الجمهورية الجزَائِرية الديمقرَاطية الشعبية
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزَارة التعليم العَالي و البحث العلمي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جَامعة عمَار ثليجي بِالأَغوَاط
UNIVERSITY OF AMAR TELIDJI LAGHOUAT

كلية العلُوم
FACULTY OF SCIENCES
قسم الَاعلَام الَالي
DEPARTMENT OF COMPUTER SCIENCES

Master Thesis

Domain: Mathematics and Computer Science

Field : Computer Science

Option: Networks, Systems and Distributed Applications

Title

Towards an Efficient in-Network Caching Using Federated Learning

by:

Manel Habib **Malek safa Djekidel**

Defended Publicly in Front of the Jury Composed of:

President: Dr. Fatima Zohra Bousbaa (Universiy of Laghouat)

Examiner: Dr. Abdelmadjid Benarfa (Universiy of Laghouat)

Examiner: Dr. Lakhdar Kamel Ouladdjedid (Universiy of Laghouat)

Advisor: Dr. Chaker Abdelaziz Kerrache (Universiy of Laghouat)

Academic Year 2022/2023

Abstract

In today's digital age, the Internet has experienced a remarkable growth accompanied by an exponential increase in both the diversity of available content and the number of users. As a consequence, the demand for server resources and the volume of server requests have surged significantly. This places significant strain on servers, diminishing their ability to handle user demands effectively. To alleviate this issue, caching is employed to store frequently requested content in memory that is closer to users. However, determining which content should be cached poses a challenge. Efficient cache management plays a vital role in enhancing data access speed and overall efficiency. This challenge has been extensively studied and applied in the context of federated learning engineering, where effective cache management techniques are crucial for optimizing the performance of distributed machine learning models. By addressing cache management challenges, researchers aim to improve scalability, efficiency, and overall system performance, ultimately enhancing the effectiveness of federated learning methodologies. In our research, we conducted a study on the topic of enhancing network caching efficiency by implementing federated learning. Our study involved the creation of four users, each of whom assigned a portion of a movie database that included movie ratings. Our goal was to identify the most popular movies using artificiel neural network and cache them for each user, thus improving delivery services within the network by bringing these movies closer to the respective users.

Keywords: Federated learning ,cache management , cache decision , Caching in network, Placement strategies, Replacement strategies , Machine learning.

Résumé

À l'ère numérique d'aujourd'hui, Internet a connu une croissance remarquable accompagnée d'une augmentation exponentielle à la fois de la diversité des contenus disponibles et du nombre d'utilisateurs. En conséquence, la demande de ressources serveur et le volume des requêtes serveur ont considérablement augmenté. Cela impose une pression importante sur les serveurs, diminuant leur capacité à gérer efficacement les demandes des utilisateurs. Pour atténuer ce problème, la mise en cache est utilisée pour stocker le contenu fréquemment demandé dans une mémoire plus proche des utilisateurs. Cependant, déterminer quel contenu doit être mis en cache pose un défi. Une gestion efficace du cache joue un rôle essentiel dans l'amélioration de la vitesse d'accès aux données et de l'efficacité globale. Ce défi a été largement étudié et appliqué dans le contexte de l'ingénierie d'apprentissage fédéré, où des techniques de gestion de cache efficaces sont cruciales pour optimiser les performances des modèles d'apprentissage automatique distribués. En relevant les défis de la gestion du cache, les chercheurs visent à améliorer l'évolutivité, l'efficacité et les performances globales du système, améliorant ainsi l'efficacité des méthodologies d'apprentissage fédéré. Dans notre recherche, nous avons mené une étude sur le thème de l'amélioration de l'efficacité de la mise en cache réseau en mettant en œuvre l'apprentissage fédéré. Notre étude a impliqué la création de quatre utilisateurs, chacun d'eux s'étant vu attribuer une partie d'une base de données de films comprenant des classements de films. Notre objectif était d'identifier les films les plus populaires à l'aide d'un réseau neuronal artificiel et de les mettre en cache pour chaque utilisateur, améliorant ainsi les services de diffusion au sein du réseau en rapprochant ces films des utilisateurs respectifs..

Mot-clé: Apprentissage fédéré, gestion de cache, décision de cache, mise en cache dans le réseau, stratégies de placement, stratégies de remplacement, apprentissage automatique.

ملخص

في العصر الرقمي اليوم ، شهدت الإنترنت نموًا ملحوظًا مصحوبًا بزيادة هائلة في كل من تنوع المحتوى المتاح وعدد المستخدمين. نتيجة لذلك ، ارتفع الطلب على وحدات التخزين وحجم طلبات بشكل كبير. يضع هذا ضغطًا كبيرًا على وحدات التخزين ، مما يقلل من قدرتها على التعامل مع طلبات المستخدم بشكل فعال. للتخفيف من هذه المشكلة ، يتم استخدام التخزين المؤقت لتخزين المحتوى المطلوب بشكل متكرر في الذاكرة القريبة من المستخدمين. ومع ذلك ، فإن تحديد المحتوى الذي يجب تخزينه مؤقتًا يمثل تحديًا. تلعب إدارة ذاكرة التخزين المؤقت الفعالة دورًا حيويًا في تعزيز سرعة الوصول إلى البيانات والكفاءة العامة. تمت دراسة هذا التحدي وتطبيقه على نطاق واسع في سياق هندسة التعلم الاتحادي ، حيث تعد تقنيات إدارة ذاكرة التخزين المؤقت الفعالة ضرورية لتحسين أداء نماذج التعلم الآلي الموزعة. من خلال معالجة تحديات إدارة ذاكرة التخزين المؤقت ، يهدف الباحثون إلى تحسين قابلية التوسع والكفاءة والأداء العام للنظام ، مما يؤدي في النهاية إلى تعزيز فعالية منهجيات التعلم الاتحادي. في بحثنا ، أجرينا دراسة حول موضوع تعزيز كفاءة التخزين المؤقت للشبكة من خلال تنفيذ التعلم الفيدرالي. تضمنت دراستنا إنشاء أربعة مستخدمين ، خصص كل منهم جزءًا من قاعدة بيانات الأفلام التي تضمنت تصنيفات الأفلام. كان هدفنا هو تحديد الأفلام الأكثر شيوعًا باستخدام الشبكة العصبية الاصطناعية وتخزينها مؤقتًا لكل مستخدم ، وبالتالي تحسين خدمات التوصيل داخل الشبكة من خلال تقريب هذه الأفلام من المستخدمين المعنيين.

:كلمات مفتاحية التعلم الموحد ، إدارة ذاكرة التخزين المؤقت ، قرار ذاكرة التخزين المؤقت ، التخزين المؤقت في الشبكة ، إستراتيجيات الموضع ، إستراتيجيات الاستبدال ، التعلم الآلي.

Dedication

I offer this humble work as a dedication to the two remarkable individuals who have consistently made sacrifices to ensure my success.

To my beloved mother naziha and father rabah, no dedication can adequately express my gratitude for the kindness, love, and unwavering support they have shown me throughout my academic journey. They are truly treasures of goodness and generosity. May God watch over them.

To my dear brothers and my sister , all members of my family, and to all my cherished friends who have supported me throughout my studies.

