UNIVERSITÉ DE LIMOGES

Falculté de Sciences et Techniques

Master 1

Sécurité de l'Information et Cryptologie
(CRYPTIS)

Parcours Informatique

Projet M1 Informatique - Semestre II

Federated learning
with Siamese Neural Network

Anh Kiet DUONG
Viet-Huy HA
Hadir BARHOUMI
Abir AROUI

Supervisor: M. KARIM Tamine

December 20, 2023

Contents

Abbreviations

ANN artificial neural networks

ASP Audio Signal Processing

CNN Convolutional Neural Network

CV Computer Vision

DFUC Diabetic Foot Ulcer Grand Challenge

DL Deep Learning

FL Federated Learning

LMCot Large Margin Cotangent Loss

ML Machine Learning

NLP Natural Language Processing

SL Similarity Learning

SNN Siamese Neural Network

Chapter 1

Introduction

1.1 General

Machine Learning is a subfield of artificial intelligence that focuses on developing algorithms capable of discovering patterns in data. These patterns can be found in various forms such as numbers, characters, images, and statistics. By analyzing and learning from these patterns, machine learning algorithms can improve their performance in performing specific tasks. This ability to automatically learn and make predictions from data is what sets machine learning apart.

In recent years, machine learning has made significant advancements and has had a profound impact on various industries. One notable breakthrough is in the field of Computer Vision, particularly with the development of self-driving cars. Tesla's Level 5 self-driving cars, for example, have showcased the remarkable capabilities of AI in enabling autonomous vehicles.

Moreover, AI has also made significant contributions to the medical field, including disease identification and diagnosis, drug discovery, medical imaging, personalized psychiatry, and health profiling. As machine learning continues to be integrated into healthcare systems, patient data can be analyzed in real-time, leading to more effective treatment options.

Furthermore, there are specific areas within machine learning that have gained attention, such as similarity learning, federated learning, and peer-to-peer (P2P) approaches. Similarity learning focuses on training models to measure and understand similarities between data points, enabling tasks like face recognition and recommendation systems. Federated learning aims to train machine learning models using decentralized data sources, addressing privacy concerns and reducing the reliance on a centralized authority. P2P approaches further alleviate the load on centralized systems by enabling direct communication and collaboration among participating clients.

These advancements in machine learning and its various applications are shaping the future of technology and revolutionizing industries worldwide.

1.2 Introduction to the problem

Driven by our passion for healthcare, we embarked on a project aimed at harnessing the power of machine learning to make advancements in the medical field, specifically in the area of disease diagnosis through medical imaging.

Our focus lies in the classification of foot ulcers, utilizing state-of-the-art machine learning techniques. The primary objective of this endeavor is to automate the classification process, enabling the early detection and prompt treatment of potentially severe diseases. It is important to note that our intention is not to replace medical professionals but rather to provide them with a valuable tool to aid in making accurate predictions and decisions.

To achieve our goals, we will leverage various machine learning methods, each tailored to address specific challenges in our research. One such approach is Federated Learning, which ensures the security and privacy of sensitive medical data by training models in a decentralized manner, utilizing local data sources without compromising individual privacy. This technique enables us to harness the collective knowledge from multiple healthcare providers while maintaining data confidentiality.

Additionally, we will employ Siamese Network, a deep learning architecture known for its ability to handle scenarios involving new labels, a concept referred to as Zero-shot learning. By incorporating Siamese Network into our framework, we aim to enhance the adaptability and robustness of our model, enabling it to effectively classify foot ulcers even in the presence of previously unseen labels.

In the subsequent chapters, we will delve deeper into the intricacies of Federated Learning and Siamese Network, exploring their mechanisms, benefits, and practical applications within our project. By combining these advanced machine learning techniques with our domain expertise in healthcare, we strive to contribute to the development of cutting-edge medical diagnostic systems that can improve patient outcomes and revolutionize the field of healthcare.

Chapter 2

Machine Learning

2.1 Kaggle datasets

2.1.1 Introduction

Kaggle, an influential platform under Google's umbrella, has transformed the data science landscape by offering a space where data enthusiasts, professionals, and organizations can collaborate, compete, and grow. With its rich repository of datasets, Kaggle extends beyond a traditional data platform, serving as a vital educational resource and a springboard for innovative machine learning projects.

The range of datasets on Kaggle is broad, with a notable segment dedicated to healthcare, a field where data-driven insights can make a tangible difference in policy-making, treatment strategies, and patient outcomes.

2.1.2 Methods

Analyzing Kaggle datasets involves a systematic approach to data collection and evaluation. Initially, a comprehensive review of available datasets is conducted by exploring Kaggle's website, considering factors such as dataset size, complexity, domain, and popularity. Upon selection, these datasets are downloaded for detailed analysis.

The datasets are then scrutinized using various data analysis techniques, ranging from descriptive statistics to more advanced exploratory data analysis (EDA), which may involve visualizing data distributions, assessing correlations, or identifying outliers.

Depending on the nature of the data and the study's objective, different machine learning models may be employed to further understand patterns, relationships, or to make predictions.

2.1.3 Types of Datasets

Kaggle offers datasets across a plethora of domains including but not limited to finance, environment, entertainment, and technology. A noteworthy subset is healthcare-related datasets, which have become increasingly crucial in the era of data-driven medicine.

These datasets encompass various healthcare aspects such as patient records, medical imaging, genomics, and drug discovery. The richness of these datasets enables researchers to uncover medical insights, develop diagnostic models, and propel healthcare towards more personalized and effective strategies.

2.1.4 Quality and Completeness

The quality and completeness of Kaggle datasets vary significantly. Some datasets are polished, with clean, well-documented, and ready-to-use data. Others may require considerable preprocessing, such as handling missing values, correcting inconsistencies, or managing unstructured data.

Despite these variations, Kaggle generally promotes good data quality by encouraging dataset publishers to provide comprehensive metadata and clear documentation. Nevertheless, it's incumbent on the user to conduct thorough data quality checks before diving into analysis or model building.

2.1.5 Case Studies

An example of a widely-used healthcare dataset on Kaggle is the "Heart Disease UCI" dataset. It contains medical attributes of individuals, such as age, sex, cholesterol levels, and the presence of heart disease. It has been used extensively for binary classification tasks, with numerous shared notebooks exploring different machine learning techniques.

Another popular dataset is the "COVID-19 Open Research Dataset," which includes thousands of scholarly articles about COVID-19 and related coronaviruses. This dataset has been used in Natural Language Processing (NLP) tasks and was the focus of the 'Covid-19 Literature Clustering' competition, which aimed to develop text mining tools to support the medical community in the fight against COVID-19.

2.1.6 Legal and Ethical Considerations

While Kaggle datasets provide a valuable resource, they also raise several legal and ethical considerations. Data privacy is of utmost concern, particularly with healthcare datasets that may contain sensitive patient information. Although

Kaggle datasets are supposed to be anonymized and de-identified, the risk of re-identification should not be overlooked.

Intellectual property rights are another consideration. Users must respect the terms of use for each dataset, which may restrict how the data can be used or shared. Lastly, biases in the data, which may reflect societal or systemic biases, can lead to skewed analysis results and reinforce existing inequalities. Therefore, users must approach the data with an awareness of these potential biases and consider their implications in any derived insights or applications.

2.1.7 COVID-19 Radiography Database

The dataset was created through a collaboration between researchers from Qatar University, Doha, Qatar, the University of Dhaka, Bangladesh, and their collaborators from Pakistan and Malaysia, in close coordination with medical professionals. This dataset comprises chest X-ray images, categorized into COVID-19 positive cases, Normal cases, and Viral Pneumonia cases.

The dataset was released in multiple stages, with each release providing an increased number of images. The initial release included 219 COVID-19, 1341 normal, and 1345 viral pneumonia chest X-ray images. Subsequently, the COVID-19 class was expanded to include 1200 additional CXR images in the first update. In the second update, the database was further expanded to include 3616 COVID-19 positive cases, 10,192 Normal cases, 6012 Lung Opacity cases (Non-COVID lung infections), and 1345 Viral Pneumonia cases, along with corresponding lung masks.

The creators of the dataset are committed to continuously updating it by incorporating new chest X-ray images from COVID-19 pneumonia patients as they become available. This ongoing effort ensures that the dataset remains comprehensive and up-to-date, facilitating further research and analysis in the field of chest X-ray imaging for COVID-19 and other respiratory conditions. [1, 2]

2.2 Convolutional Neural Network - CNN

Over the past ten years, Convolutional Neural Networks (CNNs) have revolutionized various fields that involve pattern recognition, including image processing and voice recognition. A key advantage of CNNs is their ability to reduce the parameter count in Artificial Neural Networks (ANNs). This has encouraged both researchers and developers to tackle larger models to address more complex tasks, which was not feasible with traditional ANNs. It's important to note that CNNs are most effective when solving problems where the features are not dependent on spatial location. For instance, in a face detection task, the focus isn't on the faces' locations within the image, but simply on their detection, irrespective of

their position. Another noteworthy characteristic of CNNs is their capacity to derive abstract features as the input moves deeper into the layers.

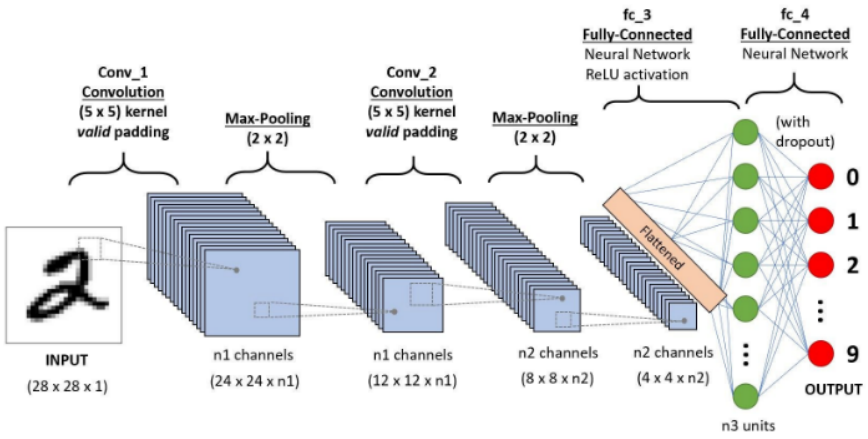


Figure 2.1: CNN Architecture

2.2.1 CNN Basic layers

Convolution Layer

The Convolution Layer is a fundamental yet critical layer in a Convolutional Neural Network (CNN). It essentially performs convolution operations, multiplying the pixel matrix derived from a given image or object to generate what's known as an activation map. One of the main benefits of an activation map is its ability to retain all the unique features of a given image while simultaneously reducing the volume of data that needs processing. The matrix used for convolution is referred to as a feature detector, a set of values that the machine can interact with. Different iterations of the image are created using varying values of this feature detector. The convolution model also undergoes a training process using backpropagation, which ensures minimal error in each layer. Based on the lowest error set, parameters such as depth and padding are established.[3]

Pooling Layer

Pooling in a Convolutional Neural Network (CNN) primarily serves the purpose of down-sampling, a process aimed at reducing the computational complexity for subsequent layers. In the context of image processing, this can be thought of as analogous to lowering the image resolution.

Contrary to some assumptions, pooling does not modify the number of filters used in the network. Among various types of pooling methods, max-pooling is one of the most frequently used. This technique involves dividing the image into smaller rectangular sub-regions. Within each of these sub-regions, only the

highest value (representing the most prominent feature) is retained while the rest are discarded.

