pretraining the model to solve the pretext tasks using an unlabeled source dataset and finetuning or transferring the pre-trained model to solve a downstream task utilizing a labeled target dataset. Formally, Given unlabeled source dataset $\mathcal{S}_U = \{\mathcal{X}_U\}$ and labeled target dataset $\mathcal{S}_L = \{\mathcal{X}_L, \mathcal{Y}_L\}$, where $\mathcal{X}_U = \{x_1, \ldots, x_U\}$, and $\mathcal{X}_L = \{x_1, \ldots, x_L\}$ with their corresponding labels $\mathcal{Y}_L = \{y_1, \ldots, y_L\}$. Note that $L << U$. Our objective is to build a predictive model; $f(x) = g_\phi(h_\theta(x))$ by utilizing both datasets, where $g_\phi(.)$ and $h_\theta(.)$ are classifier/regression and representation extractor functions, respectively.

In the pretraining phase, the model is optimized on the unlabeled dataset to solve the pretext task. In this regard, a pretext task can be defined as a process, $\mathcal{P}$, that generates pseudo labels and an objective to guide learning. Given a raw data set like $\mathcal{S}_U$, the pretext process automatically generates pseudo labels z for each sample such that $\{x_i, z_i\}_{i=1}^{U} = \mathcal{P}(\mathcal{S}_U)$. Consequently, in this stage, our goal is to optimize a self-supervised objective in a supervised way using the pseudo labels.

$$\arg\min_{\theta,\gamma} \sum_{(x_i,z_i)\in\mathcal{P}(\mathcal{S}_U)} \mathcal{L}(k_\gamma(h_\theta(x_i)), z_i), \tag{3.16}$$

where $k_\gamma(h_\theta(.))$ is the pretrained model. The pretext task enables learning general-purpose representations h_θ and provides data-efficient knowledge of downstream tasks.

In the second stage, the pretext output function k_γ is discarded, while the representation function h_θ^* is transferred to solve the target downstream task using model $g_\phi(h_\theta^*(.))$. There are two typical methods to optimize the downstream task; linear regression and fine-tuning.

In linear regression, let $k_\gamma(h_\theta(.))$ be the pre-trained model, consisting of a feature extractor, h_θ, followed by a task-specific head, k_γ. The easiest way to reuse h_θ for a new task is to replace the old head with a new one, g_ϕ, designed for the new task. This head is then trained with a frozen feature extractor. Given a target-labeled dataset of L examples, our objective is:

$$\arg\min_{\phi} \sum_{i=1}^{L} \mathcal{L}(g_\phi(h_\theta(x_i)), y_i). \tag{3.17}$$

Alternatively, one can retrain the entire network for the downstream task instead of just training a new head. Consequently, the pretext head is replaced with a new one. However, we optimized both the feature extractor and the task head as follows:

$$\arg\min_{\theta,\phi} \sum_{i=1}^{L} \mathcal{L}(g_\phi(h_\theta(x_i)), y_i). \tag{3.18}$$

3.5.3 Categories of Self-Supervised Learning

As mentioned earlier, the pretext can be shared for different downstream tasks. For example, the same pretext task, e.g., inpainting, could be used to learn visual features for two downstream tasks with other data. Because the pretext tasks can be predictive, generative, and contrastive [132], this property makes it helpful to categorize self-supervised learning approaches according to the nature of the pretext task [248]. see Fig.(3.12). The predictive methods aim to self-generate informative labels from the data as supervision and handle the data-label relationships. The generative methods focus on the intra-data information. The contrastive methods focus on the inter-data information (data-data pairs). Next, we will outline these methods focusing on contrastive-based ones since they are considered the current state-of-the-art methods.

Predictive Self-Supervised Learning

Figure (3.12). (A): The predictive self-supervised learning approach seeks to learn useful representations from unlabeled data by training the pretext task to predict pseudo labels assigned to the unlabeled data [248]. The pseudo labels are generated from the data itself, while its nature depends on the pretext of task design specifications [132], e.g., it can be either categorical or numerical. For example, we can randomly assign a set of transformations to the input data and train the model to predict which transformation is applied to a specific sample. Note that these transformations are considered pseudo labels. Because the goal of the predictive method is to generate informative labels from the data as supervision and handle the data-label relationships [288], the pseudo labels must be carefully designed to allow reasonable learning representations [248, 288].

For instance, Doersch *et al.*[79] divided the input image into nine patches, where the central patch represents the anchor patch, and the remaining patches are query patches. Then the anchor and one random query patch are fed to the network. Then, the self-supervised model is trained to predict the relative position of the query patch to the anchor patch. Inspired by this work, Noroozi *et al.*[203] introduced a model to solve jigsaw puzzles. First, the image is divided into patches. Then, these patches are randomly shuffled using a predefined set of permutations where each permutation has a specific index. Then the model is trained to solve the puzzle by predicting the index number for each patch. Gidaris*et al.*[99], on the other hand, introduced a model to predict the applied geometric transformation, *i.e.* , four random rotations, on the input image. This way, rotation prediction allows learning useful representations by recognizing the orientations of images.

Generative Self-Supervised Learning

Figure (3.12). (B): The generative self-supervised learning approaches seek to learn good representations in the input data by treating pretext tasks to regenerate the same input data or to generate new examples from the same distribution of the input data [86, 132, 183, 288, 319]. Generative adversarial networks [102] or auto-encoder-based architectures [156] are utilized in these types of methods.

Denoising auto-encoder [277] was utilized in self-supervised learning by reconstructing a noise-free output from noisy input [271]. The noisy version is created by applying certain types of noise, such as Gaussian noise. Then, the auto-encoder feeds the deteriorated image to reconstruct the original one. The intuition behind this pretext task is to enable the model to recognize the object in a noisy image correctly. Thus, the learned representations are robust and invariant to noises, while they are useful in the downstream tasks [248]. Yet, image inpainting was proposed by Pathak *et al.*[210]. The framework starts by masking or cropping part of the input image. The masking process includes central or random blocks. Then, an auto-encoder is trained to reconstruct the missing part in the masked image by minimizing reconstruction and adversarial losses. The adversarial loss is proposed to enhance the appearance of the predicted patch. An encoder-decoder architecture was proposed to produce a colored image from a grayscale one [311]. The authors have used 3-channel images; one grayscale and two colored channels. The grayscale input is fed to the network, while the remaining channels are used to supervise the training to predict the image's original colors. This task aims to enable the model to understand the coloring scheme of the objects in the input images, which results in learning wealthy representations. The image super-resolution (SR) technique was proposed as a pretext task in a self-supervised method by Ledig *et al.* [166]. The task's idea is to enhance images' resolution thanks to super-resolution GAN (SRGAN) [166], such that better and more realistic high-resolution images can be produced from low-resolution ones. Furthermore, this strategy intends to take advantage of the loss, consisting of adversarial and content losses, leading to a model which learns better semantic features of data. Eventually, the parameters of the discriminator network can be transferred to other downstream tasks [132].

Contrastive Self-Supervised Learning

Figure (3.12). (C): Contrastive self-supervised learning is recently the state-of-the-art representation learning approach that aims to produce robust representations from the input data by learning to differentiate between similar (positive) and dissimilar (negative) pairs. Positive examples are generated by applying a set of random augmentations to an input image, resulting in two transformed views of the same image. In contrast, negative examples are any other images. The positive examples are assumed to be slightly different but preserve the global features of the input image, which makes the similarity between them higher. Lastly, a contrastive model is trained to maximize the agreement between the positive pairs and minimize it with the negative pairs in case of using them [5, 86, 129, 132, 183, 248, 288, 319]. To understand better how these methods work; we start by explaining the contrastive loss before briefly discussing the current state-of-the-art methods.

Contrastive Loss. The goal of contrastive loss is to minimize the distance of the latent embedding between positive pairs while maximizing it between negative ones. Different functions

have been utilized for contrastive learning, including NCE loss [107] and InfoNCE loss [206]. Normally, these methods employ the noise-contrastive elimination (NCE) method to learn data and allow the model to pull the same images together and push unlike ones away [107]. Formally, given a mini-batch of unlabeled samples $(x_1, x_2, \ldots, x_N)$ and a stochastic augmentation $\mathcal{T}(.)$. The augmentation is implemented to generate two different views of the given sample x; x_i^+ and x_j^+. Then, the different views are fed to the encoder in the network to obtain the latent embedding vector, denoted as (z_i, z^+), as a positive pair extracted from base header $g(.)$.

The NCE loss [53] is expressed mathematically as follows:

$$\mathcal{L}_{NCE} = -log \frac{exp(sim(z_i, z_j)/\tau))}{\sum_{j=1}^{2N} \mathbb{I}_{[k \neq i]} exp(sim(z_i, z_j)/\tau)}, \tag{3.19}$$

where $sim(.)$ is a similarity or distance function, $\mathbb{I}$ is an indicator function whose value is equal to 1 if $k \neq i$ and zero otherwise, and τ temperature hyperparameter which helps in controlling penalties on hard-negative sample [280]. The total number of augmented pairs is $2N$, with $2(N-1)$ negative examples from other images. In general, the cosine function is used as similarity measurement between embedding representations z_i and z_j as follows:

$$sim(z_i, z_j) = \frac{z_i.z_j}{\parallel z_i \parallel . \parallel z_j \parallel}, \tag{3.20}$$

where $\parallel . \parallel$ is the Euclidean distance. The cosine similarity calculates the slope between two non-zero vectors in a d-dimensional space. At zero angle, the cosine similarity is one, which means the two vectors are identical, while at any other angle, the cosine value ranges from positive to negative.

InfoNCE loss [206], on the other hand, is the most widely used loss function in contrastive learning. In a self-supervised learning context, InfoNCE estimates the information shared by two images, and it is preferred because it corresponds to cross-entropy loss [5]. The InfoNCE distinguishes a positive pair (z_i, z^+) from its corresponding k negative examples $(z_i, z_1^-), \ldots, (z_i, z_k^-)$, and given as:

$$\mathcal{L}_{InfoNCE} = -log \frac{exp(sim(z_i, z^+)/\tau)}{exp(sim(z_i, z^+)/\tau) + \sum_{j=1}^{k} exp(sim(z_i, z_j^-)/\tau)}. \tag{3.21}$$

Related Works. Several works [53, 109, 112, 115, 206, 289] have applied the above concept in self-supervised learning. For instance, Momentum Contrast (MoCo) [109] proposed a framework consists of a siamese network [38], memory bank, and InfoNCE loss [206]. The Siamese network includes a query encoder and a momentum encoder. The query-encoder generates a features vector from a query image, while the momentum-encoder encodes the other images. The memory bank acts as a dictionary lookup that performs a lookup operation between a query image encoding and the dictionary, which contains other images' encoding as keys. Consequently, learning useful representations is done by maximizing the similarity between the encoding of the query image and its matching key while minimizing it with the non-matching ones. While SimCLR [53] relaxes the need for contrast momentum, it mainly relies on two simple concepts; heavy data augmentation technique, which results in correlated

views for the same input, and large batch size that includes a large set of negative examples. SimCLR [53] framework applies a set of random augmentations to generate two positive views of the same input image. Both views, then, are passed to a siamese network [38], to generate the hidden representations; (h_1, h_2), which in turn, are passed to projection heads; $g(.)$, leading to a pair of feature vectors or embeddings; (z_1, z_2). Then, the InfoNCE contrastive loss is employed to optimize the whole network by maximizing the similarity between the positive pairs and minimizing it for other images in the same batch (negative samples). The projection heads are removed when the model converges, while the convolutional encoders are used for downstream tasks. One drawback of the previous approach is that it requires expensive computations to find the negative images from a memory bank [109] or a large batch size [53].

Clustering methods. Other works overcome the necessity of distinguishing individual samples by differentiating between groups of images clustered based on their likenesses [11, 45, 46, 292]. For instance, clustering-based methods such as DeepCluster [45] utilized K-mean assignments as priors to cluster the learned representations. On the other hand, SwAV [46] applied an online clustering approach while forcing agreement between the representations from several views of the same image. Deep Cluster [45] demands a complete pass over the dataset to calculate the clusters' assignment, which becomes computationally intense in the case of large datasets. On the other hand, online methods, such as SwAV [46], calculate clusters' assignment by mapping the encoded embeddings (z_1, z_2) to a K learnable clusters (groups) on the current batch using the prototype layer (a dense layer with linear activation function). This prototype layer generates the cluster assignments (codes), *i.e.* , (Q_1, Q_2) that represent the mapping of feature vectors into clusters. Then, the Sinkhorn-Knopp distance [67] is employed to measure the similarity between the codes. Finally, a swapped prediction is performed on these codes such that it is possible to predict the codes of one view from the features vector of the other. Nonetheless, these methods require a lot of negative samples to produce reliable predictions.

Symmetric architectures methods. In a different work direction, [56, 104] trained self-supervised models without relying on negative examples. The idea is to have a Siamese architecture and online target networks. The online network, with learnable parameters, is trained to predict the presentations for the target network. BYOL [104], for example, sets the parameters of the target network as moving average parameters of the online network. However, the parameters in SimSiam [56] are shared between both networks, while backpropagation and stop-gradient trick are applied to online and target networks. Despite these tricks avoiding the collapse solutions, they lack the explainablty [20].

Information maximization methods. An elegant method, Barlow Twins [306], utilizes the redundancy-reduction principle to make the cross-correlation matrix, produced from two siamese embeddings, close to the identity matrix. In addition, the author proposed a new objective function consisting of invariance and redundancy-reduction terms. The invariance term makes the embedding fixed to the applied augmentations by encouraging the diagonal elements of the cross-correlation matrix to be 1. However, the redundancy reduction term decorrelates the different vector elements of the embedding by equating the off-diagonal components of the cross-correlation matrix to 0. This decorrelation minimizes the redundancy between output units. W-MSE [43] achieves this by whitening feature representations within

each batch via Cholesky decomposition [26]. Hua *et al.* [121] proposed shuffled decorrelated batch normalization (DBN) [123] to prevent a dimensional collapse. However, VicReg [20] proposed another method free from the normalization step by employing Variance-Invariance-Covariance regularization terms. The invariance term is the mean square distance between the embeddings, while the goal of the variance term is to maintain the standard deviation (over a batch) of each embedding variable above a given threshold. This term pushes the embedding vectors of instances within a batch to differ. Finally, the covariance term moves the covariances (over a batch) between every pair of embedding variables toward zero. This term decorrelates the variables of each embedding and prevents an informational collapse in which the variables would vary together or be highly correlated. VicReg is training joint embedding architectures based on preserving the information content of the embeddings.

3.5.4 Self-Supervised Learning in the Medical Imaging

The success of self-supervised learning in different domains, from natural language processing to computer vision, attracted other workers, with growing interest, to investigate its ability in medical images. This motivation comes from medical images always suffering from a severe scarcity of data annotations, which aligns with the problem that self-supervised learning is handling. The following paragraphs summarize the most recent works of self-supervised learning that targeted medical imaging [51, 59, 157, 248].

Predictive methods. Zhang *et al.* [310] proposed a self-supervised learning method for fine-grained body part recognition by solving slice sequences on 3D CT and MRI scans. The authors train a model to predict the spatial order of these slices as an auxiliary task. Bai *et al.* [17] proposed anatomical position prediction as a pretext task for cardiac segmentation in MRI images. The authors utilized several MRI orientations, e.g., short-axis, 2CH long-axis, and 4CH long-axis, as different views and trained the network to predict the relative positions of anatomical regions in these views concerning a specific one. A jigsaw puzzle has been utilized as a pretext task to solve different medical imaging tasks [177, 260]. For instance, Li *et al.* [177] adopted a jigsaw puzzle combined with random patches' rotation prediction to propose the so-called Self-loop uncertainty in nuclei and skin lesion segmentation. While multi-modal (e.g., T1 and T2 scans) Jigsaw puzzle pretext task has been exploited to solve four downstream tasks consisting of brain tumor segmentation, prostate segmentation, liver segmentation, and survival days regression by Taleb *et al.* [260]. The success of the jigsaw puzzle as a pretext task has motivated other researchers to investigate Rubik's cube puzzles in their works for brain hemorrhage classification and brain tumor segmentation [324, 329]. Solving the Rubik's cube enables the model to learn invariant rotation and translation features.

Generative methods. A GAN-based architecture was employed in the image colorization pretext task to segment endoscopic medical instruments as a downstream task by [223]. Chen *et al.* [51] employed a relative position prediction task to train a model for abdominal multi-organ localization and brain tumor segmentation. The idea is to create a corrupted version of the input image by swapping two random patches repeatedly. Then train a model to restore the original version from the corrupted one. A similar idea of restoring the distorted image to its original context was proposed by Zhou *et al.* [322] to solve six downstream tasks, including, for

example, Lung nodule and Liver segmentation in CT scans, Brain tumor segmentation in MRI images, and eight pulmonary diseases classification in X-rays. Holmberg *et al.*[118] designed a specific pretext task for ophthalmic disease diagnosis called cross-modal self-supervised retinal thickness prediction. First, the authors created the thickness maps extracted from a segmentation model trained on a small annotated dataset of optical coherence tomography scans (OCT). These maps serve as labels to guide the self-supervised learning process to predict the thickness maps on unlabeled infrared fundus images. A combined framework of GAN-based architecture and Rubik's cube pretext task was proposed by Tao *et al.* [262] for the pancreas and brain tissue segmentation. To learn useful representations, the generator is trained to restore the original order of the Rubik's cube before the random transformation, while the discriminator's goal is to distinguish between the correct and wrong arrangement of the generated cubes.

Contrastive learning. Some contrastive-based algorithms have been slightly modified to adapt and solve medical image problems. For example, BYOL [104] has been exploited by Xie *et al.*[294] for liver, spleen, kidney tumor, and internal abdominal organs segmentation in CT scans. Other researchers adopted MoCo [109] for tuberculosis detection [254] and pleural effusion classification [278] in chest x-rays, and COVID patient prognosis [256] and COVID classification [55] in chest CT scans. While SimCLR [53] was used by Chaitanya *et al.* [47] for cardiac and prostate segmentation, and by Azizi *et al.*[14] for chest X-ray classification and skin lesions classification in dermoscopic images. Zhang *et al.* [314] designed a self-supervised multi-tasking method that integrates rotation prediction [99], Jigsaw puzzle [203], and SimCLR [53] in one framework; the so-called twin self-supervision based semi-supervised learning (TS-SSL) for spectral-domain optical coherence tomography (SD-OCT) classification.

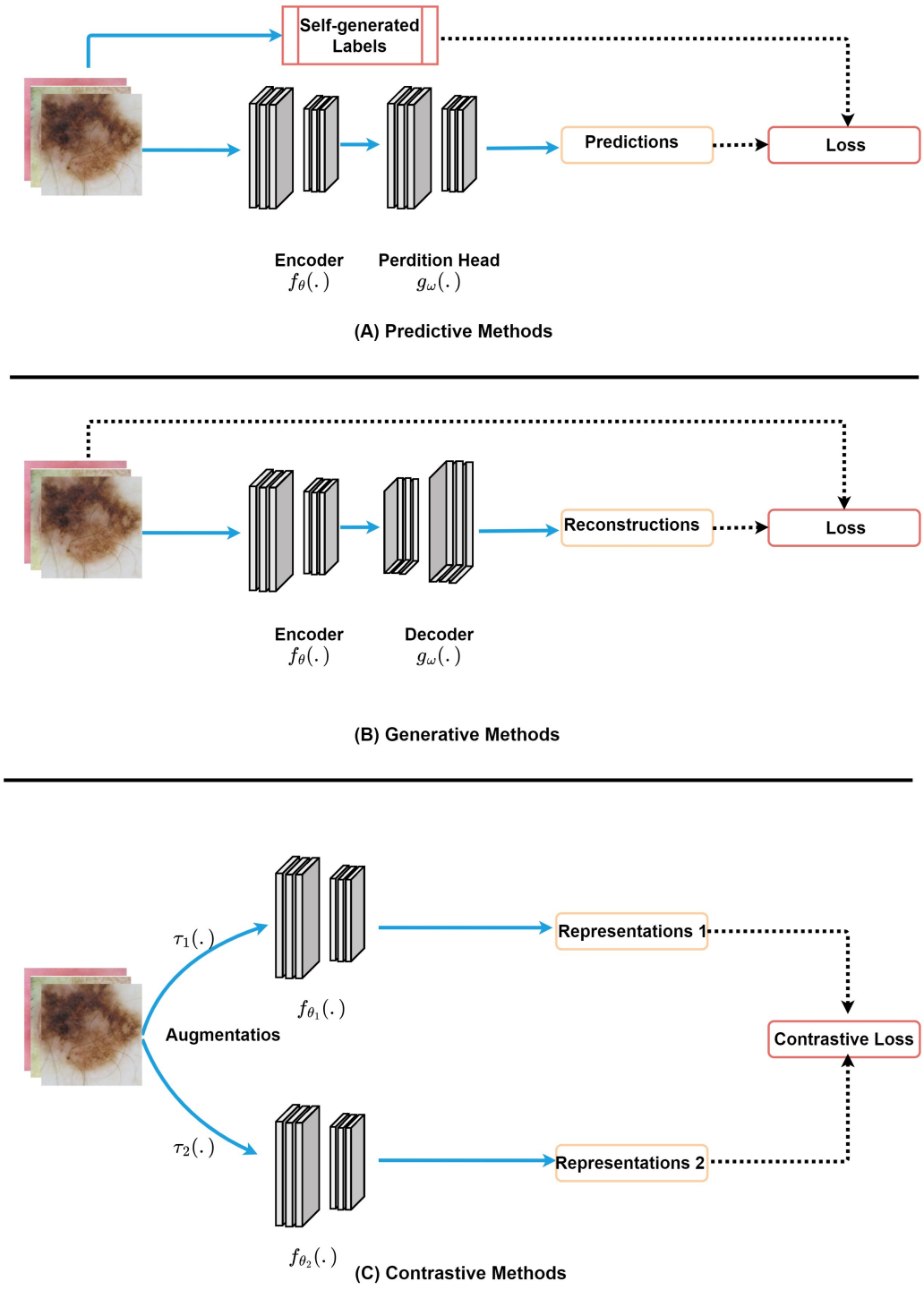


Fig. 3.12. An illustration shows the Self-Supervised Learning categories. (A) Predictive Methods, (B) Generative Methods, and (C) Contrastive Methods.

4 Data Augmentation via Random Linear Interpolation in Semi-Supervised Learning

> *All you need is lots and lots of data and lots of information about what the right answer is, and you'll be able to train a big neural net to do what you want.*
>
> **— Geoffrey Hinton**
> (British-Canadian cognitive psychologist and computer scientist. Also known as the Godfathers of Deep Learning. Hinton received the 2018 Turing Award, for his work on deep learning.)

4.1 Motivation

Medical image segmentation plays a fundamental role in the medical field [2] since it provides a tool to examine different diseases [242], quantify human organs [213], therapy planning [202], tumor development monitoring [88, 146], diagnostic aid systems [70], and intra-operative assistance [117]. Nevertheless, manual segmentation is a tedious task that requires highly experienced physicians [58] and is subject to intra-/inter-observer variability [120]. That led to a great interest in automated segmentation methods estimated at 70% of international image analysis challenges in the medical domain [188]. Recently, deep learning-based methods have achieved state-of-the-art performance in medical image segmentation [64, 105, 226], and shown their applicability to a wide range of datasets without requiring human expert [127]. However, one major drawback of this approach is the necessity for a huge amount of annotated data which is oftentimes not available in medical images. Fortunately, the semi-supervised learning (SSL) framework provides the tool to alleviate this problem by utilizing a huge amount of unlabeled data along with a few annotated ones in intelligent and efficient ways. Thus, SSL methods have proved their benefits to real cases which fit the nature of medical data, where the scarcity of labeled data is the main characteristic.

4.2 Contribution

In this part, we developed a novel and automated semi-supervised deep learning method to segment medical images. We address the task of segmenting 27 internal Brain structures in

MRI images from multiple resources and COVId-19 infection in Lung CT scans. We summarize our contributions as follows:

- We propose our data augmentation method; ***RandOm lAyer Mixup (ROAM)*** that explores the manifold by randomly selecting a subset of input and hidden layers to perform a linear interpolation of labeled and unlabeled data points and generate virtual data that fits the complexity of medical imaging segmentation in both fully and semi-supervised settings. **ROAM** overcomes the limitations of the previous methods by encouraging the network to be less confident for interpolated data points and reducing over-fitting and generalizing well to unseen data.

- We perform a comprehensive ablation study showing the importance of our design choices. Further, we discuss employing the Manifold Mixup with the presence of skip-connections in U-Net-like architectures. Also, we conduct extensive experiments, following the recommendations of Oliver *et al.*[205], to evaluate our method under the presence of domain shift, class mismatch, and different amounts of un-/labeled data.

- We utilize a unified architecture to implement different SSL methods for a fair comparison. Finally, we empirically show the effectiveness of ROAM by demonstrating a SOTA performance in both supervised and semi-supervised settings in the whole brain image segmentation and beating the baseline models in COVID-19 infection and lung segmentation.

The content of this part is based on the following publication:

Bdair T, Wiestler B, Navab N, Albarqouni S. "**ROAM: Random layer mixup for semi-supervised learning in medical images**". IET Image Processing. 2022 May 2.

4.3 Related Works

4.3.1 Semi-Supervised Learning Methods in Medical Imaging

We already have shown in section 3.3.5 the recent semi-supervised works in medical imaging that are most related to our work. In a nutshell, the current methods can be categorized into the following four main types.

Self-Supervision & Entropy Minimization: As mentioned, these methods force the decision boundary to pass through low-density regions to minimize the entropy of the predictions. One way to achieve this in the SSL setting is to generate pseudo labels for the unlabeled data using a model trained on the labeled data. Our method is similar to the methods mentioned in section 3.3.5 in the pseudo labeling step. However, we are different in two folds. First, the aforementioned methods generate pseudo labels for unlabeled data only. Yet, our method, in addition to that, generates virtual data points and their corresponding pseudo labels from

linear interpolation at a random layer of the input data. This process augments the model with novel training signals that have never been seen before; see section 4.4.2 for more details. Second, the previous methods utilize different post-processing steps to enhance the quality of the pseudo labels, yet, none of them used a sharpening operation that pushes the pseudo labels into more confident regions, which was adopted by our method, check section 4.4.2 and Fig.4.1 for more details.

Perturbation & Consistency Regularization: These methods train the model to predict the same output for different perturbations or augmentations of the input data. However, as mentioned earlier, all methods in section 3.3.5 applied data augmentation at the input space to force the consistency loss. Nonetheless, our approach augments the images at the input and the hidden layers. Although UDC-Net [176] proposed the perturbations at the features level, they introduced a sophisticated augmentation process consisting of seven decoders. In contrast, our method handles that by simply utilizing linear interpolation. Moreover, UDC-Net [176] used seven decoders fixed at one hidden space to create different variations. However, our method overcomes this limitation by randomly selecting the hidden spaces on which the augmentation is performed.

Graph-based methods: Graph methods represent labeled, unlabeled data in a graph structure, where the nodes represent the data points, and the edges represent the connectivity. Graphs can be used to propagate the labels from the labeled data to the unlabeled ones based on connectivity and similarity. The previous methods, in section 3.3.5, have shown their benefits to unstructured data, yet, it suffers the burden of graph construction and weighing steps. Moreover, graph-based approaches are transductive methods, *i.e.* , optimized on unlabeled data without separation between the training and testing phases., which results in a lack of scalability.

Generative Models: These models have been extensively used in the past few years to estimate the density distribution of the data. Major drawbacks of these approaches, as mentioned in section 3.3.5, include the computation overhead and the complexity of the architecture. For instance, [7] involves two training stages, while VTGAN [140] consists of four networks; two generators and two transformer-based discriminators.

4.3.2 Modern Regularization Methods

Modern regularization methods such as Input MixUp [307], and Manifold Mixup [275] have been recently introduced to avoid over-fitting by encouraging the model to be less confident for interpolated data points at the input space or the latent space respectively.

Input Mixup [307] is a simple data augmentation method that generates new data points (x_k, y_k) through a linear interpolation between a pair of training examples:

$$x_k = \lambda x_i + (1 - \lambda) x_j, \tag{4.1}$$

$$y_k = \lambda y_i + (1 - \lambda) y_j, \tag{4.2}$$

where $\lambda \in [0, 1]$.

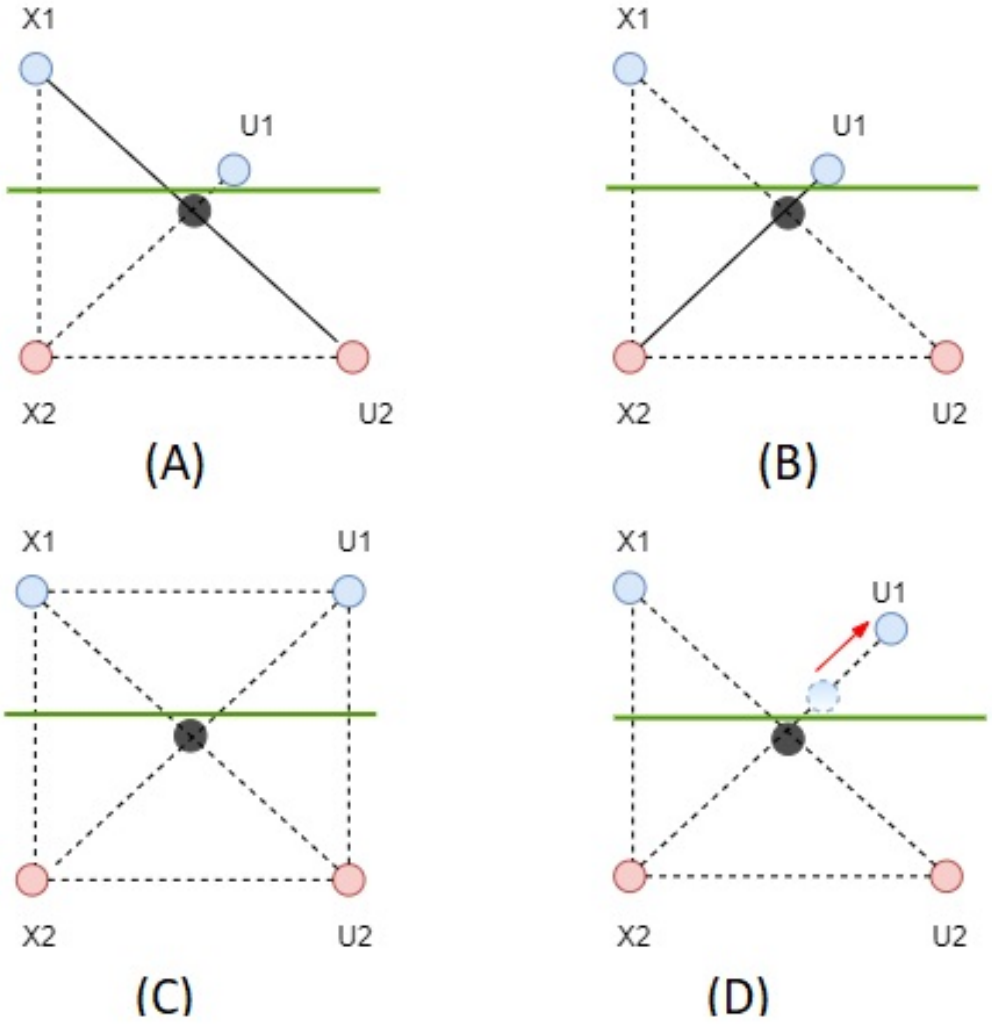


Fig. 4.1. Illustrative example. (a)-(b) Input Mixup: This shows the inconsistency of the generated soft label of grey-dot resulting from two different linear interpolations of inputs. (c) Manifold Mixup: The hidden learned states are better organized in local regions leading to the consistency of the soft labels. (d) The sharpening operation (red arrow) pushes the soft label to a more confident region.

Mixup is considered a type of data augmentation where the newly generated data points extend the training dataset following the cluster and manifold assumptions [49] that linear interpolations of input examples should lead to linear interpolations of the corresponding labels. One major drawback of this approach is that the interpolations between two samples may lead to inconsistent soft labels at interpolated points. Thus Input Mixup can suffer from underfitting and high loss. This can be better understood by examples. Fig.4.1 shows an illustrative example, where the red and the blue circles represent two classes. In Fig.4.1(A), the grey dot is generated by the linear combinations of a blue labeled example ($X1$) and a red unlabeled example ($U2$). Since the grey dot is located in the middle distance between the two classes, based on the mixing factor λ, the generated soft label has an equal probability of blue and red classes (50% each). In contrast, in Fig.4.1(B), the same data point (grey-dot) has been generated from a combination of $X2$ (red class) and $U1$ (blue class) with a probability of 90% of being blue and 10% of being red, as it is located closer to $U1$, which leads to the inconsistency of the generated soft labels between the different scenarios.

Manifold Mixup [275], on the other hand, overcomes the above limitations by performing the mixup operation at the hidden layers. Thus, training is carried out on the convex combinations of data samples hidden representations. The learned representations lead to better organization of the hidden state for each class, where it is more concentrated and organized. As a result, the inconsistency of soft labels at interpolated points can be avoided. This can be shown in Fig.4.1.(C), where the generated soft label of grey-dot is consistent, with an equal probability of each class, regardless of the interpolated data points ($X1$ and $U2$ or $X2$ and $U1$).

Both methods, *i.e.* , Input [307] and Manifold [275] Mixup, have been successfully employed for fully supervised segmentation frameworks; e.g. cardiac image segmentation [48], brain tumor segmentation [85], knee segmentation [209], and prostate cancer segmentation [135]. While the previous works have shown the effectiveness of MixUp over standard data augmentation methods in medical images, they depend heavily on fully labeled datasets, which are usually expensive and unavailable. Nevertheless, this paper addresses the scarcity of labeled data by proposing a semi-supervised learning approach.

Recently, MixMatch [29], which inspired our work, introduced Input MixUp to the SSL paradigm achieving SOTA results in image classification. MixMatch augments the model with interpolated data between labeled and unlabeled images at the input space. While this approach is exciting and provides the model with diverse data points, it is somewhat limited and suffers from inconsistent soft labels for the interpolated data points. We argue that performing the mixup operation at *randomly* selected input and hidden representations of the labeled and the unlabeled data provides the network with novel representations and additional training signals that suit the complexity of medical image segmentation tasks. Moreover, it provides durable soft labels of the augmented samples. Our method takes the advantages of both MixMatch and Manifold Mixup to boost the model's performance, leading to better generalizability.

4.4 Methodology

4.4.1 Problem Definition

In the semi-supervised learning, we are given a set of labeled $\mathcal{S}_L = \{\mathcal{X}_L, \mathcal{Y}_L\}$ and unlabeled data $\mathcal{S}_U = \{\mathcal{X}_U\}$, where $\{\mathcal{X}_L, \mathcal{X}_U\} = \{x_1, \ldots, x_L, x_{L+1}, \ldots, x_{L+U}\}$ are input images, $x \in \mathbb{R}^{H \times W}$, where H and W are the height and the width of the input image, respectively, $\mathcal{Y}_L = \{y_1, \ldots, y_L\}$ are the ground truth labels, where $y \in \mathbb{R}^{H \times W \times C}$ for the segmentation tasks, and C represents the number of classes. Usually $L << U$. Our goal is to build a model $f_\theta(x)$ that takes input image x_i and outputs its prediction $\hat{y}_i$. To leverage both labeled and unlabeled data in SSL paradigm, the objective function takes the form

$$\mathcal{L}_{Total} = \mathcal{L}_{Supervised} + \beta \mathcal{L}_{Unsupervised}, \tag{4.3}$$

where β is a weighing factor that controls the contribution of the unsupervised loss, $\mathcal{L}_{Supervised}$ denotes the supervised loss and trained using labeled data $\mathcal{S}_L$, and $\mathcal{L}_{Unsupervised}$ denotes the

unsupervised loss and trained on the unlabeled data $\mathcal{S}_U$. In this section, we will focus on the consistency-regularization approach, where its goal is to minimize the distance between the feature representations of the input data point x and its perturbed version $\hat{x}$. Formally, $L_{Unsupervised} = d(f_\theta(x), f_\theta(\hat{x}))$, where $d(\cdot,\cdot)$ is a distance metric.

4.4.2 ROAM: Random Layer Mixup for Semi-Supervised Learning in Medical Images

The core components of our method are (a) *Pseudo Labeling*: Given a pre-trained model for a few epochs on labeled data, the initial labels for the unlabeled batch were produced, then refined by applying a sharpening operation. (b) *Random Layer Mixup*: The labeled and unlabeled batches were concatenated, then passed to the network as usual. Then, a mixup operation is applied at a random layer, where the paired examples are randomly selected. At the same time, a mixup operation is applied to the corresponding labels. Finally, the process is continued from that layer to the output layer. In the following sections, we illustrate our methodology in detail, while the entire framework and the algorithm are shown in Fig.4.2 and Algorithm 1, respectively.

Pseudo Labels

First, the unlabeled data along with the labeled set are leveraged using two steps; i) sharpening the initial predictions for unlabeled data to minimize its entropy following [29], and ii) mixup the labeled and unlabeled data at random layers following [275]. The unlabeled data are first fed to the model outputting the initial predictions:

$$\hat{y}_i = f(x_i;\theta); \quad \text{where} \quad x_i \in \mathcal{X}_U, \tag{4.4}$$

before being post-processed by a sharpening operation, parameterized with T, which is highly inspired by the entropy minimization literature [29, 106]. The pseudo label set is then defined as $\tilde{\mathcal{Y}}_U = \{\tilde{y}_i, \ldots, \tilde{y}_U\}$, where

$$\tilde{y}_i = \texttt{Sharpening}(\hat{y}_i, T)_j := \hat{y}_{ij}^{\frac{1}{T}} \Big/ \sum_{j=1}^{C} \hat{y}_{ij}^{\frac{1}{T}}, \tag{4.5}$$

where $\hat{y}_i$ is given by Eq.4.4, $j \in C$, and C is the total number of classes. Note that as $T \to 0$, y_i approaches one-hot encoding. Applying the sharpening operation to the initial labels produces more stable predictions by pushing the labels away from the decision boundaries to more confident regions for each class by minimizing its entropy. This effect can be easily seen in Fig.4.1.(D), where the unlabeled data point $U1$ is moved closer to the right distribution.

Random Layers Linear Interpolation

Given the unlabeled data $\mathcal{X}_U$ and its pseudo labels $\tilde{\mathcal{Y}}_U$, along with the labeled data $\mathcal{X}_L$ and its one-hot encoding labels $\mathcal{Y}_L$, the two sets are concatenated as $\mathcal{X} = \{\mathcal{X}_L, \mathcal{X}_U\}, \mathcal{Y} = \{\mathcal{Y}_L, \tilde{\mathcal{Y}}_U\}$.