Additionally, I extend my gratitude to the faculty members and mentors who have played a pivotal role in my academic development. Your dedication to education, passion for knowledge, and willingness to share your expertise have shaped me into a more well-rounded individual.

To my friends and classmates, thank you for your camaraderie, stimulating discussions, and continuous motivation. Your presence has made this academic journey more enjoyable and memorable. We have shared countless late nights, brainstorming sessions, and moments of triumph, and I am grateful for the lifelong friendships forged during this time.

Thanks everyone

Malak Safa Djekidel

Dedication

I dedicate this work to my dear parents, for their support, their patience and their love. who gave me the strength to continue my studies. To my brother and sister to whom I wish the success. To all my best friends, the list goes on. To all my big family. To all my teachers and special dedicated to my dear friend Malak Safa Djekidel Thank you for everything. I am grateful for your unwavering friendship and the beautiful memories we've created together.

Manel Habib

Acknowledgements

We thank ALLAH, who allowed us to come this far. A big thank you to our supervisor, Dr. Chaker Abdelaziz Kerrache, for his understanding and kindness throughout our the completion of our research project. We would like to express our heartfelt appreciation to all our colleagues., speakers, and everyone who, through their words, writings, advice, and critiques, guided our thoughts, met us, and answered our questions during our research. We would like to thank everyone who participated directly or indirectly. We also thank the department head, Dr. Mohamed Elhabib MAICHA, and all the teachers of the computer science department.

Contents

List of Figures

Abbreviations

IOT *Internet of Things*

UE *User Equipment*

CSP *Cloud Service Provider*

LCE *Leave Copy Eevrywhere*

LCD *Leave Copy Down*

PC *Probabilistic Caching*

BTW *BeTweenness Centrality*

CS *Content Store*

LRU *Least Recently Used*

FIFO *First in First Out*

MRU *Most Recently Used*

SVM *Support Vector Machine*

RBF *Radial Basis Function*

PDM *Product Data Managemenr*

MDP *Markov Decision Process*

NN *Neural Network*

ANN *Artificial Neural Network*

FFNN *Feedforward Neural Network*

FBNN *Feed-backward neural networ*

RNN *Recurrent Neural Network*

MLP *Multilayer Perceptron Network*

RBFN *Radial Basis Function Networ*

CNN *Convolutional Neural Network*

RNN *Recurrent Neural Network*

FL *Federated Learning*

SGD *Stochastic Gradient Descent*

MAB *Multi-armed Bandit*

FPCC *Federated Learning based Proactive Content Caching*

BS *Base Station*

MES *Mean Squared Error*

Chapter 1

Introduction

In today's data-driven world, managing caches properly is extremely difficult due to the exponential expansion of data and the requirement for real-time processing. By keeping frequently accessed material closer to the consumers, cache management is essential for lowering latency and optimizing resource usage. Cache management is becoming even more crucial in the context of fog nodes, which serve as intermediate compute and storage nodes between the cloud and end devices, as a result of the development of cloud computing and edge computing.

1.1 Motivation

The goal of this study is to address the particular difficulties and needs associated with cache management in fog nodes, especially in the context of Federated Learning (FL). FL protects data privacy while enabling collaborative machine learning across various platforms. FL's distributed and dynamic nature, along with the resource limitations of fog nodes, necessitate the development of novel cache management strategies that can boost FL speed while lowering network overhead and protecting data privacy. Due to network traffic and data protection difficulties, it is impractical and usually wasteful to send all the data to a distant cloud[1].

1.2 Objectives

The main objectives of this study are:

1. To assess and comprehend the constraints of the cache management strategies utilized in conventional computing environments with regard to fog nodes.
2. To investigate how cache management and machine learning interact, with particular attention to FL in fog computing environments.
3. To Approximation the popular contents to the users.

1.3 Contributions

1. in-depth evaluation and comparison of the various cache management strategies used in conventional computing settings.
2. the investigation of cache management requirements and difficulties particular to FL in fog nodes
3. emphasizing the significance of taking cache management in fog nodes into account for effective and private collaborative learning.

1.4 Organization of the Thesis

Three major chapters make up this thesis. A summary of cache management strategies used in conventional computing settings is given in Chapter 1, along with a discussion of their advantages and disadvantages. In Chapter 2, machine learning methods and techniques are introduced, laying the groundwork for the study of cache management in FL that follows. The topic of cache management for FL is covered in Chapter 3, along with unique implementation techniques. The chapter also contains a review and analysis of the ideas put forth. The summary of the major discoveries, contributions.

Chapter 2

In-Network Cache Management

2.1 Introduction

the development of Internet of Things (IoT), Internet, various objects, including people, machines, things, are connected into information space in anywhere at any time generate a huge amount of data that continues to grow exponentially[2]. The applications used in these devices require access to the cloud for real-time processing. However, due to the volume of produced data at the UEs, it is impractical to send all the data to the cloud. In addition, the data privacy requirements for user data require local data processing whenever possible[3].
Edge computing is a computing paradigm where data is processed and analyzed at or near the source, rather than sent to a central location for processing. This includes deploying computing resources such as servers, storage devices, and network equipment at the network edge, closer to where data is produced or consumed[4].
Edge computing and cloud computing aren't jointly exclusive. In fact, they can be complementary. For example, edge computing can be used to process data in real-time at the edge, while the cloud can be used to store and manage large amounts of data, and to perform more complex processing and analysis on that data. By combining edge and cloud computing, organizations can create a hybrid computing environment that leverages the strengths of both approaches[4].
Cache management in edge computing refers to the strategies and techniques used to manage the caching of data at the edge of the network. Caching involves storing frequently accessed data closer to the end-user or device, rather than retrieving it from a remote server every time it is needed. This can help reduce network latency and improve application performance.In edge computing, where computing resources and data storage are distributed across multiple edge nodes, cache management becomes even more important to ensure optimal performance and efficient use of resources.