This process helps to preserve the important characteristics of the image, while reducing the overall data size and hence computational load. A common choice for the size of the rectangular sub-region in max-pooling is a 2x2 pixel window. This size is often used due to its efficiency in maintaining key features while still achieving a significant reduction in data volume. [4]

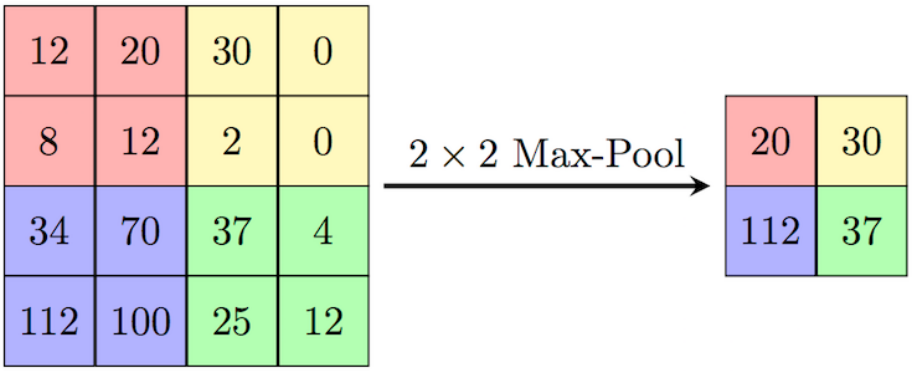


Figure 2.2: CNN Pooling Layer

Classification — Fully Connected Layer

Fully Connected layers in a neural networks are those layers where all the inputs from one layer are connected to every activation unit of the next layer. In most popular machine learning models, the last few layers are full connected layers which compiles the data extracted by previous layers to form the final output. It is the second most time consuming layer second to Convolution Layer.

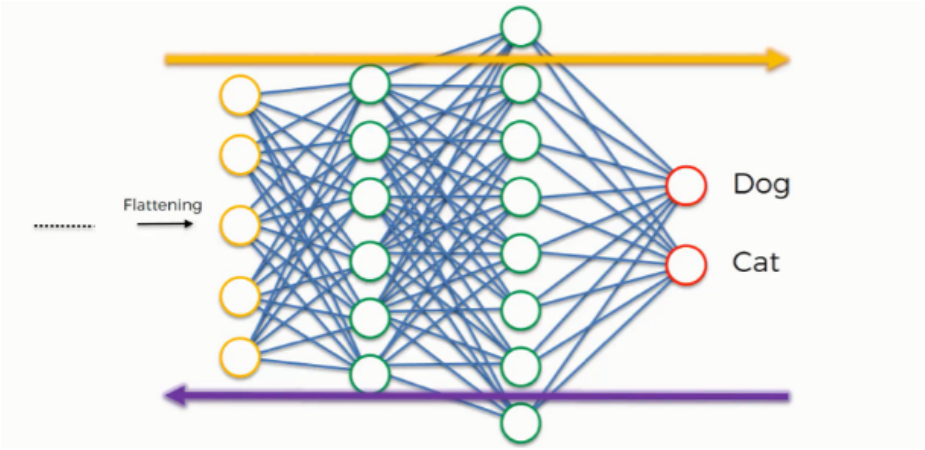


Figure 2.3: Fully Connected Layer

2.3 Transformers

2.3.1 What are Transformers

Over the past few years, the transformer model has become one of the major highlights in the field of deep learning . Transformers were introduced in "Attention is all you need" paper and were mainly developed for advanced applications in natural language processing (NLP). This is due to the transformers ability to capture long-range connections among words in the same sentence and learn complex relationships between words in it.
This model is distinguished by applying an evolving set of mathematical techniques, called attention or self-attention, which involves differentially weighting the significance of each part of the input in order to track relationships in sequential data.

2.3.2 Transformers Architecture

Transformer's architecture is primarily composed of two blocks : an encoder block and a decoder block. Each block is composed of multiple layers, and each layer contain sub-layers that uses feed-forward neural networks and self-attention.

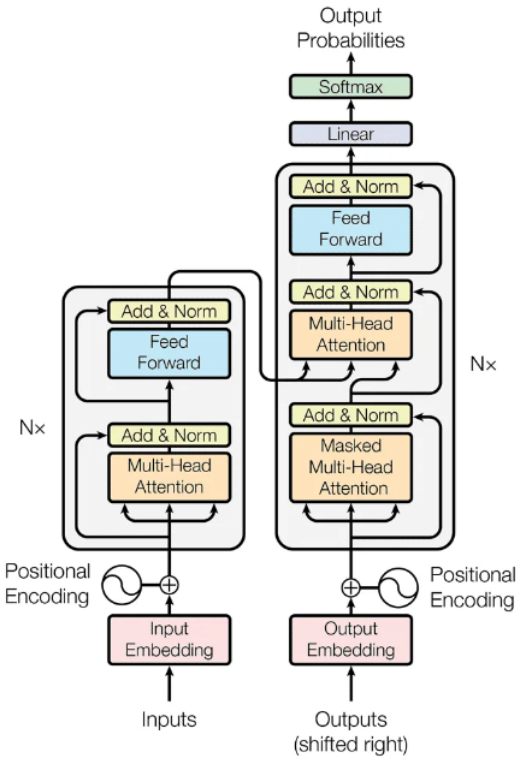


Figure 2.4: Transformers Architecture

- Encoder (on the left half of the Transformer architecture): The task of the encoder block is to extract features from an input sequence . Every layer

within the encoder block has two main sub-layers which are position-wise feed-forward networks and self-attention, these sub layers are responsible for mapping the input sequence to a sequence of continuous representations (features).
In transformers, the input sentence goes series of encoder blocks, and the resulting output of the last encoder block is used as the input to the decoder block.

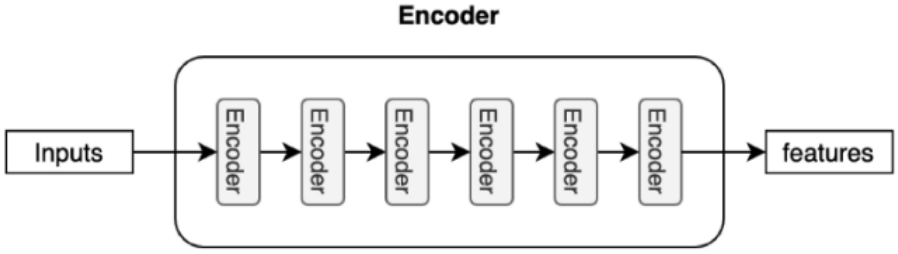


Figure 2.5: encoder block

- decoder (on the left half of the Transformer architecture): This block uses the encoded features from the encoder along with other inputs in order to generate the target output. Every layer within the decoder block has two main sub-layers which are position-wise feed-forward networks and self-attention.
  However, it also includes another attention mechanism which is encoder-decoder attention. This mechanism allows the decoder to selectively focus on relevant parts of our input during the output generation.

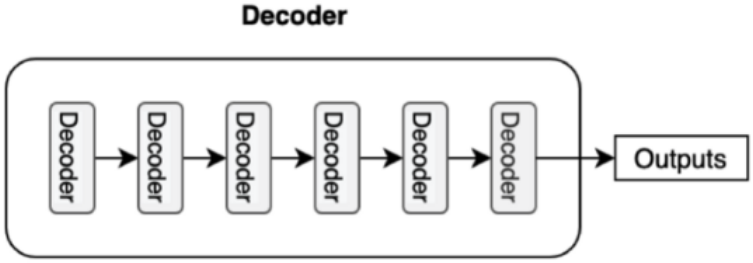


Figure 2.6: decoder block

2.3.3 Key Components Of Transformers Architecture

Self-Attention

The self-attention mechanism is the central component of the Transformer's architecture. In fact, this mechanism plays a crucial role in capturing long-range

dependencies between words for Natural Language Processing case and capturing spatial relationships between image patches for computer vision case which brings more awareness to the developed model.

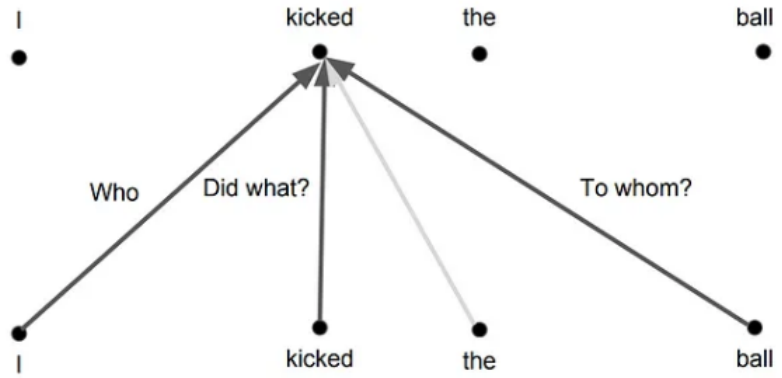


Figure 2.7: self attention simplified example

The attention mechanism consists of the following steps:

- Query, Key, and Value: The first step in calculating self-attention is transforming the input sequence into three sets of vectors known as Query vector, a Key vector, and a Value vector. These vectors are computed by multiplying the input patches by the matrices trained during the training process.
- Calculate attention Scores:It's the second step in calculating self-attention. This step involves the dot product between the queries and keys. Then, the product is normalized with the square root of the dimension of the queries in order to stabilize the gradients.
- Calculate attention Weights: This step consists of dividing the scores of the second step by 8 (default value), then passing them through a softmax function in order to normalize them and ensure that all the weights for each token are positive and add up to one.
- Calculate weighted Sum: This step consists of multiplying each value vector by the softmax score enabling the model to focus on important patches and drown-out irrelevant ones while considering the global context.Then,sum up the weighted value vectors.

After these steps, the weighted sum is processed through feed-forward layers and residual connections.

Multi-Head Attention

Multi-head attention uses self-attention mechanism and extends it by allowing the model to attend to different parts of the input sequence simultaneously. This means that the model can capture multiple perspectives on the input sequence which improves the model's ability to capture different types contextual information between the input sequence.

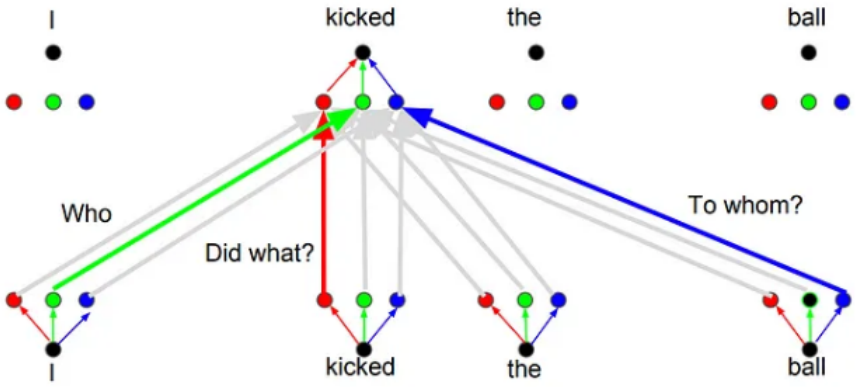


Figure 2.8: Multi-head attention simplified example

Feed-Forward Neural Network

Feed-forward layers constitute two-thirds of a transformer model's parameters, they serves as the primary mechanism for processing the information captured by the attention mechanism and generating the final output sequence.
A simple feed-forward neural network that comprises multiple layers with ReLU activation functions is applied to every attention vector.

Positional Encoding

Another important step on a transformer is Positional encoding, As transformers do not use recurrent or convolutional layers, they need a technique that injects positional information into the input sequence.
The most used positional encoding method is based on sine and cosine functions.

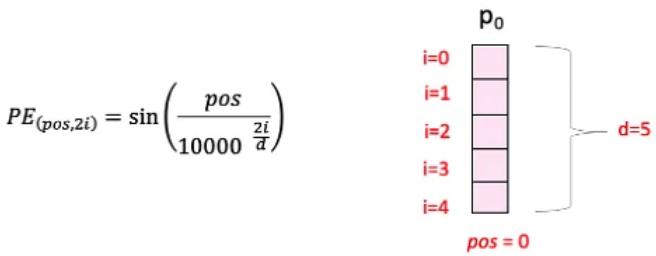


Figure 2.9: positional encoding method example

2.3.4 From Language to Vision

Vision Transformer (ViT) in Image Recognition

Although the transformer architecture was originally introduced as a groundbreaking architecture in the natural language processing domain, it's success prompted the AI crowd to ask what else they could do. The answer is unfolded as transformers found their way into computer vision, as a CNN-backbone replacement in many complicated tasks.
The vision transformer (ViT) architecture is a model that employs the transformer architecture to image recognition tasks. This model divides the input image into a set of patches, each patch of them is linearly embedded, position embeddings are added, and the resulting sequence of vectors are the input of normal transformer.
Vision transformers can be used in many image recognition tasks such as image classification, image segmentation and object detection.