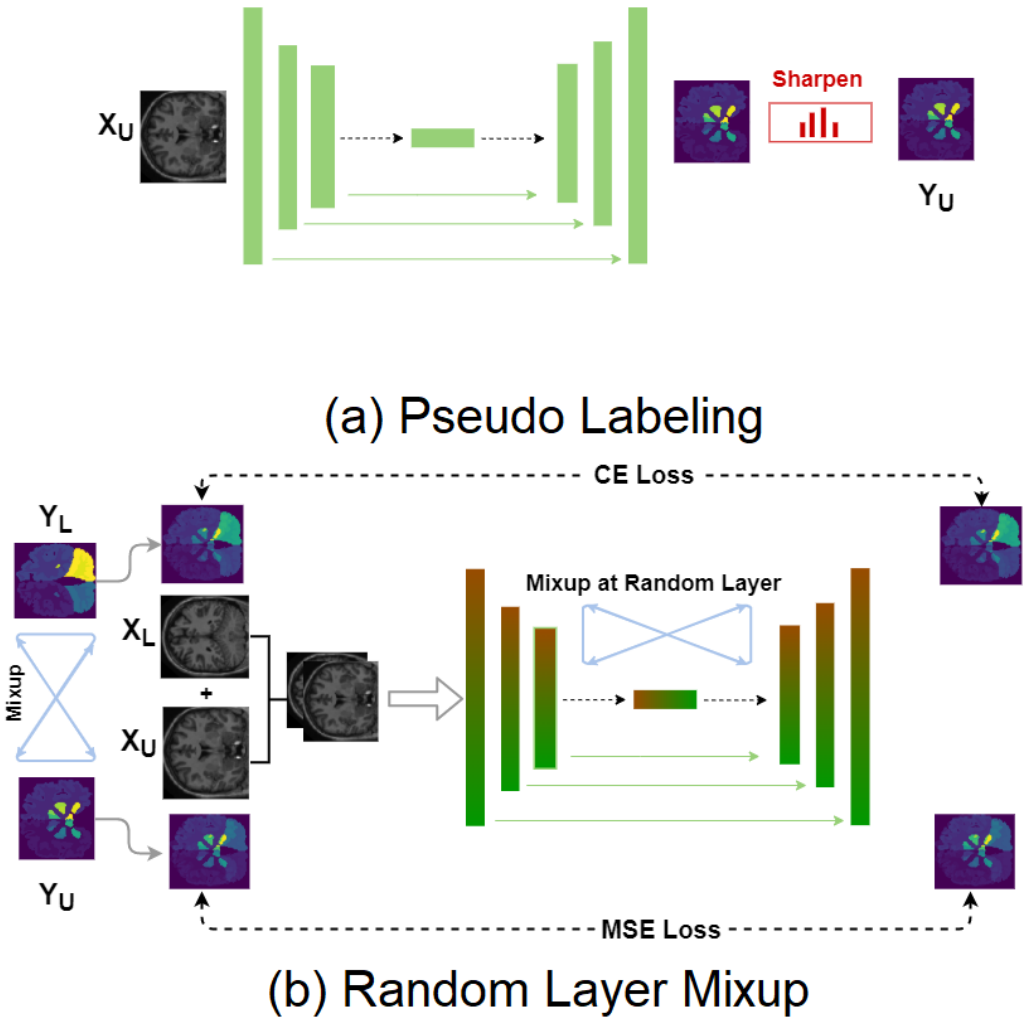


Fig. 4.2. Illustration of ROAM. (a) First, initial labels for the unlabeled batch are produced from a pre-trained model, then, a sharpening step is applied to fine-tune the labels. (b) Second, the labeled and unlabeled batches are fed to the network, and mixed at a random layer. Both models in (a) and (b) are the same, yet we freeze the parameters in step (a).

To enable running the mixup operation at a randomly selected latent space, we first define $(\mathcal{H}, \mathcal{Y})$, where

$$\mathcal{H} = \begin{cases} \mathcal{X}, & \kappa = 0 \\ f_\kappa(\mathcal{X}), & otherwise \end{cases}, \tag{4.6}$$

where $f_\kappa(\cdot)$ is the hidden representation of the input data at layer κ. Note that the input data is selected when $\kappa = 0$. To introduce noisy interpolated data, a permuted version of the original data is created $\tilde{\mathcal{H}}, \tilde{\mathcal{Y}} = \texttt{Permute}(\mathcal{H}, \mathcal{Y})$, Moreover, it fed to the `MixUp` operation as

$$\mathcal{H}' = \lambda' \mathcal{H} + (1 - \lambda')\tilde{\mathcal{H}}, \tag{4.7}$$

$$\mathcal{Y}' = \lambda' \mathcal{Y} + (1 - \lambda')\tilde{\mathcal{Y}}, \tag{4.8}$$

Algorithm 1 ROAM: Random Layer MixUp for SSL

Require: pre-trained model $f(\cdot;\theta^{(0)})$, labeled dataset $\mathcal{S}_L$, unlabeled dataset $\mathcal{S}_U$, batch size B, number of iteration K, The hyper-parameters $\{T, \alpha, \beta\}$

Initialize: $k \longleftarrow 0, \theta \longleftarrow \theta^{(0)}$

while $k \leq K$ **do**
- $\mathcal{B}_L \sim (\mathcal{X}_L, \mathcal{Y}_L)$; $\mathcal{B}_U \sim \mathcal{X}_U$ *//sample labeled and unlabeled batches*
- $\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i;\theta)$; $x_i \in \mathcal{B}_U$ *//initial labels for $\mathcal{X}_U$; Eq.4.4*
- $\tilde{\boldsymbol{y}}_i = \texttt{Sharpening}(\hat{\boldsymbol{y}}_i, T)$ *//pseudo labels; Eq.4.5*
- $\mathcal{X} = \{\mathcal{X}_L, \mathcal{X}_U\}, \mathcal{Y} = \{\mathcal{Y}_L, \tilde{\mathcal{Y}}_U\}$*//concatenate both batches, $\tilde{\mathcal{Y}}_U$ from Eq.4.5*
- $\kappa \longleftarrow$ randomly select layer
- $\mathcal{H} = f_\kappa(\mathcal{X})$*//pass the data to the network, and extract $\mathcal{H}$; Eq.5.7*
- $\tilde{\mathcal{H}}, \tilde{\mathcal{Y}} = \texttt{Permute}(\mathcal{H}, \mathcal{Y})$ *//randomly shuffle the data*
- $\mathcal{H}', \mathcal{Y}' = \texttt{Mixup}(\alpha, \mathcal{H}, \mathcal{Y}, \tilde{\mathcal{H}}, \tilde{\mathcal{Y}})$ *//perform mixup operation; Eqs.(5.8,8)*
- $\mathcal{P} \longleftarrow$ resume passing $\mathcal{H}'$ from layer κ to the output layer
- $\mathcal{P}_\mathcal{L}, \mathcal{P}_\mathcal{U} = \texttt{Split}(\mathcal{P}); \mathcal{Y}'_\mathcal{L}, \mathcal{Y}'_\mathcal{U} = \texttt{Split}(\mathcal{Y}')$ *//split the predictions and labels*
- $\theta \longleftarrow \arg\min_\theta \mathcal{L}_{CE}(\mathcal{Y}'_L, \mathcal{P}_L) + \beta\mathcal{L}_{MSE}(\mathcal{Y}'_U, \mathcal{P}_U)$ *//calculate the loss; Eq.4.9*

end while

where $\texttt{Permute}(.)$ randomly shuffles the data, $\mathcal{H}'$ and $\mathcal{Y}'$ are the interpolated mixed-up data, where the paired examples are selected randomly. To favour the original data over the permuted one, λ' is set to $\max(\lambda, 1-\lambda)$, where $\lambda \in [0,1]$ is sampled from a $\text{Beta}(\alpha, \alpha)$ distribution with α as a hyper-parameter. Further, to keep the original data flow, we run some experiments without the mixup operation and denoted as $\kappa = \Phi$. In practice, this can be achieved by setting κ and λ' to 0 and 1, respectively. To this end, the mixed-up data $\mathcal{H}'$ are fed to the model from layer κ along the way to the output layer at which the segmentation maps are predicted $\mathcal{P}$. Eventually, $\mathcal{P}$ is split back into labeled and unlabeled predictions $\mathcal{P} = \{\mathcal{P}_L, \mathcal{P}_U\}$, and similarly $\mathcal{Y}'$ into $\mathcal{Y}'_L$ and $\mathcal{Y}'_U$.

Overall Objective Function

Our overall objective function is the sum of the cross entropy loss $\mathcal{L}_{CE}$ on the mixed-up labeled data and the consistency mean squared loss $\mathcal{L}_{MSE}$ on the mixed-up unlabeled data,

$$\arg\min_\theta \mathcal{L}_{CE}(\mathcal{Y}'_L, \mathcal{P}_L) + \beta\mathcal{L}_{MSE}(\mathcal{Y}'_U, \mathcal{P}_U), \tag{4.9}$$

where β is a hyper-parameter.

4.5 Experiments & Results

Our experiments to validate our first contribution involve two parts; the whole-brain segmentation (Sec. 4.5.6) and lung segmentation (Sec. 4.5.8). First, a comparison with SSL methods for medical image segmentation is performed (Sec. 4.5.6), followed by a comparison with SOTA methods for whole-brain segmentation in a fully-supervised setting (Sec. 4.5.6). Then, extensive experiments, following the recommendations of [205], are performed (Sec. 4.5.7). Further, the performance of ROAM is investigated in the presence of the domain shift (Sec. 4.5.7). In the second part, lung segmentation results are reported in semi and fully-supervised fashions

(Sec. 4.5.8). Then, ROAM is investigated in the presence of domain shift and class mismatch (Sec. 4.5.8). Finally, the performance vs. infection size is discussed (Sec. 4.5.8)

4.5.1 Datasets

Brain

For whole-brain segmentation, we opt for three publicly available datasets as follows: (i) MALC [161], which consists of 30 T1 MRI volumes, with manual segmentation for the whole brain, which is provided by [162]. This dataset is divided into 15 training and 15 testing volumes (∼2500 slices each). The training volumes are further split into three labeled (∼500 slices), nine unlabeled volumes (∼1500 slices), and three validation volumes (∼500 slices). (ii) IBSR [221], which consists of 18 T1 MRI volumes (∼2000 slices). This dataset is provided with manual segmentation for the whole brain. (iii) CANDI [144], which consists of 13 T1 MRI volumes (∼1500 slices). Neuromorphometrics, Inc provides the manual segmentation for whole-brain for this dataset. The labels for whole-brain segmentation include 27 classes (27 internal structures); Left Cortical WM, Left Cortical GM, Right Cortical WM, Right Cortical GM, Left Lateral Ventricle, Left Cerebellar WM, Left Cerebellar GM, Left Thalamus, Left Caudate, Left Putamen, Left Pallidum, 3rd Ventricle, 4th Ventricle, Brain Stem, Left Hippocampus, Left Amygdala, Left Ventral DC, Right Lateral Ventricle, Right Cerebellar WM, Right Cerebellar GM, Right Thalamus, Right Caudate, Right Putamen, Right Pallidum, Right Hippocampus, Right Amygdala, Right Ventral DC.

Lung

Two publicly available datasets for lung segmentation are used. (i) COVID-19-CT-Seg-Benchmark [134]: which consists of 20 CT volumes with the segmentation of three classes; right lung, left lung, and infection. The data is divided into ten training and testing volumes (∼2000 slices each). The training data is further divided into two labeled volumes (∼300 slices), seven unlabeled volumes (∼1400 slices), and one validation volume (∼300 slices). (ii) MedSeg: Consists of 100 axial CT images (*i.e.* , slices) from more than 40 patients with COVID-19. The images are divided into 80 training images and 20 validation images. The labels include ground-glass, consolidation, and pleural effusion classes. The whole-lung masks for this data set are provided separately. Thus, we combined them with the previous three classes to create labels for four classes.

In all previous data settings, a patient-wise random splitting strategy was considered to avoid any overlaps, such that all slices for a specific volume/patient appear in one splitting. All images are resized to the dimension of 256×256, where the resolution is 1.5mm for the brain images and in the range of ∼ 0.86mm to 1.2mm for the lung images. The intensity values normalized to $[0, 1]$.

4.5.2 Baselines

Our baselines include: (i) The lower bound models, trained on the labeled volumes. (ii) The SSL models are trained on the labeled and the unlabeled volumes. (iii) The upper bound models trained on the labeled volumes and the nine unlabeled volumes. However, all labels

are revealed. (iv) Regularized ROAM: to evaluate our contributions, our method is introduced as a regularizer to the fully supervised lower and upper bound models, denoted as ROAM-LB, and ROAM-UB, respectively.

For the SSL setting, the following methods are selected. (i) Bai *et al.*[18], (ii) Baur *et al.*[21], (iii) Cui *et al.*[66], and (iv) Zhang *et al.*[313]. We opt for these methods based on the following criteria. First, one method from each of the SSL approaches is chosen. Second, the easiness of implementation and the compatibility with the unified architecture. Third, we rule out the 3D methods or the methods that introduce sophisticated training mechanisms, such as Multi-view training, uncertainty estimations, and domain adaptation.

4.5.3 Implementation details

2D U-Net [222] is employed as backbone architecture, where the 2D slices are the input for the network. The weights are initialized using Xavier [100] initialization and trained using Adam optimizer [147]. The learning rate, weight decay, and batch size are set to 0.0001, 0.0001, and 8, respectively. The initial models denoted lower bounds trained for 40 epochs, the other semi-supervised, and the upper bound models further trained for an additional 40 epochs. The hyper-parameters are set to $T = 0.5$, $\alpha = \{0.75, 1\}$, and $\beta = \{75, 1\}$ for the brain and lung datasets respectively. The mixup layer κ is selected randomly from the input, the first, and the last convolution layers, which is denoted as $\kappa = \{0, 1, L\}$ for the brain images and $\kappa = \{\Phi, 0, 1, L\}$ for the lung images, where Φ means no mixing of the data performed. All the experiments are performed using PyTorch framework hosted on an NVIDIA GTX 1080 8GB machine. The training time is about 6 hours. The model with the best validation accuracy is used to report the testing results

4.5.4 Evaluation Metrics

The statistical summary of the Dice score, Eq.2.17, are reported. In addition, the Hausdorff distance (HD), Eq.2.19, and the Mean Surface Distance (MSD), Eq.2.20, are reported. A Relative Improvement (RI) *w.r.t* the baseline is also reported such that RI of a over b is : $(a - b)/b$. Note that we follow the *One vs ALL* methodology for calculating previous metrics such as for the multi-class segmentation, the mean value of any metric, *i.e.* , Dice, HD, or MSD is calculated by taking the value of each class individually and averaging them.

4.5.5 Ablation Study

ROAM introduces the sharpening and concatenation operations to the Manifold Mixup. Also, it involves a set of hyper-parameters, *i.e.* , (α, β), and design choices, *i.e.* , κ, in the training process. Thus, for the model selection, an ablation study and sensitivity analysis are conducted. In all these experiments, the training is done for 80 epochs, where the model with the highest validation accuracy is selected to report the testing results. The results are presented in Table 4.1.

Tab. 4.1. Mean Dice for Brain validation and testing datasets. ROAM, with $\kappa = \{0, 1, L\}$, sharpening, concatenation, $\alpha = 0.75$, and $\beta = 75$, obtains the best validation results, hence, will be our model selection. Φ: no data mixup. All: all hidden layers. L: last layer. First, κ is examined when α and β are equal to 0.75 and 75, respectively. Based on the results, $\kappa = \{0, 1, L\}$ is used before the selection of the other parameters is investigated.

Ablation	Value	Validation	Testing
ROAM	{0, 1, L}	**0.898**	0.870
κ	0	0.881	0.852
	1	0.867	0.843
	2	0.894	0.872
	3	0.868	0.825
	4	0.863	0.828
	5	0.877	0.847
	L	0.865	0.843
	{0, 2, L}	0.884	0.851
	{1, 2, L}	0.883	0.863
	{0, 1, 5}	0.881	0.860
	{Φ, 0, 1, L}	0.882	0.864
	{All}	0.882	0.858
α	0.25	0.880	0.851
	2	0.885	0.836
β	0	0.893	0.844
Sharpening(✓)	Concatenation(×)	0.878	0.850
Sharpening(×)	Concatenation(✓)	0.861	0.823
Sharpening(×)	Concatenation(×)	0.870	0.843

The selection of the random layer κ

First, a set of layers are examined to determine which mixup operation will obtain the best results. That includes the input layer, the hidden layers, and a no-mix option, where the data is passed to the network as per the usual training procedure. Note that κ is investigated when α and β are equal to 0.75 and 75, respectively. Please refer to section 4.5.5 for why these values were selected. It is seen from the results in Table 4.1 that mixing the data at different random layers achieves better results than using only one fixed layer, except for $\kappa = 2$. This correlation emphasizes the importance of alternating the hidden space with the input space during the training process, which provides the model with novel data variations that can not be generated using either the input or the hidden layers. Based on these results, $\kappa = \{0, 1, L\}$ is fixed before the selection of the other parameters is investigated as presented the next sections.

The concatenation & the sharpening operations

In this experiment, we removed the sharpening step on the soft labels and(or) concatenated labeled and unlabeled batches, which resulted in three combinations as shown in the last rows in Table 4.1. When removing one or both steps, a drop in the Dice score was observed. However, the worst result was obtained when applying the mixup operation on a concatenated batch without the sharpening. That is attributed to mixing the initial labels without minimizing their entropy through the sharpening step, which could harm the quality of the mixed-up data.

The hyperparameters α and β

First, three values of $\alpha = \{0.25, 0.75, 2\}$ are examined, where $\alpha = 0.75$ as in [29], $\alpha = 0.25$ to favor one sample over the other, and $\alpha = 2$ to make more balance between the different samples. It is noticed from Table 4.1 that ROAM obtains the best results when selecting $\alpha = 0.75$ because this value makes the mixed-up data closer to the original data while maintaining the novelty of the generated points. In the final part of our analysis, two values of $\beta = \{0, 75\}$ are investigated, where $\beta = 75$ as in [29], and $\beta = 0$ to evaluate the effectiveness of the newly-generated data on the training where we do not propagate the unlabeled loss. The results in Table 4.1 show that ROAM effectively uses the unlabeled loss, where the accuracy when $\beta = 75$ is better than when $\beta = 0$. Furthermore, the obtained results at $\beta = 0$ show that the random layer mixup operation boosts the performance without the unlabeled loss. That is because the mixup between the labeled and the unlabeled examples augments the model with new virtual data.

In summary, the above analysis shows that ROAM, with $\kappa = \{0, 1, L\}$, sharpening, concatenation, $\alpha = 0.75$, and $\beta = 75$, obtains the highest validation accuracy. Unless stated otherwise, we opt for these selections in the next experiments. In some experiments, we report the results at the input space *i.e.* , ROAM($\kappa = 0$) to compare our method with MixMatch. Also, we report the results for ROAM($\kappa = 2$) because it obtains the second-highest validation accuracy and evaluates our method at the Manifold Mixup.

On the other hand, model selection experiments on lung validation data are conducted. Similarly, the hyperparameters $\{\kappa, \alpha, \beta\}$, concatenation, and sharpening steps are examined. The results in these experiments show that ROAM with $\kappa = \{\Phi, 0, 1, L\}$, sharpening, concatenation, $\alpha = 1$, and $\beta = 1$, obtains the highest validation accuracy. Thus, we opt for this selection for lung segmentation testing results.

The ablation study from both datasets shows the essential role of each component of our method on the segmentation task justifying its design choice.

4.5.6 Whole-brain Segmentation Results

Comparison with SSL methods.

Table 5.2 illustrates the results for whole-brain segmentation. It is apparent from this table that our method outperforms the lower bound, upper bound, and all SSL methods with a statistical significance ($p < 0.001$). The best result, with average Dice of 87.0% and RI about 16.50%,

Tab. 4.2. Mean (Median) ± Std. of different evaluation metrics are reported on the MALC testing set for baselines and different SSL methods, including ours. *: significant improvement. L: Last layer. †: MixMatch [29]. ↑(↓): The higher (lower) the better.

Model Name	Dice Coefficient ↑	RI(%) ↑	HD ↓	MSD ↓
Lower Bound	0.747(0.769)±0.071*	0	4.16±0.43	1.06±0.088
ROAM-LB	**0.823(0.841)±0.052**	**10.17**	**4.07±0.35**	**1.05±0.071**
Bai *et al.*[18]	0.800(0.815)±0.055*	7.10	4.06±0.43	1.02±0.086
Zhang *et al.*[313]	0.819(0.851)±0.060*	9.64	4.02±0.44	1.00±0.089
Cui *et al.*[66]	0.829(0.847)±0.045*	11.00	3.97±0.38	1.03±0.089
Baur *et al.*[21]	0.778(0.795)±0.071*	4.15	4.06±0.40	1.05±0.082
ROAM ($\kappa = 0$)†	**0.852(0.866)±0.037**	**14.05**	**3.91±0.35**	**0.99±0.067**
ROAM ($\kappa = 2$)	**0.872(0.881)±0.024**	**16.73**	**3.78±0.28**	**1.00±0.077**
ROAM ($\kappa = \{0, 1, L\}$)	**0.870(0.873)±0.023**	**16.50**	**3.87±0.31**	**1.00±0.061**
Upper Bound	0.871(0.886)±0.044*	16.60	3.72±0.42	0.95±0.087
ROAM-UB	**0.893(0.902)±0.024**	**19.54**	**3.56±0.34**	**0.91±0.075**

is obtained by ROAM($\kappa = \{0, 1, L\}$). Further analysis shows that ROAM($\kappa = \{0, 1, L\}$) outperforms its variant ROAM($\kappa = 0$), which is similar to MixMatch. The justification is that ROAM($\kappa = \{0, 1, L\}$) introduces a lot of variations and generates novel data points that have never been seen before via its random layer mixup. Thus, it avoids over-fitting. Interestingly, a similar performance is reported for ROAM($\kappa = 2$). Further statistical tests revealed that our method achieves the best HD and MSD scores of 3.87 and 1.00, respectively. Moreover, ROAM-LB and ROAM-UB models outperform their competitors significantly with average Dices of 82.3% and 89.3%, and RI of 10.17% and 19.54%, respectively. That is strong evidence that applying ROAM as a regularizer provides the model with new data points. Consequently, it boosts the performance without the need for any additional data. Surprisingly, ROAM-LB outperforms most SSL methods by significant margins, confirming its advantages as a strong regularizer.

Structures Level Results

The segmentation results for some internal structures are reported in Fig.4.3. The results show that ROAM significantly outperforms all SSL methods in most structures. Besides, ROAM excels over the upper bound in the Right Hippocampus and 3rd Ventricle. Additionally, the performance of our method is consistent across different structures. That is clearly shown in the Left Pallidum, 3rd Ventricle, Left Amygdala, and Right Hippocampus. Our model achieves a lower performance in Left Cortical GM, yet the difference is not statistically significant.

Qualitative Results

To provide more insights on the performance, the qualitative results are shown in Fig.4.4. The first row shows the predictions on the MALC dataset. The second row shows a cropped version, where we highlighted the right and left lateral ventricle, right thalamus, right hippocampus,

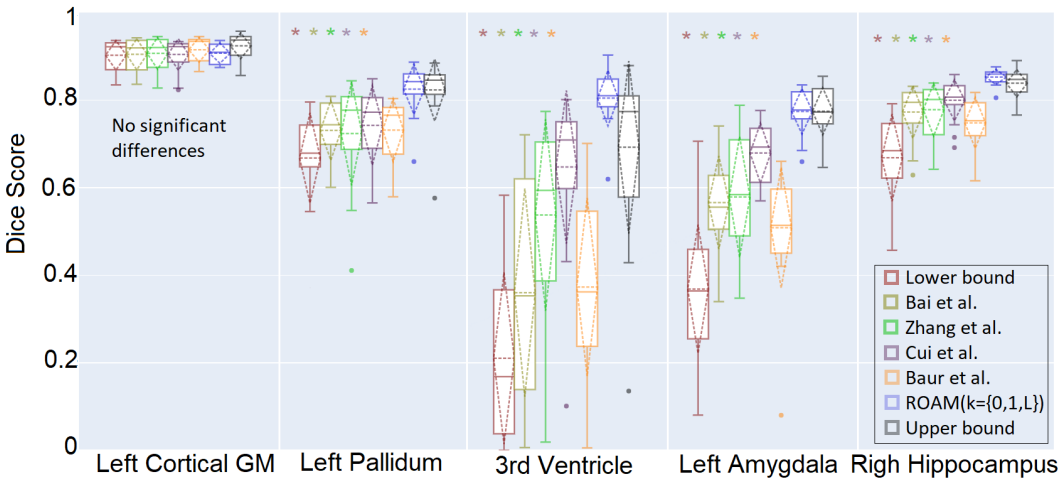


Fig. 4.3. Dice score for selected structures. Our method significantly outperforms all other SSL methods in most brain structures.

left palladium, left amygdala, and 3rd ventricle. Despite the complexity of these small structures, ROAM performs more reliably than all SSL methods. To support our findings, we also include another case from the MALC dataset in the third row. Likewise, ROAM surpasses all SSL methods. Finally, the predictions under cross-domain settings are shown for IBSR and CANDI datasets in the fourth and fifth rows, respectively. In general, ROAM predicts more accurate results than other SSL methods indicating its generalization ability to other domains. Together, the quantitative and the qualitative results show the superiority of ROAM against all SSL methods.

Comparison with SOTA for Whole Brain Segmentation

To realize the effectiveness of ROAM in a fully-supervised fashion, we run our method using the labeled data. In this experiment, the batch is mixed with its permuted version, where no sharpening nor pseudo labeling steps are performed. Also, β is set to 0 so that the unsupervised loss is not propagated. The MACL dataset is used for the training for 80 epochs, where the model at the last epoch is saved. Our method is compared with U-Net[222], and QuickNAT [226]. In contrast to U-Net and our model, QuickNAT is pre-trained using 581 labeled volumes from IXI dataset. Table 5.4 shows the testing results on the MALC dataset. All ROAM variations significantly outperform U-Net and are on par and sometimes outperform QuickNAT, without a sophisticated pre-training mechanism. Note that ROAM ($\kappa = 0$) is a special case of our method where the mixup is performed at the input space i.e. *MixMatch*. Further, the results show that our models achieve lower standard deviations compared to other methods. In summary, the results show that our simple but elegant ROAM operation leads to SOTA results without the need for large datasets.

4.5.7 Realistic Evaluation of ROAM

The purpose of the next set of experiments is to (i) assess ROAM in the presence of domain shift, (ii) show the correlation between the amount of labeled and unlabeled data on the overall performance, following the recommendations of [205].

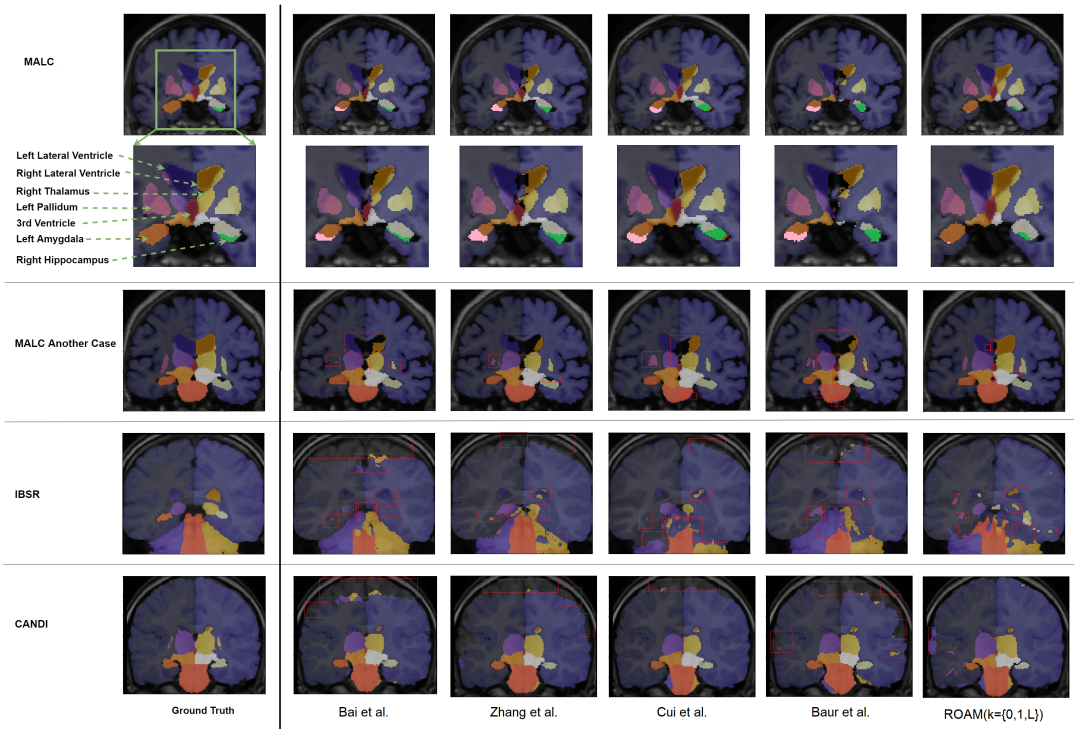


Fig. 4.4. Qualitative results of brain segmentation. The first row shows a coronal view of one case from the MALC dataset. The second row shows a cropped version highlighting selected structures for the same case. Another case is shown in the third row. Further, the segmentation results in the presence of domain shift are shown in the fourth and fifth rows of IBSR and CANDI datasets respectively. In these cases, ROAM obtains the best results, where the red boxes show the false predictions made by different models.

Tab. 4.3. Dice score for fully supervised models. ROAM significantly outperforms both U-Net and on par with QiuckNAT without sophisticated pre-training mechanism.

Model Name	Mean(median)±std	RI(%)
U-Net	0.874(0.888)±0.039	0
QuickNAT	0.895(N/A)±0.055	2.40
ROAM ($\kappa = 0$)	0.890(0.898)±0.025	1.83
ROAM ($\kappa = \{0, 1, L\}$)	**0.895(0.901)±0.022**	2.40
ROAM ($\kappa = 2$)	**0.897(0.906)±0.025**	2.63

Domain Shift Results

The trained models were picked and tested on IBSR and CANDI datasets. The results in Fig.4.5 show a drastic drop in all models, including the baseline ones. This drop is higher on the ISBR dataset. However, ROAM($\kappa = \{0, 1, L\}$) performs just as well in both cases and is less sensitive to the domain shift problem compared with other models, including ROAM($\kappa = 2$) and ROAM($\kappa = 0$). Surprisingly, although ROAM($\kappa = 2$) achieves one of the best results on the MALC dataset, it has less generalization ability than ROAM($\kappa = \{0, 1, L\}$). The results indicate that the domain shift has a lower effect on ROAM than the other methods.

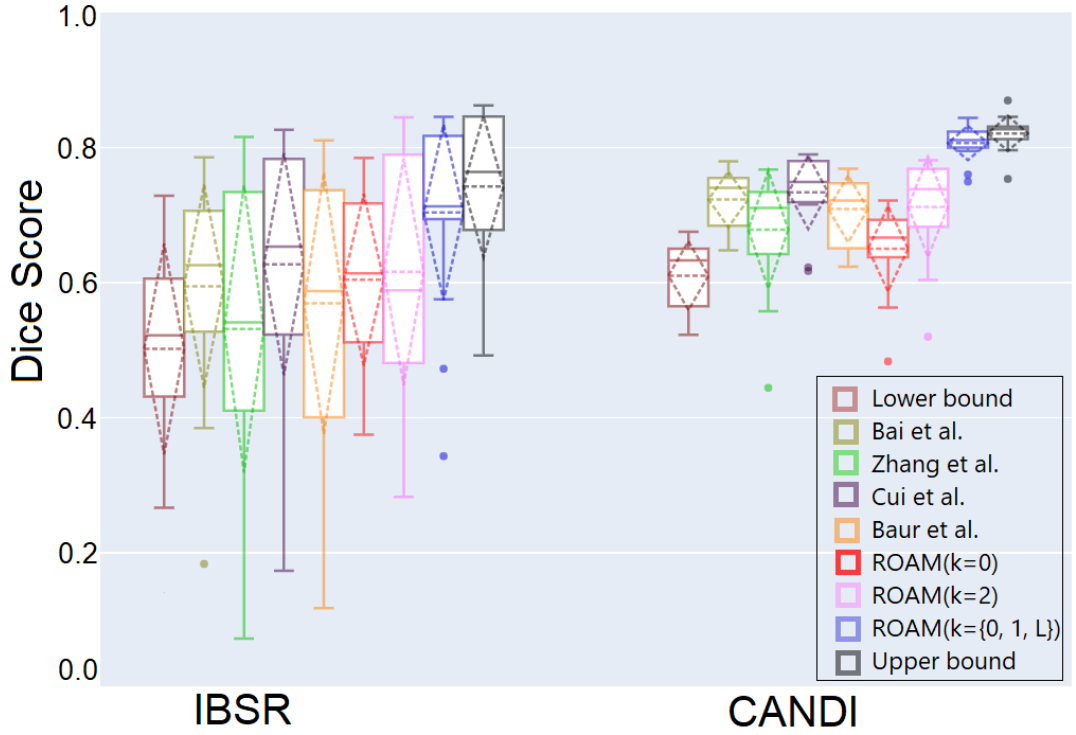


Fig. 4.5. Domain shift results. The domain shift has a lower effect on ROAM than the other methods.

Changing Amount of Labeled Data

At first, we fix the number of unlabeled data at 1500 slices while gradually increasing the amount of labeled data from 100 to 500. With successive increases in the amount of the labeled data, our model displayed a higher performance and confidence compared to other models (*cf.* Fig.4.6.(a)). This confidence level is inconsistent in other models. The same superiority is also observed at the lowest amount of labeled data (100 slices), where the obtained Dice scores are 0.622, 0.402, 0.500, 0.571, 0.400 for ROAM, Bai *et al.* [18], Zhang *et al.* [313], Cui *et al.* [66], and Baur *et al.* [21] respectively, *cf.* Fig.4.6.(a), the results on the far left.

Changing Amount of Unlabeled Data.

In this experiment, we fix the labeled data at 500 slices while gradually reducing the unlabeled from 1500 to 500. The results are shown in Fig.4.6.(b). In contrast to other methods, our model shows its superior *w.r.t* variable amount of unlabeled data. The figure shows that our approach still outperformed when the amount of unlabeled data is the lowest (500 slices) with considerable margins. The obtained Dice scores for ROAM against the other methods are 0.820, 0.795, 0.798, 0.809, and 0.760 respectively, *cf.* Fig.4.6.(b), the results on the far right. Yet, [66] achieves insignificant higher Dice at 1000 unlabeled slices. Both results confirm the superiority of our method at a low data regime.

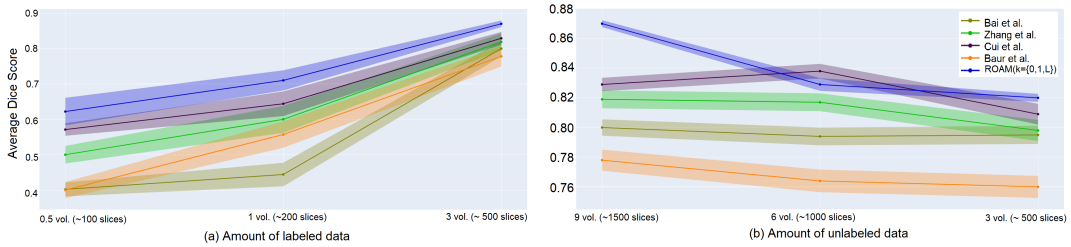


Fig. 4.6. Varying amount of data. The shaded region represent the standard deviation. The more labeled or unlabeled data being used, the higher performance and confidence of our model comparing to others.

Tab. 4.4. Lung CT images segmentation results. ROAM outperforms SSL methods for the infection and lung classes, while it outperforms the lower bound in the overall and lung results. ROAM shows lower performance in the infection segmentation comparing to U-net. The foreground column includes the infection, the left, and the right lung classes. (): negative value.

Setting	Model	Foreground		Infection		Lung	
		Mean(median)±std	RI(%)	Mean(median)±std	RI(%)	Mean(median)±std	RI(%)
Lower Bounds	U-Net	0.702(0.738)±0.176	0	**0.543(0.657)±0.254**	**0**	0.782(0.897)±0.231	0
	ROAM-LB	**0.777(0.839)±0.126**	**10.68**	0.528(0.606)±0.275	(2.76)	**0.902(0.942)±0.121**	**15.35**
SSLs	Bai *et al.*[18]	0.730(0.772)±0.154	3.99	0.552(0.599)±0.233	1.66	0.819(0.881)±0.153	4.73
	Zhang *et al.*[313]	0.736(0.775)±0.161	4.84	0.606(0.717)±0.251	11.60	0.802(0.880)±0.213	2.56
	Cui *et al.*[66]	0.810(0.873)±0.116	15.38	0.605(0.672)±0.239	11.42	0.913(0.953)±0.102	16.75
	ROAM	**0.822(0.887)±0.122**	**17.09**	**0.632(0.710)±0.252**	**16.39**	**0.918(0.957)±0.103**	**17.39**
Upper Bounds	U-Net	**0.849(0.888)±0.096**	**20.94**	**0.675(0.737)±0.229**	**24.31**	**0.936(0.974)±0.091**	**19.69**
	ROAM-UB	0.829(0.872)±0.107	18.09	0.630(0.686)±0.218	16.02	0.929(0.974)±0.102	18.80

4.5.8 Lung Segmentation Results

In the second part of our experiments, ROAM is validated on lung CT images for lung segmentation. Note that our model selection for this dataset is ROAM($\kappa = \{\Phi, 0, 1, L\}$), α and $\beta = 1$.

COVID-19-CT-Seg-Benchmark Results

Quantitative Results

The segmentation results, reported in Table 5.5, show that ROAM and ROAM-LB surpass their competitors in the overall results, see the foreground column in Table 5.5. The obtained relative improvements are $10.86\%, 17.09\%$, and 18.09% respectively for ROAM-LB, ROAM, and ROAM-UB. In line with the whole-brain segmentation results, it also observed that ROAM-LB outperforms the other SSL methods by considerable margins. In contrast to that, ROAM-UB performs just lower than the upper bound. Surprisingly, ROAM-LB's segmentation score for the infection dropped by 2.76% comparing to the U-Net. In summary, ROAM outperforms all SSL methods for all classes, and outperforms the lower bound in the overall and lung results. Yet, ROAM shows lower performance in the infection segmentation when compared to U-net.