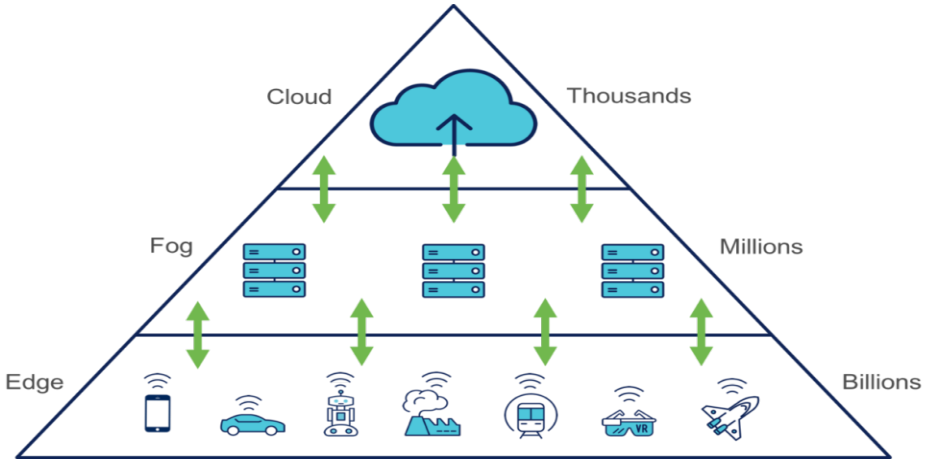

Figure 2.1: Cloud,Fog and Edge Computing

2.2 Related Work

Work 1 [5] **:** In this work *An sdn- based caching decision policy for video caching in information-centric networking.*The authors address the impact of multimedia services, particularly video-on-demand (VoD) services, on Internet traffic to improve video delivery and reduce network latency, they suggest merging software-defined networking (SDN) with information-centric networking (ICN). To define the caching choice, the study offers a 0-1 integer linear programming (ILP) issue. Unlike previous techniques, the authors take into consideration the problem's dynamic nature by accounting for time scale. They achieve more precise optimum solutions by tackling the dynamic ILP issue. Because the problem is NP-hard, they offer a unique SDN-based caching decision policy that makes use of ICN's in-network caching and the SDN controller's global view. Extensive testing shows that this strategy gives near-optimal answers in much less computing time. In terms of cache hit ratio, average number of hops, and overall performance, the SDN-based caching decision policy surpasses conventional ICN caching decision policies.

Work 2 [6] **:** In this article *cache content replacement scheme for information centric network.* Lal, Kumari Nidhi and Kumar, Anoj highlight the importance of Information Centric Networking (ICN) for accessing internet-based applications. As internet traffic increases, content-centric architectures have been implemented to meet the demands of content providers and users. These architectures incorporate network caches, significantly improving content delivery efficiency with speed and effectiveness. By utilizing ICN architectures, users can access data from nearby caches rather than downloading content directly from servers. Caching approaches make the location of data irrelevant, enabling users to access requested content regardless of how it is stored or transmitted. Numerous studies have focused on caching approaches within Content Centric Networks (CCN), as efficient caching plays a vital role in reducing delays and enhancing network performance. Additionally, an effective cache replacement scheme is needed to determine which content items should be cached and which should be evicted. This paper introduces a Cache Content Replacement (CCR) scheme for ICNs that evaluates content popularity. The CCR scheme's performance is evaluated based on cache hit ratio and cache load provision.

Work 3[7] **:** In this article *A cache management scheme for efficient content eviction and replication in cache networks.*Muhammad Bilal and Shin-Gak Kang address the need for efficient in-network caching solutions to adapt to the evolving demands of the internet. With the emergence of information-centric network (ICN) architecture, in-network caching has become a network-level solution. The unique characteristics of ICNs, such as dynamic cache states, high request arrival rates, limited cache sizes, and other factors, impose specific requirements on content eviction policies. These policies need to be fast and lightweight. In this paper, the authors propose two cache replication and eviction schemes, namely conditional leave cope everywhere (CLCE) and least frequent recently used (LFRU), designed specifically for ICN-based cache networks (CNs). The CLCE replication scheme reduces redundant content caching, thereby improving cache space utilization. LFRU combines elements of the least frequently used and least recently used schemes and is practical for rapidly changing cache networks like ICNs.

2.3 Cloud Computing

Cloud computing is a concept for offering "universal, on-demand, and convenient network access to a shared pool of configurable computing resources" that can be "rapidly provided and released with minimal administration effort or service provider engagement". A network-enabled solution that offers scalable, QoS-guaranteed services on demand that can be accessed through the Internet is referred to as cloud computing. By the use of cloud computing, resources can be shared online. A cloud service provider's infrastructure is used to distribute these resources (CSP). The cloud user can access scalable, universal resources on demand and on a pay-as-you-go basis. A degree of abstraction between the necessary computing resource and the underlying infrastructure, such as storage and network, is also made possible by cloud computing[8].

The client, the server, and the three main service delivery types are all included in cloud computing. The cloud customer make up the hardware or software abstraction layer that is used to connect to cloud services. The three main service delivery kinds are provided by the CSPs using servers. Platform as a Services, Infrastructure as a Service, and Software as a Service are the three main types of service delivery. Utilizing a cloud service provider's infrastructure,Platform as a Services offers a computing platform for consumer usage. Using the platform of a CSP, the user can create, test, and deploy an application. The user does not need to install the necessary software for this[8].

2.4 Edge Computing

Edge computing is a novel distributed computing model that blends cloud, network, end, and intelligence. Edge computing minimizes response times and bandwidth requirements by merging edge node compute, storage, and application capabilities with distributed cloud computing technologies. Edge computing provides effective capability support for edge applications such as telematics, intelligent manufacturing, and ultra-high definition video transmission.Edge computing is largely concerned with business situations such as real-time, brief-period data and local decision-making. It performs best when integrated with IoT infrastructures to provide various end users with effective and safe services[9].

Advantages of IoT Edge Computing Reference Architecture :

- Faster reaction times: When workloads are published at the edge and need local data input, processing may be done closer to the edge where the data is created. This reduces latency and improves responsiveness for real-time or near-real-time data analysis and processing.[10].
- Reduced bandwidth usage: By storing and processing data at the network's edge, edge computing makes it possible to significantly reduce both data volume and transmission distance while minimizing the network's impact from heavy traffic. This results in a local network's bandwidth usage.[10]
- Intelligent applications are those that extract and aggregate the data required through thoughtful analysis, extract and aggregate the data needed through intelligent analysis, eliminate irrelevant data, and realize automatic feedback and intelligent decision-making .[10]
- Security: By generating, processing, and storing data on edge devices, sensitive data leaks caused by data transfers between devices and the cloud are avoided. Furthermore, by keeping the data local to the device, the data's integrity is preserved.[10].
- Budget-friendly options: Large-scale data transmission over long distances results in high-cost consumption, and network bandwidth, data storage, and processing power all have upfront costs. Edge computing, in contrast, handles data processing chores locally, bringing down the overall cost of the IoT system.[10]

2.5 Fog Computing

Fog computing is a highly virtualized platform that connects endpoints to traditional cloud data centers, which are often but not always located close to the network's edge, to provide compute, storage, and networking services. Figure 1.2 depicts the envisioned information and computing infrastructure supporting future IoT applications and a demonstration of fog computing in action[4]. The foundational elements of both the Cloud and the Fog are resources for computing, storing, and networking. Nonetheless, the phrase "Edge of the Network" implies a lot of qualities that make the Fog a large extension of the Cloud. Let's list them together with references to inspiring examples[4].

- Low latency,area awareness and edge location. The Fog's origins can be traced to early concepts to provide endpoints with rich services at the edge of the network, such as games, streaming films, and augmented reality applications.
- distribution by location. The services and applications targeted by the Fog necessitate highly scattered installations, in sharp contrast to the more centralized Cloud. For instance, The Fog will actively participate in providing moving cars with high quality streaming via proxies and access points placed along highways and railroad tracks.
- As can be seen by sensor networks in general and the Smart Grid in particular, there are a very large number of nodes as a result of the widespread geo-distribution.