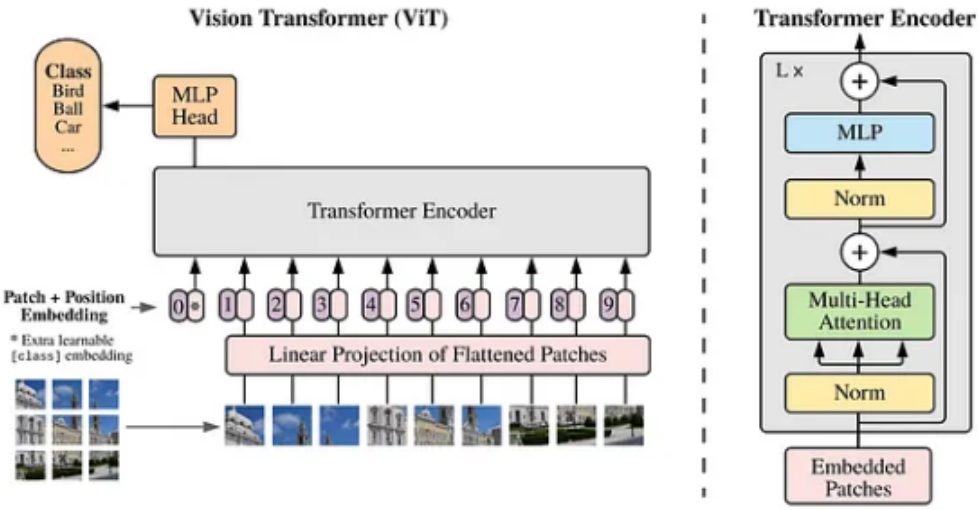


Figure 2.10: Vision transformers

Difference between CNN and ViT

Convolutional Neural Networks (CNN) have been dominant in Computer Vision tasks so far. Nowadays, transformers are performing as well as CNNs in many cases.
These two have different approaches to computer vision: while CNNs repeatedly filters small parts of the input image in order to map features , a transformer starts by connecting every element to the other ones to create the global presentation.

Chapter 3

Federated Learning

Federated Learning FL is a revolutionary approach to machine learning that addresses the challenges of data privacy and security. Traditional machine learning models rely on centralizing data from multiple sources, which can raise concerns about data privacy, especially when dealing with sensitive information. In contrast, FL allows for distributed learning across devices or servers, ensuring that data remains decentralized and secure.

The main concept behind FL is to bring the machine learning model to the data rather than bringing the data to the model. Instead of transferring data to a central server for training, FL enables training to take place on local devices or servers where the data resides. This decentralized approach ensures that data remains under the control of the data owners and minimizes the risk of unauthorized access or data breaches.

By keeping data local, FL offers several key benefits in terms of data security. Firstly, it eliminates the need to transfer sensitive data over networks, reducing the exposure to potential security threats. This is particularly important in industries such as healthcare, finance, and telecommunications, where data confidentiality is paramount. Secondly, FL employs advanced encryption techniques to protect data during transmission and ensure that only encrypted model updates are shared, further enhancing the security of the learning process.

FL also employs techniques like differential privacy to provide strong privacy guarantees. Differential privacy ensures that individual data points cannot be inferred from the model updates, safeguarding the privacy of the individual contributors. This means that even when multiple devices or servers collaborate in training a model, no sensitive information about specific data points is exposed.

The decentralized nature of FL also offers advantages in terms of data ownership and control. Data remains within the jurisdiction and control of the local entities, ensuring compliance with privacy regulations and policies. This distributed learning paradigm empowers organizations and individuals to retain ownership of their data while benefiting from collective intelligence through collaborative model training.

Overall, federated learning represents a breakthrough in machine learning, allowing organizations to harness the collective knowledge of distributed datasets while preserving data privacy and security. By eliminating the need for centralized data storage and promoting secure and privacy-preserving techniques, FL opens up new opportunities for collaboration and innovation in a wide range of domains, while prioritizing the protection of sensitive information.

3.1 Federated Learning centralized

3.1.1 FedAVG

Federated Averaging (FedAvg) is a popular algorithm used in Federated Learning (FL) to train machine learning models on distributed data without the need for data centralization. In FL, data is kept locally on individual devices or servers, and training takes place locally on these devices. FedAvg enables the aggregation of local model updates from multiple devices to create a global model that represents the collective knowledge of the distributed data.

The FedAvg algorithm works in iterations, where in each iteration, a fraction of devices (also known as clients) are selected to participate in model training. These selected clients locally train their models using their own local data and then send their updated model parameters to a central server. The central server aggregates these model updates by averaging the parameters and computes a new global model. The updated global model is then sent back to the clients, and the process continues for multiple rounds.

FedAvg incorporates a weighted averaging scheme to account for the varying amount of data and computational capabilities of the clients. Clients with more data or higher computational resources are given higher weights during the aggregation process, ensuring that their contributions have a larger impact on the global model.

By distributing the training process across multiple devices and keeping the data locally, FedAvg addresses privacy concerns associated with centralized data collection. It allows for collaborative learning while preserving data privacy. Furthermore, FedAvg has been shown to achieve comparable performance to traditional centralized learning approaches, even with the challenges of non-IID (non-independent and identically distributed) data and heterogeneous devices present in federated settings.

FedAvg is a foundational algorithm in Federated Learning and has been applied to various domains, including healthcare, finance, and IoT, where data is often sensitive and decentralized. It enables collaborative model training while maintaining privacy and data ownership, making it a promising approach for large-scale distributed learning scenarios.

3.1.2 FedSGD

Federated Stochastic Gradient Descent (FedSGD) is an optimization algorithm used in Federated Learning (FL) to train machine learning models on decentralized data. It is a variation of the traditional Stochastic Gradient Descent (SGD) algorithm, adapted to the federated setting.

In FedSGD, the training process takes place on multiple devices or servers, where each device holds its own local data. Unlike the centralized setting, where data is aggregated on a central server, FedSGD performs local model updates on each device using its local data. These local updates are then communicated to a central server, which aggregates them to create a global model.

The key difference between FedSGD and traditional SGD lies in the aggregation step. In SGD, the local updates from all devices are typically averaged to update the global model. In FedSGD, however, a fraction of devices is randomly selected to participate in each round of model updates. This selective participation helps to reduce communication overhead and computational burden.

During each round of FedSGD, the selected devices perform local model updates by computing gradients based on their local data. These gradients are then sent to the central server, which aggregates them using various methods such as weighted averaging or geometric median. The resulting aggregated gradient is used to update the global model. This process is repeated for multiple rounds until convergence or a predefined stopping criterion is met.

FedSGD enables distributed training on decentralized data while addressing privacy concerns. By keeping the data local and performing model updates on the devices themselves, FedSGD helps preserve data privacy and reduces the need for data sharing. It is particularly suitable for scenarios where data cannot be centralized due to privacy regulations, network limitations, or sensitive information.

Overall, FedSGD is a fundamental algorithm in Federated Learning, allowing collaborative training on distributed data without compromising privacy. It has been applied in various domains, including healthcare, finance, and edge computing, offering a promising solution for decentralized machine learning.

3.2 Federated Learning Peer-to-Peer

Peer-to-peer Federated Learning (P2P FL) offers a compelling alternative to Centralized FL due to several key reasons. While Centralized FL relies on a central server to orchestrate the learning process, P2P FL leverages a decentralized network of clients, eliminating the need for a single point of control and dependency.

One of the main advantages of P2P FL is increased resilience and fault tolerance. In a Centralized FL setup, the failure of the central server can disrupt the entire learning process. However, in P2P FL, the absence or failure of a single

peer does not compromise the overall system, as the learning process can continue uninterrupted among the remaining peers.

Another significant benefit of P2P FL is the potential for improved scalability and communication efficiency. In Centralized FL, the central server bears the burden of processing and aggregating updates from all participating clients, which can be resource-intensive and limit scalability. P2P FL, on the other hand, distributes the computation and communication load across the network, allowing for more efficient resource utilization and potentially faster convergence.

Moreover, P2P FL addresses privacy concerns and reduces the reliance on a trusted central authority. With Centralized FL, clients must trust and depend on the central server to handle their data securely. In contrast, P2P FL enables clients to collaborate directly with each other, minimizing the exposure of sensitive data to a central entity and enhancing privacy preservation. [5]

3.2.1 Heuristic 0: random

This approach is done in a naive manner where we simply perform random sampling. In other words, each client will randomly send its weight/vector gradient to a subset of other clients.

FedAVG

Algorithm 1 FedAVG heuristic 0: random

for round $\leftarrow 1, 2, 3, \ldots$ **do**
  for *client* $\leftarrow 1, 2, 3, \ldots$ **do**
    w_{client} = fit(w_{client}, $data_{client}$, epochs=$local_epoch)
  end for
  for client $\leftarrow 1, 2, 3, \ldots$ in parallel **do**
    $w_{\text{client}} \leftarrow$ Mean(GetRandomNeighbors(c).weight)
  end for
end for

FedSGD

Algorithm 2 FedSGD heuristic 0: random

for round $\leftarrow 1, 2, 3, \ldots$ **do**
  for *local_epoch* $\leftarrow 1, 2, 3, \ldots$ **do**
    for *step* $\leftarrow 1, 2, 3, \ldots$ **do**
      for *client* $\leftarrow 1, 2, 3, \ldots$ in parallel **do**
        $grad_{client} = Gradient(w_{client}, data_{client})$
        grad = getRandomNeighborsGrad(c)
        $w_{client} += Mean(grad)$
      end for
    end for
  end for
end for

3.2.2 Heuristic 1: n lastest

Each client in the network maintains its own identity and keeps track of the identities of the n most recent clients it has interacted with. At the end of each communication round, this information regarding the n most recent clients is disseminated throughout the network. Subsequently, each client selects its communication partners based on the level of dissimilarity in their previous interactions. Specifically, clients prioritize communication with those who have had the least amount of overlap in past interactions.