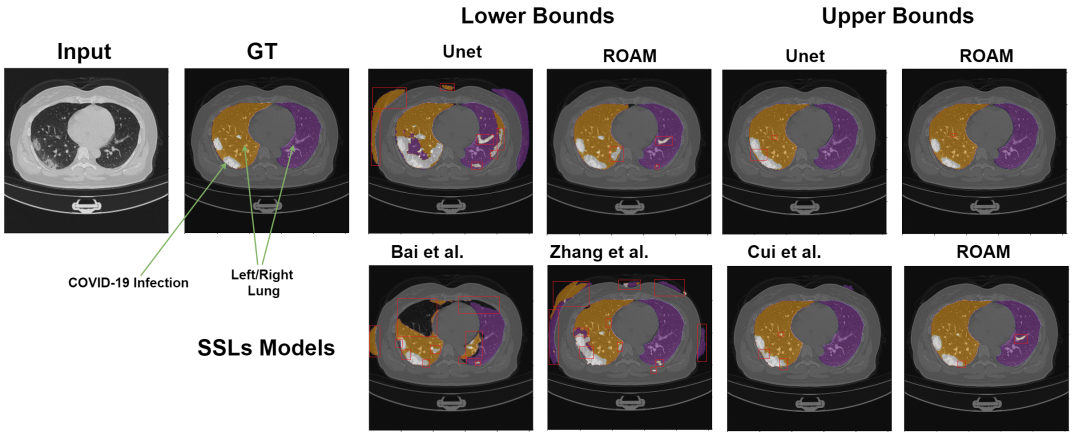


Fig. 4.7. Qualitative results of lung segmentation. Red boxes represent the false positives. ROAM generates more accurate predictions than the other models with the exception of the U-Net-UB.

Qualitative Results

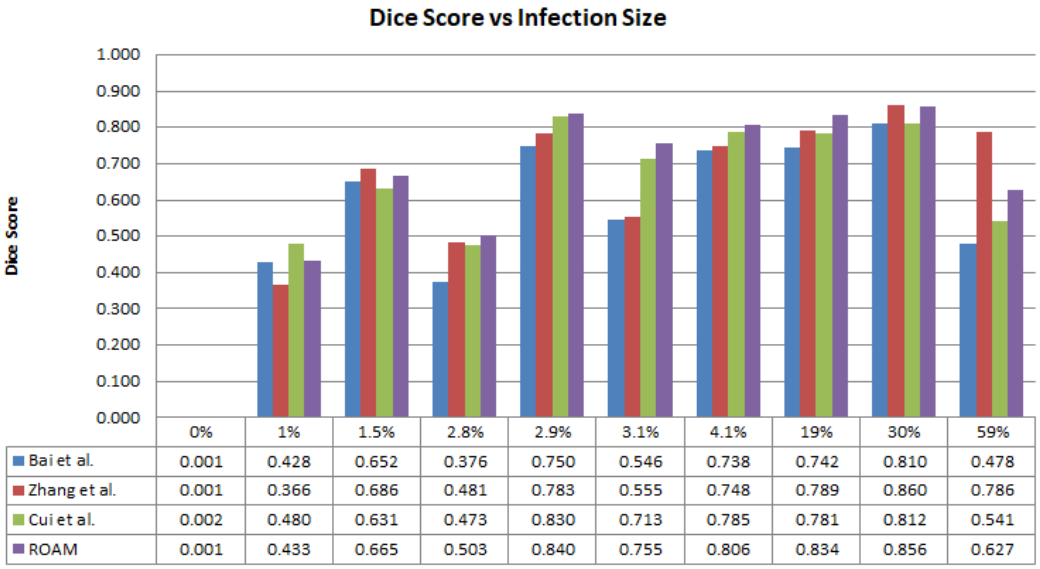
The segmentation predictions for the previous models are shown in Fig.4.7. The first two columns in the first row show the input image with its ground truth. The next four columns present the segmentation results for the lower and upper bounds, respectively. The second row shows the predictions for the SSL methods and ROAM. The red boxes are drawn to show the false predictions made by different models. Except for the upper bound, ROAM makes fewer false positives and generates more accurate predictions than the other models in all settings. Moreover, ROAM-LB performs better than U-Net lower bound and better than some SSL methods such as [18], and [313]

MedSeg: Cross Domain & Class Mismatch Results

MedSeg dataset consists of 100 CT images divided into 80 training images and 20 validation images, with four classes of lung, ground-glass opacity, consolidation, and pleural effusion. In this experiment, the model trained on MedSeg while it was tested on the COVID-19-CT-Seg-Benchmark dataset. Notice that the last dataset contains segmentation of the right lung, left lung, and infection classes. Thus, the goal of this experiment is to investigate the ability of ROAM cross domains and class mismatch conditions. Note that the training and the testing images come from different datasets with a domain shift problem. Further, the training and testing classes differ, making it a very challenging task. To resolve this issue, we perform two steps. First, after training the models, we generate the four-class predictions. Then, we assemble the predictions of ground-glass opacity, consolidation, and pleural effusion as one class called the infection class, yet the lung predictions remain without any modification. The result from the previous step is predictions of two classes; lung and infection. The second step, however, is performed on the testing data. Specifically, the right and lung masks are assembled as one class called the lung class, while the infection masks remain without any modification. The result from this step is labels with two classes; lung and infection. Having performed these two steps, our results for this experiment are generated. The results are reported in Table 4.5. We notice that ROAM enhances the predictions by 16.26% and 5.78% for the infection and foreground, respectively. The results of this experiment are in line with the results reported for brain images. That is, both are consistent and highlight the ability of ROAM to generalize to unseen data.

Tab. 4.5. Cross domain and class mismatch results. The models are trained on MedSeg dataset, while tested on COVID-19-CT-Seg-Benchmark dataset. ROAM enhances the prediction of the baseline.

Model	Foreground	Infection
	Mean(median)±std RI(%)	Mean(median)±std RI(%)
U-Net	0.675(0.684)±0.107 0	0.449(0.496)±0.233 0
ROAM	**0.714(0.728)±0.111 5.78**	**0.522(0.501)±0.224 16.26**



	0%	1%	1.5%	2.8%	2.9%	3.1%	4.1%	19%	30%	59%
Bai et al.	0.001	0.428	0.652	0.376	0.750	0.546	0.738	0.742	0.810	0.478
Zhang et al.	0.001	0.366	0.686	0.481	0.783	0.555	0.748	0.789	0.860	0.786
Cui et al.	0.002	0.480	0.631	0.473	0.830	0.713	0.785	0.781	0.812	0.541
ROAM	0.001	0.433	0.665	0.503	0.840	0.755	0.806	0.834	0.856	0.627

Fig. 4.8. Dice vs Infection. The x-axis represents the percentage of the infection size to the lung size. When the percentage below 3%, SSLs produce uncertain results. The best obtained when the classes are balanced (at 30%). The percentage of (59%) is an outlier case.

Performance vs Infection sizes

Thus far, the performance of ROAM at different data settings, domain- shift, and class mismatch has been reported. In this experiment, we try to analyze the effect of the COVID-19 infection size *w.r.t* the lung size. Fig.4.8 shows the individual Dice score for each test volume from the COVID-19-CT-Seg-Benchmark dataset. The percentage below each column represents the infection size. It stands out that the same pattern for all SSLs models is found. First, when the infection size is below 3%, all SSL methods produce uncertain results. Second, the best results are obtained when the classes are balanced (at 30%). In this case, the infection represents one-third of the lung size, while the remaining percentages are one-third for the left lung and one-third for the right lung. Third, the results at 59% represent outlier cases that fool all models because the infection represents the minor class in the image in the usual cases. In general, ROAM obtains the best results regardless of the infection percentage.

4.6 Discussion

In this dissertation, we address the insufficiency of annotated data by proposing annotation-efficient medical imaging in deep learning-based approaches. Our first solution is to augment the model with new data generated from random linear interpolation of the input and hidden spaces in a semi-supervised learning setting. Specifically, we propose ROAM as an SSL method that utilizes the modern regularization methods *i.e.* , MixUp and Manifold Mixup, to boost the model with newly-generated data points. Our approach overcomes the limitations of the previous works by exploring the manifold and performing linear interpolation at a randomly selected subset of input and hidden layers. Consequently, it generates new data points and additional training signals that suit the complexity of medical image segmentation for semi-supervised learning. Moreover, our method utilizes the better organized hidden representation of classes and produces consistent soft labels for the corresponding data points generated via the mixup operation. In this chapter, we will discuss our findings and observations of this method in detail.

4.6.1 Performance Across Different Datasets

Our method is validated using five publicly available datasets for the brain and lung images. These datasets are heterogeneous. While the structures in the brain images are almost rigid and geometrically constrained, the lung images contain highly variable sizes of COVID-19 infection. The results show that ROAM performs consistently and outperforms all SSL methods with large margins across these datasets. Further, the robustness of ROAM is significant in the brain segmentation, where ROAM always obtains the best results.

The main advantage of ROAM is the generating of new virtual data points. This process enriches our method with a wide range but free training signals which are explored not only in the input space but also at the hidden representations. Further, what makes our new data beneficial is selecting the mixing factor λ, which has been selected to keep the virtual points in the vicinity distribution of the training examples. In contrast, the other methods are limited to the original training examples such that whatever approach is used, the knowledge gain is still limited. Another advantage is that our method enhances the quality of the pseudo labels by the sharpening operation, while none of the remaining approaches make use of this post-process step.

On the other hand, ROAM ($\kappa = 2$) achieves one of the best results on the MALC dataset. We attributed this to hidden representation at this layer, where it might be the most organized and concentrated among all layers. Thus, the inconsistency in soft labels is minimized. Despite that, ROAM ($\kappa = 2$) has less generalization ability than ROAM ($\kappa = 0, 1, L$), indicating the possibility of overfitting to the training data. Further investigation might lead to more explanations.

However, one limitation has been noticed in the lower bound for the COVID-19 infection segmentation. Even though ROAM enhances the overall prediction, it fails to enhance the segmentation of the infection class. This could be attributed to the fact that mixing highly

imbalanced classes, *i.e.* infection pixels vs lung pixels, at a low data regime, could bias the model to the dominant class. In another anticipated finding, ROAM-UB achieves lower performance than the upper bound baseline model. A possible explanation might be that the amount of data- at the upper bound setting- is enough for the training. Therefore, augmenting the model with additional virtual data points may not be useful. Also, it has been shown that our lower bound model (ROAM-LB) consistently outperforms many SSL methods. That means this regularization technique, with just a few labeled data, could surpass the other SSL methods which have access to a large amount of unlabeled data. Moreover, ROAM generates new data points through its linear interpolation. The effectiveness of this operation is essential at a lower data regime where the data is crucial for the training.

Interpretability might be another limitation of our method. The generated data from the mixup operation could be hard to interpret, especially when the two mixed-up samples are randomly selected. Consider, for example, in brain experiments, an image containing the white matter was mixed with another one containing grey matter or any other brain structure. For a human or an expert, the resulting image will not be recognized as a known structure in the brain. Thus, instead of augmenting the training, this should confuse the model. Although, our experiments showed that this operation boosted the performance, the explainability of our method needs further investigation.

4.6.2 Generalizability & Domain Mismatch

One way to alleviate the need for a large amount of annotated data is to utilize datasets generated from different sources. Usually, these datasets come with many challenges, i.e. different cohorts, scanning protocols, and scanners. That leads to a technical challenge, the so-called domain shift. This problem has been investigated in this paper and have noticed that all SSL methods, including ROAM, suffer in the presence of domain shift. Yet, ROAM was less sensitive, see Fig.4.5 and Table 4.5. Nevertheless, we make no claim here that our approach is domain agnostic. Thus, further research in handling the domain shift in the SSL methods is of high importance.

4.6.3 Convergence

Manifold Mixup is guaranteed to be converged when the mixup operation is performed at a hidden layer, as long as the dimensionality of that layer is greater than the number of the classes [275]. In our paper, this condition is satisfied where the dimensionality of the hidden layers > 32, which is greater than the number of segmentation classes *i.e.* 28 for brain images and 4 for lung images.

4.6.4 Handling Skip Connections

An important question is how to handle the skip connections when mixing at a random layer. Do the skip connections get interpolated using the same lambda as the convolution layers or just forwarded without any mixup? For example, when mixing two samples x_1 and x_2 at a random hidden layer *i.e.* $\kappa = 2$, the skip connections related to that layer still hold the

Tab. 4.6. The results for Whole-brain and lung segmentation at $\kappa = 2$ with/out skip-connections mixup. ROAM works better without skip-connections mixup at SSL setting, while it performs just lower at the upper bounds. SK: Skip-Connection Miuxp.

Dataset	Model	SK	Mean(median)±std
Brain	ROAM-SSL	✓	0.834(0.853)±0.047
		×	0.872(0.881)±0.024
	ROAM-UB	✓	0.892(0.898)±0.024
		×	0.890(0.898)±0.023
Lung	ROAM-SSL	✓	0.779(0.849)±0.137
		×	0.797(0.860)±0.121
	ROAM-UB	✓	0.851(0.889)±0.100
		×	0.850(0.901)±0.107

original data from the first hidden layer. Therefore, they will not correspond to the mixed-up labels properly, which might cause a problem. One suggestion to handle this issue is to perform the mixup for a given layer and the skip-connections up to that layer with the same lambda and the same example-pairing. Practically, we investigate this solution on MALC and COVID-19 datasets when $\kappa = 2$, and report the results in Table 4.6. It is shown that ROAM performs differently, and no such approach produces consistent performance. For instance, the skip-connections mixup at SSL settings impairs the results while it has almost no effect or a negligible positive effect at the upper bounds. The issue of handling the skip-connections is intriguing and could be usefully explored in further research. Fortunately, this problem did not happen in our main scenarios, *i.e.* , performing random mixup at $\kappa = \{0, 1, L\}$ because the mixed-labels correspond to the mixed data as well. Yet, it is not the case for the manifold mixup when $\kappa = 2$, which surprisingly shows a superior result. One of the reasons could be attributed to the choice of the beta distribution parameter, *i.e.* α. For instance, when α is less than 1, the mixed data tend to preserve the original data point. Therefore, performing manifold at the bottleneck or other layers might not have such an expected negative impact.

4.6.5 Infection Size

ROAM can be affected by the highly-imbalanced dataset as can many SSL methods. Fig.4.8 shows that the best performance is obtained when the classes are equally distributed. Thus, performing mixup operations with highly-imbalanced data remains a challenging question. Our preliminary analysis in this direction paves the way for further investigation.

4.6.6 Validation Datasets

One problem of using small validation datasets is the inconsistency of the results, which may not reflect the actual performance of the model [205]. The smaller the validation set, the

larger the variations in the output. Moreover, [205] also argued that a comparison between different SSL models is possible when the validation set is equal to the training one. In this paper, we consider all these recommendations in our implementation. Consequently, the reported results fairly reflect the actual performance of each model.

4.6.7 The Unsupervised Loss

Interestingly, $\beta = 0$ shows the third-highest validation results. β is a hyperparameter that controls the contribution of the unsupervised loss. Setting $\beta = 0$ implicitly means that our model is still augmented with new data points from our random mixup yet without the unlabeled single. Based on the selected λ in Eq.(5.8) and Eq.(8), the newly-generated data points are close to the labeled data. In other words, the new data are in the vicinity distribution of the labeled data, *i.e.* high-quality data is generated, justifying the boost in the performance. On the other hand, setting $\beta > 0$ means that we propagate the training signals from the unlabeled data. These signals might be noisy and introduce uncertainty to the model because of the low quality of the pseudo labels. Hence, a decrease in performance was observed. However, after applying the sharpening, an enhancement is noticed in the model performance because the sharpening operation helps to generate more accurate pseudo labels, as shown in Fig.(**??**).d.

4.6.8 Hyper-parameters Tuning

ROAM involves a set of hyperparameters and design choices besides the standard ones. Although fine-tuning such an amount of parameters is a tedious task, our results show that ROAM outperforms all SSL methods in a wide range of hyperparameter choices. Thus, with a little effort, one can achieve SOTA performance. Our argument can be supported by the following examples. First, ROAM outperforms all SSL models regardless of the selected layer κ. Also, the lowest scores obtained by ROAM, when κ ={3 or 4}, are better than all other SSL methods, with one exception of [66]. Third, all ROAM variations, *i.e.* the sharpening and concatenation steps, outperform all other SSL models. Fourth, we show that the newly-generated data boosts the performance without the need for the unlabeled loss when $\beta = 0$. That is, the number of hyperparameters can be reduced significantly by fixing $\kappa = \{0, 1, L\}$ and just fine-tuning α and β, which is the standard procedure in many SSL methods. Consequently, our method does not require any extra effort or exhausting design choices. Based on that, our approach is easy to implement and can be generalized to different datasets, which have been shown in the brain and lung segmentation. Further, our code is publicly available for benchmarking and reproducibility.

5 Knowledge Sharing via Static & Dynamic Peer Learning in Semi-Supervised Federated Learning

> *Share your knowledge. It's a way to achieve immortality.*
>
> **— Dalai Lama (1357–1419)**
> (The high lama of Tibetan Buddhism)

> *A man can only attain knowledge with the help of those who possess it. This must be understood from the very beginning. One must learn from him who knows.*
>
> **— George Ivanovich Gurdjieff (1866–1949)**
> (Russian psychologist and scientist)

5.1 Motivation

In the recent estimation, an expectation of two hundred thousand fatal invasive and in-situ melanoma cases will be diagnosed in the USA in 2021 [249]. Yearly, millions of people are diagnosed with skin carcinoma [220]. Worldwide, skin cancer is considered one of the most expensive and fatal cancers. While most non-melanoma skin cancer cases can be cured, the melanoma ones are curable when detected in the early stages. For example, the 5-year survival rate ranges from 99% in the earliest stage to 27% for the latest stage [249]. Moreover, early detection of skin cancer can reduce the treatment expenses significantly [87]. Therefore, several attempts have investigated the automated classification of skin lesions in dermoscopic images [31, 61, 236]. Though, these attempts require handcrafted engineered features and exhausted pre-processing steps.

Yet, huge improvements in computerized methods have been achieved in recent years. For instance, the deep-learning-based methods proved to have a superior [95, 168, 187, 308] or a human-level performance [87, 269] when dealing with skin cancer classification. Nevertheless, this success comes at the cost of exhausting pre-processing steps, a prudently designed framework, or a substantial amount of labeled data assembled in one location. In real life, medical data is generated from different scanners and unevenly distributed in multiple centers

in raw formats without annotations resulting in heterogeneous data, or so-called Non-IIDness. Unfortunately, building a large repository of annotated medical data is quite challenging due to privacy burdens [138, 216], and labeling cost which is time-consuming and requires domain expert knowledge.

Federated learning (FL) [193] has been recently proposed to learn machine learning models utilizing the ample amounts of labeled data distributed in mobile devices while maintaining clients' privacy, *i.e.* , without sharing the data. Note that, the key properties of the FL are data privacy, Non-IIDness, and communication efficiency (section 3.4.2). Thus, FL goes in line with the nature of the medical setting. Consequently, federated learning has been investigated by several works in the medical domain [4, 170, 325] paving the way to training machine learning models in privacy-preserved fashion in real-world applications [89, 225, 233]. Though, in the previous works, the training demand highly accurate labeled data, e.g., ground-truth confirmed through histopathology, which often is costly and not available. In a more realistic scenario, the clients may have access to a large amount of unlabeled data along with a few annotated ones. Nevertheless, they are willing to train a reliable model to use their data. Fortunately, the semi-supervised federated learning (SSFL) paradigm can address the above scenario, which is the focus of this part of our thesis.

5.2 Contribution

In the second part of our methodology, we developed a semi-supervised federated learning method tackling the task of classifying eight skin lesions distributed over ten clients' in four realistic scenarios where the clients may or may not have a few annotated data besides many unannotated ones (e.g., not confirmed through histopathology).

Our contributions are:

- We propose **FedPerl**, an SSFL framework, inspired by Peer Learning (PL) [267] to build cooperative learning between similar clients to help each other in the pseudo labeling process and hence, better performance.

- We propose ***peers anonymization*** (PA) for the first time in the SSFL. In this work, we employ the ensemble averaging from the committee machine (CM) [12, 133, 268] for our ***peers anonymization*** (PA) technique. PA improves privacy by hiding clients' identities. Moreover, PA reduces communication costs while preserving performance. To our knowledge, no prior work has proposed the PA technique in the SSFL for the medical images. Furthermore, we show that our peer anonymization is orthogonal and can be easily integrated into other methods without additional complexity.

- We propose a ***dynamic learning policy*** that controls the contribution of peer learning in the training process. While our dynamic policy excels in the static one, it simultaneously helps the individual clients to achieve better performance.

- We introduce and test our method in four challenging scenarios, not yet been investigated thoroughly in the medical images. Moreover, we test the ability of **FedPerl** to generalize

to unseen clients. Additionally, we conduct extensive analyses on the effect of committee size on the performance at the client and community levels. Furthermore, we introduce additional evaluation metrics to evaluate the calibration of these models and their clinical applicability.

- We validate our skin lesion classification method with a database of more than 71,000 images, showing superior performance over the baselines.

The content of this part is based on the following publications:

Bdair T, Navab N, Albarqouni S. "**FedPerl: Semi-supervised peer learning for skin lesion classification**". In International Conference on Medical Image Computing and Computer-Assisted Intervention 2021 Sep 27. Springer, Cham.

Bdair T, Navab N, Albarqouni S. "**Semi-Supervised Federated Peer Learning for Skin Lesion Classification**". Journal of Machine Learning for Biomedical Imaging (MELBA) 2022 March 5.

5.3 Related Works

A very recent work [297] has shown the applicability of semi-supervised learning in a federated setting (SSFL) for COVID-19 pathology segmentation. The previous work among the firsts introduced semi-supervised learning to federated learning. Nevertheless, they have straightforwardly applied a semi-supervised learning method, *e.g.* `FixMatch` [252] locally. At first, a local model is trained in a fully supervised fashion using the labeled data. Then, the trained model is used to produce predictions for unlabeled data, where the predictions with high confidence are used to generate pseudo labels. Next, the pseudo labels are attached to the labeled data before a new training process starts. At the server, on the other hand, `FedAvg` [193] was employed to organize the training between different clients, see Sec. 5.4.2. In the previous method, clients are only trained i) globally, where the knowledge is accumulated in global model parameters, and ii) locally, where the knowledge is distilled via the local data. While this is a straightforward approach, we argue that the knowledge gained for generating pseudo labels for the local models is limited. Instead, we hypothesize that gaining extra knowledge by learning from similar clients, *i.e.* , Peer Learning (PL), is highly significant, assuming that peer learning encourages the clients' self-confidence by sharing their knowledge in a way that does not expose their identities. In contrast to [297], our method employs peer learning such that the clients gain extra knowledge by helping each other in a privacy-preserved way to leverage the unlabeled data, while [297] is limited to the local knowledge by the client itself.

Another recent work, [182] proposed an `SSFL` approach; `FedIRM` for skin lesion classification. They suggested distilling the knowledge from labeled clients to unlabeled ones through building a disease relation matrix extracted from the labeled clients and providing it to the unlabeled ones to guide the pseudo-labeling process. To share the knowledge between the clients, the previous work has assumed some clients are labeled while others are not. However, in a more challenging situation, which has not been investigated thoroughly in the medical

images or by any previous works, the labeled data is located on the server side, while the clients only have access to unlabeled data. Such a scenario is not applicable in FedIRM [182], however, it has been addressed in this thesis.

FedMatch [130] has proposed cooperative learning between clients to guide the pseudo labeling process, where similar clients share the weights. The similarities between clients are measured using K-Dimensional Tree (KD Tree) on clients' prediction of an arbitrary input. Although, FedMatch [130] has shown that such extra knowledge is helpful, it was obtained at the cost of communication and privacy. In contrast to [130], our approach is communication efficient and avoids any privacy breaches by employing an ensemble before sharing the knowledge between the peers.

Our method is inspired by peer learning from education science literature, where *peer learning* is defined as acquiring skills by sharing knowledge between peers [267]. In this thesis, the link between peer learning and federated learning is direct, where the clients have considered peers learning from each other. Further, in this contribution, we employ the ensemble averaging from the committee machine [268] for our *peers anonymization* (PA) technique. PA improves privacy by hiding clients' identities. Moreover, PA reduces communication costs while preserving performance. To our knowledge, no prior work has proposed the PA technique in the SSFL for the medical images.

5.4 Methodology

5.4.1 Problem Definition

Given M clients $\mathcal{C}_m$ who have access to their own local dataset $\mathcal{D}_m \in \mathbb{R}^{H \times W \times N_m}$, where H and W are the height and the width of the input images, and N_m is the total number of images. $\mathcal{D}_m$ consists of labeled $\mathcal{S}_L = \{\mathcal{X}_L, \mathcal{Y}_L\}$ and unlabeled data $\mathcal{S}_U = \{\mathcal{X}_U\}$, where $\mathcal{X}_L = \{x_1, \ldots, x_L, x_{L+1}, \ldots, x_{L+U}\}$ are input images; $x_i \in \mathbb{R}^{H \times W}$, and $\mathcal{Y}_L = \{y_1, \ldots, y_L\}$; $y_i \in \mathbb{R}^C$ are the corresponding categorical labels for C classes. Given query image x_q, our objective is to train a global model $f(\cdot)$ to predict the corresponding label $\tilde{y}_q$ for x_q, where the labeled and unlabeled data are leveraged in the training in a privacy-preserved fashion.

Definition. We define a model $f(\cdot)$ to be trained in a privacy-preserved fashion, if the following conditions are met:

(i) Data can not be transferred across different clients participating in the training process adhering to the General Data Protection Regulation (GDPR) [1].

(ii) Local models can not be transferred across different clients participating in the training process to avoid privacy breaches [207], or model inversion [91].

[1] https://gdpr.eu/

5.4.2 Semi-Supervised Federated Learning (SSFL)

The aforementioned conditions can be met by picking off-the-shelf SoTA SSL models, *e.g.,* `FixMatch` [252], to train the clients locally leveraging the unlabeled data, while employing `FedAvg` [193] to coordinate between the clients in a federated fashion as [297],

$$\min_{\theta} \mathcal{L}(\mathcal{D};\theta) \quad \text{with} \quad \mathcal{L}(\mathcal{D};\theta) = \sum_{m=1}^{M} w_m \, \mathcal{L}_{SSL_m}(\mathcal{D}_m;\theta), \tag{5.1}$$

where $w_m = {}^{N_m}/_{\sum_{i=1}^{M} N_i}$ is the respective weight coefficient for each client, and θ is the model parameters.

The SSL objective function appeared in `FixMatch` [252], can be used to train the client locally utilizing both labeled and unlabeled data as

$$\begin{aligned} \mathcal{L}_{SSL_m}(\mathcal{D}_m;\theta) = \arg\min_{\theta} & \mathcal{L}_{CE}(\mathcal{Y}_L, f(\alpha(\mathcal{X}_L);\theta)) \\ & + \beta \mathcal{L}_{CE}(\tilde{\mathcal{Y}}_U, f(\mathcal{A}(\mathcal{X}_U);\theta)), \end{aligned} \tag{5.2}$$

where $\mathcal{L}_{CE}(\cdot,\cdot)$ is the cross-entropy loss, β is a hyper-parameter that controls the contribution of the unlabeled loss to the total loss, $\tilde{\mathcal{Y}}_U$ is the pseudo labels for the unlabeled data $\mathcal{X}_U$, and $\alpha(\cdot)$ and $\mathcal{A}(\cdot)$ are weak and strong augmentations respectively. For an unlabeled input x_i, the pseudo label $\tilde{y}_i \in \tilde{\mathcal{Y}}_U$, is produced by applying a confidence threshold τ on the client' prediction on a weak augmented version of x_i such that

$$\tilde{y}_i = \arg\max(\mathbb{I}(f_{\mathcal{C}_m}(\alpha(x_i);\theta^*) \geq \tau)), \tag{5.3}$$

where $f_{\mathcal{C}_m}(.)$ is the local model, θ^* are frozen model parameters, and $\mathbb{I}(\cdot)$ is the indicator function.

5.4.3 Preliminaries

Peer Learning

Peer learning is defined in educational psychology as acquiring skills and knowledge through active helping among the companions. It involves people from similar social groups helping each other [267]. Peer Learning includes cooperative learning, tutoring, coaching, mentoring, and others. While distinguishing between all types of PL is out of our scope, we describe the cooperative learning that we utilized in this thesis. In cooperative learning (CL), peers work together through an interactive strategy. Usually performed in small groups of learners, CL often needs previous training to guarantee equal participation, concurrent exchange, synergy, and added value [267]. The researchers have shown clear evidence that cooperative learning can yield considerable progress, and it has also been noted to be among the most useful learning approaches [167].

Committee Machine

In the computer science literature, a similar concept to peer learning has been introduced known as Committee Machines (CM) [12, 133, 268]. In a nutshell, CM is a well-known

and active research direction and is defined as an ensemble of estimators, consisting of neural networks or committee members, that cooperate to obtain better accuracy than the individual networks. The committee prediction is generated by ensemble averaging (EA) of the individual members' predictions. CM has shown to be effective in machine learning hardware [133]. The interest in committee machines in the machine learning community started a long time ago in the last century and is still an active topic in recent deep learning methods. For instance, ensembling methods such as bagging, boosting, and averaging algorithms are considered a kind of committee machines [268]. Thus far, the idea of a committee machine is to train a group of learners to have one combined prediction aggregated from the individual ones, hoping it achieves improved performance compared to a single learner. The combined prediction takes the form of the weighted sum of the predictions or voting mechanism of the T committee members for the regression and classification tasks, which are given by 5.4 and 5.5, respectively.

$$\widehat{p(x)} = \sum_{i=0}^{T} w_i f_i(x). \tag{5.4}$$

$$\widehat{p} = \arg\max_{j} \sum_{i=0}^{T} w_i f_{i,class=j}(x). \tag{5.5}$$

Where $f_i(x)$ is the prediction of the i-th committee member at input x, w_i is the weight for that member, and $f_{i,class=j}(x)$ is the output of the learner i for j-th class.

5.4.4 FedPerl: Semi-Supervised Federated Peer Learning for Skin Lesion Classification

While the straightforward `SSFL` is simple, we argue that the learned knowledge of the individual clients could be further improved by involving similar clients in training. Therefore, inspired by peer learning, our method utilizes similar peers to help the target client in the pseudo labeling by sharing their knowledge without exposing their identities through the peer anonymization method. Our proposed `FedPerl`, illustrated in Fig. 5.1, consists of three components, namely 1) building communities, 2) peer learning, and 3) peer anonymization. While peer learning can be static or dynamic, as shown in the following sections.

Building communities

In educational social science [267], "peers" are referred to as two or more persons who share similarities and consider themselves as companions. In this work, we adopt the same concept and describe a group of clients as "peers" if they are similar. Previous work has shown that clustering can be achieved using models updates [37], While other works measure similarities between deep neural networks by comparing the representations between layers [152]. We build upon this and argue that the model weights represent and summarize the learned knowledge for each client from its training data. Thus, to measure the similarities between the clients, we represent each client $\mathcal{C}_m$ by a feature vector $f_m = \{(\mu_0, \sigma_0), \ldots, (\mu_l, \sigma_l)\} \in \mathbb{R}^{2 \cdot l}$, where (μ_l, σ_l) is the first two statistical moments, *i.e.* the mean and the standard deviation, of

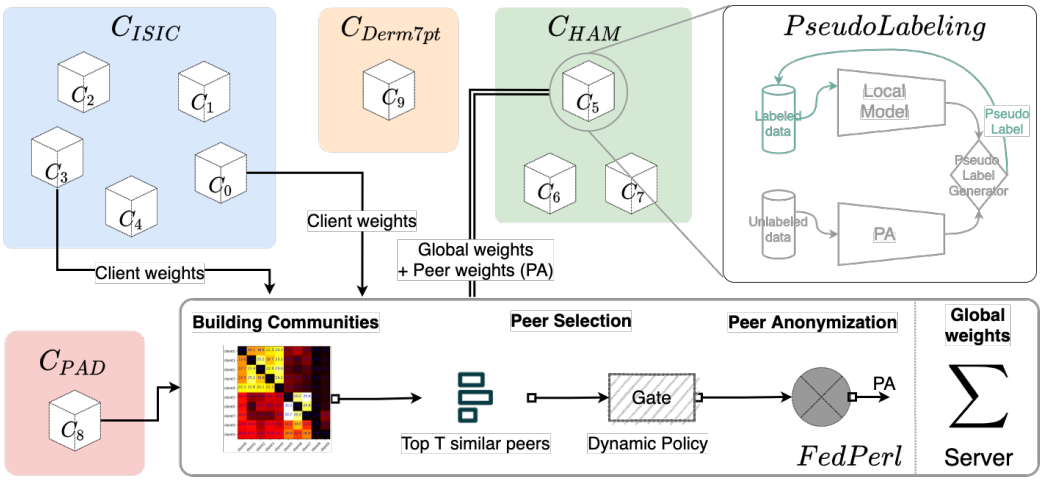


Fig. 5.1. Our semi-supervised federated learning framework (FedPerl). Our method consists of (i) Building communities: similar clients clustered into one community, (ii) Peer Learning: peers are helping in pseudo labeling, and (iii) Peer Anonymization (PA) to hide client identity, improve privacy, and reduce the communication cost. Top Right: Pseudo labeling utilizing an anonymized peer in this diagram. Bottom: Selecting the similar peers, peer learning, peers anonymization, and similarity matrix calculations are performed on the server. FedPerl exploits either static or dynamic learning policies.

the model's layer l parameters. Then, we compute the similarity ω_{mk} between clients $\mathcal{C}_m$ and $\mathcal{C}_k$ using the cosine similarity, where $\omega_{mk} = \frac{f_m^T f_k}{\|f_m\| \cdot \|f_k\|}$.

Using the cosine similarity brings the model parameters to the same behaviors without being exact, given that the means might differ, as long as they are in the same direction.

Finally, the similarity matrix between all clients is defined as

$$\mathcal{W}_{M\times M} = \begin{bmatrix} \omega_{11} & \dots & \omega_{1M} \\ \vdots & \ddots & \vdots \\ \omega_{M1} & \dots & \omega_{MM} \end{bmatrix}. \tag{5.6}$$

Our method starts with standard federated learning warm-up rounds (*e.g.* ten rounds in our case). In the next training rounds, the feature vectors are extracted after receiving the updates from the participating clients. Then, the similarity matrix is computed and updated accordingly. In `FedPerl`, The communities are formed implicitly based on the similarity matrix where similar clients are clustered into one community.

Knowledge Sharing via Peers Learning

The term "learning" is frequently defined as improved knowledge, experiences, and capabilities [267]. In peer learning, "peers" help each other by sharing their knowledge [267]. In this regard, we describe "peer learning" as the means of top T alike clients (peers) helping each other to generate pseudo labels by sharing their knowledge (model parameters). This is a helpful process since the main property of medical data is data heterogeneity. In federated learning, the clients experience different data and class distribution during the training. Thus,

accumulating and sharing distributed knowledge is useful. Particularly, it can help the local client generate pseudo labels for the unlabeled data from experiences that might never have learned from its own labeled data. To realize this, we modify the pseudo label defined in Eq.5.3 to include the predictions of the similar T peers, *i.e.* $f_t(\cdot;\theta)$ according to the similarity matrix $\mathcal{W}$ as

$$\tilde{y}_i = \arg\max \left(\mathbb{I} \left(f_{\mathcal{C}_m}(\alpha(x_i);\theta^*) + \sum_{t=0}^{T} f_t(\alpha(x_i);\theta_t^*) \geq \tau \right) \right). \tag{5.7}$$

Peers Anonymization

To improve privacy and adhere to the privacy regulations introduced in section 5.4.1, knowledge sharing among peers has to be anonymized and regulated. Thus, we propose *peers anonymization* (PA), at the server side, a simple yet effective technique. Mainly, we create an anonymized peer $f_a(\cdot;\theta_a)$ that assembles the learned knowledge from the top T similar peers where

$$f_a(\cdot;\theta_a) = \frac{1}{|T|} \sum_{t=0}^{T} f_t(\cdot;\theta_t). \tag{5.8}$$

Then, $f_a(\cdot)$ is shared with the local model to help pseudo labeling. Accordingly, Eq.5.7 is modified to

$$\tilde{y}_i = \arg\max \left(\mathbb{I} \left(f_{\mathcal{C}_m}(\alpha(x_i);\theta^*) + f_a(\alpha(x_i);\theta_a^*) \geq \tau \right) \right). \tag{5.9}$$

Notice that sharing the peers and the anonymized peer are not equivalent, *i.e.* ,

$$\frac{1}{|T|} \sum_{t=0}^{T} f_t(\alpha(x_i);\theta_t)) \neq f_a(\alpha(x_i);\theta_a). \tag{5.10}$$

Eventually, the anonymized peer is shared only once for each client at every training round, not at every local update. The advantages of the anonymized peer are; (i) it reduces the communication cost as sharing the knowledge of one peer is better than sharing two or more peers, (ii) it hides clients' identities by creating an anonymized peer. Finally, to prevent the local model from deviating from its local knowledge, we employ an MSE loss as a consistency-regularization term, which is broadly used in semi-supervised learning,

$$\mathcal{L}_{CON_m} = \| f_{\mathcal{C}_m}(x_i;\theta) - f_a(x_i;\theta_a^*) \|^2 . \tag{5.11}$$

Dynamic Learning Policy

Thus far, we have proposed a static learning policy in which the top T similar peers are used to help the local clients in the pseudo-labeling process. In peer learning, the clients are divided into groups or communities based on their similarities. A natural result of this step is also individual clients who do not belong to any community. Practically, we may have no control over the effect of applying the static peer learning policy on these clients, which could vary from one client to another where it is beneficial for some clients and not for others. For example, individual clients who do not belong to any community would be forced to learn from top T peers, based on the proposed similarity matrix, however, there is no guarantee that

they would be beneficial from the training since they may not belong to the same or similar community. Therefore, we suggest performing a dynamic policy where we could carefully involve the peers based on additional similarities or restrict the peers to a subset who are close enough. Our dynamic learning policy controls the learning stream to the clients where the peers are utilized in the learning process. In short, our goal is to maintain the gain and boost the performance of all clients. In this regard, we propose the following policies.

Validation Policy. In this policy, first, the client and its peers are validated on the global validation dataset. Then, only the peers with a validation accuracy equal to or higher than the client's accuracy are utilized. This policy can be applied with or without the peers' anonymization technique. Formally, assume that $V_{acc}(.)$ is a function that measures the accuracy on a global validation dataset, then the set of the peers that participate in peer learning for client $\mathcal{C}_m$ is defined as

$$\Omega = \{\mathcal{C}_n | V_{acc}(\mathcal{C}_n) \geq V_{acc}(\mathcal{C}_m)\}, \tag{5.12}$$

where $\mathcal{C}_n$ is a peer, $n = 1, 2, ..., T$, and T is the committee size.