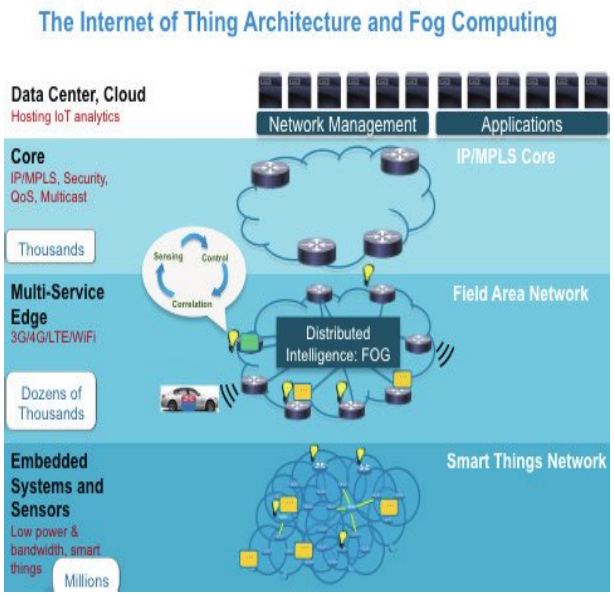

Figure 2.2: IOT and Fog Computing

- interaction in real time. Key Fog apps don't use batch processing but instead rely on real-time interactions.

2.6 Cache Decision

The decision-making strategy about the received data is one of the most crucial variables that increase the efficiency of the network because there isn't enough space in the cache to keep all of the requests from the users.A caching decision (Placement) policy plus a caching replacement policy make up a caching scheme. What content should be cached is determined by the caching decision policy, and what content should be removed from cache is determined by the caching replacement policy. The replacement strategy should be implemented as soon as possible for effectiveness[5].
Because the cache entity has a limited capacity, it is critical to forecast future popularity and store the most popular files ahead of time. Low cache efficiency is caused by the fact that conventional caching algorithms like most recently used (MRU), First-In-First-Out (FIFO), Least Recently Used (LRU), and Least Frequently Used (LFU) do not take future popularity of items into account. Since it is the fundamental barrier to proactive caching, several modern caching systems have focused on learning content popularity trends.[11]

2.7 Cache Placement

2.7.1 Introduction

The term "cache placement strategies" describes the techniques for determining where a cache should be installed in a network node or device in order to improve speed, reduce latency, and improve overall network communication efficiency. Caches play a significant role in network topologies by reducing the need to get data from distant and occasionally slower data sources by keeping frequently requested data closer to the end users or clients.

Network caching is absolutely essential when the network is dealing with high traffic volumes, capacity restrictions, or lengthy delays. To hasten information delivery, lessen network congestion, and enhance service quality, caches can be strategically placed across the network.
Cache placement solutions aim to increase cache hit rates, reduce cache misses, and maximize resource use in networks. The decision-making process is influenced by a number of variables, such as network architecture, user access patterns, popular material, cache size restrictions, and financial restraints.

2.7.2 Leave Copy Everywhere (LCE)

Leave Copy Everywhere (LCE) The majority of hierarchical caches now operate in this manner as the norm. When a hit happens at a level I cache or the origin server, a copy of the requested document is cached in all intermediate caches along the path from the location of the hit down to the requesting client[5].

2.7.3 Leave copy down (LCD)

Leave copy down (LCD) is to store the content in the immediate neighbour to the content's original creator. Although it is only one hop away from the creator, the cached content is still not ideal[12].

2.7.4 ProbCache

ProbCache (Probabilistic caching) which caches content at on-path nodes with a weighted probability p, and the probability is proportional to the number of hops from the caching node to the provider[12].With probability 1 - p, the replacement algorithm is triggered, and no copies are kept.LCE is the same as the probcache where p=1[13]. ProbCache can cache content to edge nodes at a faster speed, reducing cache redundancy to a certain extent[14].

2.7.5 Most Popular Caching Strategy (MPC)

Most Popular Caching Strategy (MPC) Each node counts locally how many times a given content name has been requested and records the pair (Content Name; Popularity Count) in a popularity table. A content name is marked as popular if it achieves a local Popularity Threshold, and if a node already has the content, it recommends to its neighbor nodes to cache it using a new Recommendation primitive. Depending on regional policies like resource availability, these proposal messages may be accepted or rejected[15].
A content's popularity can diminish over time following the suggestion process, thus the Popularity Count is reset in accordance with a Reset Value to prevent spamming neighbors with the same material[15].

2.7.6 Betweenness Centrality (Btw)

Betweenness Centrality (Btw) Each node already has the betweenness centrality parameter calculated. It counts the instances in which a node is a part of a path connecting every pair of nodes in a network topology. Only nodes with the highest betweenness

centrality are used to cache copies of content on the way to the user's response[16].

However, existing proactive caching approaches are developed for highly regulated environments where users are compelled to upload their local data to a central server, thereby causing privacy and security concerns. However, when the user base and data volume grow, those systems' scalability becomes an issue. [11].

2.8 Cache Replacement

2.8.1 Introduction

A hit event occurs when a requester node (client) makes a content data request and the caching router finds the requested material in its content store (CS)., and the caching router immediately sends the content data to the client. When the requested data cannot be found in the caching router's content store (CS), the server is contacted and the requested data is sent. When the caching router receives the requested data back from the server, it randomly chooses one of the contents in its CS and replaces it with the requested content. Random selection criteria are used to determine which material has to be replaced[6].

2.8.2 The Least Recently Used (LRU)

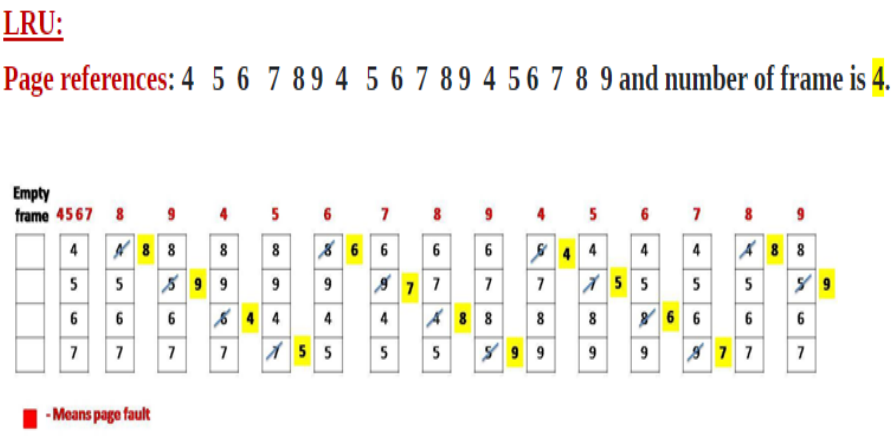

Figure 2.3: LRU Exemple

the Least Recently Used (LRU) approach. The least recently referred data item is substituted for the new data in LRU. The most popular information is viewed frequently in scenarios found in the modern world, which helps LRU achieve a high hit ratio. LRU has the benefit of being quick and easy. However, frequency information, which has a significant impact on the hit-rate figures, is not taken into account by LRU[17].
By keeping a tally for each data item to measure how many times that item is requested, the Least Frequently Used replacement approach (LFU) can achieve a high hit ratio. A low counter value means there are not many requests for that item. To replace an existing item with newly added content, LFU chooses the one with the lowest value[17].