FedAVG

Algorithm 3 FedAVG heuristic 1: n lastest

for round $\leftarrow 1, 2, 3, \ldots$ **do**
    for *client* $\leftarrow 1, 2, 3, \ldots$ **do**
        w_{client} = fit(w_{client}, $data_{client}$, epochs=$local_epoch)
    end for
    for client $\leftarrow 1, 2, 3, \ldots$ in parallel **do**
        neighbors = GetRandomNeighbors(c, without = client.last)
        $w_{\text{client}} \leftarrow$ Mean(neighbors.weight)
        client.last = (client.last + neighbors)[-n:]
    end for
end for

FedSGD

Algorithm 4 FedAVG heuristic 1: n lastest

for round $\leftarrow 1, 2, 3, \ldots$ **do**
    for *local_epoch* $\leftarrow 1, 2, 3, \ldots$ **do**
        for *step* $\leftarrow 1, 2, 3, \ldots$ **do**
            for *client* $\leftarrow 1, 2, 3, \ldots$ in parallel **do**
                $grad_{client} = Gradient(w_{client}, data_{client})$
                neighbors = GetRandomNeighbors(c, without = client.last)
                $w_{client} += Mean(neighbors.grad)$
                client.last = (client.last + neighbors)[-n:]
            end for
        end for
    end for
end for

3.2.3 Heuristic 2: F1-score

The second and third heuristics utilize the models' performances to promote communication between clients with better-performing or dissimilar models. After each round, clients calculate their models' per-class F1-scores on a test set and share them with the network. Clients then select neighbors to communicate with based on the dissimilarity or similarity scores computed using these F1-scores.

$$\text{neighbor dissimilarity score} = ndc = \sum_{\text{class i}} \left| F_k^i - F_c^i \right| \tag{3.1}$$

FedAVG

Algorithm 5 FedAVG heuristic 1: n lastest

```
for round ← 1, 2, 3, ... do
    for client ← 1, 2, 3, ... do
        w_client = fit(w_client, data_client, epochs=$local_epoch)
    end for
    for client ← 1, 2, 3, ... in parallel do
        neighbors = GetNeighbors(c, without = client.last, metric = ndc)
        w_client ← Mean(neighbors.weight)
    end for
end for
```

FedSGD

Algorithm 6 FedAVG heuristic 1: n lastest

```
for round ← 1, 2, 3, ... do
    for local_epoch ← 1, 2, 3, ... do
        for step ← 1, 2, 3, ... do
            for client ← 1, 2, 3, ... in parallel do
                grad_client = Gradient(w_client, data_client)
                neighbors = GetNeighbors(c, without = client.last, metric = ndc)
                w_client+ = Mean(neighbors.grad)
            end for
        end for
    end for
end for
```

3.2.4 Heuristic 3: Score cosine

$$\text{neighbor dissimilarity score} = ndc = \cos(F_k, F_c) = \frac{F_k . F_c}{\|F_k\| \, . \, \|F_c\|} \tag{3.2}$$

FedAVG

Algorithm 7 FedAVG heuristic 1: n lastest

```
for round ← 1, 2, 3, ... do
    for client ← 1, 2, 3, ... do
        w_client = fit(w_client, data_client, epochs=$local_epoch)
    end for
    for client ← 1, 2, 3, ... in parallel do
        neighbors = GetNeighbors(c, without = client.last, metric = ndc)
        w_client ← Mean(neighbors.weight)
    end for
end for
```

FedSGD

Algorithm 8 FedAVG heuristic 1: n lastest

```
for round ← 1, 2, 3, ... do
    for local_epoch ← 1, 2, 3, ... do
        for step ← 1, 2, 3, ... do
            for client ← 1, 2, 3, ... in parallel do
                grad_client = Gradient(w_client, data_client)
                neighbors = GetNeighbors(c, without = client.last, metric =
ndc)
                w_client+ = Mean(neighbors.grad)
            end for
        end for
    end for
end for
```

Chapter 4

Similarity Learning

In my previous publications [6, 7], I have delved into the development of siamese networks, providing a comprehensive understanding of their key knowledge and significant milestones. To gain a deeper insight and more precise explanations, I highly recommend referring to the original papers.

In the realms of computer science and statistics, the concept of similarity holds paramount importance. When comparing two-element vectors, there exist various alternative similarity techniques that can be employed, such as Euclidean distance, Pearson correlation coefficient, Spearman's rank correlation coefficient, and many others. The choice of technique depends on the specific objective of the comparison. However, when dealing with complex datasets that exhibit diverse dimensions and characteristics, and potentially require compression prior to processing, traditional similarity measurements may not yield effective results. This is where siamese neural networks emerge as a promising solution.

A siamese neural network comprises two identical artificial neural networks that have the capacity to learn the hidden representations of input vectors. These networks, which operate as feedforward perceptrons, employ error back-propagation during training to optimize their performance. The two networks work in parallel and compare their outputs at the end, often using a cosine distance metric. The output of a siamese neural network operation can be interpreted as the semantic similarity between the projected representations of the two input vectors.

One notable advantage of siamese networks is their ability to handle complex data sets with varying dimensions and characteristics. By learning the hidden representations of input vectors, siamese networks can effectively capture and compare the semantic similarity between two instances. This makes them particularly useful in tasks such as image recognition, text analysis, and recommendation systems, where the comparison of complex data is essential.

To further explore the intricacies of siamese networks and gain a more comprehensive understanding, I recommend studying the original paper [8]. There, you will find detailed explanations, experimental results, and additional insights

into the development and application of siamese networks in various domains.

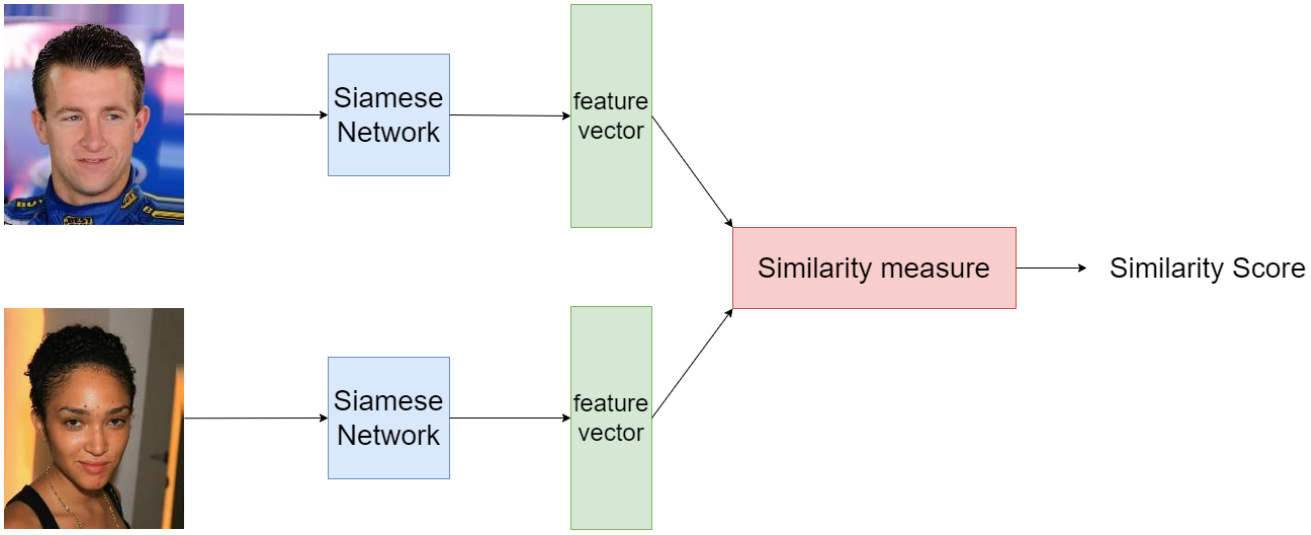


Figure 4.1: Illustrate how the siamese model works. The image is taken from the CelebA [9] dataset. The similarity measure could be cosine, euclidean, manhattan, etc.

Deep Convolutional Neural Network (DCNN) models have gained significant popularity in feature extraction tasks. Utilizing an appropriate loss function is crucial for efficient feature categorization. While Softmax loss, introduced by the One-Shot Learning approach, is commonly used, it poses challenges when applied to real-world tasks where the number of labels may frequently change, requiring re-training whenever new labels appear. To address this issue, several loss functions from the SL approach have been proposed, including Intra loss, Inter loss, Triplet loss, and Margin loss. Among these, Margin-Loss-based losses have recently been claimed to be the most effective strategies. Examples of such strategies include CosFace, which incorporates a fixed parameter as a cosine margin to remove the reliance on the cosine of weights and feature vectors' normal distribution, and ArcFace, which adds angular penalty margins in the cosine function to simultaneously enhance class compactness and class differentiation.

Siamese networks have a wide range of applications. They can be used in authentication systems, such as face recognition, fingerprint identification, and voice authentication. Additionally, siamese networks are beneficial in zero-shot, one-shot, and few-shot learning scenarios. They also find application in tasks that involve matching between text and image, as demonstrated in the work by Nguyen et al. [10]. The versatility of siamese networks makes them a valuable tool in various domains, enabling effective comparisons and similarity evaluations across different data modalities.

4.1 Distance of pairs

In addition to the previously mentioned loss functions, there are other techniques that utilize Siamese Networks to extract features and optimize the distance between outputs. Two notable examples are the **Pairwise Confusion Loss** and the **Triplet Loss**.

4.1.1 Pairwise Confusion Loss

The Pairwise Confusion Loss was introduced by Parikh and Grauman [11] as a pairwise ranking technique for relative attribute learning. This approach has gained popularity in attribute modeling tasks, and it involves training paired neural network models [12]. By comparing pairs of samples and considering their relative attributes, this loss function enables the network to learn meaningful feature representations that capture the desired similarities and differences between instances.

4.1.2 Triplet Loss

The Triplet Loss is another powerful loss function that utilizes a Siamese Network architecture. In this setup, three identical subnetworks are used, and the model is fed with three images: an anchor sample, a positive sample (similar to the anchor), and a negative sample (dissimilar to the anchor). The objective of the Triplet Loss is for the model to learn a metric space where the distance between the anchor and positive sample is smaller than the distance between the anchor and negative sample [13]. By enforcing this constraint, the network can effectively learn to measure the similarity between images and project them into a feature space where similar instances are closer to each other, while dissimilar instances are farther apart.

Both the Pairwise Confusion Loss and the Triplet Loss demonstrate the capability of Siamese Networks to optimize the distances between pairs of samples. These approaches enable the network to learn meaningful representations that can capture and quantify the similarity or dissimilarity between instances, which is valuable in various tasks such as image retrieval, face recognition, and content-based recommendation systems.

4.2 Cosine distance metric

The utilization of deep learning for learning similarities has gained significant popularity, particularly in face verification and face identification tasks. Instead of predicting a probability distribution to determine the best-fitting label for an

input image, these algorithms compare the distances between the input image and all other remaining images. This approach eliminates the dependence on the number of labels, making it unnecessary to retrain the model when new labels appear while still being able to find similar images. The key is to build a robust Deep Convolutional Neural Network (DCNN) model that can effectively project images onto an n-dimensional Euclidean space, where the distance can then be used to determine the labels.

There are two main approaches to train DCNNs more effectively: training a classification-based model using Softmax loss [14, 15, 16, 17, 18, 19], or optimizing the distance directly from the vector embeddings during training [12, 13]. Although both methods have shown decent results, they still have some shortcomings. Triplet loss, for example, compares three samples at a time, which can lead to an exponential increase in the number of triplets and a significant increase in the number of iterations as the amount of data grows. Training with triplet loss often requires the use of semi-hard sample training, which can be challenging to effectively train. On the other hand, several variants of Softmax loss have been proposed, but they each have their own drawbacks. The traditional Softmax loss, for instance, has a decision boundary dependent on both the magnitude of weights and the angles, resulting in overlapping decision regions in cosine space. The Normalized Version of Softmax Loss (NSL) is weaker in the presence of noise, and A-Softmax loss reduces its boundary as the angles decrease and may even vanish when the angles reach zero. To address these limitations, improved versions of Softmax loss, such as CosFace, ArcFace, and ElasticFace, have been proposed. However, they all share a common characteristic of applying cosine loss to calculate the angles between the feature vectors and corresponding weights.

4.2.1 Softmax Loss

First, let's examine the traditional Softmax loss, which is a common loss function used in classification-based models:

$$L_{\text{Softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{yi}^{T}xi+byi}}{\sum_{j=1}^{n}e^{W_{j}^{T}xi+bj}}, \tag{4.1}$$

where y_i is the class to which the i^{th} data belongs, n is the number of labels, and N is the number of samples. However, a model trained using the Softmax loss function can only handle a fixed number of classes. If we want to add a new label, we would need to retrain the model. The Siamese Network [20], on the other hand, is an architecture that can work with new labels without requiring retraining. Instead of classifying each input individually, a feature vector is extracted, and

then it is compared with the existing vectors in the database to make a decision. Therefore, when adding a new label, we only need to add the corresponding vector to the database.

4.2.2 Cosine-based Approaches

Next, we transform the output logit [21] as $W_j^T xi = |Wj|\,|xi| \cos(\theta ji)$, where θji is the angle between the weight vector Wj^T and the feature vector xi. By applying $l2$ normalization [22], we set $|Wj| = |xi| = 1$, $bj = 0$, and introduce a scale parameter s. With these modifications, the original Softmax loss function (4.1) becomes:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum\limits_{j=1, j\neq y_i}^{n} e^{s\cos\theta_{ji}}}. \tag{4.2}$$

Building upon this function, improved methods have been proposed that introduce a margin value m to further enhance performance.