Gated Validation Policy. As in the previous policy, the peers are validated on the global validation dataset. However, we apply a gateway on their accuracies, such that if it is equal to or higher than a pre-defined gateway threshold ρ, the peer will be involved in the pseudo labeling. Otherwise, it will be discarded from the process. In this policy, the set of peers that participate in peer learning for client $\mathcal{C}_m$ is defined as

$$\Omega = \{\mathcal{C}_n | V_{acc}(\mathcal{C}_n) \geq \rho\}. \tag{5.13}$$

Gated Similarity Policy. Like in the gated validation policy, this policy depends on a gateway that controls peers participation. Yet, no validation set is used, and a peer is allowed to participate if its similarity with the client is equal to or higher than the gateway threshold ρ. Assume that $H_{sim}(.)$ is a function that measures the similarity between two clients, then the set of the peers that participate in peer learning for client $\mathcal{C}_m$ is defined as

$$\Omega = \{\mathcal{C}_n | H_{sim}(\mathcal{C}_m, \mathcal{C}_n) \geq \rho)\}. \tag{5.14}$$

Note that regardless of the used policy, we first select the top T similar peers based on the similarity matrix. Then, one of the above policies is applied. The only difference between the last two policies is that in the gated validation policy, we used the validation accuracy as a gateway, while in the gated similarity policy, we stick to our similarity matrix. A pseudo-code summarizing our method is shown in Algorithm 2.

Overall Objective Function

The overall objective function for client m is the sum of semi-supervised and consistency-regularization losses, and given by

$$\mathcal{L}_m = \mathcal{L}_{SSL_m} + \gamma \mathcal{L}_{CON_m}, \tag{5.15}$$

Algorithm 2 Semi-Supervised Federated Peer Learning for Skin Lesion Classification

StartServer()
initialize global weights θ_G^0
for each round r=1, 2, ..., R **do:**
  $n \longleftarrow$ select random n clients from M // *i.e.* n=3
  for each client $\mathcal{C}_m$ in $1, 2, ..., n$ **do in parallel:**
    $f_{\mathcal{C}_m} \longleftarrow$ initialize client's weights with global weights
    if $r > 10$: //Peer learning starts after warm-up rounds
      $f_{\mathcal{C}_{1:T}} \longleftarrow$ GetTopSimilarPeers(T, $f_{\mathcal{C}_m}$) // Sec. 5.4.4
      if **IsDynamicPolicy** // Sec. 5.4.4
        $f_{\mathcal{C}_{1:T}} \longleftarrow$ ApplyPolicy($f_{\mathcal{C}_m}$, $f_{\mathcal{C}_{1:T}}$, ρ) //Could return from 0 up to T peers, here we assume all peers passed
      if **IsPeerAnonymization**
        $f_a \longleftarrow$ AnonymizePeers($f_{\mathcal{C}_{1:T}}$) // Eq.5.8
        $\theta_m \longleftarrow$ LocalTraining($f_{\mathcal{C}_m}$, f_a)
      else // No Peer Anonymization
        $\theta_m \longleftarrow$ LocalTraining($f_{\mathcal{C}_m}$, $f_{\mathcal{C}_{1:T}}$)
    else // No Peer learning, standard federated learning
      $\theta_m \longleftarrow$ LocalTraining($f_{\mathcal{C}_m}$)
  end for
  $\theta_G \longleftarrow \frac{1}{n}\sum_{j=1}^{n} \theta j$ // Update global weights *i.e.* FedAvg
  $\mathcal{F}_m \longleftarrow$ extract features vector for each client
  $\mathcal{W}_{M\times M} \longleftarrow$ update the similarity matrix // Eq.5.6
end for

where γ is a hyperparameter, and $\mathcal{L}_{SSL_m}$ and $\mathcal{L}_{CON_m}$ are Eq.5.2 and Eq.5.11, respectively. Note that the two terms in Eq.5.15 collaborate to achieve the balance between the local and global knowledge.

5.5 Experiments & Results

We test our method on skin dermoscopic images through a set of experiments. Before that, we show our method's proof of concept results and compare them with the current SOTA in SSFL for CIFAR10 and FMNIST in image classification tasks in section 5.5.6. FedPerl outperforms the baselines at different settings. Next, in section 5.5.7, we compare our method's skin image classification results with the baselines. The results show that peer learning enhances the performance of the models, yet applying PA enhances the communication cost in addition to the performance. After that, we show and discuss how FedPerl builds the communities in section 5.5.8. The results show that FedPerl clusters the clients into main communities and individual clients thanks to our similarity matrix. Besides, FedPerl boosts the overall performance of communities while it has a different effect on individual clients. Thus, in section 5.5.9, we comment on the impact of peer learning on individual clients. FedPerl shows superiority and less sensitivity to a noisy client. Then, we dig more deeply and present the classification results for each class in section 5.5.10. Our method enhances the classification for the individual classes, *e.g.* up to 10 times for the DF class. Further, to confirm our findings and for more validation, we present the results using different evaluation metrics in section 5.5.11. Our method is more calibrated and shows superiority over the SSFL in the area under ROC and Precision-Recall curves, risk-coverage curve, and reliably diagrams. The qualitative results are presented in the same section. In section 5.5.13, we propose a more

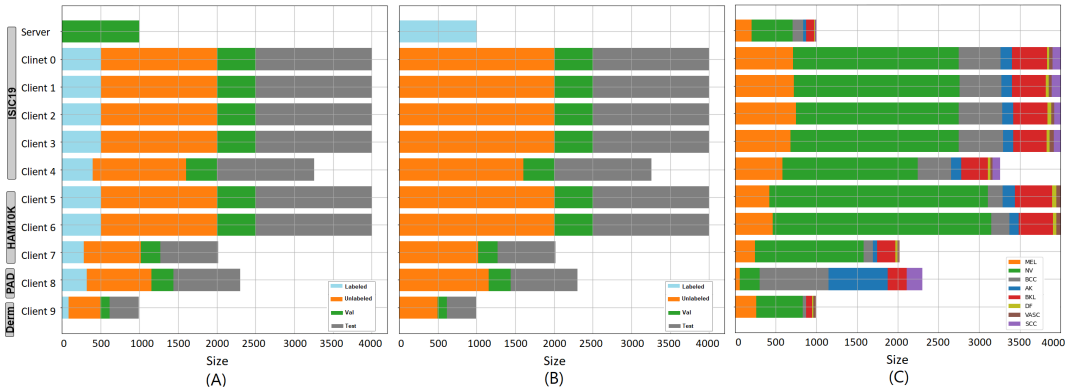


Fig. 5.2. Illustrative diagram shows the distribution of our clients. The datasets are divided randomly into ten clients besides the global model, without overlap between the clients. (A) The standard semi-supervised learning scenario. Each client data is divided into testing (gray), validation (green), labeled (blue), and unlabeled (orange) data. (B) The unlabeled/disjoint clients scenario. The labeled and unlabeled images are combined and used as unlabeled images. (C) The class distribution. The data split is designed to simulate a realistic scenario with severe class imbalance, varying data sizes, and diverse communities. The clients 5-9 missing one or more classes.

challenging scenario in which the clients do not have any labeled data. The classification results show that `FedPerl` still achieves the best performance with or without PA. We end this part of the experiments by showing the ability of `FedPerl` to generalize to unseen clients in section 5.5.14. In section 5.5.15, we conduct a comparison with `FedIRM`, a SOTA `SSFL` method in skin lesion classification, under a fourth scenario where we have few labeled clients. Both models achieve comparable results when participation rate (PR)$= 30\%$, while our method shows a lower performance when PR$= 100\%$. Note that the previous results were obtained when utilizing a static learning policy. However, in the last part of our experiments, we show the results of our dynamic peer learning policy in section 5.5.16. In general, the new policy outperforms the results from the earlier one, while at the same time, it is successfully boosting the performance of the individual clients.

5.5.1 Datasets

Our database consists of 71,000 images collected from 5 publicly available datasets as the following. (1) ISIC19 [63] which consists of 25K images with 8 classes. The classes are melanoma (MEL), melanocytic nevus (NV), basal cell carcinoma (BCC), actinic keratosis (AK), benign keratosis (BKL), dermatofibroma (DF), the vascular lesion (VASC), and squamous cell carcinoma (SCC). (2) HAM10000 dataset [270] which consists of 10K images and includes 7 classes. (3) Derm7pt [143] which consists of 1K images with 6 classes. (4) PAD-UFES [208] which consists of 2K images and includes 6 classes. The previous datasets are divided randomly into ten clients besides the global model, without overlap between datasets, *cf.* Fig.5.2. (5) ISIC20 dataset [224] which consists of more than 33K images with malignant ($\sim$ 500 images) and benign ($\sim$ 32.5K images) classes. The last dataset is used as testing data to study how `FedPerl` generalizes to unseen data. Note that testing our method on the ISIC20 is a very challenging task due to the huge class imbalance and class distribution mismatch.

5.5.2 Baselines

We conduct our experiments on the following baselines; (i) Local models: which include lower, upper, and SSL (`FixMatch` [252]) models; these models are trained on their local data without utilizing the federated learning. (ii) Federated learning models include lower, upper, `SSFLs` similar to [297], and `FedMatch` [130] models, where these models are trained locally on their data and utilizing the federated learning globally. (iii) Ablation for our method, namely `FedPerl` with(out) the PA. Note that for ease of implementation, we compare our method with one variant of `FedMatch` that do not implement weights decomposition.

5.5.3 Scenarios

Our experiments conducted under four scenarios. In the first scenario, the standard semi-supervised learning, *cf.* Fig.5.2.(A), each client data is divided into testing (gray), validation (green), labeled (blue), and unlabeled (orange) data. The data split is intended to resemble a realistic scenario with varying data size, severe class imbalance, and diverse communities, *e.g.*, the clients 0-4 originated from ISIC19, the clients 5-7 originated from HAM10000, and clients 8 and 9 originated from Derm7pt, and PAD-UFES, respectively. We train the lower bounds on the labeled data while we train `FixMatch`, `SSFLs`, and `FedPerl` on both labeled and unlabeled data. The upper bounds trained akin to SSLs, yet, all labels were exposed. We use the global data to train the global model in the second scenario, the unlabeled clients' scenario. On the client's side, however, the labeled and unlabeled images are combined and used as an unlabeled dataset, *i.e.* , the labels were excluded from the training, *cf.* Fig.5.2.(B). While the second scenario is not yet investigated thoroughly in the medical images, we address it in this paper. In the third scenario, we test the ability of our model and the baselines to generalize to an unseen client (ISIC20) with new classes that have never been seen in the training. The fourth scenario proposed by [182] in which there are few labeled clients. For this scenario, clients 1 & 9 are selected as labeled clients while the remaining are not, such that they represent the largest community and individual clients, respectively.

5.5.4 Implementation Details

We opt for EffecientNet [261] pre-trained on ImageNet[229] as a backbone architecture and trained using Adam optimizer [147] for 500 rounds. We follow FedVC [119] approach for clients' federated learning. The idea of FedVC is to conceptually split large clients into multiple smaller ones and repeat small clients multiple times such that all virtual clients are of similar sizes. Practically, this is achieved by fixing the number of training examples used for the federated learning round to be fixed for every client, resulting in exact optimization steps. The batch size B and participation rate (PR) were set to 16 & 30% (3 clients each round), respectively. The local training is performed for one epoch. The learning rate was investigated in $[0.00001, 0.0001]$ and found best at 0.00005. τ investigated in $[0.5, 0.95]$ and found best at 0.6 & 0.9 for the federated and local models, respectively. β investigated in $[0.1, 5]$, and found best at 0.5. γ investigated in $[0.01, 0.1]$, and found best at 0.01. T investigated in $\{2, 3, 4, 5\}$, and found best at $T = 2$. The dynamic learning policy threshold ρ tested at three values: 0.75, 0.85, and 0.95. All images were resized to 224×224, and normalized to intensity

values of $[0, 1]$. Random flipping and rotation were considered weak augmentations, whereas RandAugment [65] was used as strong augmentation. We opt for the PyTorch framework for the implementation hosted on a standalone NVIDIA Titan Xp 12 GB machine. As the followed procedures in semi-supervised learning, `FedPerl` starts with warm-up rounds, e.g., ten rounds in our case. The testing results are reported for the models with the best validation accuracy. The average training time takes around 7 hours for each run for `FedPerl` models (w/o PA), about 5.85 hours for `FedPerl` (with PA), about 5.5 hours for `SSFL`, and about 6.25 hours for `FedMatch` shedding light on the cost-effectiveness of our approach. All the hyperparameters tuning was performed on a validation detest.

5.5.5 Evaluation Metrics

We report the statistical summary of precision, recall, and F1-score. A Relative Improvement (RI) *w.r.t* the baseline is also reported, where RI of a over b is : $(a - b)/b$. To highlight more in the model's performance at various threshold settings, we plot Area Under Receiver Operating Characteristic (AUROC) and Area Under Precision-Recall (AUPR) curves. Note that we follow the *One vs. ALL* methodology for plotting. AUROC shows the model's ability to discriminate between positive and negative examples, assuming balance data. Nevertheless, AUPRC is a useful performance metric for imbalanced data, such as in our case, where we care about finding positive examples. Further, we investigate the uncertainty evaluation and model confidence. Thus, we report Risk-Coverage (RC) curve [94], Reliability Diagram (RD) [106], and Expected and Maximum Calibration errors [78], denoted as ECE and MCE respectively. RC curve plots the risk as a function of the coverage. The coverage denotes the percentage of the input processed by the model without rejection, while the risk denotes the level of risk of the model's prediction [94]. To calculate the reliability diagrams and calibration errors, we adopted an adaptive binning strategy [78] that depends on fixable intervals in the calculations. This strategy is more accurate than using fixed intervals [78]. Practically, we can realize the intervals used from the figure itself. For example, the width of the bars in figures 5.6 and 5.8 represents the ranges used to calculate ECE and MCE.

5.5.6 Proof-Of-Concept Results

First, we show proof of concept of our method on CIFAR-10 and FMNIST datasets and compare it with `FedMatch` [130]; a very recent work of `SSFL`. We follow the codebases and the experimental setup they used to have a fair comparison. The results, reported in Table 5.1, show that `FedPerl` Outperforms `FedMatch` [130] in all experiments setup indicating the effectiveness of our method on finding the similarity (Sec 5.4.4), without introducing extra complexity, e.g., weight decomposition [130]. Additionally, our *peers anonymization* (PA) improves the accuracy and privacy at a low communication cost. Note that PA employs one anonymized peer while `FedMatch` uses two clients in training. Interestingly, the FMNIST dataset results show that our method outperforms `FedProx-SL`, which is inconsistent with the CIFAR10 dataset results. Although, these results are not comparable because of `FedProx-SL` results were taken from the original paper, whereas `FedPerl` results were generated by our environment. However, this could be attributed to the fact that FMNIST images are much

simpler than the ones in CIFAR10, yielding more robust similarities and producing more accurate pseudo labels.

Tab. 5.1. The classification accuracy for the Proof-Of-Concept experiment on CIFAR10 & FMNIST datasets. *: Results as in FedMatch [130]. SL: Fully supervised. The SSL methods use 10% of the labeled data.

PA	Method (SSFL)	CIFAR10 IID	CIFAR10 NonIID	FMNIST NonIID
*	FedAvg-SL	58.60±0.42	55.15±0.21	-
*	FedProx-SL	59.30±0.31	57.75±0.15	82.06±0.26
*	FedAvg-UDA	46.35±0.29	44.35±0.39	-
*	FedProx-UDA	47.45±0.21	46.31±0.63	73.71±0.17
*	FedAvg-FixMatch	47.01±0.43	46.20±0.52	-
*	FedProx-FixMatch	47.20±0.12	45.55±0.63	62.40±0.43
*	FedMatch	52.13±0.34	52.25±0.81	77.95±0.14
w/o PA	FedMatch (Our run)	53.12±0.65	53.10±0.99	76.48±0.18
w/ PA	FedMatch (Our run)	**53.32±0.59**	**53.80±0.39**	**76.72±0.44**
w/o PA	FedPerl	**53.37±0.11**	**53.75±0.40**	**76.52±0.08**
w/ PA	FedPerl	**53.98±0.06**	**53.50±0.71**	**82.75±0.44**

5.5.7 Skin Lesion Results

Federated Learning Results

In this section, we present the federated learning classification results before applying our method *i.e.* , without peer learning or PA. Table 5.2 proves the current findings that `FedAvg` outperforms the local models significantly. For example, see Local/`FixMatch` vs. `FedAvg`, the obtained F1-score are 0.647 and 0.698, 0.664 and 0.734, and 0.726 and 0.773, respectively, with relative improvement (RI) up to 19.74%. Interestingly, both lower `FedAvg` and `FedAvg` ‡ (`SSFL`) models exceed the local SSL and upper bound models. That implies aggregating knowledge across different clients is more beneficial than individually exploiting local unlabeled or labeled data. Next, we discuss `FedPerl` results at different values of T.

FedPerl Results without PA

The results of `FedPerl` without applying peer anonymization is shown in Table 5.2 (denoted as w/o PA). The first concluding remarks reveal that peer learning enhances the local models. For illustration, our method outperforms the lower model with RI between 14.53% and 15.46%. Further, `FedPerl` exceeds (`SSFL`) `FedAvg`† [297] and `FedMatch` [130] by 1.8% and 1.08%, respectively. Moreover, our approach is better than the local upper bound by 2.9%. Note that `SSFL` is considered a special case of `FedPerl` when $T = 0$. In addition, `FedPerl` results at a different number of peers T (committee size) are comparable, while the communication cost, compared to the standard `SSFL`, increases proportionally with the increasing value of T (see AC in Table 5.2). Note that the additional cost is calculated with respect to the baseline

Tab. 5.2. The results under the standard semi-supervised learning scenario. Mean (Median) ± Std. of different evaluation metrics. †:∼[297]. ‡:∼ FedMatch[130]. RI: Relative Improvement. AC: Additional Cost. The AC is calculated *w.r.t* the baseline (SSFL). For simplicity, we assume the initial cost for the SSFL is 0%. +: with PA.

Setting	Model	F1-score	Precision	Recall	RI(%)	AC(%)
Lower	Local	0.647(0.632)±0.053	0.644(0.622)±0.053	0.666(0.650)±0.053	-	
	FedAvg	0.698(0.690)±0.084	0.711(0.702)±0.072	0.709(0.700)±0.077	7.88	
SSL	FixMatch	0.664(0.636)±0.060	0.666(0.645)±0.063	0.692(0.671)±0.052	2.63	
SSFL	FedAvg†	0.734(0.725)±0.065	0.744(0.730)±0.064	0.739(0.728)±0.061	13.44	0
	FedMatch‡	0.739(0.729)±0.076	0.751(0.745)±0.068	0.744(0.732)±0.071	14.22	200
w/o PA	FedPerl(T=1)	**0.746(0.741)±0.071**	**0.753(0.744)±0.069**	**0.748(0.744)±0.069**	**15.30**	100
w/o PA	FedPerl(T=2)	**0.747(0.736)±0.071**	**0.756(0.741)±0.067**	**0.750(0.739)±0.069**	**15.46**	200
w/o PA	FedPerl(T=3)	**0.746(0.741)±0.072**	**0.757(0.743)±0.066**	**0.747(0.743)±0.070**	**15.30**	300
w/o PA	FedPerl(T=4)	**0.741(0.731)±0.077**	**0.751(0.735)±0.069**	**0.745(0.736)±0.072**	**14.53**	400
w/o PA	FedPerl(T=5)	**0.744(0.734)±0.073**	**0.753(0.744)±0.071**	**0.747(0.739)±0.069**	**15.00**	500
	FedMatch+‡	**0.745(0.737)±0.071**	**0.750(0.737)±0.067**	**0.750(0.746)±0.069**	**15.15**	100
	FedPerl(T=2)	**0.746(0.737)±0.075**	**0.754(0.741)±0.071**	**0.749(0.742)±0.073**	**15.30**	100
	FedPerl(T=3)	**0.746(0.738)±0.066**	**0.756(0.743)±0.060**	**0.748(0.740)±0.065**	**15.30**	100
	FedPerl(T=4)	**0.746(0.736)±0.077**	**0.755(0.745)±0.072**	**0.750(0.740)±0.074**	**15.30**	100
	FedPerl(T=5)	**0.749(0.739)±0.068**	**0.758(0.744)±0.065**	**0.750(0.742)±0.066**	**15.77**	100
Upper	Local	0.726(0.701)±0.044	0.729(0.705)±0.045	0.732(0.710)±0.042	12.21	
	FedAvg	0.773(0.757)±0.068	0.779(0.765)±0.065	0.773(0.759)±0.069	19.47	

(SSFL). For simplicity, we assume the initial cost for the SSFL is 0%. Finally, the results imply that employing one similar peer ($T = 1$) is adequate to obtain remarkable enhancement with minimal communication cost, yet, at the loss of privacy. To address this, we propose **peers anonymization** technique.

FedPerl Results

After applying the peer anonymization, all models show a similar or slightly better performance when compared to the previous results (*i.e.* , w/o PA), *cf.* Table 5.2. Nevertheless, the new models enhance the baseline's performance while still being better at hiding clients' identities and reducing the communication cost $O(1)$ regardless of the committee size T. Interestingly, applying peer anonymization not only enhances FedPerl, but also the FedMatch method. Specifically, the F1-score increases from 0.739 to 0.745, see FedMatch vs. FedMatch+ in Table 5.2. Note that the additional advantages of FedMatch+ over FedMatch is the anonymized peer and the communication cost. The performance improvement is attributed to the carefully designed strategy of creating the anonymized peer, such that the learned knowledge from many models is ensembled into a single model. The results confirm the superiority of FedPerl, and show that our peer anonymization is orthogonal and can be easily integrated into other methods without additional complexity.

In Fig. 5.3, we show the performance accuracy during the training. While we notice that similar clients have achieved similar training behavior, no further improvement in the last stages of the training was observed for all clients. For example, the accuracy for the clients is 0-4 between 75-65, while it is between 80-90 for the clients 5-7. Client 9 achieved accuracy

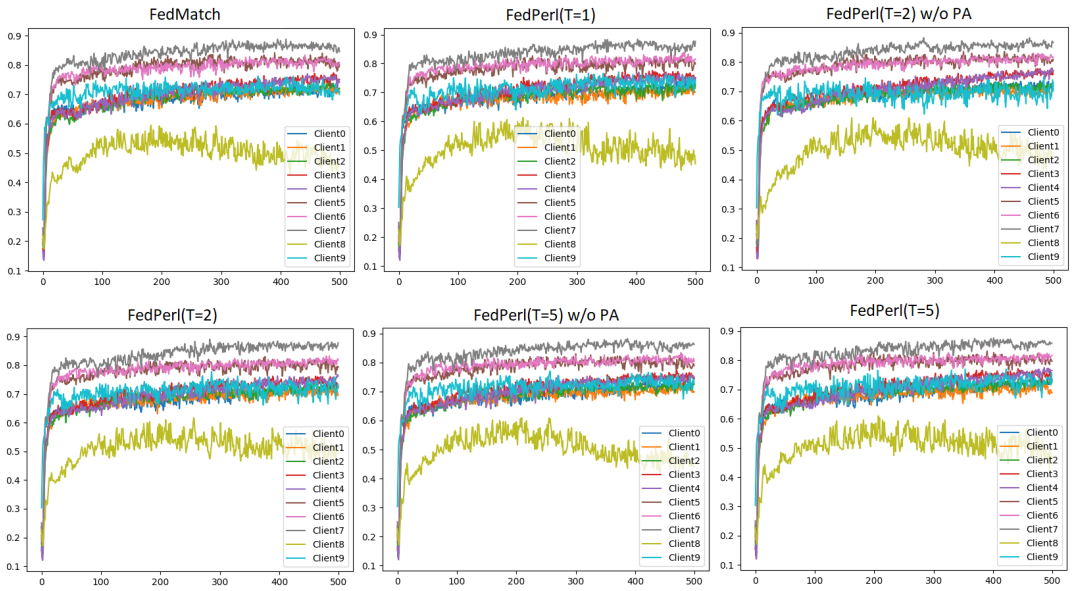


Fig. 5.3. The accuracy performance during the training. Due to the large number of curves that can be presented, we opt for FedMatch and FedPerl at different community sizes. Similar clients have achieved similar training performance.

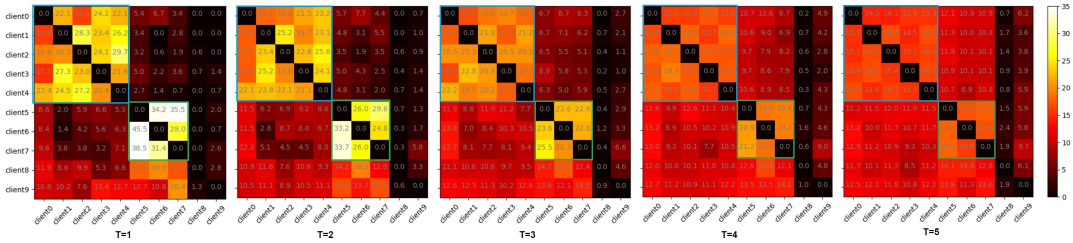


Fig. 5.4. FedPerl clusters clients into two main communities (blue & green rectangles), while clients 8 & 9 do not belong to any community. As we increase the committee size T, the frequency of selecting peers within the same community decreases. The numbers and colors correspond to the frequency, where the brighter colors or higher numbers values represent higher frequencies.

that is similar to clients 0-4. However, the best accuracy for client 8 was achieved in the middle of the training. This suggests that handling Out-of-Distribution clients in federated learning has to be further investigated.

5.5.8 Building Communities Results

This experiment investigates the importance of the similarity matrix used to rank similar clients and cluster them into communities. In Fig.5.4, we present the percentage of selecting peers during the training at different T values. Let us consider the community size (T=2) to gain more insights. For instance, the percentage of 33.2% between clients 6 and 5 reflects how often client 5 has chosen as a similar peer for client 6. The blue & green rectangles show that the clients clustered into two main communities. Interestingly, the clustering matches the clients' distribution we designed in our experiment, *cf.* Fig.5.2.

For further analysis on community 1 (blue rectangle), we find the frequency of selecting peers from the same community for each client by calculating the horizontal summations (columns 0-4). The frequencies are $81.6\%, 85.6\%, 89.5\%, 86.4\%$, and 88.1% for the clients 0-4, respectively. That suggests that client 0 learns from its community with a percentage of 81.6% of the training time. On average, 86.24% of the time, the first community members learn from each other, while it is 57.77% for community 2 (green rectangle). The same clustering also is shown for `FedPerl` at different committee sizes; $T = \{1, 3, 4, 5\}$. Note that the frequency values gradually decrease when a larger committee size is used for communities 1 & 2. The decrease in frequencies is expected because the likelihood of selecting peers from outside the community increases as we use a bigger committee size. Hence, the frequencies are distributed among the clients. In contrast, the frequencies for selecting peers for the individual clients (8 & 9) are comparable at different T values.

Tab. 5.3. The mean F1-score is reported to show the influence of peer learning on the community and individual clients. M: number of clients. *: SSFL.

Model	C_{ISIC} (M=5)	C_{HAM} (M=3)	8 (M=1)	9 (M=1)
FedPerl(T=0)*	0.718	0.816	0.602	0.703
FedPerl(T=1)	0.738	0.829	0.584	0.699
FedPerl(T=2)	0.736	0.833	0.567	0.717
FedPerl(T=3)	0.735	0.828	0.594	0.725
FedPerl(T=4)	0.735	0.826	0.562	0.727
FedPerl(T=5)	0.737	0.824	0.588	0.731

For further analysis of the community results, we average the classification results in each community and report them in Table 5.3. The first note from the results indicates that peer learning boosts the overall performance of the communities, compare the values of $T = 0$ vs. $T = \{1, 2, 3, 4, 5\}$ for C_{ISIC} and C_{HAM} respectively. Note that peer learning is not applied when $T = 0$. Further, we notice a stable performance for the C_{ISIC} community after applying the peer learning regardless of T values, yet with slight changes. However, an increasing then a decreasing performance is observed for the C_{HAM} at increasing values of T. This performance inconsistency is attributed to the community size. For instance, C_{ISIC} community includes 5 clients, while C_{HAM} community contains 3 clients. At first, let us consider C_{ISIC}. Based on their similarities, the probability of selecting peers from the outside community for different values of $T \geq 1$ is very low, and most likely the peers coming from the same community *i.e.* , internal peers. For the case when $T = 5$, selecting an external peer is guaranteed. However, its effect is negligible compared to the other clients, who most likely are internal peers. Now let us consider C_{HAM}. We notice that the performance increases gradually and reaches its best at $T = 2$. Based on our similarity matrix, the peers until this value most likely are internal peers, yielding enhancement in the performance. Nevertheless, involving external peers is confirmed after that (*i.e.* , $T > 2$). Consequently, the local model is distracted by increasing the number of external peers when using larger T values. Hence, the decrease in performance.

Tab. 5.4. The classification results for each client (mean F1-score). C_{ISIC}, C_{HAM}: the results at the community level. $Avg_{/C8}$: the results after excluding client 8. †:∼[297]. ‡:∼FedMatch[130]. Diff.% = $Avg_{/C8}$ - Avg. +: with PA.

Setting	Model/Client	0	1	2	3	4	C_{ISIC}	5	6	7	C_{HAM}	8	9	$Avg_{/C8}$	Avg	Diff.%
Lower	Local	0.581	0.618	0.603	0.622	0.596	0.604	0.742	0.738	0.670	0.717	0.656	0.641	0.646±0.056	0.647±0.053	-
	FedAvg	0.678	0.687	0.667	0.703	0.692	0.685	0.794	0.796	0.787	0.792	0.492	0.684	0.731±0.047	0.698±0.084	3.3
SSL	FixMatch	0.634	0.635	0.608	0.637	0.626	0.628	0.752	0.783	0.716	0.751	**0.650**	0.602	0.666±0.063	0.664±0.060	-
SSFL	FedAvg †	0.727	0.718	0.686	0.723	0.735	0.718	0.812	0.831	0.806	0.816	0.602	0.703	0.764±0.045	0.734±0.065	3.0
	FedMatch‡	0.724	0.724	0.722	0.734	0.751	0.731	0.801	0.850	0.813	0.822	0.553	0.717	0.760±0.046	0.739±0.076	2.1
	FedMatch+‡	**0.733**	**0.740**	**0.729**	**0.734**	0.744	**0.736**	**0.813**	0.843	**0.826**	**0.827**	0.581	0.703	**0.768±0.053**	**0.745±0.071**	2.3
w/o PA	FedPerl(T=2)	**0.735**	**0.731**	**0.725**	**0.737**	0.739	**0.733**	0.805	**0.850**	**0.839**	**0.831**	0.582	**0.729**	**0.769±0.047**	**0.747±0.071**	2.2
	FedPerl(T=2)	**0.737**	**0.737**	**0.724**	0.730	**0.751**	**0.736**	**0.818**	0.846	**0.834**	**0.833**	0.567	**0.717**	**0.765±0.046**	**0.746±0.075**	1.9
Upper	Local	0.698	0.698	0.677	0.700	0.696	0.694	0.806	0.804	0.752	0.787	0.702	0.722	0.728±0.046	0.726±0.044	-
	FedAvg	0.736	0.747	0.735	0.753	0.761	0.746	0.859	0.855	0.861	0.858	0.630	0.789	0.797±0.054	0.773±0.068	2.4

On the other hand, the individual clients' results are interesting (*i.e.* , clients 8 & 9). While an enhancement is noticed for client 9, a reduction is observed for client 8. We note that the accuracy of client 9 is increased as the committee size increases, thanks to peer learning. In general, the large the committee size, the better the performance. However, peer learning harms client 8. One explanation is attributed to the class distribution mismatch between client 8 and the others, *cf.* Fig.5.2.(C). Further analysis is discussed in the next section concerning the individual clients' performance.

Random Peers

To investigate the importance of peer learning and our similarity matrix, we perform an additional experiment where the peers for the clients are selected randomly. The obtained F1-score is 0.736, with RI equals 13.75% and 0.27% *w.r.t.* the lower bound and `SSFL`, respectively. These results imply two conclusions. (i) Even with random clients, peer learning still benefits training; compare this experiment results with the `SSFL`. (ii) Utilizing our similarity matrix brings extra knowledge by picking more accurate peers; compare this experiment results with the `FedMatch` models.

5.5.9 The Influence of Peer Learning on Clients

This experiment aims to gain more insights into the individual results, realize the influence of peer learning on clients, and compare it with the baselines. The results are shown in Table 5.4. First, we observe that `FedPerl` exceeds the baselines, including the local upper bounds with salient margins, *e.g.*, for client 7, it is about 16.4% (Lower Local vs. `FedPerl`). In the same direction, `FedPerl` steadily surpasses `FedMatch` at the community's level and in all individual clients' results except for client 4. Yet, thanks to our PA, `FedMatch+` shows better results than `FedMatch` at all communities and clients except for clients 4, 6, and 9. Surprisingly, `FedPerl` excels the upper `FedAvg` for client 0. The performance improvement is observed for all clients except client 8. One explanation is that `FedPerl` does not find suitable peers for client 8 to learn from due to the class distribution mismatch (*cf.* Fig. 5.2.(C)).

For further investigation on the impact of client 8, we explore excluding it from the training. Then we compare the new and the previous results, reported as $Avg_{/C8}$ and Avg respectively in Table 5.4. The comparison unveils that all federated learning models (*i.e.* , `FedAvg`, `FedMatch`, and `FedPerl`) obtain better performance after excluding client 8. Still, the best performance

Tab. 5.5. The classification results for the eight classes (mean F1-score). +: with PA. †:∼[297]. ‡:∼FedMatch[130]

Setting	Model	MEL	NV	BCC	AK	BKL	DF	VASC	SCC
Lower	Local	0.430	0.811	0.502	0.293	0.357	0.099	0.318	0.124
	FedAvg	0.501	0.834	0.646	0.377	0.507	0.173	0.642	0.111
SSL	FixMatch	0.451	0.831	0.540	0.304	0.374	0.052	0.292	0.135
SSFL	FedAvg †	0.565	0.852	0.680	**0.396**	0.570	0.416	0.707	0.253
	FedMatch†	0.573	0.852	0.700	0.366	0.565	0.462	0.701	0.275
	FedMatch+	**0.579**	**0.853**	**0.701**	0.376	**0.574**	**0.506**	**0.708**	**0.302**
w/o PA	FedPerl(T=2)	**0.576**	**0.854**	**0.706**	0.393	**0.589**	**0.552**	0.702	**0.305**
	FedPerl(T=2)	**0.602**	**0.854**	0.687	0.390	**0.592**	**0.493**	**0.712**	**0.315**
Upper	Local	0.551	0.853	0.651	0.428	0.520	0.308	0.654	0.308
	FedAvg	0.617	0.867	0.750	0.510	0.637	0.672	0.804	0.282

is observed for FedPerl over the local upper and the (SSFL) FedAvg models. Note that the performance reduction after including client 8 in training (see Avg in Table 5.4) implies a negative impact of this client. To realize that, we calculate the difference in performance before and after including client 8, *i.e.* $Avg_{/C8} - Avg$, and report the results in column $Diff.$ in Table 5.4. The resulting values show the negative impact of client 8 on the results. Where the higher the difference is, the higher the negative impact is. For example, it negatively impacted the smallest on FedPerl (1.9%), moderate on FedPerl w/o PA (2.2%) and on both FedMatch methods (2.1% and 2.3%), and the largest on FedAvg (3%). Such negative behavior could represent a threat in federated learning, where a noisy and out-of-distribution client might hurt other clients and mislead the global model. Nevertheless, the most interesting observation from this experiment is that FedPerl is less prone to the negative and noisy impact than SSFLs thanks to the training schema we proposed. We do not claim that FedPerl is robust against class distribution mismatch but is less sensitive to a noisy client. Nevertheless, the inconsistency in behavior between clients 8 & 9 could be further investigated.

On the other side, we notice that the enhancement after applying peer learning was also observed at the community level; C_{ISIC} and C_{HAM} with 13.2% and 11.6%, respectively, confirming the finding in the previous section.

Note that our final objective consists of two terms that try to balance the local and global benefits. Experimentally, we have shown that client 8 harms the clients. However, this impact was the minimum on FedPerl who is utilizing peer learning. Thus, we argue that involving peers, who influence the local models through participating in the pseudo labeling, has two advantages; (i) it restricts client 8 to sending more reliable updates, and (ii) it reduces the negative influence of that client. Also, the T peers learn and coach the local client and guide it to be more accurate, where a noisy client could be fixed by averaging with more reliable clients.

Tab. 5.6. The area under ROC curve for the eight classes. +: with PA. †:∼[297]. ‡:∼FedMatch[130]

Setting	Model	MEL	NV	BCC	AK	BKL	DF	VASC	SCC
Lower	Local	0.662	0.777	0.760	0.677	0.644	0.529	0.676	0.540
	FedAvg	0.709	0.834	0.827	0.634	0.739	0.575	0.909	0.528
SSL	FixMatch	0.670	0.804	0.785	0.685	0.658	0.515	0.644	0.540
SSFL	FedAvg†	0.737	0.846	0.827	**0.692**	0.777	0.675	0.889	0.583
	FedMatch‡	0.749	0.851	0.843	0.650	0.772	0.690	0.897	0.586
	FedMatch+	**0.751**	**0.854**	**0.847**	0.655	**0.778**	**0.717**	**0.904**	**0.596**
w/o PA	FedPerl(T=2)	**0.750**	**0.859**	**0.853**	0.659	**0.785**	**0.732**	**0.900**	**0.608**
	FedPerl(T=2)	**0.758**	**0.860**	0.838	0.660	**0.791**	**0.717**	**0.907**	**0.608**
Upper	Local	0.728	0.838	0.831	0.733	0.733	0.648	0.824	0.606
	FedAvg	0.773	0.869	0.848	0.750	0.805	0.876	0.958	0.594

5.5.10 Class Level Results

Because our setting is heterogeneous and suffers from severe class imbalance (*cf.* Fig.5.2.(C)), it is important to validate our method in that setting. Thus, we report the class level performance in Table 5.5. `FedPerl` obtains skin lesion classification accuracy better than local models (`FedPerl` vs. Local/`FixMatch`). For example, the improvement reaches ten times in the DF class. Moreover, `FedPerl` enhances the accuracy for BCC, BKL, DF, VASC, and SCC lesions by 16.6%, 21.8%, 50.4%, 42.0%, and 18.0%, respectively, in the SSL setting. The comparison with `FedMatch` reveals the same behavior seen in the previous results. First, our method, in general, outperforms `FedMatch` in all lesions. Second, applying PA to `FedMatch` (denoted as `FedMatch+`) boosts its accuracy. On the other hand, we observe an insignificant decrease in the accuracy of the AK lesion. The key factor of `FedPerl` advantage is attributed to the knowledge exchanged through peer learning.