2.8.3 First in First Out (FIFO)

First in First Out (FIFO) is a straightforward approach that swaps out old data for fresh data. If data is kept in the cache storage for a long time, it is deemed to be old, and the older the data, the higher the probability.that a replacement will be made. Yet, FIFO faces a significant problem because the less popular item is substituted for the chosen older item when it is popular.[17]LFU and LRU are more effective caching techniques than.FIFO. When there is no more cache space, it discards the oldest information, regardless of how well-liked it may be[12].

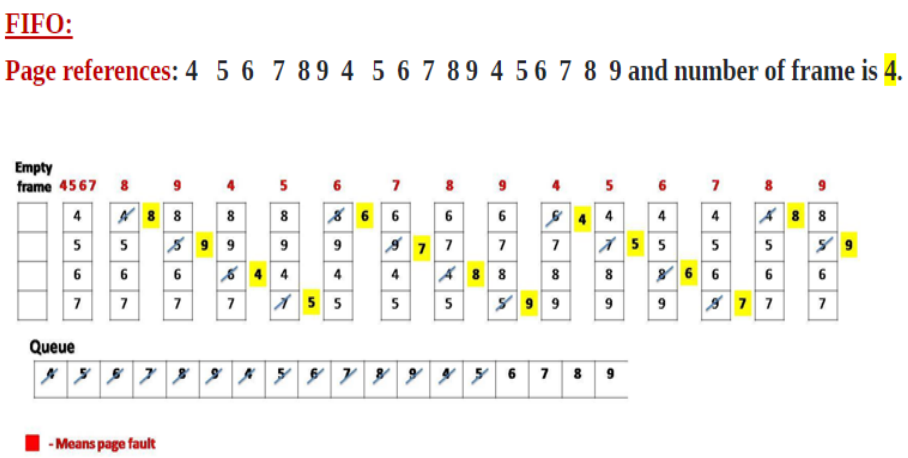

Figure 2.4: FIFO exemple

2.8.4 Most recently used (MRU)

MRU Exemple

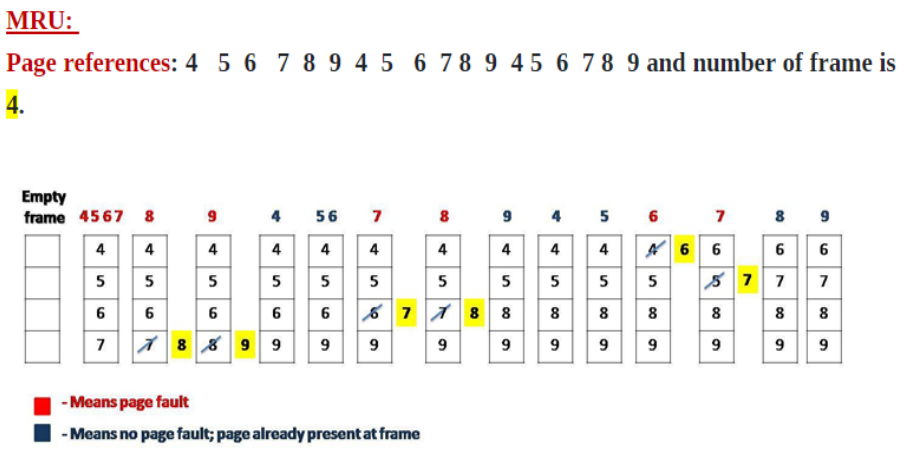

Figure 2.5: MRU Exemple

Most recently used (MRU) approach, as its name suggests, is the exact antithesis of least recently used replacement method.A hit event occurs when a requester node (client) makes a request for content data and the caching router finds the requested content in its content store (CS). At that point, the caching router provides the requested content

data to the client right away. When the requested data cannot be located in the caching router's content store (CS), the server is contacted and the caching router chooses a particular piece of material in its CS based on how recently it has been used[6]. The most popular material may be replaced with the least popular one because of the way LRU and LFU work. The Time Aware Least Recent Used (TLRU) policy was put forth in to address this issue. It is a basic LRU addon that uses a popularity-based content lifetime awareness eviction policy. The time stamp of an arriving piece of content is determined locally by a caching node under this policy. If the average request time is less than the time stamps of the saved contents, the incoming content is cached[7].

2.9 Conclusion

In conclusion, the chapter on Cache Management provides an overview of various concepts and techniques related to caching in computing systems. It explores the role of caching in improving system performance and reducing data access latency. The chapter also discusses different caching strategies, including cache placement and cache replacement.

Overall, the chapter provides a comprehensive understanding of cache management principles and techniques, equipping readers with valuable insights into optimizing data caching in modern computing environments. The next chapter will present machine learning ,types and algorithms.

Chapter 3

Machine Learning

3.1 Introduction

The science of creating computer algorithms that can mimic human intelligence is known as machine learning. It draws inspiration from concepts found in a variety of fields, including computer science, information theory, psychology, control theory, philosophy, probability, statistics, and psychology. A few of the many disciplines in which this technology has been applied include pattern recognition, computer vision, spacecraft engineering, economics, entertainment, ecology, computational biology, and biomedical and medical applications. The most important characteristic of these algorithms is their singular ability to learn the environment from incoming data, either with or without a teacher[18].
Generally speaking, machine learning can be divided into "supervised," "unsupervised," "semi-supervised," and "reinforcement" learning [19].
Machine learning techniques are frequently used to examine massive volumes of data and gather pertinent information for the detection, classification, and prediction of future events. Machine learning has a fairly broad definition, encompassing everything from straightforward data summarization using linear regression to multiclass classification using deep neural networks[1].

3.2 Related work

- **Work 1** [20] **:** In this work, the authors present a structured and concise overview of the studies conducted on the k-means algorithm with the aim of addressing its limitations. They delve into variants of the k-means algorithm and discuss recent developments in the field. To evaluate their effectiveness, the researchers perform an experimental analysis using diverse datasets. This analysis provides detailed insights into the performance of these variants and developments. What sets this work apart from other survey papers is its meticulous and in-depth experimental analysis, coupled with a comprehensive comparison among various k-means clustering algorithms. By conducting such a thorough examination, the authors differentiate their research and contribute to a clearer understanding of the k-means algorithm and its diverse research direction.

- **Work 2** [21] **:** In this work, the authors establish the foundation of their analysis of supervised learning by drawing upon the theory of risk minimization. They emphasize the importance of minimizing risk in the context of supervised learning.

To illustrate this concept, they provide an overview of two widely used supervised learning techniques in multimedia research: support vector machines and nearest neighbor classifiers. These two methods are acknowledged as the most popular approaches in the field. By introducing these techniques, they demonstrate their relevance and applicability within the context of multimedia research, highlighting their significance in tackling supervised learning problems.

3.3 Supervised Learning

3.3.1 Description of The Model

The concept of learning from examples has simply been formalized via supervised learning.While learning is supervised, the learner (usually a computer program) has access to two datasets: a training dataset and a testing dataset. The goal is for the train-test set learner to "learn" from a set of labeled instances in the training set to identify as accurately as possible the unlabeled examples in the test set. In other words, The learner's objective is to create a rule, procedure, or method to classify new samples (in the test set) by examining samples that have been assigned class labels[22]. Supervised machine learning is ideal for binry classification , multi-class classification , regression modeling and ensembling.