4.2.3 SphereFace

Introduced by Liu et al. [14] in 2017, the SphereFace loss function is defined as:

$$L_{\text{SphereFace}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s\cos\left(m*\theta_{y_i}\right)}}{e^{s\cos\left(m*\theta_{y_i}\right)} + \sum\limits_{j=1, j\neq y_i}^{n} e^{s\cos\theta_{ji}}}. \tag{4.3}$$

4.2.4 CosFace

Introduced by Wang et al. [16] in 2018, the CosFace loss function is defined as:

$$L_{\text{CosFace}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s\left(\cos\left(\theta_{y_i}\right)-m\right)}}{e^{s\left(\cos\left(\theta_{y_i}\right)-m\right)} + \sum\limits_{j=1, j\neq y_i}^{n} e^{s\cos\theta_{ji}}}. \tag{4.4}$$

4.2.5 ArcFace

Introduced by Deng et al. [17] in 2019, the ArcFace loss function is defined as:

$$L_{ArcFace} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s\left(\cos\left(\theta_{y_i}+m\right)\right)}}{e^{s\left(\cos\left(\theta_{y_i}+m\right)\right)} + \sum\limits_{j=1, j\neq y_i}^{n} e^{s\cos\theta_{ji}}}. \tag{4.5}$$

4.2.6 ElasticFace

In the context of ElasticFace [19], a novel approach is introduced where the margin value is a random sample from a normal distribution.

Elastic-Cos

Based on the CosFace loss function, the Elastic-Cos loss incorporates a random margin value from a normal distribution. It is defined as:

$$L_{ElasticCos} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\left(\cos\left(\theta_{y_i}\right)-\aleph\left(\mu,\sigma^2\right)\right)}}{e^{s\left(\cos\left(\theta_{y_i}\right)-\aleph(\mu,\sigma^2)\right)}+\sum\limits_{j=1,j\neq y_i}^{n}e^{s\cos\theta_{ji}}}. \tag{4.6}$$

Elastic-Arc

Similar to Elastic-Cos, Elastic-Arc is based on the ArcFace loss function and incorporates a random margin value from a normal distribution. It is defined as:

$$L_{ElasticArc} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\left(\cos\left(\theta_{y_i}\right)-\aleph\left(\mu,\sigma^2\right)\right)}}{e^{s\left(\cos\left(\theta_{y_i}\right)-\aleph(\mu,\sigma^2)\right)}+\sum\limits_{j=1,j\neq y_i}^{n}e^{s\cos\theta_{ji}}}. \tag{4.7}$$

These cosine-based approaches, including SphereFace, CosFace, ArcFace, Elastic-Cos, and Elastic-Arc, have been introduced to enhance the performance of DCNN models for SL. By incorporating margin values and utilizing the cosine distance, these methods aim to improve class separation and compactness in the learned feature space.

4.3 LMCot

4.3.1 Introduction

In 2023, Duong et al. [6] proposed the Large Margin Cotangent Loss (LMCot) as a novel approach for enhancing performance in verification and identification tasks. The LMCot loss utilizes the cotangent function instead of the cosine function, as the cotangent function has a broader range of values, allowing for better optimization. Experimental results demonstrated that LMCot outperformed existing methods in various benchmark datasets and achieved state-of-the-art performance.

4.3.2 Cotangent-based Loss Function

The primary motivation behind LMCot is to address the limitation of the cosine function used in existing methods such as ArcFace [17]. The cosine function returns values between $[-1, 1]$, which limits its ability to accurately reflect the angle between vectors. In contrast, the cotangent function has an unrestricted range of values, making it more suitable for measuring angles.

The LMCot loss function is defined as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cot(\theta yi+m))}}{e^{s(\cot(\theta yi+m))} + \sum_{j=1,j\neq yi}^{n} e^{s\cot\theta ji}}, \tag{4.8}$$

where L represents the LMCot loss, N is the number of samples, s is a scale parameter, m is the margin, θyi is the angle between the weight and feature vector of the ground truth class, and θji is the angle between the weight and feature vector of class j.

To calculate the cotangent values, the LMCot loss function utilizes the l_2-normalized feature vectors and weights. The loss function penalizes the difference between the cotangent of the ground truth angle θyi and the cotangent of the angles θji for other classes. This encourages the model to optimize the decision boundary to improve classification accuracy.

4.3.3 Pseudo-Code and Training Process

The pseudo-code below demonstrates the implementation of LMCot in a Siamese network with a backbone network denoted as B and weights denoted as W. The variable ε represents a small value that prevents division by zero.

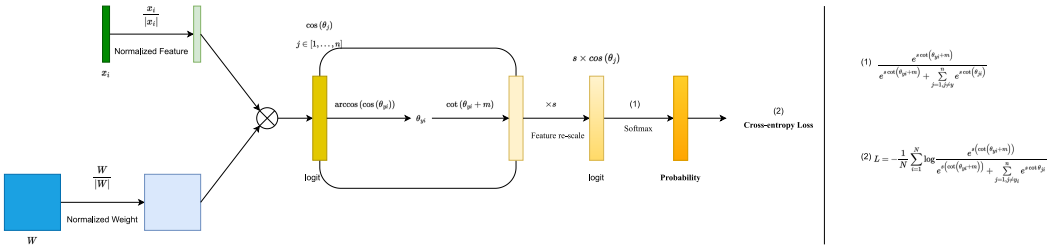


Figure 4.2: Training process of the DCNN model with Large Margin Cotangent Loss based on ArcFace.

With Data x, scale s, margin m, epsilon ε, class number n, Ground-Truth ID gt.

In the circumstance that the arc-cosine computation becomes complicated, we can use the following pseudo-code to skip the theta computing step.

Algorithm 9 Pseudo-code for LMCot in Siamese Network

$f = \|B(x)\|$
$W = \|W\|$
$\cos\_theta = Wf$
$theta = \arccos(\cos\_theta)$
$\cot\_t = \cos\_theta / \max(\sin(theta), \varepsilon)$
$\cot\_t\_m = \cos(theta + m) / \max(\sin(theta + m), \varepsilon)$
$one\_hot = onehot(gt)$
$\text{logit} = one\_hot * \cot\_t\_m + (1 - one\_hot) * \cot\_t$
$\text{logit} = s * \text{logit}$

Algorithm 10 Pseudo-code for LMCot in Siamese Network without Arc

$f = |B(x)|$
$W = |W|$
$\cos_t heta = Wf$
$\sin_t heta = \sqrt{1 - \cos_t heta^2}$
$\cot_t heta = \cos_t heta / \max(\sin_t heta, \varepsilon)$
$\cos_t heta_m = \cos_t heta * \cos(m) - \sin_t heta * \sin(m)$
$\sin_t heta_m = \sin_t heta * \cos(m) + \cos_t heta * \sin(m)$
$\cot_t heta_m = \cos_t heta_m / \max(\sin_t heta_m, \varepsilon)$
$one_hot = onehot(gt)$
$\text{logit} = one_hot * \cot_t heta_m + (1 - one_hot) \cot_t heta$
$\text{logit} = s * \text{logit}$

The pseudo-code demonstrates the calculation of the class-wise affinity score logit using the LMCot loss function. It involves the computation of the cotangent values based on the cosine values and angle calculations. The output logit represents the scores for each class, which can be used for classification or similarity comparison.

4.3.4 Ensemble Methods

In addition to the LMCot loss function, ensemble methods can be used to further enhance performance. By combining different loss functions, such as SphereFace, CosFace, ArcFace, and ElasticFace, with the use of cotangent instead of cosine, improved results can be achieved.

The ensemble method can be formulated as follows:

$$
\begin{aligned}
L &= -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cot(E(m_1,\sigma_1)\theta_{yi}+E(m_2,\sigma_2))+E(m_3,\sigma_3))}}{e^{s(\cot(E(m_1,\sigma_1)\theta_{yi}+E(m_2,\sigma_2))+E(m_3,\sigma_3))}+I}, \\
I &= \sum_{j=1,j\neq y_i}^{n} e^{s\cot\theta_{ji}},
\end{aligned}
\tag{4.9}
$$

where $E(\overline{x},\sigma)$ represents the normal function that returns a random value from a Gaussian distribution with mean $\overline{x}$ and standard deviation σ. This ensemble method incorporates different margin values and standard deviations for cotangent calculations, resulting in improved model performance.

Alternatively, a combination of cotangent and cosine functions can be used:

$$
\begin{aligned}
&L = \alpha L_{\cot} + \beta L_{\cos}, \\
&L_{\cot} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cot(E(m_1,\sigma_1)\theta_{yi}+E(m_2,\sigma_2))+E(m_3,\sigma_3))}}{e^{s(\cot(E(m_1,\sigma_1)\theta_{yi}+E(m_2,\sigma_2))+E(m_3,\sigma_3))}+I}, \\
&L_{\cos} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(E(m_1,\sigma_1)\theta_{yi}+E(m_2,\sigma_2))+E(m_3,\sigma_3))}}{e^{s(\cos(E(m_1,\sigma_1)\theta_{yi}+E(m_2,\sigma_2))+E(m_3,\sigma_3))}+I}, \\
&\text{and } I = \sum_{j=1,j\neq y_i}^{n} e^{s\cot\theta_{ji}}.
\end{aligned}
\tag{4.10}
$$

Lastly, the training process can be divided into multiple stages, where each stage utilizes a different loss function. This staged training approach can further improve the model's performance by gradually refining the decision boundaries.

In summary, the Large Margin Cotangent Loss (LMCot) introduces the cotangent function as a replacement for the cosine function in loss calculations, enabling better optimization and enhanced performance in verification and identification tasks. The use of ensemble methods and staged training can further enhance the model's capabilities.

4.4 Applications of Siamese Networks

Siamese networks have found numerous applications in various domains due to their ability to measure similarity and compare pairs of instances. Some notable applications include:

4.4.1 Authentication Applications

Siamese networks have been widely used in authentication applications such as face recognition, fingerprint identification, voice authentication, and signature

verification. These tasks require comparing biometric features of an individual against a database to verify their identity. Siamese networks excel in this area by learning the representations of biometric data and providing accurate similarity measurements for authentication purposes.

4.4.2 Zero-Shot, One-Shot, and Few-Shot Learning

Zero-shot, one-shot, and few-shot learning scenarios involve training a model with limited labeled data or even without any labeled examples. Siamese networks have proven to be effective in such settings. By learning a metric space that captures the similarity between instances, siamese networks can leverage the available labeled examples to generalize to unseen classes or perform accurate classification with minimal training data.

4.4.3 Text-Image Matching

Matching text and image data is another application where siamese networks have shown promise. By learning the representations of both textual and visual features, siamese networks can measure the semantic similarity between text and image pairs. This capability is valuable in tasks such as cross-modal retrieval, content-based image search, and image captioning.

4.4.4 Content-Based Recommendation Systems

Siamese networks can be employed in content-based recommendation systems, where the goal is to suggest relevant items to users based on their preferences or past behavior. By learning the similarity between user profiles and item representations, siamese networks can generate accurate recommendations that align with users' preferences and interests.

These applications represent just a few examples of how siamese networks can be utilized. Their ability to compare and measure similarity between instances across different domains opens up possibilities for various tasks, ranging from biometrics to multimedia analysis and personalized recommendation systems.

4.4.5 Bio-Hasing

BioHashing is a biometric feature extraction technique that is used to transform biometric data into a compact and secure binary representation. It was introduced by Teoh et al. [23] in 2005 as a method to protect sensitive biometric information while still allowing for efficient and accurate matching.

The main idea behind BioHashing is to use a random projection matrix to project the biometric data onto a lower-dimensional space. This projection is

specifically designed to preserve the similarity information between biometric samples while minimizing the information leakage about the original data.

BioHashing is considered an application of similarity learning, which is the task of learning a similarity function that can measure the similarity or dissimilarity between pairs of data points. In the case of BioHashing, the goal is to learn a similarity function that can accurately measure the similarity between biometric samples.