5.5.11 Additional Evaluation Metrics

Area Under ROC & Precision-Recall Curves

For more validation, we report the area under ROC curve (AUROC) and the area under Precision-Recall curve (AUPRC) in Table 5.6 and Table 5.7 respectively. It is clearly shown that `FedPerl` exceeds `SSFLs` in all classes results except for the AK class. For instance, in AUROC results, the enhancement of `FedPerl` over `SSFL` around 2.1%, 1.4%, 1.1%, 1.4%, 4.2%, 1.8%, and 2.5% for the MEL, NV, BCC, BKL, DF, VASC, and SCC classes respectively. Moreover, the boosting of `FedPerl` over the lower `FedAvg` reaches 14% for the DF class. Interestingly, `FedPerl` outperforms both upper bounds for the SCC class. On the other side, comparing the AUPRC results reveals the same observations. Finally, the superiority of our method is still found over `FedMatch`.

Tab. 5.7. The area under Precision-Recall curve for the eight classes. +: with PA. †:∼[297]. ‡:∼FedMatch[130]

Setting	Model	MEL	NV	BCC	AK	BKL	DF	VASC	SCC
Lower	Local	0.505	0.864	0.622	0.457	0.409	0.287	0.524	0.164
	FedAvg	0.582	0.894	0.702	0.443	0.556	0.394	0.712	0.313
SSL	FixMatch	0.527	0.879	0.646	0.476	0.453	0.358	0.534	0.216
SSFL	FedAvg†	0.620	0.903	0.730	**0.494**	0.617	0.574	0.762	0.348
	FedMatch‡	0.640	0.906	0.745	0.460	0.615	0.559	0.753	0.344
	FedMatch+	**0.645**	**0.908**	**0.752**	0.479	**0.632**	**0.598**	0.761	**0.349**
w/o PA	FedPerl(T=2)	**0.642**	**0.910**	**0.751**	0.476	**0.630**	**0.618**	0.754	**0.368**
	FedPerl(T=2)	**0.651**	**0.911**	0.744	0.475	**0.629**	**0.627**	**0.769**	**0.356**
Upper	Local	0.596	0.899	0.710	0.561	0.555	0.456	0.690	0.361
	FedAvg	0.668	0.916	0.762	0.574	0.670	0.719	0.847	0.373

Risk Coverage curve

We show the Risk-Coverage curves for FedPerl and our baselines in Fig.5.5. Each plot in the figure depicts a model. Inside each plot, we draw the curves for all clients. The numbers next to a client name represent the risk value at the full coverage of the input data, *i.e.* (risk: coverage). It is shown from the figures that FedPerl achieves the lowest risk with the best coverage amongst all models, and this is for all clients except for clients 5 & 8. Note that the coverage of client 8 in all federated models is worse than the local models, which is attributed to class mismatch. Please refer to sections 5.5.8 and 5.5.9 for more details. Nevertheless, if we consider the clients 0, 4, and 9 as examples, we observe that FedPerl obtains the maximum coverage at risks of 25.7%, 24.4%, and 26.0%, respectively. These values are better than all local models, including the upper local model, and better than FedAvg SSL (SSFL) model. Though, an insignificant drop in the coverage is noticed for client 5 compared to SSFL. A detailed comparison between FedPerl and SSFL at 10% risk shows the superiority of FedPerl over SSFL in all clients, except client 8. For instance, the coverage jumps from $(33\%$ to $49\%)$ for client 2, from the range of $(46.5\% - 55\%)$ to the range of $(53\% - 65\%)$ for clients 0, 1, 3, 4, and 9, and from the range of $(79.8\% - 86\%)$ to the range of $(79.5\% - 90\%)$ for clients 5, 6, and 7. Note that the minimum coverage of the client is 7 in FedPerl (at 0 risk) is 30%, while it is 0 coverage at 0 risk for SSFL. Client 6, on the other hand, achieves a minimum coverage of 56% at 2% risk. Utilizing our method achieves lower risk and better coverage in skin lesion classification.

Reliability Diagram and Calibration Error

To investigate the uncertainty and models' calibration, we draw reliability diagrams and the expected and the maximum calibration errors in Fig.5.6. Then, we show the results for the federated models, including ours. The numbers inside the sub-figures show the calibration error for each client. The numbers next to the model name show all clients' averaged ECE and MCE errors. In each figure, we present the models' accuracy at different confidence intervals, such that the width of each bin represents the difference between the highest and lowest confidences. The figures show that our method improves the calibration for all models and reduces the errors significantly *cf.* Fig.5.6 (FedPerl vs. FedAvg models). While the most

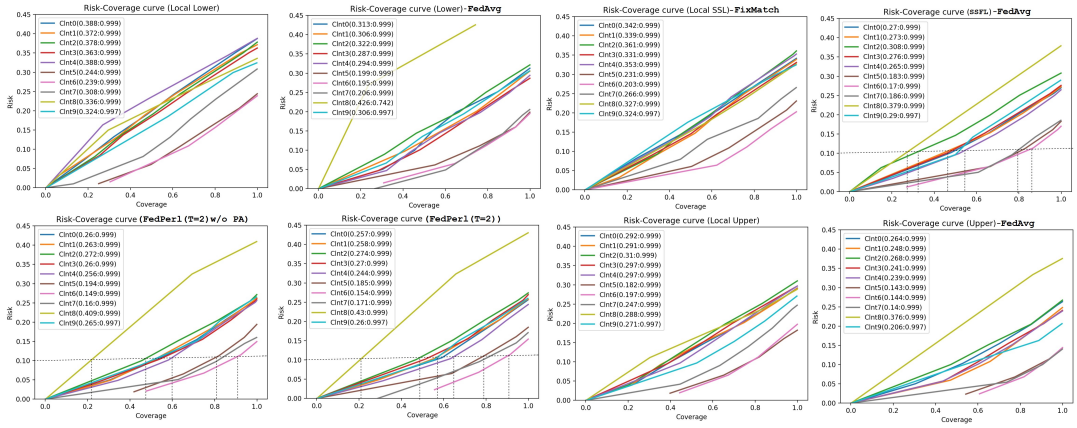


Fig. 5.5. The area under Risk-Coverage curve for FedPerl and the baselines. The numbers that appear next to a client name represent the risk at the full coverage respectively, *i.e.* , (risk: coverage). In general, FedPerl obtains the lowest risk with the best coverage among all models.

interesting and surprising results reveal that the lower federated model (Lower FedAvg) is the most calibrated model after the upper model (Upper FedAvg), such that it is better than SSFL and FedPerl respectively. We can attribute this issue to the uncertainty of using unlabeled data during the training of both models (SSFL and FedPerl). In contrast to that, the lower and the upper FedAvg models only trained on high-quality labeled data. Nonetheless, our model has better calibration errors than the SSFL, where the ECE and MCE are 0.144 and 0.277 for FedPerl, and 0.152 and 0.287 for SSFL, respectively. Besides, FedPerl outperforms the remaining baselines with considerable margins. Such lower calibration errors indicate more reliable and confident predictions for the FedPerl over the other methods. Moreover, our experiments showed that peer learning produced a more calibrated model than SSFL, *cf.* Fig.5.6 (FedPerl vs. SSFL models). Yet, after applying peer anonymization, a better calibration error is obtained, *cf.* Fig.5.6 (FedPerl(T=2) vs. FedPerl(T=2) w/o PA models). That implicitly means that the used peers are calibrated enough to produce more accurate pseudo labels than the ones generated from the clients individually.

5.5.12 Skin Lesion Qualitative Results

Sample predictions of FedPerl are shown in Fig.5.7. The first row shows that sample cases were classified correctly by FedPerl, yet, misclassified by the other methods. Below each case, we show the prediction confidence. The first row shows the confidence for FedPerl, while the second row shows the confidence for both FedPerl and SSFL respectively. It is noticed that there are challenging cases, still, FedPerl could classify them correctly, e.g., AK and SCC classes. The remaining cases were classified correctly with high confidence by FedPerl, while the others misclassify them. On the other hand, the second row shows cases that were classified correctly by both FedPerl and SSFL, yet, FedPerl achieves higher confidence. For instance, in BKL, DF, and SCC classes, the confidence margins are 35.9, 39.2, and 39.8, respectively.

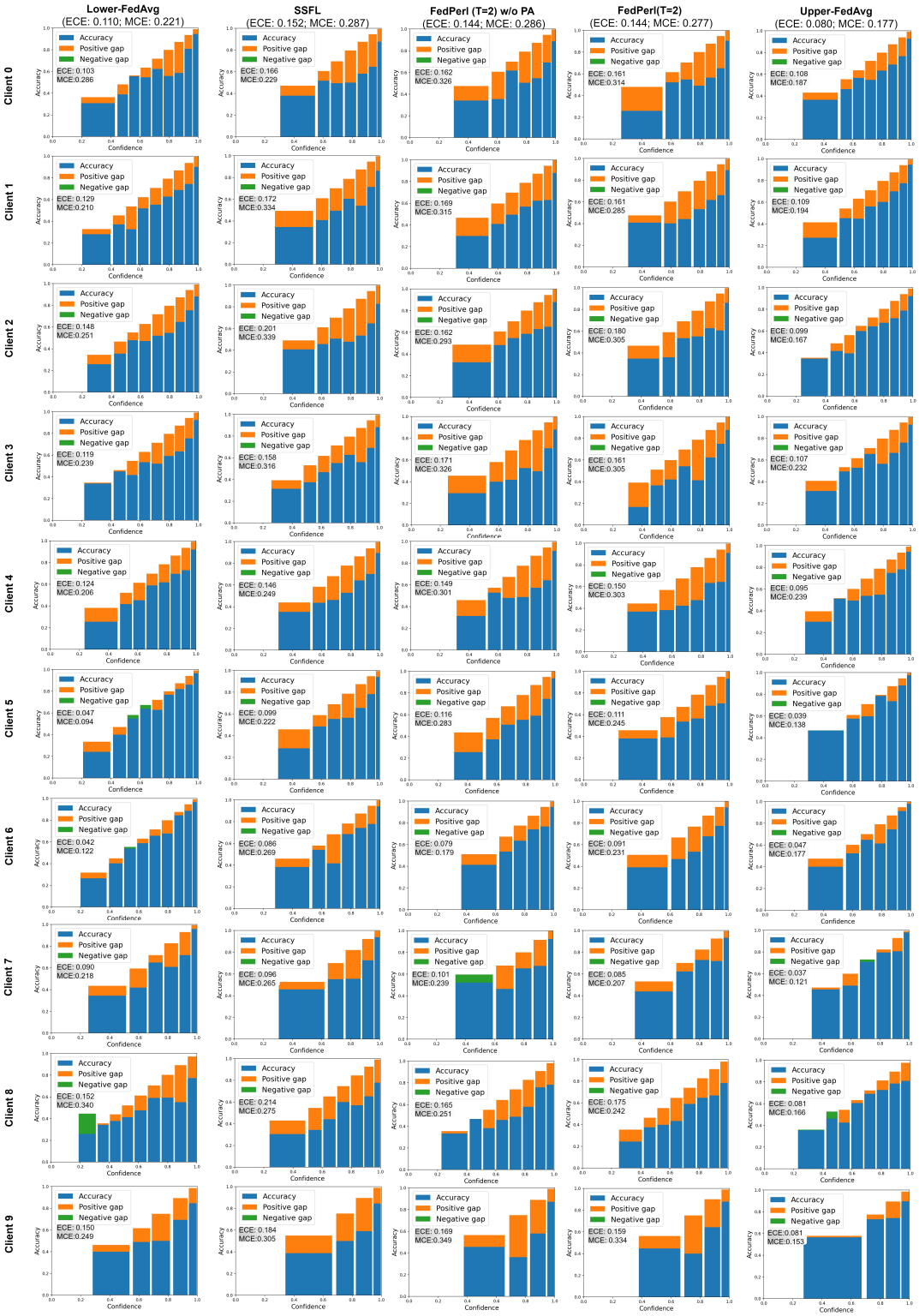


Fig. 5.6. Reliability diagrams and calibration errors. FedPerl is more calibrated than SSFL and local upper models indicating better and more confident predictions. The local models are shown in the supplementary materials.

5.5.13 Unlabeled Clients Scenario

Till this experiment, we have trained our models to exploit the labeled and unlabeled data at each client. The previous setting is widely studied in the literature, a.k.a, the standard semi-

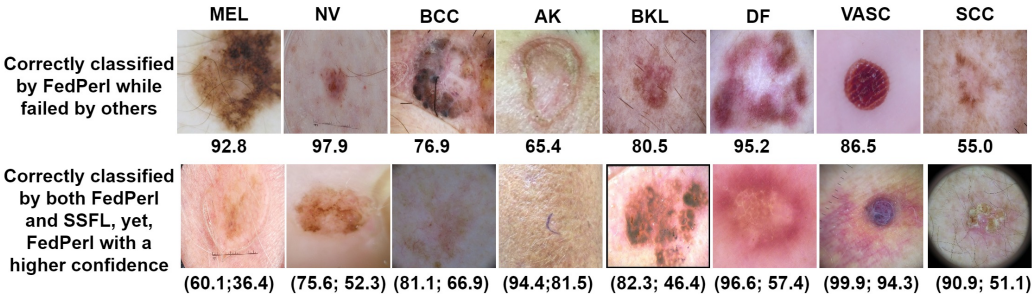


Fig. 5.7. Qualitative results. Sample predictions of FedPerl and SSFL for skin lesion. FedPerl confidence is shown below the images in the first raw, while the second raw shows the confidence for FedPerl and SSFL respectively.

Tab. 5.8. The classification results for unlabeled clients scenario. +: with PA. †:∼[297]. ‡:∼FedMatch[130]

Setting	Model	F1-score	Precision	Recall
SSFL	FedAvg†	0.637(0.649)±0.121	0.647(0.649)±0.099	0.670(0.678)±0.120
	FedMatch‡	0.641(0.662)±0.131	0.653(0.657)±0.099	0.667(0.693)±0.134
w/o PA	FedPerl(T=1)	**0.644(0.662)±0.115**	**0.658(0.660)±0.078**	**0.674(0.688)±0.118**
w/o PA	FedPerl(T=2)	**0.644(0.670)±0.126**	0.651(0.664)±0.100	**0.671(0.691)±0.130**
w/o PA	FedPerl(T=3)	**0.645(0.654)±0.117**	**0.655(0.657)±0.094**	**0.670(0.677)±0.123**
w/o PA	FedPerl(T=4)	**0.644(0.660)±0.124**	**0.655(0.665)±0.103**	0.668(0.678)±0.129
w/o PA	FedPerl(T=5)	**0.641(0.659)±0.129**	**0.655(0.660)±0.098**	0.668(0.681)±0.134
	FedMatch+	**0.649(0.662)±0.118**	**0.655(0.659)±0.102**	**0.677(0.688)±0.121**
	FedPerl(T=2)	**0.645(0.662)±0.119**	**0.654(0.659)±0.103**	**0.673(0.687)±0.119**
	FedPerl(T=3)	**0.648(0.663)±0.118**	**0.660(0.669)±0.102**	**0.678(0.693)±0.120**
	FedPerl(T=4)	**0.649(0.666)±0.124**	**0.656(0.663)±0.102**	**0.678(0.692)±0.125**
	FedPerl(T=5)	**0.645(0.659)±0.114**	0.652(0.653)±0.096	**0.675(0.687)±0.118**

supervised learning paradigm. In federated learning, however, a more challenging situation may appear to the surface in which the clients only have access to unlabeled data without knowing their annotations, see *Scenarios* in sec. 5.5.3 for more details. The results of applying this scenario to `FedPerl` and our baselines are reported in Table 5.8. Thanks to peer learning, our method enhances the performance of the baselines up to 0.8% and 0.4% compared to `FedAvg` and `FedMatch`, respectively. Moreover, an additional improvement of about 1.2% is obtained after applying peer anonymization (see last four rows in Table 5.8). That also holds for `FedMatch` where `FedMatch+` shows a relative improvement of about 0.8% after applying PA. The better results are attributed to the aggregated knowledge from distributed similar clients who help the local models overcome the missing labeled data.

5.5.14 Generalization to Unseen Client Scenario

The goal of this experiment is to investigate the generalization ability of the federated models to unseen clients. To achieve this, we collect the previously trained global models, including the baselines and `FedPerl`, then we perform inference on the ISIC20 dataset. Note that this

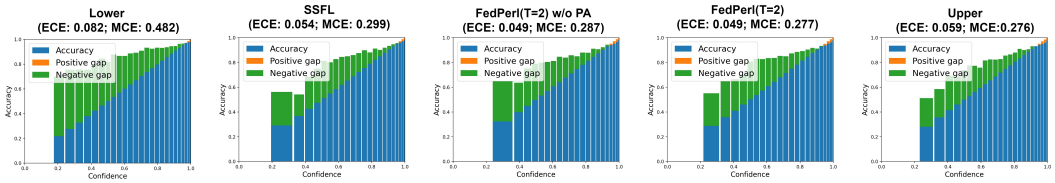


Fig. 5.8. Reliability diagrams and calibration errors on the ISIC20 dataset. FedPerl is more calibrated with lower calibration errors than the baselines.

Tab. 5.9. The unseen client scenario. The global models' classification results for FedPerl and the baselines on the ISIC20 dataset. +: with PA. †:∼[297]. ‡:∼FedMatch[130]

		Malignant			Benign		
Setting	Model	F1-score	Precision	Recall	F1-score	Precision	Recall
Lower	FedAvg	0.131	0.097	0.204	0.976	0.985	0.966
SSFL	FedAvg†	0.161	0.114	0.274	0.974	0.987	0.962
	FedMatch‡	0.160	0.113	0.278	0.972	0.987	0.954
w/o PA	FedPerl(T=1)	0.160	0.112	**0.279**	0.973	0.987	0.960
w/o PA	FedPerl(T=2)	**0.178**	**0.126**	**0.305**	0.974	0.987	0.962
w/o PA	FedPerl(T=3)	**0.166**	0.110	**0.339**	0.969	**0.988**	0.951
w/o PA	FedPerl(T=4)	**0.169**	**0.117**	**0.308**	0.972	0.987	0.958
w/o PA	FedPerl(T=5)	**0.166**	**0.120**	0.269	**0.975**	0.987	**0.965**
	FedMatch+	0.146	0.099	**0.281**	0.970	0.987	0.954
	FedPerl(T=2)	**0.163**	0.113	**0.295**	0.973	0.987	0.959
	FedPerl(T=3)	**0.167**	0.114	**0.308**	0.972	0.987	0.957
	FedPerl(T=4)	**0.170**	**0.115**	**0.324**	0.971	0.987	0.956
	FedPerl(T=5)	0.150	0.099	**0.305**	0.968	0.987	0.950
Upper	FedAvg	0.153	0.095	0.382	0.961	0.988	0.935

dataset consists of more than 33K images with two classes; malignant and benign. Considering that the class distribution is highly imbalanced, around 500 images contain malignant cases, while the remaining images have benign cases. Also, the models were trained to distinguish between 8 classes making the direct inference a very challenging task. To resolve this issue, we perform two steps. First, we generate the eight-class predictions from the models. Then, we assemble these predictions into two groups. The malignant group contains melanoma, basal cell carcinoma, actinic keratosis, and squamous cell carcinoma classes. The benign group includes melanocytic nevus, benign keratosis, dermatofibroma, and vascular lesions. Then, we generate our metrics as a binary classification task.

The results are reported in Table 5.9. Interestingly, `FedPerl` obtains the best malignant-class classification results outperforming the lower, the `SSFL` including `FedMatch`, and the upper bounds, with F1-score up to 0.178 for `FedPerl` models. Note that, for clinical applications, the ability of a model to detect the true positive cases (malignant) is high relevant than detecting

the true negative cases (benign) because the early detection of cancerous lesions reduces the treatment cost and the death rate. The ability of FedPerl to classify the malignant and benign classes is also shown in the reliability diagrams and calibration errors, *cf.* Fig.5.8. We can see from the figure that FedPerl is more calibrated and achieves better expected and maximum calibration errors than SSFL. From these results, we show that FedPerl has a better generalization ability to detect malignant cases than the baselines and FedMatch. While we have seen in all previous experiments that applying PA to FedMatch (denoted as FedMatch+) always boosts its performance; this observation does not hold in this experiment—specifically, the F1-score drops from 0.160 to 0.146. The same observation is found for some FedPerl models.

5.5.15 Comparison with SOTA in the Few Labeled Clients Scenario

In this experiment, we conduct a comparison with FedIRM [182]; very recent work in SSFL for the skin lesion classification. Notice that FedIRM introduced a scenario where some clients are labeled while others are not. In addition, the training paradigm in FedIRM assumed that all clients participate in the training in each round, *i.e.* , $PR = 100\%$, which is not applicable in many cases. The vast majority of federated learning approaches assume that a random set of clients will participate in the training each round, which was our selection in this paper where the $PR = 30\%$. Thus, to cover both cases, we present the results at $PR = \{30\%, 100\%\}$. For our comparison, we opt FedAvg and FedPerl(T=2) models. Note that the hyperparameters are kept as in the previous experiments, while the results are reported in Fig.5.9. First, let us consider when $PR = 30\%$. FedAvg obtains F1-score equals 62.3 while FedIRM, FedPerl w/o PA, and FedPerl achieve comparable results at 66.3, 66.1, and 66.1, respectively. Although our method outperforms FedAvg ($PR = 100\%$), we observe a slight relative drop in the performance when we compare to FedIRM, by 1.4% and 0.3% for FedPerl w/o PA and FedPerl, respectively. That could be attributed to that FedIRM only transfers the knowledge from labeled to unlabeled clients to guide the pseudo labeling process. However, this is not the case in our method, where we utilize similar peers (regardless of their labels). Note that around 80% of the clients are unlabeled in this particular scenario favoring the FedIRM method. Still, FedPerl outperforms FedAvg, and extensive hyperparameters tuning could yield better performance for our method. For the same reasons, FedPerl w/o PA, when $PR = 100\%$, achieves the lower results among FedPerl models with F1-score equals 65.7, where more unlabeled clients were involved in the training. Nevertheless, averaging the unlabeled peers might cancel their negative impact on the local model, as shown by FedPerl with peer anonymization (PA) at F1-score = 68.7.

5.5.16 Dynamic Learning Policy Results

Previously, we have shown that the static peer learning policy constantly benefits clients and communities. For instance, see the results in Table 5.4. Also, we have shown that for the individual clients, who do not belong to any community, our method is still profitable, as for client 9. However, for other clients, *i.e.* , client 8, we have seen that peer learning, FedMatch, and FedAvg perform lower than the local model, even though our model is better than the

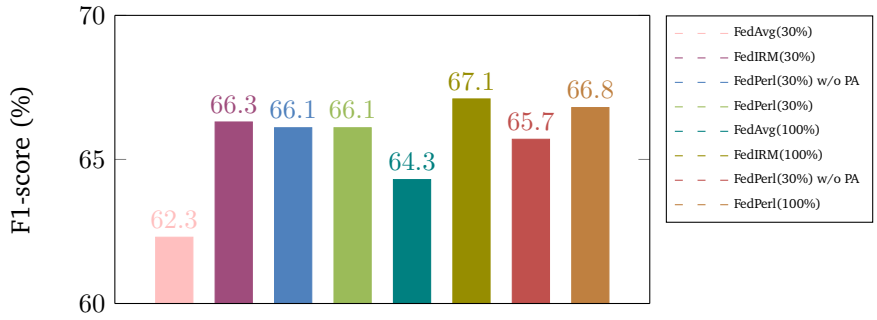


Fig. 5.9. Comparison between our Method and FedIRM. While both methods achieve comparable results when the participation rate =30%, ours show lower performance when 100% of the participation rate. Still, FedPerl outperforms FedAvg.

Tab. 5.10. Dynamic learning polices results. The classification results under two different settings. C_{ISIC}, C_{HAM}: the results at the community level. (mean F1-score). PA+/-: with/out Peers anonymization.

The classification results when the clients contain labeled and unlabeled data (the standard SSL setting)

Policy	Model/Client	0	1	2	3	4	C_{ISIC}	5	6	7	C_{HAM}	8	9	Avg
No Policy (baselines)	PA- FedPerl(T=2)	0.735	0.731	0.725	0.737	0.739	0.733	0.805	0.850	0.839	0.831	0.582	0.729	0.747
	PA+ FedPerl(T=2)	0.737	0.737	0.724	0.730	0.751	0.736	0.818	0.846	0.834	0.833	0.567	0.717	0.746
Validation Policy	PA- FedPerl(T=2)	0.729	0.727	0.724	**0.737**	0.746	0.732	0.814	0.845	0.819	0.826	0.571	**0.732**	0.744
	PA+ FedPerl(T=2)	**0.743**	0.729	**0.736**	0.732	0.749	**0.738**	0.806	0.845	0.822	0.824	0.572	0.724	0.746
Gated Validation Policy	PA-(75) FedPerl(T=2)	0.729	0.727	**0.738**	0.732	0.750	0.735	0.814	0.841	0.828	0.827	**0.598**	0.725	**0.748**
	PA+(75) FedPerl(T=2)	**0.746**	0.727	**0.737**	0.731	0.748	**0.738**	0.814	0.844	0.834	0.831	0.571	0.725	**0.748**
	PA-(85) FedPerl(T=2)	0.728	0.734	**0.740**	**0.738**	**0.760**	**0.740**	0.814	0.842	0.830	0.829	0.535	0.710	0.743
	PA+(85) FedPerl(T=2)	**0.738**	0.732	0.723	**0.742**	**0.755**	**0.738**	**0.818**	**0.850**	0.838	**0.835**	**0.596**	**0.729**	**0.752**
	PA-(95) FedPerl(T=2)	**0.739**	0.732	**0.735**	**0.737**	**0.754**	**0.739**	0.815	**0.850**	**0.839**	**0.834**	**0.583**	**0.730**	**0.751**
	PA+(95) FedPerl(T=2)	**0.747**	**0.745**	**0.731**	**0.738**	**0.752**	**0.743**	**0.818**	**0.851**	**0.839**	**0.836**	**0.596**	**0.731**	**0.755**
Gated Similarity Policy	PA-(75) FedPerl(T=2)	0.734	0.721	**0.742**	**0.739**	**0.764**	**0.740**	**0.821**	0.843	0.832	0.832	**0.593**	0.714	**0.750**
	PA+(75) FedPerl(T=2)	**0.740**	0.725	**0.735**	**0.742**	**0.754**	**0.739**	0.811	0.841	0.827	0.826	0.580	0.699	0.745
	PA-(85) FedPerl(T=2)	0.727	**0.741**	**0.731**	**0.742**	**0.751**	**0.739**	0.812	0.846	0.825	0.828	**0.586**	0.716	**0.748**
	PA+(85) FedPerl(T=2)	**0.738**	0.735	**0.735**	**0.742**	**0.752**	**0.741**	**0.820**	**0.850**	**0.839**	**0.836**	**0.588**	**0.730**	**0.753**
	PA-(95) FedPerl(T=2)	0.734	0.728	**0.732**	**0.739**	**0.765**	**0.740**	**0.820**	0.845	**0.839**	**0.835**	**0.617**	**0.731**	**0.755**
	PA+(95) FedPerl(T=2)	**0.737**	**0.740**	**0.731**	**0.739**	**0.764**	**0.742**	**0.819**	**0.853**	**0.836**	**0.836**	**0.618**	**0.732**	**0.757**

The classification results when the labeled data is only available on the server while the clients have no labeled data (the unlabeled clients or the disjoint setting)

Policy	Model/Client	0	1	2	3	4	C_{ISIC}	5	6	7	C_{HAM}	8	9	Avg
No Policy (baseline)	PA+ FedPerl(T=2)	0.649	0.642	0.671	0.645	0.654	0.652	0.730	0.751	0.729	0.737	0.308	0.670	0.645
Validation Policy	PA+ FedPerl(T=2)	0.642	0.638	0.669	**0.647**	**0.678**	**0.655**	0.721	**0.752**	**0.740**	**0.738**	0.267	0.656	0.641
Gated Validation Policy	PA+(75) FedPerl(T=2)	0.642	0.639	0.667	**0.647**	**0.659**	0.651	**0.740**	0.750	**0.745**	**0.745**	0.303	**0.678**	**0.647**
	PA+(85) FedPerl(T=2)	**0.669**	**0.657**	0.664	0.643	**0.666**	**0.660**	**0.740**	0.740	**0.744**	**0.740**	**0.340**	0.687	**0.655**
	PA+(95) FedPerl(T=2)	**0.653**	**0.650**	0.662	0.630	**0.669**	**0.653**	**0.743**	**0.751**	**0.733**	**0.742**	**0.343**	0.664	**0.650**
Gated Similarity Policy	PA+(75) FedPerl(T=2)	**0.659**	**0.660**	**0.671**	**0.650**	**0.657**	**0.659**	**0.732**	0.749	**0.744**	**0.742**	**0.334**	**0.671**	**0.653**
	PA+(85) FedPerl(T=2)	**0.649**	**0.645**	**0.679**	0.636	**0.656**	**0.653**	0.728	**0.757**	**0.751**	**0.745**	**0.327**	**0.670**	**0.650**
	PA+(95) FedPerl(T=2)	**0.658**	**0.647**	**0.678**	**0.655**	**0.676**	**0.663**	**0.732**	**0.759**	**0.738**	**0.743**	**0.336**	**0.672**	**0.655**

others. To resolve this issue, in section 5.4.4, we propose a dynamic learning policy that controls the learning stream of the clients. The results are reported in Table 5.10. Due to the enormous amount of models that could be examined in this experiment, we opt for ρ at $\{0.75, 0.85, 0.95\}$ and $T = 2$. We generate the results for the standard semi-supervised and unlabeled client scenarios. Our baseline in this experiment is our model `FedPerl(T=2)` because our goal is to compare with the static policy, and we do not see any need to include the previous models that already compared with `FedPerl(T=2)`.

The Standard Semi-supervised Learning Results

Validation Policy. First, by comparing overall results, denoted as Avg in Table 5.10, we notice no significant improvement in the performance for both models; PA($\pm$) `FedPerl(T=2)`. On the other hand, lower results are obtained for the C_{HAM} community. For instance, the F1-score dropped from 0.831 and 0.833 to 0.826 and 0.824, respectively. In contrast, a comparable result at 0.732 or a slight enhancement at 0.738 are obtained for C_{ISIC}. Besides, the clients' results are inconsistent regardless of whether they belong to a community. While we notice boosting for clients 0, 2, 3, and 9, the remaining clients have lower results. Further, we notice no positive influence on the results when applying PA.

Gated Validation Policy. While there is not much benefit in the previous policy, the results in this experiment show a consistent improvement as the gateway threshold ρ increases. For instance, the overall results boosted up to 0.2%, 0.6%, and 0.9% when $\rho = 0.75$, 0.85, and 0.95, respectively. A consistent improvement was also found at the community level when ρ is larger than 0.75, with better results at $\rho = 0.95$. While an increase reaches 1% is noticed for C_{ISIC} clients starting from $\rho = 0.75$ with PA model *i.e.* PA+(75) `FedPerl(T=2)`, the increase is seen starting from PA+(85) model for C_{HAM} clients with F1-score reaches 0.836. In general, the clients' results get boosted by our gated validation policy. In the beginning, when $\rho = 0.75$, clients 0, 2, and 8, show better performance than the baseline. Then, more clients are included when $\rho = 0.85$ until all clients show improvement with our model PA+(95) with F1-score at 0.752 at client 4. These results confirm the same behavior found in communities' results. A more discussion on the individual clients' results, *i.e.* , 8 & 9, reveals that the combination of PA with values of $\rho = \{0.85, 0.95\}$ achieves more reliable F1-scores. Even though our model PA-(75) obtains the highest score for client 8, the results for other clients are not of the same quality. In summary, we present in this experiment that our gated validation policy improves the overall, communities', and clients' results demonstrating its advantage. More importantly, the results of client 8 were boosted from 0.567 to 0.596 at PA+(95) model.

Gated Similarity Policy. This policy differs from the earlier one in using the similarity between the client and its peers as a gateway to control peers' participation instead of using the global validation dataset. We notice that the general behavior is similar to the preceding one. Though, better results are obtained at different levels, especially for client 8, whose reported F1-scores are equal to 0.617 and 0.618 on models PA$\pm$(95), which are better than the former ones by 1.9% and 4.7% respectively. The similarity in the results is justified because both policies proposed to manage the learning stream of the clients, especially the individual ones, shown in both strategies. A gated similarity policy brings more stability to all clients and better accuracy for client 8.

The Unlabeled Clients' Results

We have shown in the past section that the validation policy has no potential improvement while combining the gated methods with PA usually obtains the best performance. Therefore, and for simplicity, we only report the results with the PA technique.

After analyzing the second part of Table 5.10, we notice that the results of the validation policy are improved by F1-score equals to 0.655 and 0.738, for C_{ISIC} and C_{HAM} respectively. However, the overall results decreased by 0.4%. The individual results, on the other hand, vary

between the clients. While clients $3, 4, 6,$ and 7 show an enhancements, clients' $0, 1, 2, 5, 8,$ and 9 accuracies are decreased. In contrast to the previous results, we observe a constant improvement of gated policies in the overall accuracy from 0.647 to 0.655 for the validation with PA+(75) to similarity with PA+(95) gated models, respectively. Note that all models from both policies accomplish better results than the baseline model. On the other hand, the communities' results show comparable results, yet better than the baseline, for both strategies with some advantages for similar models. While the individual improvement is distributed among the clients in the gated validation approach except for client 2, it is intelligible in similarity models, especially in PA+(95) model. Moreover, both individual clients, 8 & 9, show steady improvements in all similarity models, yet, client 9 suffers from lower performance in gate validation with ρ larger than 0.75. However, the maximum gain appears for client 8 in the PA+(95) gated validation policy with F1-score equaling 0.343.

5.6 Discussion

Thus far, we have investigated the distributed data in remote locations by a novel approach that enables individual clients to share their knowledge through different peer learning policies in a semi-supervised federated learning setting. Thus, we propose FedPerl. This new approach compiles semi-supervised learning, federated learning, peer learning, committee machine, and learning policies to devise a novel framework for skin lesion classification tasks. We show through extensive experiments and evaluation metrics that our method performs better over the baselines in the standard semi-supervised labeled and unlabeled client settings. In this chapter, we will discuss our findings and observations of this method in detail.

5.6.1 Simplicity & Performance

A key feature of our method is simplicity. Implementing and applying our method is direct and can be implemented with a few lines of code. The computational cost to calculate the similarity between the clients is negligible, thanks to our strategy which computes the similarity on extracted features rather than on the whole weight parameters. Such that for a model with l layers and θ weights parameters, where $\theta \ggg l$, the cost of our similarity is $\mathrm{O}(l) \lll \mathrm{O}(\theta)$, note that θ could be millions of parameters. From another perspective, the experiments show that `FedPerl` is more calibrated and outperforms the baselines, including the `SSFL`, thanks to the peer learning we propose, where `FedPerl` exploits other clients by interacting with their experiences. As a core component in our method, peer anonymization reduces communication cost while enhancing performance. Additionally, it improves the clients' privacy by hiding their identities. Yet, a non-avoidable cost is still property in peer learning.

5.6.2 Similarity

Clients' communities are shaped implicitly based on the similarities between the clients. To measure the similarities, we exploited model parameters to profile the clients. Yet another approach to quantify the similarity is to use a server-side validation set as it has been utilized in `FedMatch`. While we have shown through different experiments that our method of finding

the similarities outperformed the one that depends on validation set *i.e.* FedMatch, another drawback is that the availability of validation datasets at the server side is challenging. Further, we have shown the importance of peer learning and our similarity in the random peers experiment. Still, the representational similarity is an open research direction in federated learning.

5.6.3 Orthogonality

Another main property of the PA technique is that it can be implemented directly to other methods, which are similar to ours, with negligible effort. We have shown through different experiments that applying PA to FedMatch is resulted in a better model, *i.e.* FedMatch+. While the new model achieves better accuracy, it also reduces the communication cost comparing to the original one.

5.6.4 Privacy

Our anonymized peer is designed by aggregating/averaging the model parameters of the top T similar peers. This process generates a virtual model that is not related to a specific client and offers a harder target for attackers seeking information about individual training instances [193, 207]. Nevertheless, a privacy guarantee for aggregated models (not individuals) is an open issue and has not been thoroughly investigated in the community and mathematical analysis is yet to be proven.

5.6.5 Local Updates

While the local models' weights are continually updated during the training, the peers' ones remain intact. A natural question could be if such a procedure might poison the models, especially with larger iteration updates? While such concern is of high importance, we have designed our method to alleviate this problem by training the local model and keeping the peer models intact to avoid any poisoning. Also, we employed an MSE loss as a consistency-regularization, FedVC approach in the federated learning, and our dynamic policy, especially if the local model is quite different from the peers and has been trained for more local iterations.

5.6.6 Communities & Committee Size

Figure 5.4 shows that FedPerl clusters the clients into communities based on their similarities. The overall performance for each community gets boosted by FedPerl, *cf.* Tables 5.3 and 5.4, which is attributed to the knowledge sharing. We have noticed that the community performance is related to the committee size *i.e.* T. While changing T has an insignificant effect on C_{ISIC} performance, its effect is clear on C_{HAM}. Thus, a natural question would be, what is the ideal committee size? Our experiments show that as long as T below the actual community size, the overall performance is rather stable, *cf.* C_{ISIC} in Table 5.3. Once T exceeds the community size, the performance starts decreasing, *cf.* C_{HAM} in Table 5.3.

We associate this with the probability of including external peers as we increase T, which might negatively influence the local models/sites of the community, see sec.5.5.8. While the cluster/community size can be defined by the cardinality of the spectral clustering of the similarity matrix, yet in more practical scenarios, setting T to a value larger than the community size is impractical. The trade-off between the committee size and the performance needs further investigation.

5.6.7 Clustering

The clustering in the literature means grouping similar data and assigning labels to them. Because we use this word frequently in our paper and to resolve any ambiguity, we provide the following interpretation. First, we do not use any clustering method nor it is defined heuristically or fixed at the beginning of the federated learning. Hence, we do not assign labels to the clusters, but rather we want to highlight that our similarity matrix works effectively to force similar peers to learn from each other. At the beginning of the training, the clusters, *i.e.* learning from similar peers, are dynamically changed. This is explained by the small numbers in each row in Fig. 5.4, where the darker colors or smaller numbers represent lower frequencies. However, as the training proceeds, these clusters are evolved to force the similar peers to learn from each other more frequently, which is shown by the brighter colors or higher numbers values in the same figure.