3.3.2 Linear regression

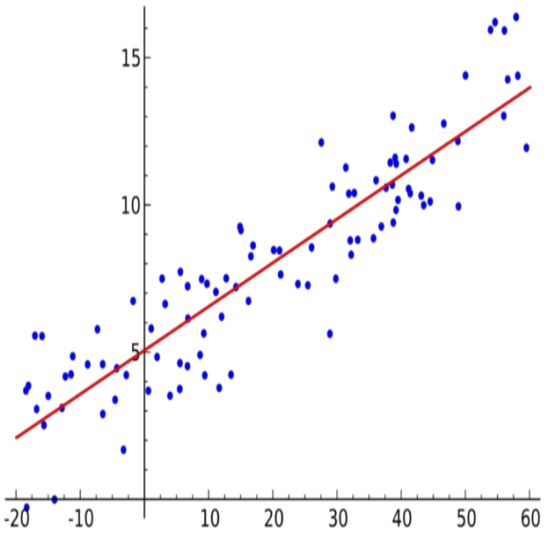

Figure 3.1: Linear regression

linear regression includes only a independent variable and a linear connection between the independent (x) and dependent (y) variables. The red line in the graph above is referred to as the straight line that fits the data the best[23].
The main goal is to fit a best line represented by a linear equation $Y = f(x) + \delta$ in order

to predict $f(\tilde{X}) = \tilde{a} + \tilde{b}X$ for a value of X
The linear equation below can be used to represent the line.

$$Y = f(x) + \delta$$

δ represents the error or disturbance term.
Linear regression is mainly divided into two categories: There are two types of linear regression: basic and multiple. An independent variable is a defining characteristic of simple linear regression. Nevertheless, multiple linear regression is distinguished by the presence of at least two independent variables[23].
In statistics, the classic method is given by ordinary least squares to solve simple regression problems; the best linear estimator according to the Gauss-Markov theorem, consists in calculating the sum of the squares of the residuals which we will denote scr (f):

$$scr(f) = \sum_{i=1}^{n}(Y_i - f(X_i))^2 = \sum_{i=1}^{n}\delta_i$$

In the simple linear regression:

$$scr(f) = scr(a, b) = \sum_{i=1}^{n}(Y_i - f(a - bX_i))^2$$

the function f is a polynomial of degree 1 of X , P = a,bis the set of parameters of the model and we are searching for the estimates $\tilde{a}$ and $\tilde{b}$ which minimize scr(f).
Identifying the critical points normal equation solutions that cancel their first derivative products important. We arrive at an analytical conclusion :

$$\tilde{a} = \bar{y} - \tilde{b}\bar{x} \qquad and \qquad \tilde{b} = \frac{\sum_{i=1}^{n}((x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$with \qquad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

The prediction model is then given by:

$$f(\tilde{X}) = \tilde{a} + \tilde{b}X$$

3.3.3 Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a common classification, regression, and detection technique. It employs statistical learning theory-based heuristic algorithms. It utilizes heuristic algorithms based on statistical learning theory. One key advantage of SVM is that it solves model parameter determination as a convex optimization problem, guaranteeing global optimum solutions.SVM employs a collection of data vectors with predefined class labels to create a linear hyperplane for class separation. These data

vectors serve as the training set and have unique features used for classification. In complex cases,Finding the best separator is a challenge that SVM simplifies into a linear issue by utilizing kernel functions to translate the data to a higher-dimensional feature space. The best classifier can then be used to categorize fresh data with unidentified class labels based on its attributes [24].

To describe the method, we must first discuss the issue in terms of the two-class

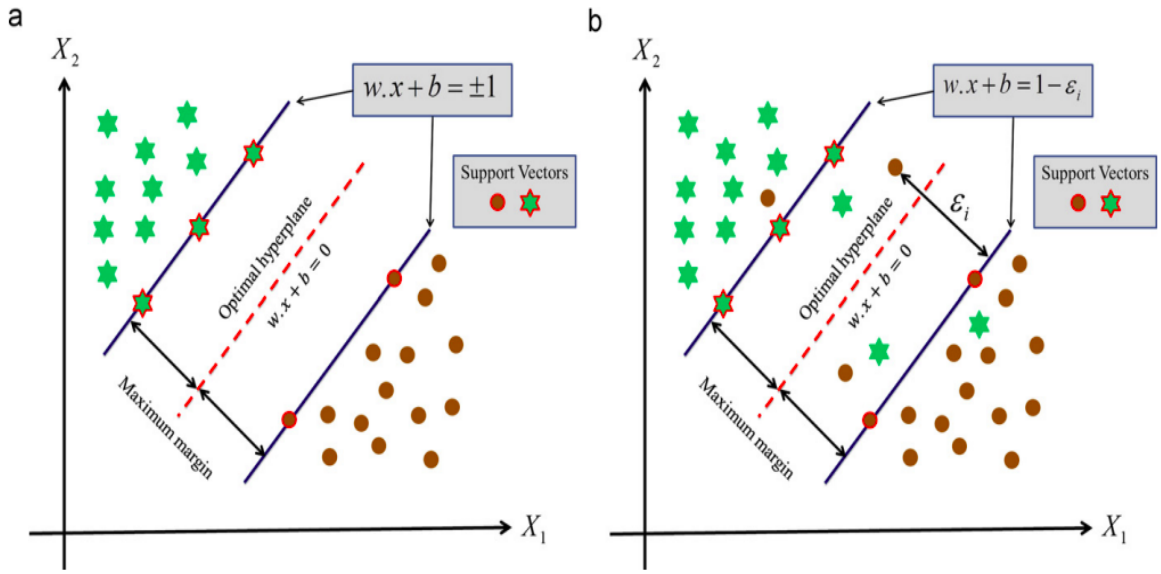

Figure 3.2: The optimal hyperplane: a separable and b non-separable

problem., Let's consider a training set consisting of l feature vectors as our data vectors $x_i \in R^n$ where $i(= 1, 2, ..., n)$ is the number of samples , Each sample in the training set is assigned a class label y_i, which can take the value of 1 for one class or -1 for the other class (i.e., $y_i \in$ -1,1)[24].