Siamese networks, which are a type of deep neural network architecture, are commonly used in BioHashing. Siamese networks consist of two identical subnetworks that share weights and learn to extract meaningful features from biometric data. These networks are trained using pairs of biometric samples, where the network learns to minimize the distance between similar samples and maximize the distance between dissimilar samples.

By leveraging siamese networks and similarity learning, BioHashing can generate secure binary representations of biometric data that are resilient to attacks and protect the privacy of individuals. These binary representations can be used for efficient and accurate matching in various biometric applications, such as fingerprint recognition, face recognition, and iris recognition, while minimizing the risk of unauthorized access to sensitive biometric information.

4.5 Conclusion

In conclusion, while Softmax loss is a commonly used loss function for classification problems, it has limitations when it comes to handling dynamic label sets. Siamese Networks provide an alternative approach that allows for the incorporation of new labels without the need for retraining the entire model. By extracting feature vectors and comparing them with existing ones in the database, Siamese Networks can make decisions based on similarity rather than fixed class labels.

This capability becomes especially valuable in domains like healthcare, where new diseases and conditions continuously emerge. Siamese Networks enable long-term lifetime learning by addressing the challenge of adding new labels without requiring extensive retraining. This flexibility and adaptability make Siamese Networks well-suited for evolving applications.

Furthermore, Siamese Networks not only offer high performance in classification tasks but also provide enhanced security by producing feature vectors instead of probability vectors for each label. This makes Siamese Networks more robust and resistant to adversarial attacks.

Additionally, Siamese Networks offer improved interpretability. With feature vectors as outputs, it becomes possible to compute pairwise distances and gain insight into the exact conditions of patients or instances under consideration. This level of granularity enhances the explanatory power of Siamese Networks.

An intriguing avenue for future research is the combination of Siamese Networks with federated learning. This approach presents the opportunity to propagate feature vector changes between models instead of relying solely on gradient descent, potentially leading to cost savings and improved performance. To the best of our knowledge, this particular combination has not been explored extensively, leaving room for novel investigations and discoveries in this area.

In summary, Siamese Networks provide a powerful and versatile framework for handling dynamic label sets, offering both excellent performance and unique advantages such as adaptability, security, interpretability, and the potential for synergies with federated learning. Further exploration and research in these areas hold great promise for advancing the field of machine learning and its applications.

Chapter 5

Result

5.1 COVID-19 Radiography Database

5.1.1 CNN VGG19

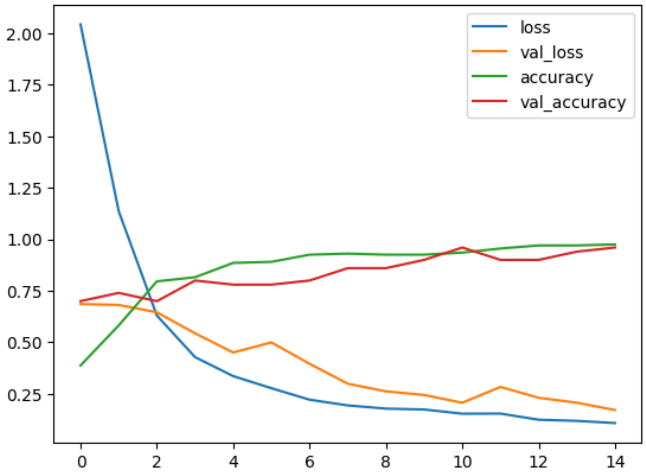


Figure 5.1: CNN VGG19 Training curve

```
7/7 [==============================] - 195s 28s/step - loss: 2.0420 - accuracy: 0.3881 - val_loss: 0.6861 - val_accuracy: 0.7000
Epoch 2/15
7/7 [==============================] - 193s 28s/step - loss: 1.1362 - accuracy: 0.5821 - val_loss: 0.6816 - val_accuracy: 0.7400
Epoch 3/15
7/7 [==============================] - 194s 31s/step - loss: 0.6301 - accuracy: 0.7960 - val_loss: 0.6459 - val_accuracy: 0.7000
Epoch 4/15
7/7 [==============================] - 191s 28s/step - loss: 0.4285 - accuracy: 0.8159 - val_loss: 0.5440 - val_accuracy: 0.8000
Epoch 5/15
7/7 [==============================] - 198s 29s/step - loss: 0.3364 - accuracy: 0.8856 - val_loss: 0.4510 - val_accuracy: 0.7800
Epoch 6/15
7/7 [==============================] - 190s 27s/step - loss: 0.2778 - accuracy: 0.8905 - val_loss: 0.4998 - val_accuracy: 0.7800
Epoch 7/15
7/7 [==============================] - 191s 27s/step - loss: 0.2220 - accuracy: 0.9254 - val_loss: 0.3963 - val_accuracy: 0.8000
Epoch 8/15
7/7 [==============================] - 197s 29s/step - loss: 0.1943 - accuracy: 0.9303 - val_loss: 0.2995 - val_accuracy: 0.8600
Epoch 9/15
7/7 [==============================] - 192s 31s/step - loss: 0.1788 - accuracy: 0.9254 - val_loss: 0.2626 - val_accuracy: 0.8600
Epoch 10/15
7/7 [==============================] - 192s 31s/step - loss: 0.1745 - accuracy: 0.9254 - val_loss: 0.2449 - val_accuracy: 0.9000
Epoch 11/15
7/7 [==============================] - 196s 29s/step - loss: 0.1542 - accuracy: 0.9353 - val_loss: 0.2073 - val_accuracy: 0.9600
Epoch 12/15
7/7 [==============================] - 190s 27s/step - loss: 0.1547 - accuracy: 0.9552 - val_loss: 0.2840 - val_accuracy: 0.9000
Epoch 13/15
7/7 [==============================] - 191s 31s/step - loss: 0.1248 - accuracy: 0.9701 - val_loss: 0.2314 - val_accuracy: 0.9000
Epoch 14/15
7/7 [==============================] - 232s 37s/step - loss: 0.1193 - accuracy: 0.9701 - val_loss: 0.2076 - val_accuracy: 0.9400
Epoch 15/15
7/7 [==============================] - 190s 28s/step - loss: 0.1088 - accuracy: 0.9751 - val_loss: 0.1721 - val_accuracy: 0.9600
```

Figure 5.2: Training result

5.1.2 ViT (Transformer)

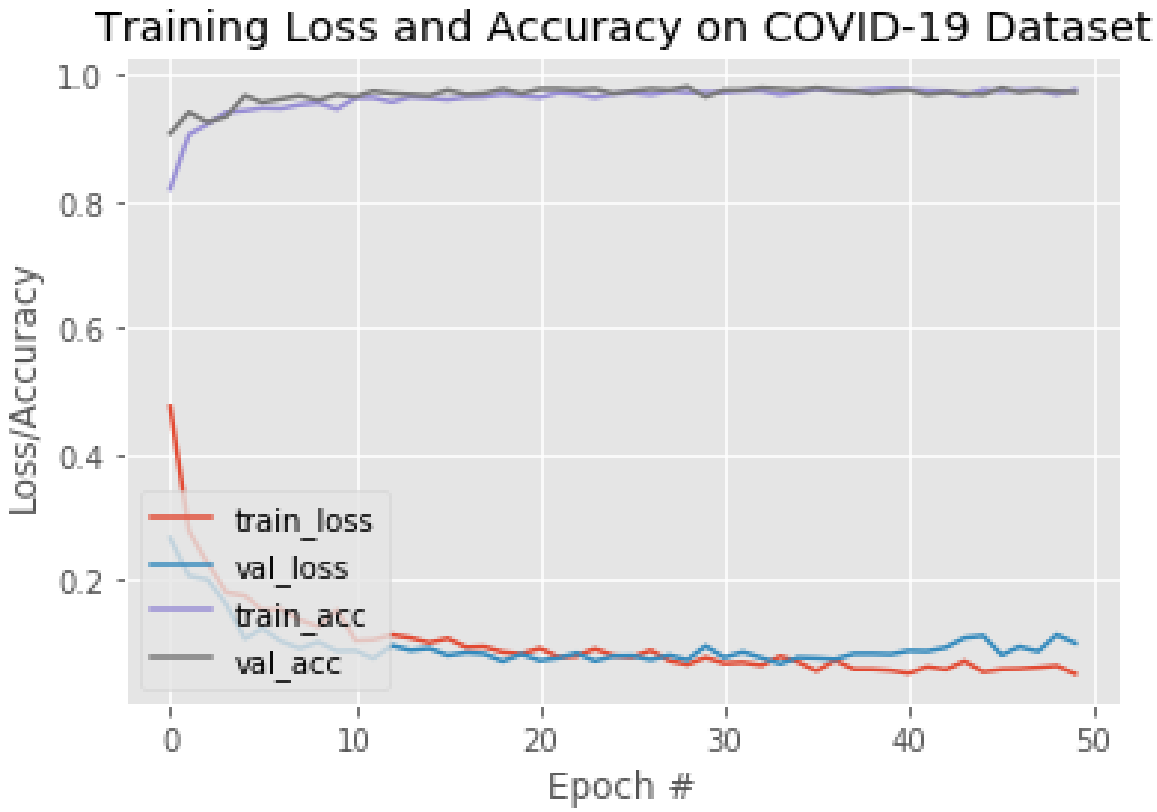


Figure 5.3: Training curve

	precision	recall	f1-score	support
Covid-19	0.99837	1.00000	0.99918	1225
Normal	0.98330	0.98788	0.98559	1073
Pneumonia	0.98877	0.98234	0.98555	1076
accuracy			0.99052	3374
macro avg	0.99015	0.99008	0.99011	3374
weighted avg	0.99052	0.99052	0.99051	3374

Table 5.1: Result table on validation

5.2 Similarity Learning

	Macro F1-Score	Micro F1-Score
Top 1	0.6216	0.6801
Top 2	0.6077	0.6532
Top 3	0.5959	0.6714
Our	**0.6295**	**0.6842**

Table 5.2: We evaluated our solution on the DFU dataset [24]. Our approach utilizes a Siamese model with a backbone composed of Bi-Linear (EfficientNetV2S CNN + BeITV2Base transformer) architecture and incorporates test time augmentation (TTA) techniques [25].

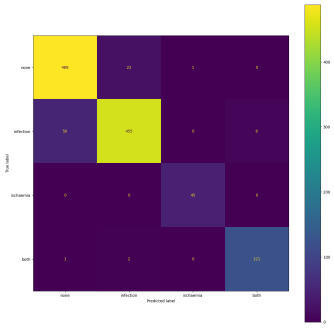


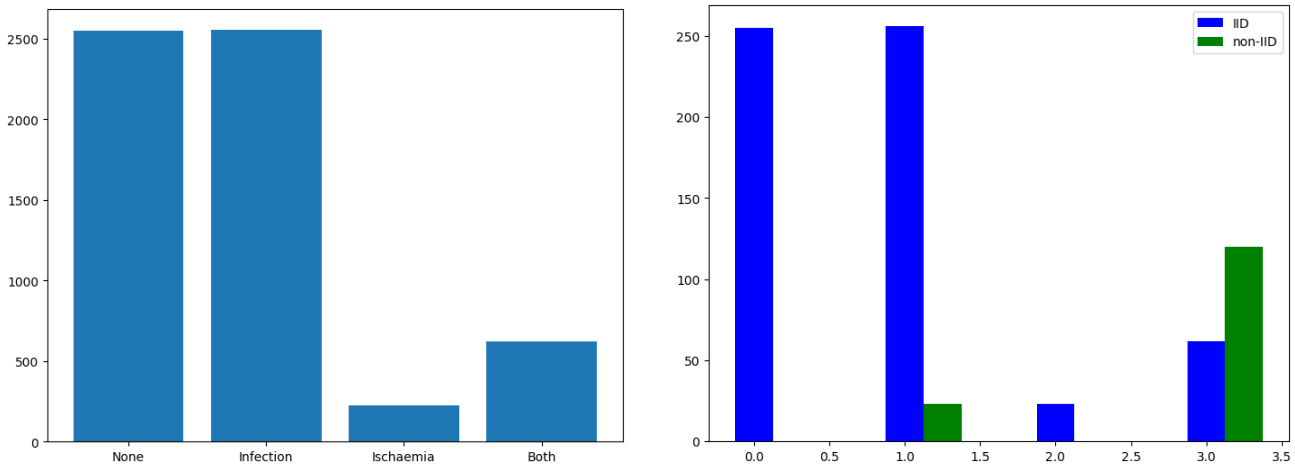
Figure 5.4: Validation confusion matrix

	precision	recall	f1-score	support
None	0.91	0.95	0.93	511
Infection	0.95	0.89	0.92	511
Ischaemia	0.98	1.00	0.99	45
Both	0.95	0.98	0.96	124
accuracy			0.93	1191
macro avg	0.95	0.96	0.95	1191
weighted avg	0.93	0.93	0.93	1191