5.6.8 Individual Clients

The clustering produces individual clients who do not belong to a specific community *i.e.* clients 8 & 9, which confirms the reality. The effect of `FedPerl` is diversified between those two clients. While client 9 makes use of `FedPerl`, a drastic drop in the performance of client 8 was noticed, which could be attributed to the class distribution mismatch. This indicates that `FedPerl` may not fit non-iid scenarios. Yet, combining `FedPerl` with works that are handling the distribution mismatch (non-iid) problem would be a promising direction of research [171, 175, 309, 318]. On the other side, one nice property has been shown by our experiments that `FedPerl` is less sensitive to the noisy clients than the standard `SSFL` and `FedAvg` methods (*cf.* Table. 5.4), which could be attributed to the learning schema of selecting similar peers in `FedPerl`. In our experiments, we found out that inductive bias coming from similar in-distribution clients did not hurt the global model, it rather improved the global model performance. Having said that, Out-of-Distribution (OOD) client, e.g., client 8, has shown to harm the model's performance. If there is a strong inductive bias from a couple of OOD clients, this potentially might hurt the global model. One might need to consider a smarter way of aggregation for such OOD clients. However, this is out of the scope of this manuscript.

5.6.9 Unlabeled Clients

In a more challenging experiment, which is unique in federated learning, we trained our models utilizing labeled global data and unlabeled local ones. `FedPerl` also shows the best results comparing to the baselines thanks to our peer learning strategy, which enforces

additional knowledge to the clients besides the global one exploited via federated learning. Further, applying PA produced more stable results and higher accuracy.

5.6.10 Unseen Clients

In another part of our experiments, we have tested our method on the ISIC20 dataset. The low performance for all models can be attributed to two things; i) Class Mismatch: the models were trained on 8 classes while ISIC20 contains only two classes with severe class imbalance (500 malignant vs. 32.5K benign), and ii) Domain Shift: none of the models proposed to address the domain shift problem between ISIC19 and ISIC20. In this experiment, we tried to show how SSFL models perform in such a challenging situation. The results showed that our model is still better than all baseline models in skin cancer classification shedding the light on the generalization capability. This is attributed to the `FedPerl` is learning more powerful and discriminative representations of the minority class by aggregating the peers' knowledge and experiences. At the same time, we attributed the better performance of our model to `FedMatch` to the similarity matrix that we utilized such that our method picks more accurate peers to the local model than the `FedMatch` approach. In the current version, `FedPerl` does not have a specific property that handles the class imbalance.

5.6.11 Learning from Few Labeled Clients

The comparison with a SOTA method reveals that our method is on par with `FedIRM` when part of the unlabeled clients participate in the training. Yet, that is not the case when all clients are involved, which is a rare setup. We attributed that to the quality of the pseudo labels generated with the help of unlabeled peers to ones generated with the help of labeled clients. Nevertheless, combining both approaches in a joint or a co-training setup could be an interesting research direction and might lead to better performance.

5.6.12 Learning Policy

Our first strategy depends on a static peer learning policy that involves best T peers based on their similarities. While this policy is effective in the communities and clients, it suffers in performance when countered by an ODD client. To resolve this issue, we proposed, in this paper, more dynamic and adaptive policies. Specifically, the successful policies employed a gateway to control the learning rate from peers. The participation is measured based on either a global dataset or how similar the peer is to the client. Only the clients who pass a predefined threshold can participate in the training. We found that the results of the two policies are somehow similar with advantages to the one based on similarity. Yet, most importantly, the performance of the OOD client gets boosted by both policies. Because we do not have control or can not anticipate the in-distribution from out-of-distribution clients, the selection between the static and dynamic methods goes toward the dynamic ones. Even if we know the clients, the results show that the dynamic policy betters the static one. On the other hand, our preference between the validation or similarity gated policies goes toward the similarity. In most cases, the global validation data is not available, which prevents us from applying the validation policy. Further, the gated similarity policy produces more

consistent and stable results. Though, the trade-off between the global and local benefits could be the decision-maker in real-life scenarios. Our dynamic learning policies are considered heuristic ones, however, they were proposed to address a problem that we noticed in the static learning policy, where the performance of some individual clients has not improved by the federated learning. We could achieve that by utilizing the global validation dataset or the client similarities. However, to provide a comprehensive study addressing any potential questions from the reader, we tested three different policies. Note that the three policies are separate and work independently. Besides, they have shown to be effective (*cf.* Table 5.10).

6

Representations Learning via Virtual Embeddings and Self-Consistency in Self-Supervised Learning

> *Most of what we learn as humans and most of what animals learn is in a self-supervised mode, not a reinforcement mode. It's basically observing the world and interacting with it a little bit, mostly by observation in a test-independent way. This is the type of learning that we don't know how to reproduce with machines*
>
> **— Yann LeCun**
> (French computer scientist, a Professor at New York University, and Vice President, Chief AI Scientist at Meta. LeCun received the 2018 Turing Award for his work on deep learning.)

6.1 Motivation

Although the unlabeled data is missing their annotations, they are still wealthy with another type of information hidden in their representations. The last is what we researched in the third perspective by employing representation learning in a self-supervised learning paradigm. In this regard, self-supervised learning has recently gained much attention due to the high cost and data limitation in training supervised learning models. The current paradigm in self-supervised learning is to utilize data augmentation at the input space to create different views of the same images and train a model to maximize the representations between similar images and minimize them for different ones. While this approach achieves state-of-the-art (SOTA) results in various downstream tasks, it still lakes the opportunity to investigate latent space augmentation.

6.2 Contribution

In this part of our thesis, we propose **TriMix**, a novel concept for self-supervised learning that leverages the virtual data augmentation at the input and hidden embeddings, at the same time forcing the model to predict the percentage of such compositions in the virtual data from its original ones. Our method performs linear interpolation on the input images and their corresponding hidden embeddings. Likewise, the mixed samples are fed to the model to generate virtual features. During the training, the model learns to decompose the mixed-up augmented features to their original components through our ***virtual embeddings loss***. Note that the virtual embeddings are generated from a series of non-linear operations of the mixed-up data at the network. At the same time, the mixed-up data is produced from the linear process of the input images. Thus, matching the mixed-up data with its virtual embeddings might not be straightforward. To resolve this issue, we propose our ***self-consistency loss***, which ensures the linearity and forces both embeddings to be consistent. To this end, our **contributions** are:

- We propose **TriMix**, a novel method for self-supervised learning that leverages the augmentation at hidden embeddings in training and guides the model to decompose the mixed data to its original components through our ***virtual embeddings loss***. Furthermore, the newly generated representations are fine-tuned via redundancy reduction techniques to learn better discriminative features.

- We propose ***self-consistency loss*** to force the linearity and consistency between virtual embeddings and mixed-up embeddings for better training.

- We compare **TriMix** with recent self-supervised learning methods on eight benchmark datasets of natural and medical datasets while showing superior performance.

The content of this part is based on the following manuscript:

Bdair T, Abdelhamid H, Navab N, Albarqouni S. "**TriMix: Virtual embeddings and self-consistency for self-supervised learning**". Submitted to "IEEE Transactions on Neural Networks and Learning Systems". 2022 Jun 13.

6.3 Related Works

The current self-supervised learning approaches have shown remarkable advances in downstream tasks such as computer vision [15, 20, 43, 46, 53, 54, 56, 104, 109, 255, 264, 265, 306], natural language processing [74, 302], and speech recognition [206, 295]. Self-supervised learning methods do not rely on a massive amount of annotated data. Instead, they train a model to produce good representations of the unsupervised data, a.k.a pretext task that help in a supervised task such as image classification and segmentation, a.k.a downstream task.

An evolving direction in self-supervised learning methods known as contrastive learning utilizes siamese networks [38] to maximize agreement between different views of the same

image, known as positive samples, while decreasing it with other images, *i.e.* the negative examples as proposed in SimCLR [53] and MoCO [109]. One drawback of the previous works requires expensive computations to find the negative images from a memory bank [109] or a large batch size [53].

SwAV [46] overcomes this limitation by clustering the samples based on the similarities of their features while forcing a consistency between cluster assignments produced for the positive examples. However, BYOL [104] and SimSiam [56] relax the necessity for negative samples by employing asymmetric network tricks to avoid model failure while achieving state-of-the-art results.

Recent methods were proposed based on information maximization to avoid features collapse via (i) whiting approaches; W-MSE [43], (ii) redundancy reduction; Barlow Twins [306], (iii) features decorrelation and normalization; Shuffled-DBN [121], and (iv) variance-preservation term; VICReg [20]. While the previous methods work properly on highly curated datasets for pretraining such as ImagNet [73], DnC [264] alternates between the contrastive learning and clustering-based methods to improve the performance on less curated datasets.

So far, the above methods utilize augmentation at the input space to create different views of the same image to learn better representations. However, a couple of non-self-supervised works have shown a boost in performance in image classification [29, 275] or medical image segmentation [24, 48, 85, 135, 209] via the augmentation at the input space [29, 48, 85, 209], the hidden representations [275], or randomly at the input and hidden layers [24, 135]. Although the latter process provides virtual data points that benefit the model during the training, none of the former self-supervised learning methods has investigated that. Therefore, we propose our method that provides the model with virtual embeddings created at the hidden layers to learn better representations.

6.4 Methodology

Our method has the same architecture employed in the recent self-supervised learning methods [20, 43, 56, 104, 121, 306] where a siamese network [38] is trained on joint embeddings of distorted images. In addition, our strategy proposes a mutual training of redundancy reductions and latent space augmentation approaches. Thus, we build upon information maximization approaches to include our contributions. Specifically, we borrow the redundancy-reduction principle from Barlow Twins [306] before combining it with augmented embeddings and self-consistency methods. An illustrative diagram showing our method in Fig. 6.1, while PyTorch-style pseudocode for TriMix is shown in Algorithm 3.

6.4.1 Redundancy-Reduction

Given a batch of input images I sampled from a dataset $\mathcal{D}$, two different views (X, X') are generated by applying two transformations t and t' on I, where $X = t(I)$, $X' = t'(I)$, and t and t' are sampled from a distribution of data augmentations $\mathcal{T}$. Then, X and X' are encoded to a deep neural network with trainable parameters f_θ to produce the hidden representations

$Y = f_\theta(X)$ and $Y' = f_\theta(X')$, respectively. Next, these representations are fed to projector h_ϕ to create the embeddings Z and Z', where $Z = h_\phi(Y)$ and $Z' = h_\phi(Y')$. The embeddings are normalized along the batch dimension to produce unit vectors with 0 mean. The training loss appeared in Barlow Twins [306], consists of two terms; invariance $\mathcal{L}_{inv}$ and redundancy reduction $\mathcal{L}_{rr}$ terms as follows.

$$\mathcal{L}_{\mathcal{BT}} \triangleq \mathcal{L}_{inv} + \alpha\mathcal{L}_{rr}, \tag{6.1}$$

where α is a hyperparameter, and $\mathcal{L}_{inv}$ and $\mathcal{L}_{rr}$ are given by 6.2 and 6.3 , respectively.

$$\mathcal{L}_{inv} = \sum_i (1 - \mathcal{C}_{ii})^2, \tag{6.2}$$

$$\mathcal{L}_{rr} = \sum_i \sum_{i \neq i} {\mathcal{C}_{ij}}^2, \tag{6.3}$$

where $\mathcal{C}$ is the cross-coloration matrix between the two outputs of the network and given by

$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z_{b,i} \;\; z'_{b,j}}{\sqrt{\sum_b (z_{b,i})^2} \sqrt{\sum_b (z'_{b,j})^2}}, \tag{6.4}$$

where i, j index to the vector dimension of the networks' outputs, and b indexes to the batch samples. Note that $\mathcal{C} \in \mathbb{R}^{d \times d}$ is a square matrix with a size equal to the output dimension, with entries between (1) for perfect correlation and (-1) for perfect anti-correlation. Barlow Twins' objective function tries to find the best representations that preserve as much information about the samples. At the same time, the distortions applied to these samples are agnostic or less informative.

6.4.2 TriMix: Virtual Embeddings and Self-Consistency in Self-Supervised Learning

Augmented Embeddings

To augment the model with new data points, our method, shown in Fig.(6.1), takes one view of the input, *i.e.* X, and flips it to create a reversed version X^r, where $X^r = \texttt{flip}(X)$. Then, virtual data is generated by applying linear interpolation, *i.e.* Mixup [307], between the original and the reversed versions as follows.

$$X^{vrt} = \lambda * X + (1 - \lambda) * X^r, \tag{6.5}$$

where the mixup factor λ is randomly sampled from the $Uniform$ distribution; *i.e.* $\lambda \in [0, 1]$. Note that the Mixup is performed on one arm of the siamese network with its reversed version to guarantee that no sample is mixed-up with itself. Once we generate the mixed-up data, we pass it to the model to produce virtual embeddings; $Z^{vrt} = h_\phi(f_\theta(X^{vrt}))$. Then, Z^{vrt} is normalized along the features dimension and batch size to produce unit vectors with 0 mean.

Embeddings Decomposition

We train the model to decompose the virtual embeddings to their original components to learn useful representations from the new virtual data points and their embeddings. In other words,

we train the model to predict the values of the mixup factor λ. To achieve this, first, we create a cross-correlation matrix computed between the original embeddings of the input X and virtual embeddings, *i.e.* (Z, Z^{vrt}).

$$\mathcal{M}_{mn} \triangleq \frac{\sum_a z_{a,m} \; z^{vrt}_{a,n}}{\sqrt{\sum_a (z_{a,m})^2}\sqrt{\sum_a (z^{vrt}_{a,n})^2}}, \tag{6.6}$$

where m, n index to the batch samples, and a indexes the embeddings' vector dimension. Note that $\mathcal{M} \in \mathbb{R}^{B\times B}$ is a square matrix with a size equal to the batch size B. To this end, each row represents the similarities between an image m in the original batch and all the images in the virtual batch. Then, $\mathcal{M}$ is normalized using softmax operation to generate distributions with probabilities between $[0, 1]$ along its rows.

$$\mathcal{M}_{m,:} = \texttt{softmax}(\mathcal{M}_{m,:}/\tau), \tag{6.7}$$

where m indexes the batch sample, and τ is the temperature hyperparameter.

Virtual mbeddings loss. To enforce the model to regress the percentage of mixed-up data composition, we propose our virtual embeddings loss as the absolute mean difference between the matrix $\mathcal{M}$ and our ground truth matrix $\mathcal{GT}$, and given by

$$\mathcal{L}_{vrt} = \|\mathcal{M} - \mathcal{GT}\| \tag{6.8}$$

where $\mathcal{GT} = \lambda I + (1-\lambda)(I * R), \mathcal{GT} \in \mathbb{R}^{B\times B}$, where $I \in \mathbb{R}^{B\times B}$ is a square identity matrix with a size equal to the batch size B, and R is a transformation matrix that rotates I counterclockwise by 90 degree, see $\mathcal{GT}$ in Fig.(6.1).

Self-Consistency

Thus far, we train the model to decompose the virtual embeddings to their original components. However, the mixed-up data X^{vrt} are generated from linear interpolation of the input images (X, X^r). In contrast, the virtual embeddings Z^{vrt} are generated from a series of non-linear operations of X^{vrt} at the network. Thus, training the model to predict this linear operation of the mixed-up data from non-linear virtual embeddings might be challenging. To resolve this issue, we force the linearity and consistency between the virtual embeddings Z^{vrt} and *mixed-up embeddings*; $\tilde{Z}$. Such that we define $\tilde{Z}$ as the result of linear interpolation of the original inputs embeddings and their reversed version (Z, Z^r), and given by

$$\tilde{Z} = \lambda * Z + (1 - \lambda) * Z^r, \tag{6.9}$$

where $Z^r = \texttt{flip}(Z)$, and λ is the same one used in Eq.(6.5).

Self-consistency loss. Consequently, we propose our loss as the mean absolute difference between the two embeddings and the mixed-up embeddings to force the linearity and consistency between the virtual embeddings.

$$\mathcal{L}_{con} = \|\tilde{Z} - Z^{vrt}\|. \tag{6.10}$$

Algorithm 3 PyTorch-style pseudocode for TriMix

```
# net: f_θ(h_φ())
# α,β,γ: hyperparameters
# uniform: uniform distribution
# B: batch size
# D: embeddings dimensionality
# mm: matrix-matrix multiplication
# eye: identity matrix
for tuple in dataloader:
  x, x' = tuple # sample two augmented views
  z, z' = net(x), net(x') # produce embeddings
  # Barlow Twins
  z = (z - z.mean(0))/z.std(0) #BxD
  z' = (z' - z'.mean(0))/z'.std(0) #BxD
  # DxD cross-correlation matrix
  c = mm(z.T, z')/B
  # loss calculation for Barlow Twins
  c_diff = (c-eye(D)).pow(2)
  off_diagonal(c_diff).mul_(α)
  L_BT = c_diff.sum()
  # TriMix: 1. Augmented Embeddings
  λ= uniform() # sample the mixing factor
  x^r= flip(x)# reversed version
  z^r= flip(z)# reversed embeddings
  x^vrt = λ*x + (1-λ)*x^r # mixed-up/virtual data
  z^vrt = net(x^vrt) # virtual embeddings
  z̃ = λ*z + (1-λ)*z^r # mixed-up embeddings
  # TriMix: 2. Features Decomposition
  # Normalization along D and B
  z^vrt = (z^vrt - z^vrt.mean(0))/z^vrt.std(0)
  z^vrt = ((z^vrt.T - z^vrt.mean(1))/z^vrt.std(1)).T
  m = mm(z, z^vrt.T)/D # BxB matrix
  m = softmax(m(0)/τ) # softmax normalization
  # Create our ground truth
  gt = λ*eye(B)+(1 − λ)*rotation90(eye(B))
  # Virtual embeddings loss
  L_vrt = L1Loss(gt-m)
  # TriMix: 3. Self Consistency
  L_con = L1Loss(z̃-z^vrt)
  # Over all loss #
  loss = L_BT + βL_vrt + γL_con
  loss.backward()
  optimizer.step()
```

Overall Objective Function

The overall objective function is the sum of Barlow Twins, virtual embeddings, and self-consistency losses, and given by

$$\mathcal{L} = \mathcal{L}_{\mathcal{BT}} + \beta\mathcal{L}_{vrt} + \gamma\mathcal{L}_{con} \tag{6.11}$$

where β and γ are hyperparameters.

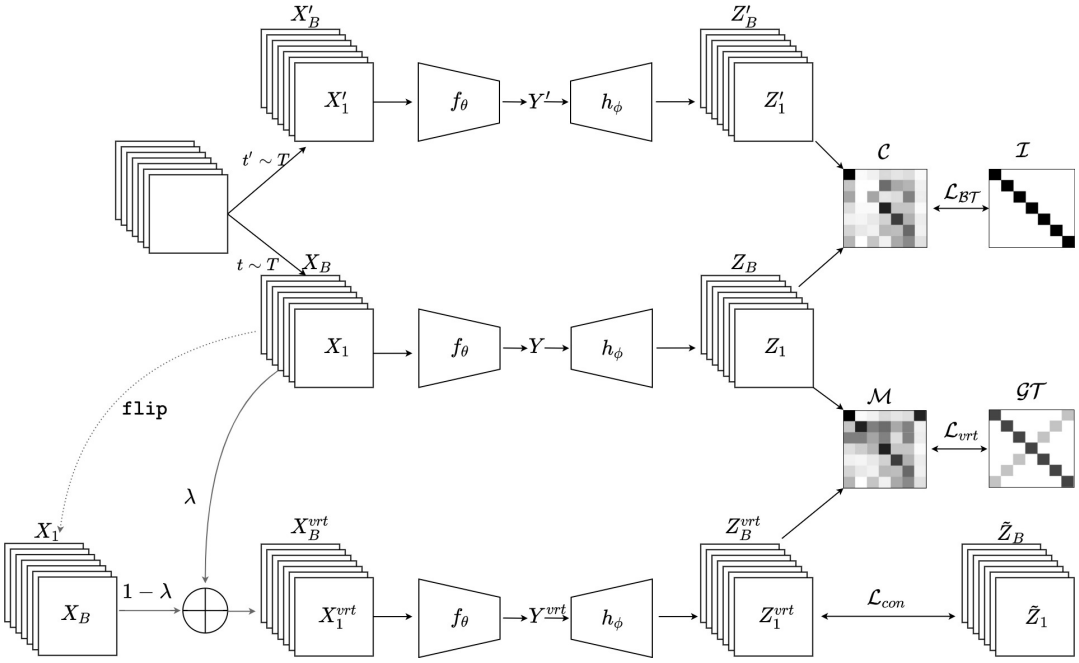


Fig. 6.1. TriMix mainly consists of virtual embeddings and self-consistency. Virtual embeddings: First, virtual data X^{vrt} is created by linear interpolation of the input images. X^{vrt} is then fed to the network to create the virtual embeddings Z^{vrt}. The model is trained to decompose the virtual data to the original ones using $\mathcal{L}_{vrt}$ loss, see Eq.(6.8). Self-consistency: to force the consistency between the virtual embeddings Z^{vrt} and augmented embeddings $\tilde{Z}$, we propose $\mathcal{L}_{con}$, see Eq.(6.10), where $\tilde{Z}$ is created using Eq.(6.9). PyTorch alike pseudo code is presented in Algorithm 3.

6.5 Experiments & Results

6.5.1 Datasets

We conduct our experiments on eight public benchmarks. (i) **CIFAR-10** [158] and (ii) **CIFAR-100** [158]. Both datasets consists of 32×32 images with 10 and 100 classes, respectively. (iii) **STL10** [62], consists of 96×96 images with 10 classes. (iv) **Tiny ImageNet** [163], consists of 64×64 images with 200 classes. Four medical datasets from **MedMNIST**[298, 299]. **MedMNIST** provides an MNIST-like set of standardized biomedical images consisting of 18 datasets with different scales and tasks. In this work, we randomly opt for four 2D multi-class datasets as follow. (i) **PathMNIST**: 107,180 colon pathology images of 9 classes. (ii) **DermaMNIST**: 10,015 dermatoscopic images of 7 classes. (iii) **OCTMNIST**: 109,309 retinal OCT images of 4 classes. (iv) **BloodMNIST**: 17,092 blood cell microscopic images of 8 classes. All the medical images are provided with the size of 28×28.

6.5.2 Baselines

Our baselines are the current state-of-the-art self-supervised learning methods. We opt for the following approaches; (i) SimCLR [53], (ii) BYOL [104], (iii) Barlow Twins [306], and (iv) VicReg [20].

Tab. 6.1. Top-1 accuracy(%) of TriMix and the baselines for KNN (with 200-epoch pretraining) and linear evaluations (with 100-epoch supervised training) on the benchmark datasets. All models use a ResNet-18 encoder and the same projector and augmentations. The best results are in bold. Our method outperforms other methods at different datasets in both KNN and linear evaluations.

	KNN				Linear			
Method	CIFAR10	CIFAR100	STL10	Tiny ImageNet	CIFAR10	CIFAR100	STL10	TinyImageNet
SimCLR	82.17	48.06	79.47	30.33	86.14	59.27	86.35	42.74
BYOL	81.13	45.73	81.19	30.75	85.43	57.31	86.33	41.94
Barlow Twins	84.37	52.47	80.98	**36.70**	86.93	60.66	86.29	**45.50**
VICReg	77.73	44.58	74.06	26.20	81.38	53.46	77.45	34.17
TriMix(ours)	**86.35**	**54.01**	**81.59**	34.66	**88.39**	**63.37**	**87.06**	45.15

6.5.3 Implementation Details

Adam optimizer [147] is utilized for training the models for 200 epochs, with a learning rate of 1×10^{-3}, weigh decay of 1×10^{-6}, and batch size of 256. We adopt ResNet-18 [110] as back-bone encoder with 512 output units. A 3-layer MLP with hidden layers of the size of 1024 is used as a projector. All hidden layers are followed by the batch normalization layer and ReLU activation. We set $\alpha = 5 \times 10^{-3}$ as in Barlow Twins [306], and perform grid search for β and γ where it found best at 1000 and 200, respectively. τ set to 2 as widely adopted in the literature. We opt for the PyTorch framework as an implementation environment hosted on a standalone NVIDIA Tesla V100 (Volta) with a 32 GB machine. The average training time takes around $7 - 8$ hours for each approach.

6.5.4 Image Augmentations

To produce the two views of the images, we follow the standard data augmentations used in the community. Specifically, random cropping, color jittering, horizontal flipping, and grayscaling were applied.

6.5.5 Results

KNN and Linear Evaluations on Natural Images

We evaluate the pre-trained representations using a supervised linear classifier on the frozen representations following the standard procedures. Specifically, after an unsupervised pre-training on the training sets for 200 epochs, the features were frozen, then a supervised linear classifier, consisting of a fully-connected layer followed by a softmax layer, was trained on the extracted features for 100 epochs. We use a learning rate of 1×10^{-3}, weight decay of 1×10^{-6}, a momentum of 0.9, and a batch size of 256. The results are reported in Table 6.1. Our method obtains the best top-1 accuracy of 88.39%, 63.37%, and 87.06% for the linear evaluation on CIFAR10, CIFAR100, and STL10 datasets, respectively, which are better than all baseline methods. Note that the KNN results reveal the same superiority of our approach over the baselines. On the other hand, TriMix achieves the second-best results and is on

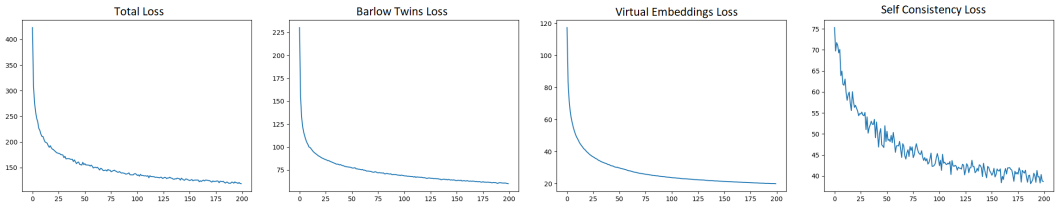


Fig. 6.2. TriMix training curves show that all losses converged and contributed significantly to the training on the TinyImage dataset. While tiny oscillations are noticed in the self-consistency loss, the overall trend is decreased.

Tab. 6.2. Top-1 accuracy(%) of the semi-supervised learning results (with 100 fine tuning epochs) using 1%, 10% and 100% training examples on the benchmark datasets. The best results are in bold. Our method outperforms three out of 4 datasets at lower data regimes (at 1% and 10%), while it obtains the second-best results at 100% data split.

	1%				10%				100%			
Method	CIFAR10	CIFAR100	STL10	TinyImageNet	CIFAR10	CIFAR100	STL10	TinyImageNet	CIFAR10	CIFAR100	STL10	TinyImageNet
SimCLR	79.03	29.94	65.23	16.14	86.16	51.65	79.98	35.5	**92.68**	**70.09**	88.94	**55.36**
BYOL	77.27	28.04	61.05	15.68	85.01	50.36	79.03	34.13	92.29	69.56	88.74	54.26
Barlow Twins	79.19	33.14	63.82	**20.28**	86.55	54.86	80.15	**39.83**	92.00	69.15	88.95	54.62
VICReg	70.07	22.51	53.79	12.23	81.94	45.93	71.93	31.35	91.97	69.18	85.41	52.38
TriMix(ours)	**81.03**	**34.08**	**66.80**	19.32	**87.56**	**56.23**	**81.08**	39.52	92.16	69.72	**89.54**	54.89

par with the Barlow Twins in the linear evaluation for the Tiny ImageNet dataset. For more illustration, we draw the losses during the training of our method on TinyImage in Fig.6.2. The curves show that our losses (virtual embeddings and consistency) are beneficial and contribute significantly to the training. Despite that, we notice negligible oscillations in the consistency loss, which is attributed to the complexity of its task. However, its overall curve decreases during the training. The above results demonstrate the importance of utilizing the manifold embedding augmentations and the self-consistency tasks to achieve outstanding results in self-supervised learning.

Semi-supervised Evaluation on Natural images

In this experiment, our method and the baselines were fine-tuned on subsets of 1%, 10%, and 100% of the benchmark datasets for semi-supervised learning. We use a learning rate of 1×10^{-3}, weight decay of 1×10^{-6}, a momentum of 0.9, and a batch size of 256 for 100 epochs. The obtained semi-supervised results are reported in Table 6.2. Our approach achieves the best results for the data splits at 1% and 10% on 3 out of 4 datasets. Nevertheless, our method gets the second-best results for the 100% of the data split. This experiment shows the effectiveness of our strategy in a lower data regime.

KNN and Linear Evaluations on Medical images

Thus far, we have shown the performance of TriMix in four natural datasets. In the following experiments, we validate our method on four publicly available medical datasets from MedMNIST. Note that we kept the same setup from previous experiments. In addition, we evaluated the pre-trained representations using a supervised linear classifier on the frozen representations from the pre-training step on the medical data. The results are reported in Table 6.3. Starting with the KNN results, TriMix outperforms all other methods in the four

Tab. 6.3. Top-1 accuracy(%) of TriMix and the baselines for KNN (with 200-epoch pretraining) and linear evaluations (with 100-epoch supervised training) on the medical datasets. Best results are in bold. Our method outperforms other methods in the medical datasets in the KNN and linear evaluations, confirming the results in the previous experiments.

	KNN				Linear			
Method	PathMNIST	DermaMNIST	OCTMNIST	BloodMNIST	PathMNIST	DermaMNIST	OCTMNIST	BloodMNIST
SimCLR	91.56	71.87	69.90	89.42	92.69	73.41	74.5	92.90
BYOL	91.50	71.16	70.00	86.14	92.72	72.47	75.80	90.73
Barlow Twins	91.36	71.68	71.20	87.89	92.42	73.36	**77.50**	92.28
VICReg	90.40	70.77	69.50	86.14	90.87	72.24	72.39	89.16
TriMix(ours)	**91.92**	**72.07**	**72.30**	**89.92**	**92.88**	**73.82**	76.60	**93.16**

Tab. 6.4. Top-1 accuracy(%) of the semi-supervised learning results (with 100 fine tuning epochs) using 1%, 10% and 100% training examples on 4 medical datasets. Best results are in bold. In general, our method shows the best results in all data split.

	1%				10%				100%			
Method	PathMNIST	DermaMNIST	OCTMNIST	BloodMNIST	PathMNIST	DermaMNIST	OCTMNIST	BloodMNIST	PathMNIST	DermaMNIST	OCTMNIST	BloodMNIST
SimCLR	90.47	66.88	77.30	84.68	91.95	71.22	80.69	**92.15**	**93.50**	75.86	**83.89**	95.81
BYOL	89.07	67.11	78.40	80.64	92.09	71.32	80.01	91.25	93.34	75.31	83.60	**96.11**
Barlow Twins	89.90	67.13	78.30	82.61	92.08	70.97	80.20	91.34	92.54	75.86	82.30	95.90
VICReg	89.77	66.98	79.30	78.54	91.61	**71.37**	80.59	88.97	93.38	76.00	81.50	95.32
TriMix(ours)	**90.81**	**67.23**	**78.80**	**84.89**	**92.15**	**71.37**	**80.78**	**92.15**	**93.50**	**76.21**	81.90	95.73

medical data with a classification accuracy of 91.92, 72.07, 72.30, and 89.92 for PathMNIST, DermaMNIST, OCTMNIST, and BloodMNIST, respectively, with improvement between 0.2% and 1.1% better than the second-best models, confirming the results in the previous experiments. The same superiority also is found in the results of the linear evaluation. For example, except for OCT images, our approach outperforms all methods with accuracy reaching 93.16 in blood cell microscopic images.

Semi-Supervised Evaluation on Medical images

The semi-supervised learning for the medical data is presented in Table 6.4. As in the previous setting, we fine-tune the pre-trained models on subsets of 1%, 10%, and 100% of datasets. The table shows that our method achieves the best results in all datasets at all data splits. Exception from that is the results of OCT and Blood datasets at 100% of the data. For instance, the accuracy of our approach at 1%, 10%, and 100% of the data are 90.81, 92.15, and 93.50 in PathMNIST, 67.23, 71.37, and 76.21 in DermaMNIST, 78.80, 80.78, and 81.90 in OCTMNIST, and 84.89, 92.15, and 95.73 in BloodMNIST, respectively. These experiments in both medical and natural datasets reveal the generalizability and applicability of our method to various types of images in self/semi-supervised settings. Nearly in all these experiments, TriMix has superiority over others. Still, another critical issue is to investigate our method in a transfer learning setting.

Transfer Learning from Natural to Medical Images

This experiment aims to build linear classifiers on top of fixed representations of the pre-trained models on the natural images. Then, we fine-tune these models on the four medical datasets simulating the transfer learning setting as in the literature, which resulted in 80 models shown in Table 6.5. One can notice that our approach generalizes better when using a pre-trained model on CIFAR10 and TinyImage. Nevertheless, Barlow Twins works better

Tab. 6.5. Top-1 accuracy(%) of transfer learning experiments of the pre-trained models on natural data to the medical data. The best results are in bold. Our learned representations capture more beneficial information and generalize better than the remaining approaches.

	CIFAR10				CIFAR100			
Method	PathMNIST	DermaMNIST	OCTMNIST	BloodMNIST	PathMNIST	DermaMNIST	OCTMNIST	BloodMNIST
SimCLR	77.17	71.57	63.90	77.81	76.43	71.57	60.50	76.23
BYOL	78.19	72.21	65.00	79.42	80.78	71.52	57.30	75.85
Barlow Twins	78.30	71.67	62.70	**80.70**	77.60	**73.01**	58.40	**80.35**
VICReg	77.47	71.77	63.50	77.19	80.87	71.22	**65.10**	76.79
TriMix(ours)	**78.43**	**72.27**	**67.20**	78.49	**80.92**	71.17	57.40	78.84
	STL10				TinyImageNet			
Method	PathMNIST	DermaMNIST	OCTMNIST	BloodMNIST	PathMNIST	DermaMNIST	OCTMNIST	BloodMNIST
SimCLR	74.27	70.22	40.90	61.47	77.43	70.92	46.00	69.48
BYOL	72.93	69.52	**57.90**	**67.72**	77.42	70.97	52.10	**73.98**
Barlow Twins	**77.40**	70.12	50.20	67.17	76.50	71.06	**56.20**	73.77
VICReg	71.10	70.02	44.60	60.07	72.86	71.02	47.50	64.42
TriMix(ours)	76.56	**70.32**	49.30	63.84	**77.77**	**71.87**	51.20	72.46

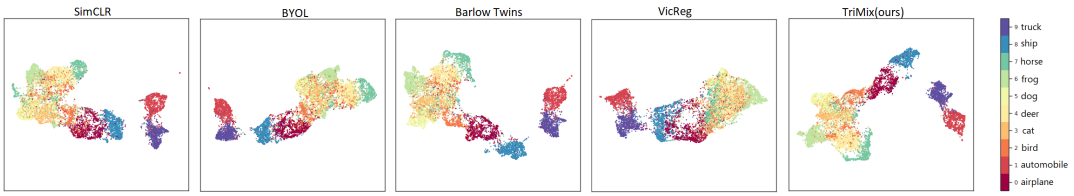


Fig. 6.3. 2D UMAP projection of CIFAR10 testing dataset using different self-supervised learning methods. Our method (TriMix) is better and less noisy in clustering the ten classes than the remaining methods.

using CIFAR100, and BYOL outperforms others using STL10. Generally, TriMix is among the best models regardless of the dataset used. This experiment reveals that our learned representations capture more beneficial information and generalize better than the remaining approaches.

Clustering the Learned Representations Y Analysis

To gain more insights into the effect of our approach on the learned representations and realize the differences between ours and the baselines, we visualize the learned representations using UMAP [191], an open-source library for dimensionality reduction. As an illustration, we cluster the learned representations for the CIFAR10 testing dataset in Figure 6.3. It is noticed that our method clusters the ten classes more precisely than the others. For example, the classes of "truck" and "automobile" are more compact by ours than the others, while the classes of "ship" and "airplane" are less overlapped with each other. On the other hand, the animals' classes (*i.e.* , bird, cat, deer, dog, frog, and horse) represent a challenge to all models; nevertheless, our method clusters them with lower noises.

Objective Function Components Analysis

This experiment investigates the effect of our main contributions. We train different versions of TriMix with(out) virtual embeddings and self-consistency terms on the CIFAR10 dataset for 200 epochs. Then, we conduct a linear evaluation for 100 epochs and register the results in

Tab. 6.6. Objective function components analysis. Linear evaluation on the CIFAR10 and STL10 data. Our contributions enhance the baseline while combining them boosts the performance significantly.

Configuration	CIFAR10
Barlow Twins	86.29
TriMix: Virtual embeddings loss only	87.16
TriMix: Self-consistency loss only	87.32
TriMix: Virtual embeddings loss + self-consistency loss	87.74
TriMix: Virtual embeddings loss + self-consistency loss + feature norms	**88.39**

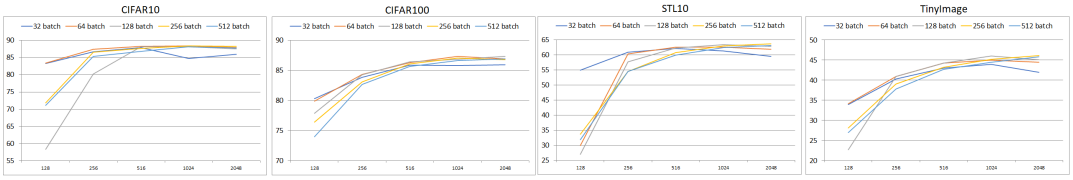


Fig. 6.4. Top-1 accuracy(%) of TriMix when changing the batch size and features dimension. (i) Batch size: increasing the size of the features enhances the results. (ii) Features dimension: smaller batches work better with smaller dimensions and vice versa.

Table 6.6. First, it is shown that utilizing either of our contributions enhances total accuracy with 87.16 and 87.32 when using the virtual and self-consistency losses, respectively. While combining both terms adds more enhancements up to 87.74. Finally, joining all contributions, including the features normalization, plays a significant role in the overall performance at 88.39.

Batch and Projector Sizes Analysis

Further, we test the effect of changing the batch size and projector dimension on TriMix. Specifically, we investigate batches of size {32, 64, 128, 256, 512}, with features dimensions of {128, 256, 512, 1024, 2048} on CIFAR10/100, STL10, and TinyImage. The results are curves for 100 models, presented in Fig.6.4. In general, the accuracy curves show that for fixed batch size, using more extensive features achieves better results, which agrees with the literature. Also, for most models, no significant difference in the accuracy for dimensions 1024 and 2048. Further, for fixed feature size, the smaller batches work better with smaller dimensions, while bigger batches favor bigger dimensions.