When it is possible to split the two classes with a straight line and they comply to a specified set of equations, a collection of linear separators known as separating hyperplanes exists.

$$w.x_i + b \geq +1 \qquad for \qquad y_i = +1.....(1)$$
$$w.x_i + b \leq -1 \qquad for \qquad y_i = -1.....(1)$$

which is equivalent to

$$y_i(w.x_i + b) \geq 1 \qquad i = 1, 2..., n.....(2)$$

A decision function can then be formulated for the separating hyperplane:

$$f(x) = sgn(wx + b).....(3)$$

where sgn is a sign function that has the following definition

$$f(x) = \begin{cases} 1 & if \quad x > 0 \\ 0 & if \quad x = 0 \\ -1 & if \quad x < 0 \end{cases} .....(4)$$

The parameters w and b, which define the separating hyperplane as a decision function, can be found by solving the following optimization function:

$$minimize\tau(w) = \frac{1}{2}\|w\|^2 .....(5)$$

depending on:

$$y_i(w.x_i + b) \geq 1 \qquad i = 1, 2..., L.....(6)$$

The saddle point of the Lagrange function provides the answer to this optimization dilemma

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i(y_i(w.x_i + b) - 1).....(7)$$

$$\frac{\delta}{b\delta}L(w, b, \alpha) = 0, \frac{\delta}{w\delta}L(w, b, \alpha) = 0.....(8)$$

The Lagrange multiplier α plays a part in the example. The objective is to maximize the Lagrange function with respect to $\alpha_1 > 0$ while minimising it with respect to the variables w and b. A particular optimization function determines the values of the Lagrange multipliers α [24].

maximize
$\sum_{i=1}^{l} \alpha_1 - \frac{1}{2}\sum_{ij=1}^{l} \alpha_i\alpha_j y_i y_j (x_i x_j).....(9)$

subject to
$\alpha \geq 0$, i= 1.. l and $\sum_{i=1}^{l} \alpha_i y_i = 0.....(10)$

The decision function for the separating approach based on the optimal hyperplane can be expressed as follows:

$$f(x) = sng(\sum_{i=1}^{l} y_i\alpha_i(xx_i) + b.....(11)$$

The solution mentioned above can be utilized for problems that exhibit separability within the feature space. In order to enhance the classification process, the approach incorporates the use of a penalty value C to address misclassification errors, along with the inclusion of positive slack variables ϵ_i as depicted in Figure 2.2 These variables were introduced and included within constraint (1) in the following manner[24].
$wx_i + b \geq 1 - \epsilon_i \qquad for \qquad y_i = +1$
$wx_i + b \leq -1 + \epsilon_i \qquad for \qquad y_i = -1$
$\epsilon_i \geq 0, \qquad i = 1, 2, ..., n \qquad .....(12)$
The following function has to be optimized at the end
minimize :
$\tau(w) = \frac{1}{2}\|w\|^2(\sum i = 1^l \epsilon_i)^k .....(13)$
subject to:
$y_i(wx_i + b) \geq 1\epsilon_i, \qquad i = 1, 2, ..., n \qquad .....(14)$

The penalty value "C" is associated with misclassification errors and can be determined through a cross-validation process. When k=1, the optimization procedure for the aforementioned function is analogous to the problem of separable classes When data cannot be linearly separated, a projection function$\phi(x)$is employed to map the training

data from the original data space x to a Hilbert space X.
Within the SVM optimization function, the feature information present in the training data is represented through dot products (xixj) as shown in Equations (9) and (11) The training procedure in the Hilbert space X primarily relies on data within this space through a dot product because the training data is only given as two vector dot products. As a result, a function of the kind $\phi(x_i) \times \phi(x_j)$ is created to indicate the algorithm's dependence on the dot product. As a result, the kernel function K is introduced[24]. Its definition is :

$$K(x_i, x_j) = \phi(x_i) \times \phi(x_j).....(15)$$

In the training program, it is sufficient to utilize the kernel function K without requiring knowledge of the explicit form of $\phi(x)$. This approach extends to the decision function as well. Hence, the dual formulation of the problem can be expressed as follows[24].

maximize :
$\sum_{i=1}^{l} \alpha_1 - \frac{1}{2} \sum_{ij=1}^{l} \alpha_i \alpha_j y_i y_j (x_i x_j).....(16)$

subject to

$\alpha \geq 0$, i= 1.. l and $\sum_{i=1}^{l} \alpha_i y_i = 0.....(17)$

The decision function for the new separation rule, which is based on the best hyperplane, is as follows:

$$f(x) = sng(\sum_{i=1}^{l} y_i \alpha_i k(x x_i) + b.....(18)$$

For this, a variety of kernel functions can be used. We take into consideration the following two kernel functions in this study:

$$K(x_i, x_j) = (x_i x_j + 1)^d.....(19)$$

the radial basis function (RBF):

$$K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}.....(20)$$

The kernel parameters, represented as d and g, are critical in defining how the kernel functions behave. The parameter d denotes a polynomial function's degree, whereas g is an inner product coefficient that controls the width of the Radial Basis Function (RBF). Figure 2.2 shows how a projection may be used to efficiently classify non-linear features in the feature space.
The SVM is primarily designed as a two-class classifier, but many real-world problems involve multiple classes. To address this, several methods have been proposed to construct a multiclass classifier by combining multiple two-class SVMs. One common approach is known as "one-against-one" (also referred to as pairwise classification). In this approach,$L(L-1)/2$ different two-class SVMs are trained on all possible pairs of classes, and during the classification of test points, the class with the highest number of "votes" is selected. This enables the SVM to handle multiclass classification tasks effectively[24].

3.4 Unsupervised Learning

3.4.1 Description of The Model

Unsupervised data learning is a method for discovering patterns without using a target attribute. This shows that all of the study's variables were inputs, and that the methods may be used to cluster data in accordance with the methodology. additionally connection mining methods. The labels in the data that are later utilized to perform supervised learning tasks can be generated using unsupervised learning techniques. In other words, unsupervised clustering algorithms apply labels to each data value and find organic groups in the unlabeled data[25].

3.4.2 k-Means Algorithm

The algorithm has been used in various studies to create groups or classes unlabeled records based on mean distance between classes[26]. These kinds of unsupervised algorithms are often used in data mining and pattern recognition. The square-error and error criterion foundations of this technique seek to reduce the cluster performance index[27]. Based on the mean distance between classes, the approach has been utilized in numerous research to group or classify unlabeled datasets. The method starts and develops the categories or labels that are later employed in other prospective analyses. Here are the basic steps of the algorithm:

- Initialize the k cluster centers at random[28].
- Each data point should be assigned to the closest cluster center[28].
- The cluster centers should now be recalculated as the mean of all the data points included in that cluster[28].
- Repeat steps 2 and 3 until convergence, i.e., until the cluster assignments no longer change or a maximum number of iterations is reached[28].

Algorithm 1 k-means algorithm

Dataset $\mathcal{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, number of clusters K
Cluster assignments $\mathcal{C} = C_1, C_2, \ldots, C_K$
Initialization: Randomly select K data points as initial centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K$;

while not converged **do**
Step 1: Assign each data point to the nearest centroid

$$C_i = \mathbf{x}_j \in \mathcal{X} \mid \arg\min_k |\mathbf{x}_j - \mathbf{c}_k|_2 = i, \quad i = 1, 2, \ldots, K$$

Step 2: Update the centroids based on the new cluster assignments

$$\mathbf{c}i = \frac{1}{|C_i|} \sum \mathbf{x} \in C_i \mathbf{x}, \quad i = 1, 2, \ldots, K$$

3.4.3 Algorithm Explanation

Input:

- X: Data Points: The initial data set is a compilation of data points, denoted as X, that can be depicted as a set of vectors in a multidimensional space. Typically, each data point is characterized by a collection of attributes or features.
- k: The parameter k, which signifies the number of centroids, is typically predetermined by the user before executing the algorithm.

Output:

- C:The algorithm identifies center points of clusters known as centroids, represented as c1, c2, ..., ck.
- c_i: Set of n cluster assignments, indicating which cluster each data point belongs to.