Table 5.3: Result table on validation

5.3 Federated Learning

5.3.1 Data distribution

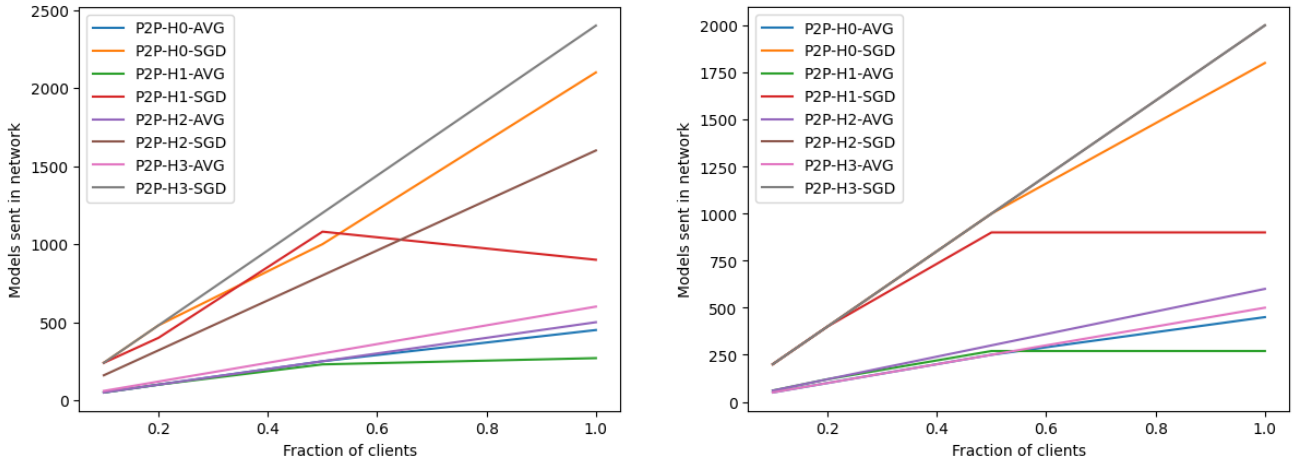


(a) Distribution of training data

(b) Data distribution of 1^{st} client

Figure 5.5: Data distribution

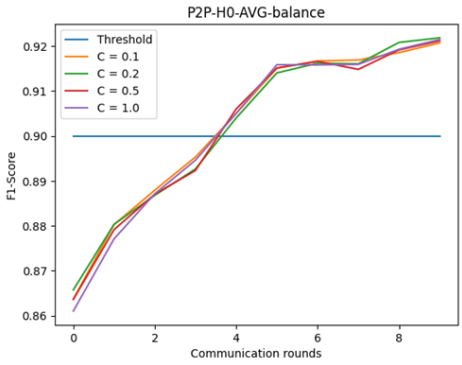
5.3.2 Compare heuristics



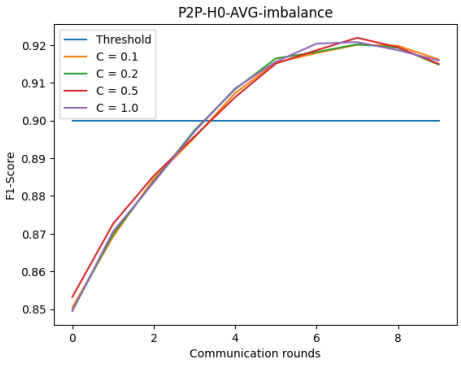
(a) IID

(b) non-IID data

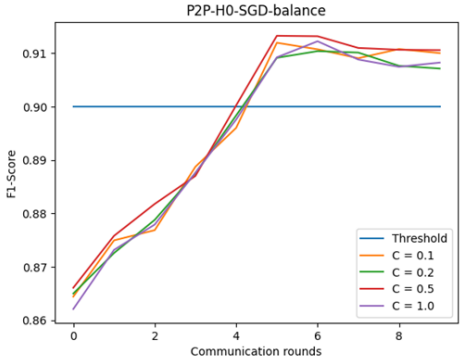
Figure 5.6: A comparison of Fedavg to FedavgP2P considering models sent in the network when 90% model accuracy had been reached. C is the fraction of clients the central server (or every client with FedavgP2P) had received updates from. According to the graph above, it can be said that Heuristic 1 has the best learning ability when we increase the factor C.