Virtual Data Analysis

This experiment aims to realize how our method decomposes and regresses the augmented data from its mixture. In Fig.6.5, we present sample images from the STL10 dataset. Fig.6.5.(A) shows a sample ground truth matrix ($\mathcal{GT}$), which is created at each epoch, see Eq.(6.8). To illustrate, we also show the corresponding batch in the above raw. To the left of the matrix, we display the augmented images. For example, the one in the red rectangle is generated by mixing the two images from the original batch, *i.e.* , the images in green rectangles in the above raw, using λ and $1-\lambda$ (0.51 and 0.49, respectively in this sample). Figures 6.5.(B) and 6.5.(C),

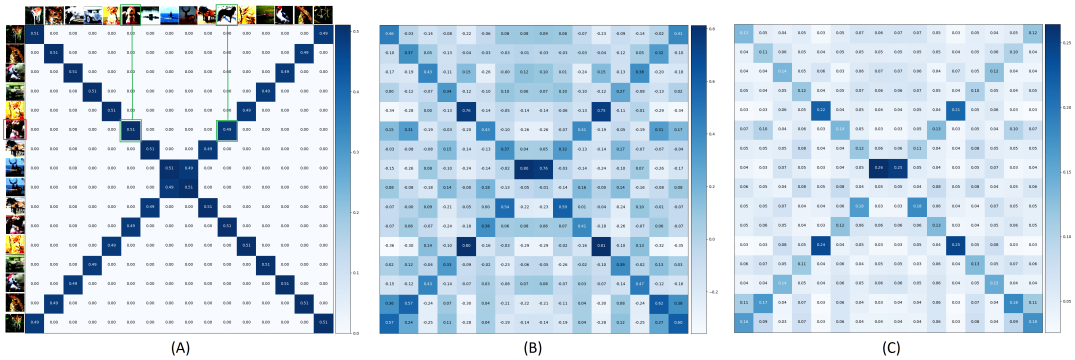


Fig. 6.5. Mixed data analysis. (A) sample ground truth matrix. We depict the corresponding batch in the above raw. To the left of the matrix, we display the virtual images. For example, the image in the red rectangle is generated from the images in the green rectangles. (B) and (C) show the predicted matrix before and after softmax operation. By raining on this task, our method gains more information from the images and hence, better performance.

on the other hand, show the predicted matrix before and after softmax operation, which are corresponding to Eq.(6.6) and Eq.(6.7), respectively. Our method attempts to anticipate the ground truth matrix and decomposes the mixed images into the original parts, as shown in Fig.6.5 (B). Thus, by training on this auxiliary task, our method learns to distinguish between different images and gain more information, boosting performance. Notice that the predicted matrix, in Fig.6.5 (B), contains some noises, *i.e.* , negative values. However, the model produces more stable predictions and better results after applying the softmax operation, Eq.(6.7).

TriMix Variants Analysis

Our initial setting depends on Z to train the model. While in this experiment, we explore alternative ways of implementing our loss functions. Specifically, we attempt to use the hidden feature representation Y in our objective functions. The first alternative is to define virtual embeddings loss and self-consistency loss on feature representation Y. The second choice is to optimize virtual embeddings loss on embeddings Z and self-consistency loss on hidden feature Y. Eventually, we compared both experiments with our baseline model and reported the results in Table 6.7. In short, our initial choice for the overall loss function that depends on Y is still the most reasonable approach. However, using our losses on different data types, *i.e.* , virtual embeddings loss on embeddings Z and self-consistency loss on hidden feature Y achieve the lowest accuracy. That is related to the fact that each term works on incompatible features (*i.e.* , Y and Z), confusing our objective function.

6.6 Discussion

Our last contribution is to address the deficiency of the labeled data by investigating the representations of learning in a self-supervised learning setting. To achieve that, we propose our method, TriMix, which incorporates; i) virtual embeddings loss: first, we augment the network with novel embeddings generated from the original ones, then we train the model to decompose these virtual embeddings to their original components, 2) self-consistency term

Tab. 6.7. TriMix: Loss function based on feature representation Y analysis. Linear evaluation on the CIFAR10. Our initial setting is the most valid option. While using different features in both terms confuses the model.

Configuration	Accuracy
Virtual embeddings loss on Y + self-consistency loss on Y	86.32
Virtual embeddings loss on Z + self-consistency loss on Y	85.37
Virtual embeddings loss on Z + self-consistency loss on Z (Baseline)	**88.39**

that enforces the consistency between the virtual and the original data. In this chapter, we will discuss our findings and observations of this method in detail.

6.6.1 Applicability and Transferability

Our method has shown to be effective in all experiments beating recent SSL methods in most tasks. First, we have demonstrated the applicability of our approach to eight public datasets, including natural and medical images. Also, our strategy was very beneficial in semi-supervised learning settings, especially at lower data regimes *i.e.* , 1% and 10%, where the models suffer from scarcity in the labeled data. That corresponds to the need, in this particular setting, for additional and novel representations to augment the model with new training data. Even though no specific approach was dominated in the transfer learning experiments due to the complexity of transferring the pre-trained models, using any natural dataset, to all medical data and achieving SOTA performance. Still, among all the pre-trained models, ours were the most successful.

6.6.2 Manifold and Hidden Embeddings Augmentation

The current self-supervised learning methods heavily depend on two views or augmentations of the input data to train the network. While this is an essential step for any successful self-supervised approach, none of the previous efforts have investigated the augmentation at the manifold or hidden representations. We have shown in this thesis that attaching the manifold augmentation to the training boosts performance. Our augmentation methodology depends on mixing the original embeddings with random percentages while training the model to predict these percentages and decompose the augmented data to the original elements. Analyzing the hidden representations shows that our method is better at clustering the classes with fewer noises, which justifies its top performance over other baselines. Still, an important question addressed by our thesis was where to inject this function. We have shown that placing the two losses at the hidden embeddings, *i.e.* , Z, achieved the best results. That is attributed to the homogeneity of used data in both terms. On the other hand, placing the two losses at different locations, *i.e.* , embedding Z or hidden representations Y, might confuse the model and reduce its performance. While we kept our augmentation methodology simple, one can investigate more sophisticated augmentation approaches or inject more extra tasks. Model failure is one of the most challenging tasks in that situation.

6.6.3 Interpretability

TriMix performance is related to the training strategy we introduced. Our auxiliary task serves two purposes. First, augment the model with novel data generated from the mixup operation at the manifold layers. It is known that data augmentation plays a fundamental role in the performance of any deep learning method. Second, train the model to distinguish between the images by predicting the mixing ratio used to generate the new data. For illustration, consider a case when mixing a malignant sample with a benign one. Now, when we train our model to decompose the combined image to the original ones. The model implicitly learns the characteristic of each class. Thus, more information is being realized by achieving this task.

7 Conclusion & Future Works

7.1 Conclusion

This thesis addresses the insufficiency of annotated data by proposing annotation-efficient methods in medical imaging. We tackle the problem from three related perspectives that handle the scarcity of labeled data. First, we investigate data augmentation via random linear interpolation in a semi-supervised learning setting. Our proposal uses linear interpolations at the input and hidden spaces to boost the model's performance with newly generated data points that fit the complexity of the medical images. Further, we show the applicability of our method to medical image segmentation tasks for brain and lung images. Our comprehensive analysis shows that our method efficiently utilizes labeled and unlabeled data, proving its stability, superiority, and consistency. Thus, it addresses our first research question, to what extent does the massive available non-annotated data in training help build robust deep learning methods suitable to the complexity of medical images?

In some cases, the availability of high-quality annotated data is impossible. Thus, we ask whether we can incorporate and seek other available data resources from remote locations to overcome the labeled data's limitation and build powerful deep learning models. Our second contribution addresses this point by proposing a federated semi-supervised learning method inspired by peer learning, knowledge sharing from educational science, and ensemble averaging from committee machines. Specifically, our strategy is based on knowledge sharing via static & dynamic peer learning policies. Finally, we show a real-life application of our approach that suits the characteristics of the medical data, i.e., data heterogeneity, severe class imbalance, and an abundant amount of unlabeled data on skin lesion classification in dermoscopic images. Furthermore, our testing environment comprises the standard semi-supervised setting and a more challenging and less investigated scenario where clients can access the unlabeled data. Our method is on par with the state-of-the-art process in skin classification in the standard federated learning and outperforms all other baselines. Moreover, our solution demonstrates less sensitivity to noisy clients and better generalizes to unseen data. Besides, we propose the peer anonymization (PA) technique. PA is a simple and efficient approach to creating an anonymized peer and hiding clients' identities. PA enhances performance while reducing communication costs. We show that our method is orthogonal and easy to implement to other methods without additional complexity. By achieving most of the best results, we believe that our proposed method answers the second research question.

Our last research question was whether we could mine and extract the knowledge from unlabeled data to perform the same tasks achieved by the labeled ones. In this regard, we suggested to utilized representation learning and proposed virtual embeddings and self-consistency in our self-supervised learning method, TriMix. While the current works depend on

data augmentations at the input space in Siamese-based architectures, our approach proposed an auxiliary task that generates new virtual data from hidden embeddings. Besides providing new data, our training strategy is to learn the model to predict the mixing factor in the data so that the model can distinguish between the images and decompose the mixed data to the original components. Further, we propose a loss term to force self-consistency in the data. We have shown the applicability of our method on eight public datasets consisting of natural and medical images, with notable improvements in both data types. Moreover, our approach demonstrated superior performance in the low amount of data of the semi-supervised learning setting while on par with the best models when utilizing the whole training data. Although none of the methods excels in all transfer learning experiments, our pre-trained models showed the best accuracy. Our strategy highlights the importance of embedding augmentations and additional tasks to achieve leading results. Accordingly, we found that utilizing this approach provides a decent solution that mitigates the scarcity of annotated data.

7.2 Future Works

Although this work allowed us to achieve promising results and outstanding performances, several possible future directions and extensions can be studied and addressed.

Generating The New Data. In the first approach, the quality of the generated data and pseudo labels mainly depends on the initial guess and the mixup coefficient λ. Also, we opt for a simple augmentation methodology in the third method to avoid any potential model collapsing. However, studying different ones, including sophisticated techniques, could be a future research direction. For instance, one could think of modeling this coefficient as a function of uncertainty measures. Also, to generate more realistic mixed-up data, one could investigate performing the mixup operation on disentangled representations [113]. Therefore, instead of naive mixups, one could explore more intelligent ways of data mixing. Further, in the last approach, we build a ground truth matrix based on an arbitrary factor at the beginning of each epoch. However, we could research a more intelligent and adaptive way that evolved during the training. On the other hand, we already enhanced the pseudo-labeling process in the second approach by seeking help from similar clients. Still, we could refine it by including post-processing steps such as conditional random fields [155] or prior knowledge extracted from the data [93, 182].

Learning Policies. In this thesis, we investigate two learning policies; a fixed approach that selects similar top peers and a dynamic and more adaptive one that controls the learning stream of the clients. We have shown that both strategies are effective with advantages to the dynamic one. Thus far, we exploited the model parameters as similarity measurement, while we could employ different techniques to profile the clients such as graph-based methods[40, 50].

Privacy. While we encourage our method to exchange knowledge in FedPerl (second contribution), the ensembling approach is utilized to build the anonymized peer, which hides clients' identities and improves privacy. Yet, we could investigate the privacy guarantee for aggregated models as future work.

List of Authored and Co-authored Publications

A

2021

[1] **Bdair T**, Navab N, Albarqouni S. "FedPerl: Semi-supervised peer learning for skin lesion classification". In International Conference on Medical Image Computing and Computer-Assisted Intervention 2021 Sep 27. Springer, Cham.

2022

[2] **Bdair T**, Navab N, Albarqouni S. "Semi-Supervised Federated Peer Learning for Skin Lesion Classification". Journal of Machine Learning for Biomedical Imaging (MELBA) 2022 March 5.

[3] **Bdair T**, Wiestler B, Navab N, Albarqouni S. "ROAM: Random layer mixup for semi-supervised learning in medical images". IET Image Processing. 2022 May 2.

[4] **Bdair T**, Abdelhamid H, Navab N, Albarqouni S. "TriMix: Virtual embeddings and self-consistency for self-supervised learning". Submitted to "IEEE Transactions on Neural Networks and Learning Systems". 2022 Jun 13.

Bibliography

[1] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti. “A survey on homomorphic encryption schemes: Theory and implementation”. In: *ACM Computing Surveys (Csur)* 51.4 (2018), pp. 1–35 (cit. on p. 49).

[2] H. J. Aerts, E. R. Velazquez, R. T. Leijenaar, et al. “Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach”. In: *Nature communications* 5.1 (2014), pp. 1–9 (cit. on pp. 5, 63).

[3] A Agrawala. “Learning with a probabilistic teacher”. In: *IEEE Transactions on Information Theory* 16.4 (1970), pp. 373–379 (cit. on p. 34).

[4] S. Albarqouni, S. Bakas, K. Kamnitsas, et al. *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning: Second MICCAI Workshop, DART 2020, and First MICCAI Workshop, DCL 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4-8, 2020, Proceedings*. Vol. 12444. Springer Nature, 2020 (cit. on pp. 50, 88).

[5] S. Albelwi. “Survey on Self-Supervised Learning: Auxiliary Pretext Tasks and Contrastive Learning Methods in Imaging”. In: *Entropy* 24.4 (2022), p. 551 (cit. on pp. 52, 53, 56, 57).

[6] D. Alexey, P. Fischer, J. Tobias, M. R. Springenberg, and T. Brox. “Discriminative unsupervised feature learning with exemplar convolutional neural networks”. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 99 (2015) (cit. on p. 53).

[7] R. Alizadehsani, D. Sharifrazi, N. H. Izadi, et al. “Uncertainty-Aware Semi-Supervised Method Using Large Unlabeled and Limited Labeled COVID-19 Data”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 17.3s (2021), pp. 1–24 (cit. on pp. 45, 65).

[8] M. Andreux, J. O. d. Terrail, C. Beguier, and E. W. Tramel. “Siloed federated learning for multi-centric histopathology datasets”. In: *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*. Springer, 2020, pp. 129–139 (cit. on p. 50).

[9] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier. “Federated Learning for Healthcare: Systematic Review and Architecture Proposal”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 13.4 (2022), pp. 1–23 (cit. on p. 50).

[10] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan. “Medical image analysis using convolutional neural networks: a review”. In: *Journal of medical systems* 42.11 (2018), pp. 1–13 (cit. on p. 32).

[11] Y. M. Asano, C. Rupprecht, and A. Vedaldi. “Self-labelling via simultaneous clustering and representation learning”. In: *arXiv preprint arXiv:1911.05371* (2019) (cit. on p. 58).

[12] B. Aubin, A. Maillard, J. Barbier, F. Krzakala, N. Macris, and L. Zdeborová. “The committee machine: Computational to statistical gaps in learning a two-layers neural network”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2019.12 (2019), p. 124023 (cit. on pp. 88, 91).

[13] S. Aumann, S. Donner, J. Fischer, and F. Müller. “Optical coherence tomography (OCT): principle and technical realization”. In: *High Resolution Imaging in Microscopy and Ophthalmology* (2019), pp. 59–85 (cit. on p. 8).

[14] S. Azizi, B. Mustafa, F. Ryan, et al. “Big self-supervised models advance medical image classification”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3478–3488 (cit. on p. 60).

[15] P. Bachman, R. D. Hjelm, and W. Buchwalter. “Learning representations by maximizing mutual information across views”. In: *arXiv preprint arXiv:1906.00910* (2019) (cit. on p. 122).

[16] T. Bai, Z. Zhang, C. Zhao, and X. Luo. “A Novel Pseudo-Labeling Approach for Cell Detection Based on Adaptive Threshold”. In: *International Symposium on Bioinformatics Research and Applications*. Springer. 2021, pp. 254–265 (cit. on p. 43).

[17] W. Bai, C. Chen, G. Tarroni, et al. “Self-supervised learning for cardiac mr image segmentation by anatomical position prediction”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 541–549 (cit. on p. 59).

[18] W. Bai, O. Oktay, M. Sinclair, et al. “Semi-supervised learning for network-based cardiac MR image segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 253–260 (cit. on pp. 43, 72, 75, 78–80).

[19] S. Bakas, M. Reyes, A. Jakab, et al. “Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge”. In: *arXiv preprint arXiv:1811.02629* (2018) (cit. on p. 50).

[20] A. Bardes, J. Ponce, and Y. LeCun. “Vicreg: Variance-invariance-covariance regularization for self-supervised learning”. In: *arXiv preprint arXiv:2105.04906* (2021) (cit. on pp. 58, 59, 122, 123, 127).

[21] C. Baur, S. Albarqouni, and N. Navab. “Semi-supervised deep learning for fully convolutional networks”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 311–319 (cit. on pp. 44, 72, 75, 78).

[22] C. Baur, S. Denner, B. Wiestler, N. Navab, and S. Albarqouni. “Autoencoders for unsupervised anomaly segmentation in brain MR images: a comparative study”. In: *Medical Image Analysis* 69 (2021), p. 101952 (cit. on p. 10).

[23] C. Baur, B. Wiestler, S. Albarqouni, and N. Navab. “Deep autoencoding models for unsupervised anomaly segmentation in brain MR images”. In: *International MICCAI brainlesion workshop*. Springer. 2018, pp. 161–169 (cit. on p. 10).

[24] T. Bdair, B. Wiestler, N. Navab, and S. Albarqouni. “ROAM: Random Layer Mixup for Semi-Supervised Learning in Medical Imaging”. In: *arXiv preprint arXiv:2003.09439* (2020) (cit. on p. 123).

[25] Y. Bengio, A. Courville, and P. Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828 (cit. on pp. 2, 3).

[26] C. Benoit. “Note sur une méthode de résolution des équations normales provenant de l’application de la méthode des moindres carrés à un système d’équations linéaires en nombre inférieur à celui des inconnues (Procédé du Commandant Cholesky)”. In: *Bulletin géodésique* 2.1 (1924), pp. 67–77 (cit. on p. 59).

[27] C. I. Bercea, B. Wiestler, D. Rueckert, and S. Albarqouni. “Federated disentangled representation learning for unsupervised brain anomaly detection”. In: *Nature Machine Intelligence* (2022), pp. 1–11 (cit. on p. 51).

[28] T. Bernecker, A. Peters, C. L. Schlett, et al. “FedNorm: Modality-Based Normalization in Federated Learning for Multi-Modal Liver Segmentation”. In: *arXiv preprint arXiv:2205.11096* (2022) (cit. on pp. 47, 51).

[29] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel. “Mixmatch: A holistic approach to semi-supervised learning”. In: *arXiv preprint arXiv:1905.02249* (2019) (cit. on pp. 67, 68, 74, 75, 123).

[30] C. Bettini, G. Civitarese, and R. Presotto. “Personalized semi-supervised federated learning for human activity recognition”. In: *arXiv preprint arXiv:2104.08094* (2021) (cit. on p. 50).

[31] M Binder, H Kittler, A Seeber, A Steiner, H Pehamberger, and K Wolff. “Epiluminescence microscopy-based classification of pigmented skin lesions using computerized image analysis and an artificial neural network.” In: *Melanoma research* 8.3 (1998), pp. 261–266 (cit. on p. 87).

[32] T Birsan and D. Tiba. “One hundred years since the introduction of the set distance by Dimitrie Pompeiu”. In: *IFIP Conference on System Modeling and Optimization*. Springer. 2005, pp. 35–39 (cit. on p. 25).

[33] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006 (cit. on pp. 14, 15).

[34] A. Blum and T. Mitchell. “Combining labeled and unlabeled data with co-training”. In: *Proceedings of the eleventh annual conference on Computational learning theory*. 1998, pp. 92–100 (cit. on p. 43).

[35] K. Bonawitz, V. Ivanov, B. Kreuter, et al. “Practical secure aggregation for privacy-preserving machine learning”. In: *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1175–1191 (cit. on p. 49).

[36] G. Bortsova, F. Dubost, L. Hogeweg, I. Katramados, and M. de Bruijne. “Semi-supervised medical image segmentation via learning consistency under transformations”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 810–818 (cit. on p. 44).

[37] C. Briggs, Z. Fan, and P. Andras. “Federated learning with hierarchical clustering of local updates to improve training on non-IID data”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–9 (cit. on p. 92).

[38] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. “Signature verification using a" siamese" time delay neural network”. In: *Advances in neural information processing systems* 6 (1993) (cit. on pp. 57, 58, 122, 123).

[39] L. Cai, J. Gao, and D. Zhao. “A review of the application of deep learning in medical image classification and segmentation”. In: *Annals of translational medicine* 8.11 (2020) (cit. on p. 9).

[40] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, and B. Caputo. “Cluster-driven graph federated learning over multiple domains”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2749–2758 (cit. on p. 138).

[41] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar. “Expanding the reach of federated learning by reducing client resource requirements”. In: *arXiv preprint arXiv:1812.07210* (2018) (cit. on p. 49).

[42] S. Candemir, S. Jaeger, K. Palaniappan, et al. “Lung segmentation in chest radiographs using anatomical atlases with nonrigid registration”. In: *IEEE transactions on medical imaging* 33.2 (2013), pp. 577–590 (cit. on p. 9).

[43] Y. Cao, Z. Xie, B. Liu, Y. Lin, Z. Zhang, and H. Hu. “Parametric instance classification for unsupervised visual feature learning”. In: *arXiv preprint arXiv:2006.14618* (2020) (cit. on pp. 58, 122, 123).

[44] G. Campos-do Carmo and M. Ramos-e Silva. “Dermoscopy: basic concepts”. In: *International journal of dermatology* 47.7 (2008), pp. 712–719 (cit. on p. 9).

[45] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. “Deep clustering for unsupervised learning of visual features”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 132–149 (cit. on p. 58).

[46] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. “Unsupervised learning of visual features by contrasting cluster assignments”. In: *arXiv preprint arXiv:2006.09882* (2020) (cit. on pp. 58, 122, 123).

[47] K. Chaitanya, E. Erdil, N. Karani, and E. Konukoglu. “Contrastive learning of global and local features for medical image segmentation with limited annotations”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12546–12558 (cit. on p. 60).

[48] K. Chaitanya, N. Karani, C. F. Baumgartner, A. Becker, O. Donati, and E. Konukoglu. “Semi-supervised and task-driven data augmentation”. In: *International Conference on Information Processing in Medical Imaging*. Springer. 2019, pp. 29–41 (cit. on pp. 67, 123).

[49] O. Chapelle, B. Scholkopf, and A. Zien. “Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]”. In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 542–542 (cit. on pp. 2, 3, 34–37, 39–41, 43, 66).

[50] F. Chen, G. Long, Z. Wu, T. Zhou, and J. Jiang. “Personalized federated learning with graph”. In: *arXiv preprint arXiv:2203.00829* (2022) (cit. on p. 138).

[51] L. Chen, P. Bentley, K. Mori, K. Misawa, M. Fujiwara, and D. Rueckert. “Self-supervised learning for medical image analysis using image context restoration”. In: *Medical image analysis* 58 (2019), p. 101539 (cit. on pp. 52, 59).

[52] S. Chen, G. Bortsova, A. G.-U. Juarez, G. van Tulder, and M. de Bruijne. “Multi-Task Attention-Based Semi-Supervised Learning for Medical Image Segmentation”. In: *arXiv preprint arXiv:1907.12303* (2019) (cit. on p. 45).

[53] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607 (cit. on pp. 57, 58, 60, 122, 123, 127).

[54] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton. “Big self-supervised models are strong semi-supervised learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 22243–22255 (cit. on pp. 52, 122).

[55] X. Chen, L. Yao, T. Zhou, J. Dong, and Y. Zhang. “Momentum contrastive learning for few-shot COVID-19 diagnosis from chest CT images”. In: *Pattern recognition* 113 (2021), p. 107826 (cit. on p. 60).

[56] X. Chen and K. He. “Exploring simple siamese representation learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15750–15758 (cit. on pp. 58, 122, 123).

[57] V. Cheplygina, M. de Bruijne, and J. P. Pluim. “Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis”. In: *Medical image analysis* 54 (2019), pp. 280–296 (cit. on pp. 10, 35).

[58] S. J. Chiu, X. T. Li, P. Nicholas, C. A. Toth, J. A. Izatt, and S. Farsiu. “Automatic segmentation of seven retinal layers in SDOCT images congruent with expert manual segmentation”. In: *Optics express* 18.18 (2010), pp. 19413–19428 (cit. on pp. 1, 32, 63).

[59] A. Chowdhury, J. Rosenthal, J. Waring, and R. Umeton. “Applying self-supervised learning to medicine: review of the state of the art and medical implementations”. In: *Informatics*. Vol. 8. 3. MDPI. 2021, p. 59 (cit. on p. 59).

[60] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. “3D U-Net: learning dense volumetric segmentation from sparse annotation”. In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432 (cit. on p. 22).

[61] W. H. Clark Jr, D. E. Elder, D. Guerry IV, et al. “Model predicting survival in stage I melanoma based on tumor progression”. In: *JNCI: Journal of the National Cancer Institute* 81.24 (1989), pp. 1893–1904 (cit. on p. 87).

[62] A. Coates, A. Ng, and H. Lee. “An analysis of single-layer networks in unsupervised feature learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 215–223 (cit. on p. 127).

[63] N. Codella, V. Rotemberg, P. Tschandl, et al. “Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic)”. In: *arXiv preprint arXiv:1902.03368* (2019) (cit. on p. 97).

[64] P. Coupé, B. Mansencal, M. Clément, et al. “AssemblyNet: A Novel Deep Decision-Making Process for Whole Brain MRI Segmentation”. In: *arXiv preprint arXiv:1906.01862* (2019) (cit. on pp. 32, 63).

[65] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. “Randaugment: Practical automated data augmentation with a reduced search space”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 702–703 (cit. on p. 99).

[66] W. Cui, Y. Liu, Y. Li, et al. “Semi-supervised Brain Lesion Segmentation with an Adapted Mean Teacher Model”. In: *International Conference on Information Processing in Medical Imaging*. Springer. 2019, pp. 554–565 (cit. on pp. 44, 72, 75, 78, 79, 85).

[67] M. Cuturi. “Sinkhorn distances: Lightspeed computation of optimal transport”. In: *Advances in neural information processing systems* 26 (2013) (cit. on p. 58).

[68] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. R. Salakhutdinov. “Good semi-supervised learning that requires a bad gan”. In: *Advances in neural information processing systems*. 2017, pp. 6510–6520 (cit. on p. 42).

[69] E. Darzidehkalani, M. Ghasemi-Rad, and P. van Ooijen. “Federated Learning in Medical Imaging: Part II: Methods, Challenges, and Considerations”. In: *Journal of the American College of Radiology* (2022) (cit. on p. 50).

[70] J. De Fauw, J. R. Ledsam, B. Romera-Paredes, et al. “Clinically applicable deep learning for diagnosis and referral in retinal disease”. In: *Nature medicine* 24.9 (2018), pp. 1342–1350 (cit. on pp. 5, 63).

[71] M. Deeley, A Chen, R Datteri, et al. “Comparison of manual and automatic segmentation methods for brain structures in the presence of space-occupying lesions: a multi-expert study”. In: *Physics in Medicine & Biology* 56.14 (2011), p. 4557 (cit. on pp. 1, 32).

[72] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22 (cit. on p. 41).

[73] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on pp. 28, 123).

[74] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on p. 122).

[75] A. P. Dhawan. *Medical image analysis*. John Wiley & Sons, 2011 (cit. on p. 6).

[76] E. Diao, J. Ding, and V. Tarokh. “SemiFL: Communication efficient semi-supervised federated learning with unlabeled clients”. In: *arXiv preprint arXiv:2106.01432* (2021) (cit. on p. 50).

[77] L. Ding, K. Zhao, X. Zhang, X. Wang, and J. Zhang. “A lightweight U-Net architecture multi-scale convolutional network for pediatric hand bone segmentation in X-ray image”. In: *IEEE Access* 7 (2019), pp. 68436–68445 (cit. on p. 9).

[78] Y. Ding, J. Liu, J. Xiong, and Y. Shi. “Revisiting the evaluation of uncertainty estimation and its application to explore model complexity-uncertainty trade-off”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 4–5 (cit. on pp. 27, 99).

[79] C. Doersch, A. Gupta, and A. A. Efros. “Unsupervised visual representation learning by context prediction”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1422–1430 (cit. on pp. 53, 55).

[80] L. Dora, S. Agrawal, R. Panda, and A. Abraham. “State-of-the-art methods for brain tissue segmentation: A review”. In: *IEEE reviews in biomedical engineering* 10 (2017), pp. 235–249 (cit. on p. 9).

[81] A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020) (cit. on pp. 28, 45).

[82] J. S. Duncan and N. Ayache. “Medical image analysis: Progress over two decades and the challenges ahead”. In: *IEEE transactions on pattern analysis and machine intelligence* 22.1 (2000), pp. 85–106 (cit. on p. 11).

[83] A. A. Duquette, P.-M. Jodoin, O. Bouchot, and A. Lalande. “3D segmentation of abdominal aorta from CT-scan and MR images”. In: *Computerized Medical Imaging and Graphics* 36.4 (2012), pp. 294–303 (cit. on p. 9).

[84] C. Dwork. “Differential privacy: A survey of results”. In: *International conference on theory and applications of models of computation*. Springer. 2008, pp. 1–19 (cit. on p. 48).

[85] Z. Eaton-Rosen, F. Bragman, S. Ourselin, and M. J. Cardoso. “Improving data augmentation for medical image segmentation”. In: *International Conference on Medical Imaging with Deep Learning*. 2018 (cit. on pp. 67, 123).

[86] L. Ericsson, H. Gouk, C. C. Loy, and T. M. Hospedales. “Self-Supervised Representation Learning: Introduction, advances, and challenges”. In: *IEEE Signal Processing Magazine* 39.3 (2022), pp. 42–62 (cit. on pp. 52, 53, 56).

[87] A. Esteva, B. Kuprel, R. A. Novoa, et al. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *nature* 542.7639 (2017), pp. 115–118 (cit. on pp. 32, 87).

[88] T. Falk, D. Mai, R. Bensch, et al. “U-Net: deep learning for cell counting, detection, and morphometry”. In: *Nature methods* 16.1 (2019), pp. 67–70 (cit. on pp. 5, 63).

[89] M. Flores, I. Dayan, H. Roth, et al. “Federated Learning used for predicting outcomes in SARS-COV-2 patients”. In: (2021) (cit. on pp. 51, 88).

[90] S Fralick. “Learning to recognize patterns without a teacher”. In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 57–64 (cit. on p. 34).

[91] M. Fredrikson, S. Jha, and T. Ristenpart. “Model inversion attacks that exploit confidence information and basic countermeasures”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 1322–1333 (cit. on p. 90).

[92] Y. Fu, Y. Lei, T. Wang, W. J. Curran, T. Liu, and X. Yang. “Deep learning in medical image registration: a review”. In: *Physics in Medicine & Biology* 65.20 (2020), 20TR01 (cit. on p. 10).

[93] P.-A. Ganaye, M. Sdika, and H. Benoit-Cattin. “Semi-supervised learning for segmentation under semantic constraint”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 595–602 (cit. on pp. 44, 138).

[94] Y. Geifman and R. El-Yaniv. “Selective classification for deep neural networks”. In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 25, 99).

[95] N. Gessert, M. Nielsen, M. Shaikh, R. Werner, and A. Schlaefer. “Skin lesion classification using ensembles of multi-resolution EfficientNets with meta data”. In: *MethodsX* 7 (2020), p. 100864 (cit. on pp. 32, 87).

[96] R. C. Geyer, T. Klein, and M. Nabi. “Differentially private federated learning: A client level perspective”. In: *arXiv preprint arXiv:1712.07557* (2017) (cit. on p. 48).

[97] M. Ghorbani, M. Bahrami, A. Kazi, M. Soleymani Baghshah, H. R. Rabiee, and N. Navab. “GKD: Semi-supervised Graph Knowledge Distillation for Graph-Independent Inference”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2021, pp. 709–718 (cit. on p. 44).

[98] M. Ghorbani, A. Kazi, M. S. Baghshah, H. R. Rabiee, and N. Navab. “Ra-gcn: Graph convolutional network for disease prediction problems with imbalanced data”. In: *Medical Image Analysis* 75 (2022), p. 102272 (cit. on p. 44).

[99] S. Gidaris, P. Singh, and N. Komodakis. “Unsupervised representation learning by predicting image rotations”. In: *arXiv preprint arXiv:1803.07728* (2018) (cit. on pp. 53, 55, 60).

[100] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256 (cit. on pp. 18, 72).

[101] I. Goodfellow, Y. Bengio, and A. Courville. *Softmax units for multinoulli output distributions. Deep Learning*. 2018 (cit. on p. 15).

[102] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680 (cit. on pp. 41, 45, 56).

[103] Y. Grandvalet and Y. Bengio. “Semi-supervised learning by entropy minimization”. In: *Advances in neural information processing systems*. 2005, pp. 529–536 (cit. on pp. 37, 43).

[104] J.-B. Grill, F. Strub, F. Altché, et al. “Bootstrap your own latent: A new approach to self-supervised learning”. In: *arXiv preprint arXiv:2006.07733* (2020) (cit. on pp. 58, 60, 122, 123, 127).

[105] Z. Gu, J. Cheng, H. Fu, et al. “CE-Net: context encoder network for 2D medical image segmentation”. In: *IEEE transactions on medical imaging* 38.10 (2019), pp. 2281–2292 (cit. on pp. 32, 63).

[106] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. “On calibration of modern neural networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330 (cit. on pp. 27, 68, 99).

[107] M. Gutmann and A. Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 297–304 (cit. on p. 57).

[108] C. He, S. Li, J. So, et al. “Fedml: A research library and benchmark for federated machine learning”. In: *arXiv preprint arXiv:2007.13518* (2020) (cit. on p. 50).

[109] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. “Momentum contrast for unsupervised visual representation learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9729–9738 (cit. on pp. 57, 58, 60, 122, 123).

[110] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on pp. 21, 28, 128).

[111] Y. He, A. Carass, Y. Liu, et al. “Fully convolutional boundary regression for retina OCT segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 120–128 (cit. on p. 9).

[112] O. Henaff. “Data-efficient image recognition with contrastive predictive coding”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4182–4192 (cit. on p. 57).

[113] I. Higgins, L. Matthey, A. Pal, et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework.” In: *Iclr* 2.5 (2017), p. 6 (cit. on p. 138).

[114] D. L. Hill, P. G. Batchelor, M. Holden, and D. J. Hawkes. “Medical image registration”. In: *Physics in medicine & biology* 46.3 (2001), R1 (cit. on p. 10).

[115] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, et al. “Learning deep representations by mutual information estimation and maximization”. In: *arXiv preprint arXiv:1808.06670* (2018) (cit. on p. 57).

[116] S. Hochreiter. “Untersuchungen zu dynamischen neuronalen Netzen”. In: *Diploma, Technische Universität München* 91.1 (1991) (cit. on p. 21).

[117] T. C. Hollon, B. Pandian, A. R. Adapa, et al. “Near real-time intraoperative brain tumor diagnosis using stimulated Raman histology and deep neural networks”. In: *Nature medicine* 26.1 (2020), pp. 52–58 (cit. on pp. 5, 63).

[118] O. G. Holmberg, N. D. Köhler, T. Martins, et al. “Self-supervised retinal thickness prediction enables deep learning from unlabelled data to boost classification of diabetic retinopathy”. In: *Nature Machine Intelligence* 2.11 (2020), pp. 719–726 (cit. on p. 60).

[119] T.-M. H. Hsu, H. Qi, and M. Brown. “Federated visual classification with real-world data distribution”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 76–92 (cit. on pp. 47, 98).

[120] S. Hu, E. A. Hoffman, and J. M. Reinhardt. “Automatic lung segmentation for accurate quantitation of volumetric X-ray CT images”. In: *IEEE transactions on medical imaging* 20.6 (2001), pp. 490–498 (cit. on pp. 33, 63).

[121] T. Hua, W. Wang, Z. Xue, Y. Wang, S. Ren, and H. Zhao. “On Feature Decorrelation in Self-Supervised Learning”. In: *arXiv preprint arXiv:2105.00470* (2021) (cit. on pp. 59, 123).

[122] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. “Densely connected convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708 (cit. on p. 28).

[123] L. Huang, D. Yang, B. Lang, and J. Deng. “Decorrelated batch normalization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 791–800 (cit. on p. 59).

[124] Q. Huang, Y. Luo, and Q. Zhang. “Breast ultrasound image segmentation: a survey”. In: *International journal of computer assisted radiology and surgery* 12.3 (2017), pp. 493–507 (cit. on p. 9).

[125] I. A. M. Ikhsan, A. Hussain, M. A. Zulkifley, N. M. Tahir, and A. Mustapha. “An analysis of x-ray image enhancement methods for vertebral bone segmentation”. In: *2014 IEEE 10th International Colloquium on Signal Processing and its Applications*. IEEE. 2014, pp. 208–211 (cit. on p. 9).

[126] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456 (cit. on p. 28).

[127] F. Isensee, P. F. Jaeger, S. A. Kohl, J. Petersen, and K. H. Maier-Hein. “nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation”. In: *Nature methods* 18.2 (2021), pp. 203–211 (cit. on pp. 32, 63).

[128] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto. “Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data”. In: *arXiv preprint arXiv:2008.06180* (2020) (cit. on p. 50).

[129] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. “A survey on contrastive self-supervised learning”. In: *Technologies* 9.1 (2020), p. 2 (cit. on pp. 52, 53, 56).

[130] W. Jeong, J. Yoon, E. Yang, and S. J. Hwang. “Federated semi-supervised learning with inter-client consistency & disjoint learning”. In: *arXiv preprint arXiv:2006.12097* (2020) (cit. on pp. 3, 49, 50, 90, 98–101, 104–107, 110, 111).

[131] A. Jiménez-Sánchez, M. Tardy, M. A. G. Ballester, D. Mateus, and G. Piella. “Memory-aware curriculum federated learning for breast cancer classification”. In: *arXiv preprint arXiv:2107.02504* (2021) (cit. on p. 51).

[132] L. Jing and Y. Tian. “Self-supervised visual feature learning with deep neural networks: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020), pp. 4037–4058 (cit. on pp. 3, 52, 53, 55, 56).

[133] D. Joksas, P Freitas, Z Chai, et al. “Committee machines—a universal method to deal with non-idealities in memristor-based neural networks”. In: *Nature communications* 11.1 (2020), pp. 1–10 (cit. on pp. 88, 91, 92).

[134] M. Jun, W. Yixin, A. Xingle, et al. “Towards Efficient COVID-19 CT Annotation: A Benchmark for Lung and Infection Segmentation”. In: *arXiv preprint arXiv:2004.12537* (2020) (cit. on p. 71).

[135] W. Jung, S. Park, K.-H. Jung, and S. I. Hwang. “Prostate cancer segmentation using manifold mixup U-Net”. In: *International Conference on Medical Imaging with Deep Learning–Extended Abstract Track*. 2019 (cit. on pp. 67, 123).

[136] Y. Kabir, M. Dojat, B. Scherrer, F. Forbes, and C. Garbay. “Multimodal MRI segmentation of ischemic stroke lesions”. In: *2007 29th annual international conference of the IEEE engineering in medicine and biology society*. IEEE. 2007, pp. 1595–1598 (cit. on p. 9).

[137] P. Kairouz, H. B. McMahan, B. Avent, et al. “Advances and open problems in federated learning”. In: *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210 (cit. on p. 45).

[138] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren. “Secure, privacy-preserving and federated machine learning in medical imaging”. In: *Nature Machine Intelligence* 2.6 (2020), pp. 305–311 (cit. on p. 88).

[139] K. Kamnitsas, D. C. Castro, L. L. Folgoc, et al. “Semi-supervised learning via compact latent space clustering”. In: *arXiv preprint arXiv:1806.02679* (2018) (cit. on p. 39).

[140] S. A. Kamran, K. F. Hossain, A. Tavakkoli, S. L. Zuckerbrod, and S. A. Baker. “Vtgan: Semi-supervised retinal image synthesis and disease prediction using vision transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3235–3245 (cit. on pp. 45, 65).

[141] Y. Kang, Y. Liu, and T. Chen. “Fedmvt: Semi-supervised vertical federated learning with multiview training”. In: *arXiv preprint arXiv:2008.10838* (2020) (cit. on p. 50).

[142] H. Kassem, D. Alapatt, P. Mascagni, A. Consortium, A. Karargyris, and N. Padoy. “Federated Cycling (FedCy): Semi-supervised Federated Learning of Surgical Phases”. In: *arXiv preprint arXiv:2203.07345* (2022) (cit. on p. 52).

[143] J. Kawahara, S. Daneshvar, G. Argenziano, and G. Hamarneh. “Seven-point checklist and skin lesion classification using multitask multimodal neural nets”. In: *IEEE Journal of Biomedical and Health Informatics* 23.2 (2019), pp. 538–546 (cit. on p. 97).

[144] D. N. Kennedy, C. Haselgrove, S. M. Hodge, P. S. Rane, N. Makris, and J. A. Frazier. *CANDIShare: a resource for pediatric neuroimaging data*. 2012 (cit. on p. 71).

[145] J. Ker, L. Wang, J. Rao, and T. Lim. “Deep learning applications in medical image analysis”. In: *Ieee Access* 6 (2017), pp. 9375–9389 (cit. on p. 11).

[146] P. Kickingereder, F. Isensee, I. Tursunova, et al. “Automated quantitative tumour response assessment of MRI in neuro-oncology with artificial neural networks: a multicentre, retrospective study”. In: *The Lancet Oncology* 20.5 (2019), pp. 728–740 (cit. on pp. 5, 63).

[147] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on pp. 16, 46, 72, 98, 128).

[148] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on p. 41).

[149] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. “Semi-supervised learning with deep generative models”. In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 41).

[150] T. N. Kipf and M. Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016) (cit. on p. 40).

[151] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. “Federated learning: Strategies for improving communication efficiency”. In: *arXiv preprint arXiv:1610.05492* (2016) (cit. on p. 49).

[152] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. “Similarity of neural network representations revisited”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3519–3529 (cit. on p. 92).

[153] E. Korot, Z. Guan, D. Ferraz, et al. “Code-free deep learning for multi-modality medical image classification”. In: *Nature Machine Intelligence* 3.4 (2021), pp. 288–298 (cit. on p. 9).

[154] S. B. Kotsiantis, I. Zaharakis, P Pintelas, et al. “Supervised machine learning: A review of classification techniques”. In: *Emerging artificial intelligence applications in computer engineering* 160.1 (2007), pp. 3–24 (cit. on p. 31).

[155] P. Krähenbühl and V. Koltun. “Efficient inference in fully connected crfs with gaussian edge potentials”. In: *Advances in neural information processing systems* 24 (2011) (cit. on pp. 43, 138).

[156] M. A. Kramer. “Nonlinear principal component analysis using autoassociative neural networks”. In: *AIChE journal* 37.2 (1991), pp. 233–243 (cit. on p. 56).

[157] R. Krishnan, P. Rajpurkar, and E. J. Topol. “Self-supervised learning in medicine and healthcare”. In: *Nature Biomedical Engineering* (2022), pp. 1–7 (cit. on p. 59).

[158] A. Krizhevsky, G. Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009) (cit. on p. 127).

[159] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012) (cit. on pp. 21, 28).

[160] S. Laine and T. Aila. “Temporal ensembling for semi-supervised learning”. In: *arXiv preprint arXiv:1610.02242* (2016) (cit. on p. 38).

[161] B Landman and S Warfield. “MICCAI 2012 workshop on multi-atlas labeling”. In: *Medical image computing and computer assisted intervention conference*. 2012 (cit. on p. 71).

[162] B. A. Landman and S. K. Warfield. *MICCAI 2012: Workshop on Multi-atlas Labeling*. éditeur non identifié, 2019 (cit. on p. 71).

[163] Y. Le and X. Yang. “Tiny imagenet visual recognition challenge”. In: *CS 231N* 7.7 (2015), p. 3 (cit. on p. 127).

[164] Y. LeCun, B. Boser, J. S. Denker, et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551 (cit. on p. 21).

[165] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 18).

[166] C. Ledig, L. Theis, F. Huszár, et al. “Photo-realistic single image super-resolution using a generative adversarial network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4681–4690 (cit. on p. 56).

[167] H. M. Levin, G. V. Glass, and G. R. Meister. “Cost-effectiveness of computer-assisted instruction”. In: *Evaluation review* 11.1 (1987), pp. 50–72 (cit. on p. 91).

[168] H. Li, Y. Wang, R. Wan, S. Wang, T.-Q. Li, and A. C. Kot. “Domain Generalization for Medical Imaging Classification with Linear-Dependency Regularization”. In: *arXiv preprint arXiv:2009.12829* (2020) (cit. on pp. 32, 87).

[169] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen. “Medical image classification with convolutional neural network”. In: *2014 13th international conference on control automation robotics & vision (ICARCV)*. IEEE. 2014, pp. 844–848 (cit. on p. 9).

[170] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. “Federated learning: Challenges, methods, and future directions”. In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60 (cit. on pp. 45, 47–49, 88).

[171] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. “Federated optimization in heterogeneous networks”. In: *arXiv preprint arXiv:1812.06127* (2018) (cit. on p. 117).

[172] W. Li, F. Milletarì, D. Xu, et al. “Privacy-preserving federated brain tumour segmentation”. In: *International workshop on machine learning in medical imaging*. Springer. 2019, pp. 133–141 (cit. on pp. 46, 50).

[173] X. Li, L. Yu, H. Chen, C.-W. Fu, L. Xing, and P.-A. Heng. “Transformation-Consistent Self-Ensembling Model for Semisupervised Medical Image Segmentation”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2020) (cit. on p. 44).

[174] X. Li, Y. Gu, N. Dvornek, L. H. Staib, P. Ventola, and J. S. Duncan. “Multi-site fMRI analysis using privacy-preserving federated learning and domain adaptation: ABIDE results”. In: *Medical Image Analysis* 65 (2020), p. 101765 (cit. on p. 51).

[175] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou. “Fedbn: Federated learning on non-iid features via local batch normalization”. In: *arXiv preprint arXiv:2102.07623* (2021) (cit. on pp. 47, 48, 117).

[176] Y. Li, L. Luo, H. Lin, H. Chen, and P.-A. Heng. “Dual-consistency semi-supervised learning with uncertainty quantification for COVID-19 lesion segmentation from CT images”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2021, pp. 199–209 (cit. on pp. 44, 65).

[177] Y. Li, J. Chen, X. Xie, K. Ma, and Y. Zheng. “Self-loop uncertainty: A novel pseudo-label for semi-supervised medical image segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2020, pp. 614–623 (cit. on pp. 44, 59).

[178] X. Liang, Y. Lin, H. Fu, L. Zhu, and X. Li. “RSCFed: Random Sampling Consensus Federated Semi-supervised Learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10154–10163 (cit. on p. 51).

[179] H. Lin, J. Lou, L. Xiong, and C. Shahabi. “Semifed: Semi-supervised federated learning with consistency and pseudo-labeling”. In: *arXiv preprint arXiv:2108.09412* (2021) (cit. on p. 50).

[180] G. Litjens, T. Kooi, B. E. Bejnordi, et al. “A survey on deep learning in medical image analysis”. In: *Medical image analysis* 42 (2017), pp. 60–88 (cit. on pp. 9, 10).

[181] J. Liu, J. Huang, Y. Zhou, et al. “From distributed machine learning to federated learning: A survey”. In: *Knowledge and Information Systems* (2022), pp. 1–33 (cit. on pp. 45, 46).

[182] Q. Liu, H. Yang, Q. Dou, and P.-A. Heng. “Federated Semi-supervised Medical Image Classification via Inter-client Relation Matching”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2021, pp. 325–335 (cit. on pp. 50, 51, 89, 90, 98, 112, 138).

[183] X. Liu, F. Zhang, Z. Hou, et al. “Self-supervised learning: Generative or contrastive”. In: *IEEE Transactions on Knowledge and Data Engineering* (2021) (cit. on pp. 52, 53, 56).

[184] X. Liu, L. Faes, A. U. Kale, et al. “A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis”. In: *The lancet digital health* 1.6 (2019), e271–e297 (cit. on p. 10).

[185] Y. Liu, M. Jin, S. Pan, et al. “Graph self-supervised learning: A survey”. In: *IEEE Transactions on Knowledge and Data Engineering* (2022) (cit. on p. 52).

[186] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440 (cit. on p. 22).

[187] A. R. Lopez, X. Giro-i Nieto, J. Burdick, and O. Marques. “Skin lesion classification from dermoscopic images using deep learning techniques”. In: *2017 13th IASTED international conference on biomedical engineering (BioMed)*. IEEE. 2017, pp. 49–54 (cit. on pp. 32, 87).

[188] L. Maier-Hein, M. Eisenmann, A. Reinke, et al. “Why rankings of biomedical image analysis competitions should be interpreted with care”. In: *Nature communications* 9.1 (2018), pp. 1–13 (cit. on p. 63).

[189] J. A. Maintz and M. A. Viergever. “A survey of medical image registration”. In: *Medical image analysis* 2.1 (1998), pp. 1–36 (cit. on p. 10).

[190] C. R. Maurer and J. M. Fitzpatrick. “A review of medical image registration”. In: *Interactive image-guided neurosurgery* 1 (1993), pp. 17–44 (cit. on p. 10).

[191] L. McInnes, J. Healy, and J. Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018) (cit. on p. 131).

[192] G. McLachlan. *Discriminant analysis and statistical pattern recognition*. Vol. 544. John Wiley & Sons, 2004 (cit. on p. 41).

[193] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. “Communication-efficient learning of deep networks from decentralized data”. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282 (cit. on pp. 2, 3, 45–47, 88, 89, 91, 116).

[194] A. Meyer, S. Ghosh, D. Schindele, et al. “Uncertainty-aware temporal self-learning (UATS): Semi-supervised learning for segmentation of prostate zones and beyond”. In: *Artificial Intelligence in Medicine* 116 (2021), p. 102073 (cit. on p. 44).

[195] A. Meyer-Bäse, A. Meyer-Baese, and V. J. Schmid. *Pattern Recognition and Signal Analysis in Medical Imaging*. Academic Press, 2004 (cit. on pp. 6, 8).

[196] F. Milletari, N. Navab, and S.-A. Ahmadi. “V-net: Fully convolutional neural networks for volumetric medical image segmentation”. In: *2016 fourth international conference on 3D vision (3DV)*. IEEE. 2016, pp. 565–571 (cit. on p. 22).

[197] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii. “Virtual adversarial training: a regularization method for supervised and semi-supervised learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.8 (2018), pp. 1979–1993 (cit. on p. 39).

[198] V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Icml*. 2010 (cit. on pp. 15, 19).

[199] D. Ng, X. Lan, M. M.-S. Yao, W. P. Chan, and M. Feng. “Federated learning: a collaborative effort to achieve better medical imaging models for individual sites that have small labelled datasets”. In: *Quantitative Imaging in Medicine and Surgery* 11.2 (2021), p. 852 (cit. on p. 50).

[200] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, et al. “Federated learning for smart healthcare: A survey”. In: *ACM Computing Surveys (CSUR)* 55.3 (2022), pp. 1–37 (cit. on pp. 48, 50).

[201] D. Nie, Y. Gao, L. Wang, and D. Shen. “ASDNet: Attention based semi-supervised deep networks for medical image segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 370–378 (cit. on p. 45).

[202] S. Nikolov, S. Blackwell, A. Zverovitch, et al. “Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy”. In: *arXiv preprint arXiv:1809.04430* (2018) (cit. on pp. 5, 63).

[203] M. Noroozi and P. Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles”. In: *European conference on computer vision*. Springer. 2016, pp. 69–84 (cit. on pp. 53, 55, 60).

[204] A. Odena. “Semi-supervised learning with generative adversarial networks”. In: *arXiv preprint arXiv:1606.01583* (2016) (cit. on p. 41).

[205] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow. “Realistic evaluation of deep semi-supervised learning algorithms”. In: *Advances in neural information processing systems* 31 (2018) (cit. on pp. 42, 43, 64, 70, 76, 84, 85).

[206] A. v. d. Oord, Y. Li, and O. Vinyals. “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748* (2018) (cit. on pp. 3, 57, 122).

[207] T. Orekondy, S. J. Oh, Y. Zhang, B. Schiele, and M. Fritz. “Gradient-leaks: Understanding and controlling deanonymization in federated learning”. In: *arXiv preprint arXiv:1805.05838* (2018) (cit. on pp. 90, 116).

[208] A. G. Pacheco, G. R. Lima, A. S. Salomão, et al. “PAD-UFES-20: a skin lesion benchmark composed of patient data and clinical images collected from smartphones”. In: *arXiv preprint arXiv:2007.00478* (2020) (cit. on p. 97).

[209] E. Panfilov, A. Tiulpin, S. Klein, M. T. Nieminen, and S. Saarakkala. “Improving Robustness of Deep Learning Based Knee MRI Segmentation: Mixup and Adversarial Domain Adaptation”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019, pp. 0–0 (cit. on pp. 67, 123).

[210] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. “Context encoders: Feature learning by inpainting”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544 (cit. on pp. 53, 56).

[211] C. Petitjean and J.-N. Dacher. “A review of segmentation methods in short axis cardiac MR images”. In: *Medical image analysis* 15.2 (2011), pp. 169–184 (cit. on p. 9).

[212] D. L. Pham, C. Xu, and J. L. Prince. “A survey of current methods in medical image segmentation”. In: *Annual review of biomedical engineering* 2.3 (2000), pp. 315–337 (cit. on p. 11).

[213] D. L. Pham, C. Xu, and J. L. Prince. “Current methods in medical image segmentation”. In: *Annual review of biomedical engineering* 2.1 (2000), pp. 315–337 (cit. on pp. 5, 63).

[214] N. Qian. “On the momentum term in gradient descent learning algorithms”. In: *Neural networks* 12.1 (1999), pp. 145–151 (cit. on p. 16).

[215] P. Regulation. “Regulation (EU) 2016/679 of the European Parliament and of the Council”. In: *Regulation (eu)* 679 (2016), p. 2016 (cit. on p. 45).

[216] N. Rieke, J. Hancox, W. Li, et al. “The future of digital health with federated learning”. In: *NPJ digital medicine* 3.1 (2020), pp. 1–7 (cit. on pp. 1, 45, 48, 50, 88).

[217] R. L. Rivest, L. Adleman, M. L. Dertouzos, et al. “On data banks and privacy homomorphisms”. In: *Foundations of secure computation* 4.11 (1978), pp. 169–180 (cit. on p. 49).

[218] H. Robbins and S. Monro. “A stochastic approximation method”. In: *The annals of mathematical statistics* (1951), pp. 400–407 (cit. on pp. 16, 46).

[219] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*. Vol. 317. Springer Science & Business Media, 2009 (cit. on p. 25).

[220] H. W. Rogers, M. A. Weinstock, A. R. Harris, et al. “Incidence estimate of nonmelanoma skin cancer in the United States, 2006”. In: *Archives of dermatology* 146.3 (2010), pp. 283–287 (cit. on p. 87).

[221] T. Rohlfing. “Image similarity and tissue overlaps as surrogates for image registration accuracy: widely used but unreliable”. In: *IEEE transactions on medical imaging* 31.2 (2011), pp. 153–163 (cit. on p. 71).

[222] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241 (cit. on pp. 22, 72, 76).

[223] T. Ross, D. Zimmerer, A. Vemuri, et al. “Exploiting the potential of unlabeled endoscopic video data with self-supervised learning”. In: *International journal of computer assisted radiology and surgery* 13.6 (2018), pp. 925–933 (cit. on p. 59).

[224] V. Rotemberg, N. Kurtansky, B. Betz-Stablein, et al. “A patient-centric dataset of images and metadata for identifying melanomas using clinical context”. In: *Scientific data* 8.1 (2021), pp. 1–8 (cit. on p. 97).

[225] H. R. Roth, K. Chang, P. Singh, et al. “Federated Learning for Breast Density Classification: A Real-World Implementation”. In: *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*. Springer, 2020, pp. 181–191 (cit. on pp. 51, 88).

[226] A. G. Roy, S. Conjeti, N. Navab, C. Wachinger, A. D. N. Initiative, et al. “QuickNAT: A fully convolutional network for quick and accurate segmentation of neuroanatomy”. In: *NeuroImage* 186 (2019), pp. 713–727 (cit. on pp. 9, 32, 63, 76).

[227] S. Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on p. 16).

[228] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536 (cit. on p. 15).

[229] O. Russakovsky, J. Deng, H. Su, et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115.3 (2015), pp. 211–252 (cit. on p. 98).

[230] P. K. Sahoo, S. Soltani, and A. K. Wong. “A survey of thresholding techniques”. In: *Computer vision, graphics, and image processing* 41.2 (1988), pp. 233–260 (cit. on p. 11).

[231] M. Sajjadi, M. Javanmardi, and T. Tasdizen. “Regularization with stochastic transformations and perturbations for deep semi-supervised learning”. In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 37).

[232] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. “Improved techniques for training gans”. In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 41).

[233] K. V. Sarma, S. Harmon, T. Sanford, et al. “Federated learning improves site performance in multicenter deep learning without data sharing”. In: *Journal of the American Medical Informatics Association* (2021) (cit. on pp. 51, 88).

[234] F. Sattler, T. Korjakow, R. Rischke, and W. Samek. “Fedaux: Leveraging unlabeled auxiliary data in federated learning”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2021) (cit. on p. 50).

[235] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek. “Robust and communication-efficient federated learning from non-iid data”. In: *IEEE transactions on neural networks and learning systems* 31.9 (2019), pp. 3400–3413 (cit. on p. 49).

[236] T Schindewolf, W. Stolz, R. Albert, W. Abmayr, and H. Harms. “Classification of melanocytic lesions with color and texture analysis using digital image processing.” In: *Analytical and Quantitative Cytology and Histology* 15.1 (1993), pp. 1–11 (cit. on p. 87).

[237] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. “Unsupervised anomaly detection with generative adversarial networks to guide marker discovery”. In: *International conference on information processing in medical imaging*. Springer. 2017, pp. 146–157 (cit. on p. 10).

[238] H. Scudder. “Probability of error of some adaptive pattern-recognition machines”. In: *IEEE Transactions on Information Theory* 11.3 (1965), pp. 363–371 (cit. on p. 34).

[239] M. P. Sendak, J. D’Arcy, S. Kashyap, et al. “A path for translation of machine learning products into healthcare delivery”. In: *EMJ Innov* 10 (2020), pp. 19–00172 (cit. on p. 10).

[240] R. G. Sepuya, E. T. Dozeman, J. E. Prittie, A. J. Fischetti, and J. G. Weltman. “Comparing diagnostic findings and cost of whole body computed tomography to traditional diagnostic imaging in polytrauma patients”. In: *Journal of Veterinary Emergency and Critical Care* 32.3 (2022), pp. 334–340 (cit. on pp. 1, 32).

[241] L. Shao, F. Zhu, and X. Li. “Transfer learning for visual categorization: A survey”. In: *IEEE transactions on neural networks and learning systems* 26.5 (2014), pp. 1019–1034 (cit. on p. 28).

[242] N. Sharma and L. M. Aggarwal. “Automated medical image segmentation techniques”. In: *Journal of medical physics/Association of Medical Physicists of India* 35.1 (2010), p. 3 (cit. on pp. 5, 63).

[243] M. J. Sheller, B. Edwards, G. A. Reina, et al. “Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data”. In: *Scientific reports* 10.1 (2020), pp. 1–12 (cit. on p. 48).

[244] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas. “Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation”. In: *International MICCAI Brainlesion Workshop*. Springer. 2018, pp. 92–104 (cit. on pp. 46, 50).

[245] D. Shen, G. Wu, and H.-I. Suk. “Deep learning in medical image analysis”. In: *Annual review of biomedical engineering* 19 (2017), p. 221 (cit. on pp. 12, 32, 33).

[246] C. Shorten and T. M. Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of big data* 6.1 (2019), pp. 1–48 (cit. on p. 29).

[247] E. Shortliffe. *Computer-based medical consultations: MYCIN*. Vol. 2. Elsevier, 2012 (cit. on p. 10).

[248] S. Shurrab and R. Duwairi. “Self-supervised learning methods and applications in medical imaging analysis: A survey”. In: *PeerJ Computer Science* 8 (2022), e1045 (cit. on pp. 52, 53, 55, 56, 59).

[249] R. L. Siegel. *Cancer Statistics, 2021. Published early online January 12, 2021 in CA Cancer Journal for Clinicians. MPH, American Cancer Society, Atlanta, Ga.* 2021 (cit. on p. 87).

[250] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on pp. 21, 28).

[251] D. I. M. Size and D. I. M. S. Growth. “Share & Trends Analysis Report By Product (Titanium Implants, Zirconium Implants), By Region (North America, Europe, Asia Pacific, Latin America, MEA), And Segment Forecasts, 2018-2024”. In: *Personalized Medicine Market Analysis By Product And Segment Forecasts To 2022* (2018) (cit. on p. 5).

[252] K. Sohn, D. Berthelot, C.-L. Li, et al. “Fixmatch: Simplifying semi-supervised learning with consistency and confidence”. In: *arXiv preprint arXiv:2001.07685* (2020) (cit. on pp. 89, 91, 98).

[253] T. A. Sorensen. “A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons”. In: *Biol. Skar.* 5 (1948), pp. 1–34 (cit. on p. 24).

[254] H. Sowrirajan, J. Yang, A. Y. Ng, and P. Rajpurkar. “Moco pretraining improves representation and transferability of chest x-ray models”. In: *Medical Imaging with Deep Learning*. PMLR. 2021, pp. 728–744 (cit. on p. 60).

[255] A. Srinivas, M. Laskin, and P. Abbeel. “Curl: Contrastive unsupervised representations for reinforcement learning”. In: *arXiv preprint arXiv:2004.04136* (2020) (cit. on p. 122).

[256] A. Sriram, M. Muckley, K. Sinha, et al. “Covid-19 prognosis via self-supervised representation learning and multi-image prediction”. In: *arXiv preprint arXiv:2101.04909* (2021) (cit. on p. 60).

[257] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958 (cit. on p. 28).

[258] P. Suetens. *Fundamentals of medical imaging*. Cambridge university press, 2017 (cit. on p. 5).

[259] 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*. 2016, pp. 2818–2826 (cit. on p. 28).

[260] A. Taleb, C. Lippert, T. Klein, and M. Nabi. “Multimodal self-supervised learning for medical image analysis”. In: *International Conference on Information Processing in Medical Imaging*. Springer. 2021, pp. 661–673 (cit. on p. 59).

[261] M. Tan and Q. Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114 (cit. on p. 98).

[262] X. Tao, Y. Li, W. Zhou, K. Ma, and Y. Zheng. “Revisiting Rubik’s cube: self-supervised learning with volume-wise transformation for 3D medical image segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2020, pp. 238–248 (cit. on p. 60).

[263] A. Tarvainen and H. Valpola. “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results”. In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 38, 44).

[264] Y. Tian, O. J. Henaff, and A. v. d. Oord. “Divide and Contrast: Self-supervised Learning from Uncurated Data”. In: *arXiv preprint arXiv:2105.08054* (2021) (cit. on pp. 122, 123).

[265] Y. Tian, D. Krishnan, and P. Isola. “Contrastive multiview coding”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer. 2020, pp. 776–794 (cit. on p. 122).

[266] E. J. Topol. “High-performance medicine: the convergence of human and artificial intelligence”. In: *Nature medicine* 25.1 (2019), pp. 44–56 (cit. on p. 10).

[267] K. J. Topping. “Trends in peer learning”. In: *Educational psychology* 25.6 (2005), pp. 631–645 (cit. on pp. 3, 88, 90–93).

[268] V. Tresp. “Committee machines”. In: *Handbook for neural network signal processing* (2001), pp. 1–18 (cit. on pp. 3, 88, 90–92).

[269] P. Tschandl, N. Codella, B. N. Akay, et al. “Comparison of the accuracy of human readers versus machine-learning algorithms for pigmented skin lesion classification: an open, web-based, international, diagnostic study”. In: *The Lancet Oncology* 20.7 (2019), pp. 938–947 (cit. on pp. 32, 87).

[270] P. Tschandl, C. Rosendahl, and H. Kittler. “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions”. In: *Scientific data* 5.1 (2018), pp. 1–9 (cit. on p. 97).

[271] M. Tschannen, O. Bachem, and M. Lucic. “Recent advances in autoencoder-based representation learning”. In: *arXiv preprint arXiv:1812.05069* (2018) (cit. on pp. 53, 56).

[272] J. E. Van Engelen and H. H. Hoos. “A survey on semi-supervised learning”. In: *Machine Learning* 109.2 (2020), pp. 373–440 (cit. on pp. 35–37).

[273] W. G. Van Panhuis, P. Paul, C. Emerson, et al. “A systematic review of barriers to data sharing in public health”. In: *BMC public health* 14.1 (2014), pp. 1–9 (cit. on p. 45).

[274] A. Vaswani, N. Shazeer, N. Parmar, et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 45).

[275] V. Verma, A. Lamb, C. Beckham, et al. “Manifold mixup: Better representations by interpolating hidden states”. In: *International Conference on Machine Learning*. 2019, pp. 6438–6447 (cit. on pp. 65, 67, 68, 83, 123).

[276] M. A. Viergever, J. A. Maintz, S. Klein, K. Murphy, M. Staring, and J. P. Pluim. *A survey of medical image registration–under review*. 2016 (cit. on p. 10).

[277] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103 (cit. on p. 56).

[278] Y. N. T. Vu, R. Wang, N. Balachandar, C. Liu, A. Y. Ng, and P. Rajpurkar. “Medaug: Contrastive learning leveraging patient metadata improves representations for chest x-ray interpretation”. In: *Machine Learning for Healthcare Conference*. PMLR. 2021, pp. 755–769 (cit. on p. 60).

[279] B. Wang, A. Li, H. Li, and Y. Chen. “Graphfl: A federated learning framework for semi-supervised node classification on graphs”. In: *arXiv preprint arXiv:2012.04187* (2020) (cit. on p. 50).

[280] F. Wang and H. Liu. “Understanding the behaviour of contrastive loss”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 2495–2504 (cit. on p. 57).

[281] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni. “Federated learning with matched averaging”. In: *arXiv preprint arXiv:2002.06440* (2020) (cit. on pp. 47, 48).

[282] W. Wang, D. Liang, Q. Chen, et al. “Medical image classification using deep learning”. In: *Deep learning in healthcare*. Springer, 2020, pp. 33–51 (cit. on p. 9).

[283] X. Wang, H. Chen, H. Xiang, H. Lin, X. Lin, and P.-A. Heng. “Deep virtual adversarial self-training with consistency regularization for semi-supervised medical image classification”. In: *Medical image analysis* 70 (2021), p. 102010 (cit. on p. 44).

[284] K. Weiss, T. M. Khoshgoftaar, and D. Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), pp. 1–40 (cit. on p. 28).

[285] Wikipedia contributors. *CT scan — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-June-2022]. 2022 (cit. on p. 6).

[286] Wikipedia contributors. *Medical ultrasound — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Medical_ultrasound&oldid=1089398967`. [Online; accessed 21-June-2022]. 2022 (cit. on p. 6).

[287] H. Wu, S. Chen, G. Chen, W. Wang, B. Lei, and Z. Wen. “FAT-Net: Feature adaptive transformers for automated skin lesion segmentation”. In: *Medical Image Analysis* 76 (2022), p. 102327 (cit. on p. 9).

[288] L. Wu, H. Lin, C. Tan, Z. Gao, and S. Z. Li. “Self-supervised learning on graphs: Contrastive, generative, or predictive”. In: *IEEE Transactions on Knowledge and Data Engineering* (2021) (cit. on pp. 52, 53, 55, 56).

[289] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. “Unsupervised feature learning via non-parametric instance discrimination”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3733–3742 (cit. on p. 57).

[290] Y. Xia, D. Yang, Z. Yu, et al. “Uncertainty-aware multi-view co-training for semi-supervised medical image segmentation and domain adaptation”. In: *Medical Image Analysis* (2020), p. 101766 (cit. on p. 43).

[291] G. Xie, J. Wang, Y. Huang, et al. “FedMed-GAN: Federated Domain Translation on Unsupervised Cross-Modality Brain Image Synthesis”. In: *arXiv preprint arXiv:2201.08953* (2022) (cit. on p. 51).

[292] J. Xie, R. Girshick, and A. Farhadi. “Unsupervised deep embedding for clustering analysis”. In: *International conference on machine learning*. PMLR. 2016, pp. 478–487 (cit. on p. 58).

[293] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji. “Self-supervised learning of graph neural networks: A unified review”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022) (cit. on pp. 52, 53).

[294] Y. Xie, J. Zhang, Z. Liao, Y. Xia, and C. Shen. “PGL: prior-guided local self-supervised learning for 3D medical image segmentation”. In: *arXiv preprint arXiv:2011.12640* (2020) (cit. on p. 60).

[295] W. Xiong, J. Droppo, X. Huang, et al. “Achieving human parity in conversational speech recognition”. In: *arXiv preprint arXiv:1610.05256* (2016) (cit. on p. 122).

[296] S. S. Yadav and S. M. Jadhav. “Deep convolutional neural network based medical image classification for disease diagnosis”. In: *Journal of Big Data* 6.1 (2019), pp. 1–18 (cit. on p. 9).

[297] D. Yang, Z. Xu, W. Li, et al. “Federated Semi-Supervised Learning for COVID Region Segmentation in Chest CT using Multi-National Data from China, Italy, Japan”. In: *Medical Image Analysis* (2021), p. 101992 (cit. on pp. 50, 51, 89, 91, 98, 100, 101, 104–107, 110, 111).

[298] J. Yang, R. Shi, and B. Ni. “MedMNIST Classification Decathlon: A Lightweight AutoML Benchmark for Medical Image Analysis”. In: *IEEE 18th International Symposium on Biomedical Imaging (ISBI)*. 2021, pp. 191–195 (cit. on p. 127).

[299] J. Yang, R. Shi, D. Wei, et al. “MedMNIST v2: A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification”. In: *arXiv preprint arXiv:2110.14795* (2021) (cit. on p. 127).

[300] Q. Yang, Y. Liu, T. Chen, and Y. Tong. “Federated machine learning: Concept and applications”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19 (cit. on pp. 45, 48).

[301] X. Yang, Z. Song, I. King, and Z. Xu. “A survey on deep semi-supervised learning”. In: *arXiv preprint arXiv:2103.00550* (2021) (cit. on pp. 35–37, 41).

[302] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. “Xlnet: Generalized autoregressive pretraining for language understanding”. In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 122).

[303] Y. Yeganeh, A. Farshad, N. Navab, and S. Albarqouni. “Inverse distance aggregation for federated learning with non-iid data”. In: *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*. Springer, 2020, pp. 150–159 (cit. on p. 50).

[304] L. Yu, S. Wang, X. Li, C.-W. Fu, and P.-A. Heng. “Uncertainty-Aware Self-ensembling Model for Semi-supervised 3D Left Atrium Segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 605–613 (cit. on p. 44).

[305] Y. Yuan, M. Chao, and Y.-C. Lo. “Automatic skin lesion segmentation using deep fully convolutional networks with jaccard distance”. In: *IEEE transactions on medical imaging* 36.9 (2017), pp. 1876–1886 (cit. on p. 9).

[306] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. “Barlow twins: Self-supervised learning via redundancy reduction”. In: *arXiv preprint arXiv:2103.03230* (2021) (cit. on pp. 58, 122–124, 127, 128).

[307] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. “mixup: Beyond empirical risk minimization”. In: *arXiv preprint arXiv:1710.09412* (2017) (cit. on pp. 65, 67, 124).

[308] J. Zhang, Y. Xie, Y. Xia, and C. Shen. “Attention residual learning for skin lesion classification”. In: *IEEE transactions on medical imaging* 38.9 (2019), pp. 2092–2103 (cit. on pp. 32, 87).

[309] L. Zhang, Y. Luo, Y. Bai, B. Du, and L.-Y. Duan. “Federated Learning for Non-IID Data via Unified Feature Learning and Optimization Objective Alignment”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 4420–4428 (cit. on p. 117).

[310] P. Zhang, F. Wang, and Y. Zheng. “Self supervised deep representation learning for fine-grained body part recognition”. In: *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. IEEE. 2017, pp. 578–582 (cit. on p. 59).

[311] R. Zhang, P. Isola, and A. A. Efros. “Colorful image colorization”. In: *European conference on computer vision*. Springer. 2016, pp. 649–666 (cit. on pp. 53, 56).

[312] R. Zhang, P. Isola, and A. A. Efros. “Split-brain autoencoders: Unsupervised learning by cross-channel prediction”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1058–1067 (cit. on p. 53).

[313] Y. Zhang, L. Yang, J. Chen, M. Fredericksen, D. P. Hughes, and D. Z. Chen. “Deep adversarial networks for biomedical image segmentation utilizing unannotated images”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 408–416 (cit. on pp. 45, 72, 75, 78–80).

[314] Y. Zhang, M. Li, Z. Ji, et al. “Twin self-supervision based semi-supervised learning (TS-SSL): Retinal anomaly classification in SD-OCT images”. In: *Neurocomputing* 462 (2021), pp. 491–505 (cit. on p. 60).

[315] Z. Zhang, Y. Yang, Z. Yao, et al. “Improving semi-supervised federated learning by reducing the gradient diversity of models”. In: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE. 2021, pp. 1214–1225 (cit. on p. 50).

[316] Z. Zhang, Z. Yao, Y. Yang, Y. Yan, J. E. Gonzalez, and M. W. Mahoney. “Benchmarking semi-supervised federated learning”. In: *arXiv preprint arXiv:2008.11364* 17 (2020), p. 3 (cit. on p. 50).

[317] Y. Zhao, H. Liu, H. Li, P. Barnaghi, and H. Haddadi. “Semi-supervised federated learning for activity recognition”. In: *arXiv preprint arXiv:2011.00851* (2020) (cit. on p. 50).

[318] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. “Federated learning with non-iid data”. In: *arXiv preprint arXiv:1806.00582* (2018) (cit. on p. 117).

[319] Y. Zheng, M. Jin, Y. Liu, L. Chi, K. T. Phan, and Y.-P. P. Chen. “Generative and contrastive self-supervised learning for graph anomaly detection”. In: *IEEE Transactions on Knowledge and Data Engineering* (2021) (cit. on p. 56).

[320] S. K. Zhou, H. Greenspan, C. Davatzikos, et al. “A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises”. In: *Proceedings of the IEEE* 109.5 (2021), pp. 820–838 (cit. on p. 10).

[321] Y. Zhou, L. Xie, W. Shen, E. Fishman, and A. Yuille. “Pancreas segmentation in abdominal CT scan: a coarse-to-fine approach”. In: *arXiv preprint arXiv:1612.08230* (2016) (cit. on p. 9).

[322] Z. Zhou, V. Sodha, M. M. Rahman Siddiquee, et al. “Models genesis: Generic autodidactic models for 3d medical image analysis”. In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2019, pp. 384–393 (cit. on p. 59).

[323] H. Zhu, H. Zhang, and Y. Jin. “From federated learning to federated neural architecture search: a survey”. In: *Complex & Intelligent Systems* 7.2 (2021), pp. 639–657 (cit. on p. 45).

[324] J. Zhu, Y. Li, Y. Hu, K. Ma, S. K. Zhou, and Y. Zheng. “Rubik’s cube+: A self-supervised feature learning framework for 3d medical image analysis”. In: *Medical image analysis* 64 (2020), p. 101746 (cit. on p. 59).

[325] W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng. “Privacy-Preserving Federated Brain Tumour Segmentation”. In: *Machine Learning in Medical Imaging: 10th International Workshop, MLMI 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 13, 2019, Proceedings*. Vol. 11861. Springer Nature. 2019, p. 133 (cit. on p. 88).

[326] X. Zhu and A. B. Goldberg. “Introduction to semi-supervised learning”. In: *Synthesis lectures on artificial intelligence and machine learning* 3.1 (2009), pp. 1–130 (cit. on pp. 34, 35, 37).

[327] X. J. Zhu. “Semi-supervised learning literature survey”. In: (2005) (cit. on pp. 34, 35, 37).

[328] Y. Zhu, Y. Liu, J. James, and X. Yuan. “Semi-supervised federated learning for travel mode identification from gps trajectories”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.3 (2021), pp. 2380–2391 (cit. on p. 50).

[329] X. Zhuang, Y. Li, Y. Hu, K. Ma, Y. Yang, and Y. Zheng. “Self-supervised feature learning for 3d medical images by playing a rubik’s cube”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 420–428 (cit. on p. 59).

[330] D. Zimmerer, F. Isensee, J. Petersen, S. Kohl, and K. Maier-Hein. “Unsupervised anomaly localization using variational auto-encoders”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 289–297 (cit. on p. 10).

[331] M. Zinkevich, M. Weimer, L. Li, and A. Smola. “Parallelized stochastic gradient descent”. In: *Advances in neural information processing systems* 23 (2010) (cit. on p. 46).

[332] G. Zizzo, A. Rawat, M. Sinn, and B. Buesser. “Fat: Federated adversarial training”. In: *arXiv preprint arXiv:2012.01791* (2020) (cit. on p. 50).

List of Figures

List of Tables