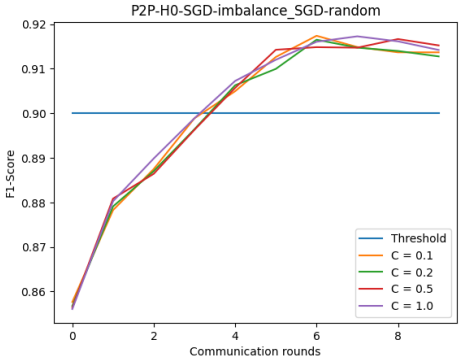
(a) P2P H0 FedAVG IID



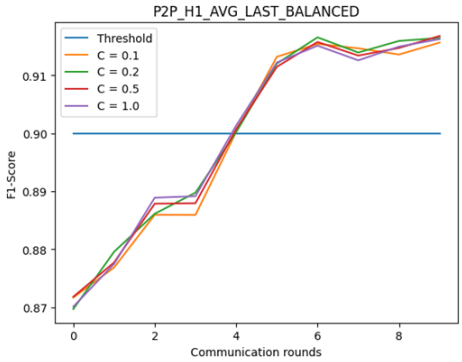
(b) P2P H0 FedAVG non-IID



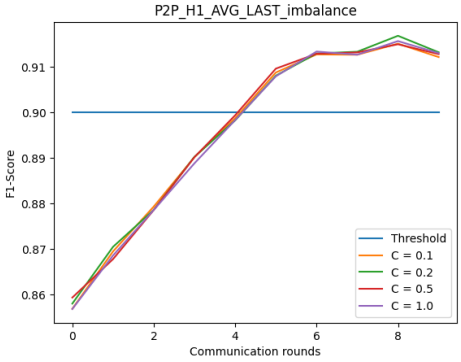
(a) P2P H0 FedSGD IID



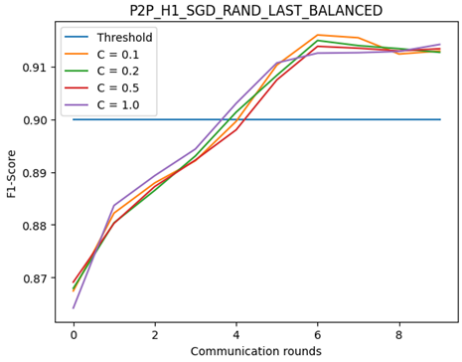
(b) P2P H0 FedSGD non-IID



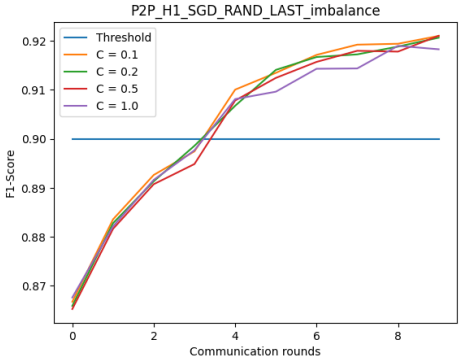
(a) P2P H1 FedAVG IID



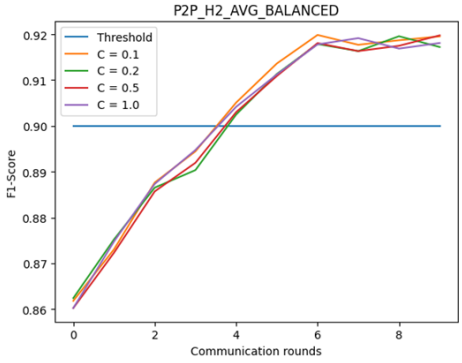
(b) P2P H1 FedAVG non-IID



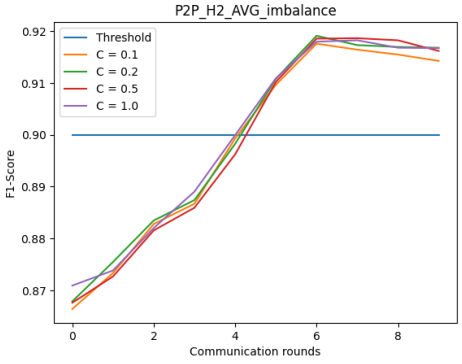
(a) P2P H1 FedSGD IID



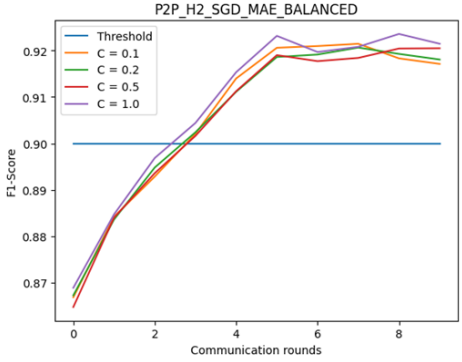
(b) P2P H1 FedSGD non-IID



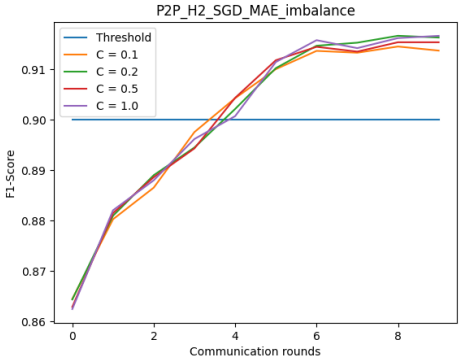
(a) P2P H2 FedAVG IID



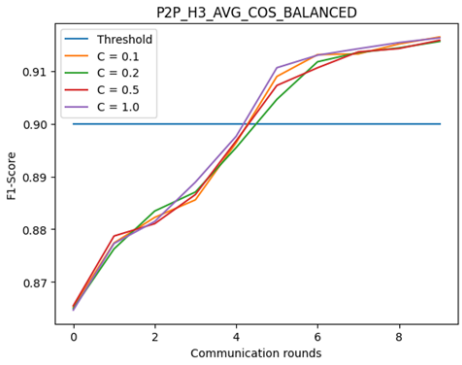
(b) P2P H2 FedAVG non-IID



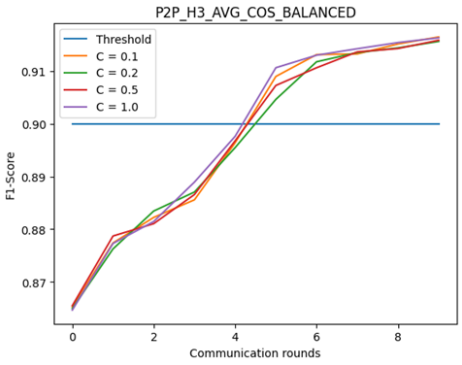
(a) P2P H2 FedSGD IID



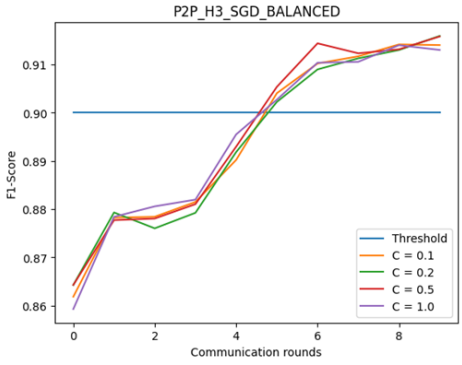
(b) P2P H2 FedSGD non-IID



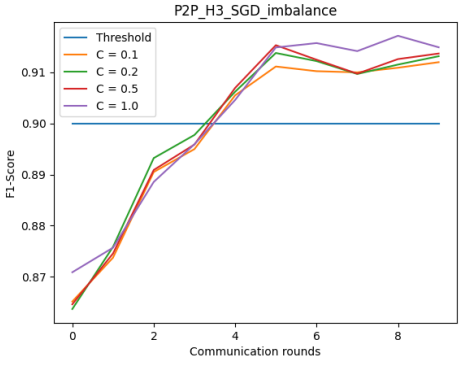
(a) P2P H3 FedAVG IID



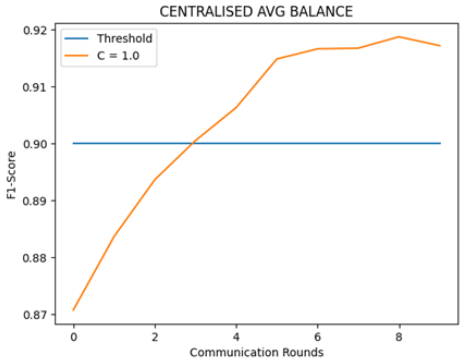
(b) P2P H3 FedAVG non-IID



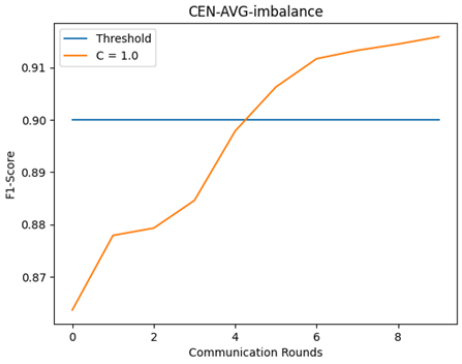
(a) P2P H3 FedSGD IID



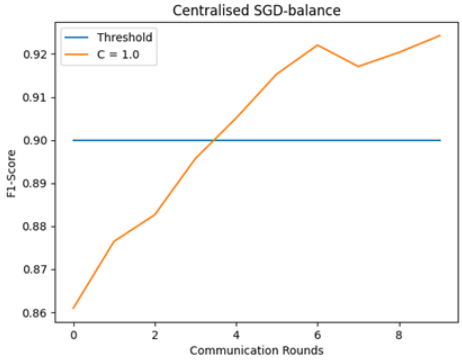
(b) P2P H3 FedSGD non-IID



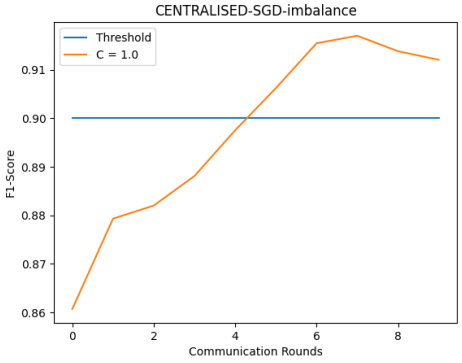
(a) Centralised FedAVG IID



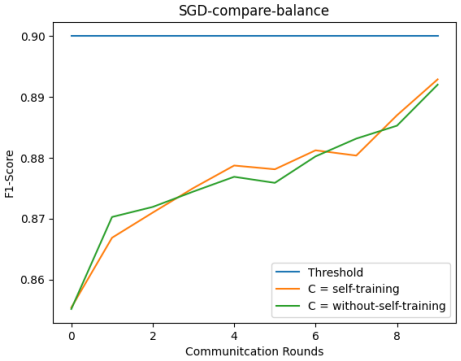
(b) Centralised FedAVG non-IID



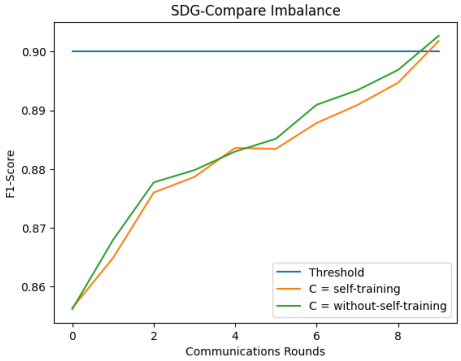
(a) Centralised FedSGD IID



(b) Centralised FedSGD non-IID



(a) IID



(b) non-IID

Figure 5.7: Compare a model that uses gradient vectors from its neighbors and both its gradient vectors (orange) and a model that uses only gradient vectors from its neighbors (green). Here we set the number of steps per round to 1.

5.4 Our related works

We have some achievements in related fields.

- Medical: BiLinear CNNs Model and Test Time Augmentation for Screening Viral and COVID-19 Pneumonia [25].
- Siamese Neural Network (SNN):
  - Large Margin Cotangent Loss for Deep Similarity Learning [6].
  - Silver medal **Google Landmark Retrieval 2021** [1].
  - Silvr medal **Google Landmark Recognition 2021**[2].
  - Bronze medal **Google Landmark Recognition 2020**[3].
  - HCMUS at MediaEval 2020: Image-Text Fusion for Automatic News-Images Re-Matching [10].
  - Remove Non-landmark to Improve Landmark Recognition [26].

[1] https://www.kaggle.com/c/landmark-retrieval-2021

[2] https://www.kaggle.com/competitions/landmark-recognition-2021

[3] https://www.kaggle.com/c/landmark-recognition-2020

Bibliography

[1] M. E. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. Al Emadi, *et al.*, "Can ai help in screening viral and covid-19 pneumonia?," *Ieee Access*, vol. 8, pp. 132665–132676, 2020.

[2] T. Rahman, A. Khandakar, Y. Qiblawey, A. Tahir, S. Kiranyaz, S. B. A. Kashem, M. T. Islam, S. Al Maadeed, S. M. Zughaier, M. S. Khan, *et al.*, "Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images," *Computers in biology and medicine*, vol. 132, p. 104319, 2021.

[3] A. Ajit, K. Acharya, and A. Samanta, "A review of convolutional neural networks," in *2020 international conference on emerging trends in information technology and engineering (ic-ETITE)*, pp. 1–5, IEEE, 2020.

[4] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, pp. 1–6, Ieee, 2017.

[5] D. Mäenpää, "Towards peer-to-peer federated learning: Algorithms and comparisons to centralized federated learning," 2021.

[6] A.-K. Duong, H.-L. Nguyen, and T.-T. Truong, "Large margin cotangent loss for deep similarity learning," in *2022 International Conference on Advanced Computing and Analytics (ACOMPA)*, pp. 40–47, IEEE, 2022.

[7] A.-K. Duong, H.-L. Nguyen, and T.-T. Truong, "Enhanced face authentication with separate loss functions," *arXiv preprint arXiv:2302.11427*, 2023.

[8] D. Chicco, "Siamese neural networks: An overview," *Artificial Neural Networks*, pp. 73–94, 2021.

[9] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

[10] T. Nguyen-Quang, T.-D. H. Nguyen, T.-L. Nguyen-Ho, A.-K. Duong, N. Hoang-Xuan, V.-T. Nguyen-Truong, H.-D. Nguyen, and M.-T. Tran, "Hcmus at mediaeval 2020: Image-text fusion for automatic news-images re-matching," 2020.

[11] D. Parikh and K. Grauman, "Relative attributes," in *2011 International Conference on Computer Vision*, pp. 503–510, IEEE, 2011.

[12] A. Dubey, O. Gupta, P. Guo, R. Raskar, R. Farrell, and N. Naik, "Pairwise confusion for fine-grained visual classification," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 70–86, 2018.

[13] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[14] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 212–220, 2017.

[15] X. Wang, S. Wang, S. Zhang, T. Fu, H. Shi, and T. Mei, "Support vector guided softmax loss for face recognition," *arXiv preprint arXiv:1812.11317*, 2018.

[16] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5265–5274, 2018.

[17] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4690–4699, 2019.

[18] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, "Adacos: Adaptively scaling cosine logits for effectively learning deep face representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10823–10832, 2019.

[19] F. Boutros, N. Damer, F. Kirchbuchner, and A. Kuijper, "Elasticface: Elastic margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1578–1587, 2022.

[20] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, vol. 6, 1993.

[21] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.

[22] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "Normface: L2 hypersphere embedding for face verification," in *Proceedings of the 25th ACM international conference on Multimedia*, pp. 1041–1049, 2017.

[23] A. B. Teoh, A. Goh, and D. C. Ngo, "Random multispace quantization as an analytic mechanism for biohashing of biometric and random identity inputs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 12, pp. 1892–1901, 2006.

[24] J. M. Pappachan, C. O'Shea, D. G. Armstrong, and M. H. Yap, "Diabetic foot ulcer grand challenge 2021: Evaluation and summary," *Diabetic Foot Ulcers Grand Challenge: Second Challenge, DFUC 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, September 27, 2021: Proceedings*, vol. 13183, p. 90, 2022.

[25] A.-K. Duong and V.-H. Huynh, "Bilinear cnns model and test time augmentation for screening viral and covid-19 pneumonia," *SSICT2020*, p. 112, 2020.

[26] A.-K. Duong, P. SeshuRaju, and C. X. Nam, "Remove non-landmark to improve landmark recognition," *VNUHCM-US-Conf'20*, 2020.

[27] J. Bell, "What is machine learning?," *Machine Learning and the City: Applications in Architecture and Urban Design*, pp. 207–216, 2022.

[28] M. A. Ahmad, C. Eckert, and A. Teredesai, "Interpretable machine learning in healthcare," in *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, pp. 559–560, 2018.

[29] W. B. Tesfay, P. Hofmann, T. Nakamura, S. Kiyomoto, and J. Serna, "I read but don't agree: Privacy policy benchmarking using machine learning and the eu gdpr," in *Companion Proceedings of the The Web Conference 2018*, pp. 163–166, 2018.

[30] N. Johnson, S. Kotz, and N. Balakrishnan, "Normal distributions," *Continuous univariate distributions*, vol. 1, pp. 156–157, 1987.

[31] T. O. Ayodele, "Types of machine learning algorithms," *New advances in machine learning*, vol. 3, pp. 19–48, 2010.

[32] F. Osisanwo, J. Akinsola, O. Awodele, J. Hinmikaiye, O. Olakanmi, and J. Akinjobi, "Supervised machine learning algorithms: classification and comparison," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp. 128–138, 2017.

[33] D. of Computer Science, "Charles elkan," *Technical Report No. CS97-557, Department of Computer Science and Engineering, University of California, San Diego, September 1997.*, 1997.

[34] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[35] D. A. Pisner and D. M. Schnyer, "Support vector machine," in *Machine learning*, pp. 101–121, Elsevier, 2020.

[36] D. R. Hush and B. G. Horne, "Progress in supervised neural networks," *IEEE signal processing magazine*, vol. 10, no. 1, pp. 8–39, 1993.

[37] Z. Ghahramani, "Unsupervised learning," in *Summer school on machine learning*, pp. 72–112, Springer, 2004.

[38] M. E. Celebi and K. Aydin, *Unsupervised learning algorithms*. Springer, 2016.

[39] 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*, pp. 248–255, 2009.

[40] K. P. Sinaga and M.-S. Yang, "Unsupervised k-means clustering algorithm," *IEEE access*, vol. 8, pp. 80716–80727, 2020.

[41] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[42] M. Reddy, V. Makara, and R. Satish, "Divisive hierarchical clustering with k-means and agglomerative hierarchical clustering," *Int J of Comp Science Trands and Tech (IJCST)*, vol. 5, no. 5, pp. 5–11, 2017.

[43] M. F. A. Hady and F. Schwenker, "Semi-supervised learning," *Handbook on Neural Information Processing*, pp. 215–239, 2013.

[44] X. J. Zhu, "Semi-supervised learning literature survey," 2005.

[45] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.