Step 1:
To assign a data point to its corresponding centroid, the algorithm computes the distance between each data point and each centroid. Typically, a distance metric like the Euclidean distance, which is the square root of the sum of squared differences between corresponding features of two points, is used for this calculation. In the equation

$$|\mathbf{x}_j - \mathbf{c}_k|^2$$

represents the squared Euclidean distance between the jth data point x_j and the kth centroid c_k.
Nearest centroid assignment: The goal of this step is to assign each data point to the nearest centroid based on the calculated distances.
The phrase:

$$|\arg\min_k |\mathbf{x}_j - \mathbf{c}_k|^2$$

finding the centroid k that minimizes the squared Euclidean distance between the data point x_j and the centroid c_k. In other words, it identifies the centroid that is closest to the data point in terms of Euclidean distance.

Cluster assignment: Each data point is allocated to the cluster that is represented by its nearest centroid once that centroid has been identified.

$$|\arg\min_k |\mathbf{x}_j - \mathbf{c}_k|^2 = i$$

is used to signify this, where Ci stands for the cluster assignment for the jth data point, x_j, and I for the index of the centroid that is closest to x_j.
Step 2:
$C_(i, x)$ represents the $x-th$ data point belonging to the $i-th$ cluster. can be interpreted as the sum of all the data points x in the $i-th$ cluster, divided by the cardinality or the number of data points in the $i-th$ cluster. This operation is performed for each cluster, indexed by i, ranging from 1 to K.

This step entails finding the average of all the data points in each cluster and utilizing that result as the new centroid for that cluster in the context of updating centroids in a clustering algorithm. Up until the centroids settle and the algorithm converges, this step is performed iteratively. The algorithm's subsequent iteration uses the modified centroids to improve the cluster allocations. This stage aids in locating the central or most representative point in each cluster, which can be used to make predictions or conduct additional research.

3.5 Semi-supervised

3.5.1 Description of The Model

Traditionally, supervised and unsupervised learning have been the two main objectives in machine learning. Semi-supervised learning algorithms make an effort to enhance performance in one of these two tasks using data typically related to the other[29]. It's a unique type of categorization. In order to train, traditional classifiers require labeled data (feature/label pairs). Yet, labeled cases are frequently expensive, time-consuming, or difficult to get since they need the work of skilled human annotators. Unlabeled data can be gathered quite easily, but there aren't many applications for them now. Semi-supervised learning solves this issue by improving classifiers using both data with and without labels. Semi-supervised learning is quite intriguing in both concepts and applications because it needs less human effort and produces higher accuracy.[30]
Both labeled and unlabeled data are used for training in semi-supervised learning. It contrasts supervised and unsupervised learning (data that has all been tagged) (data all unlabeled). The phrases "learning from labeled and unlabeled data" and "learning from partially labeled/classified data" are also used. Be aware that semi-supervised learning might be inductive or transductive.[30]
When additional unlabeled data is present and classification is the desired outcome, this is referred to as "semi-supervised classifica-tion." Its cousin, "semi-supervised clustering," aims to cluster unlabeled data with certain pairwise constraints.[30]
The majority of semi-supervised learning algorithms combine supervised and unsupervised techniques.For example co-training7

3.5.2 Co-training algorithm

Several models are trained cooperatively on a labeled dataset using the machine learning technique known as co-training in order to enhance their performance on an unlabeled dataset. The essential concept is that each model learns from the data that the other models offer, effectively utilizing their aggregate knowledge to enhance generalization and accuracy.[31]

Algorithm 2 Co-training Algorithm

Require: Labeled dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
Ensure: Trained classifiers C_1 and C_2
0: Split D into two disjoint sets D_1 and D_2
0: Initialize C_1 and C_2 with their respective classifiers
0: **while** not converged **do**
0: Train C_1 on D_1
0: Train C_2 on D_2
0: Predict labels for unlabeled data using C_1 and C_2
0: Select a subset of most confident predictions from both classifiers
0: Add the selected subset to the labeled dataset
0: Update D_1 and D_2 with the updated labeled dataset
0: **end while**=0

3.5.3 Algorithm explanation

Here's a high-level overview of the co-training algorithm:
Input:

- Labeled dataset: $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, where x_i is the input feature vector and y_i is a label that corresponds to the ith data point.
- Unlabeled dataset: $x_1, x_2, ..., x_n$, which consists of the same input feature vectors as the labeled dataset, but without the corresponding labels.

Output:

- Trained classifier(s) for making predictions on new, unseen data.

1. Create two disjoint sets, D1 and D2, from the labeled dataset (D), which will be used to train C1 and C2, respectively.
2. Initialize C1 and C2 with their respective classifiers, which could be any suitable machine learning algorithm.
3. While not converged (meaning the algorithm has not reached a stopping criterion, such as a maximum number of iterations or a certain level of accuracy), perform the following steps iteratively:
   (a) Train C1 on D1, and train C2 on D2, using the labeled data in each set.
   (b) Predict labels for unlabeled data using both C1 and C2, which could be instances that were not part of the original labeled dataset D.
   (c) Select a subset of most confident predictions from both classifiers. This could be done based on a confidence threshold or other criteria.
   (d) Add the selected subset of confident predictions to the labeled dataset, effectively expanding the labeled dataset and incorporating the predicted labels.
   (e) Update D1 and D2 with the updated labeled dataset, which now includes the added subset of confident predictions.
   (f) Repeat the process from step 4 until a convergence criterion is met.

3.6 Reinforcement learning

3.6.1 Description of The Model

A type of machine learning known as reinforcement learning teaches a model to solve a problem at its best by making a series of independent decisions[32].

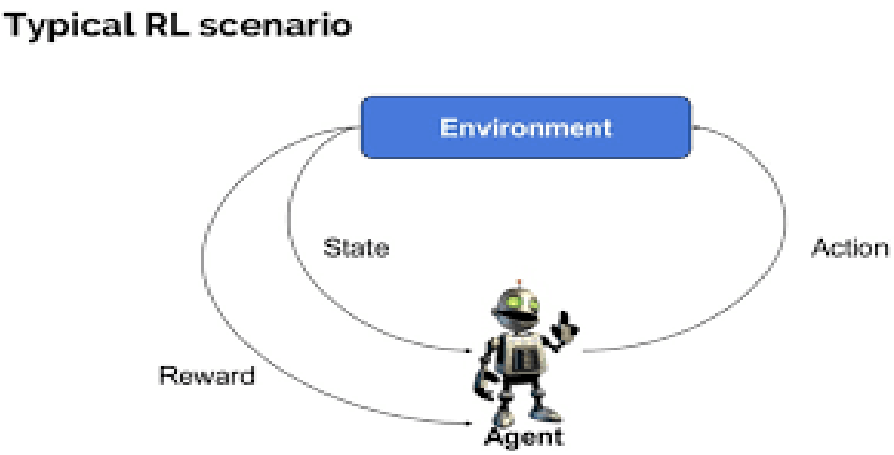

Figure 3.3: Reinforcement Learning

The objective feedback from the environment serves as the reward function. Reward variables can be either scalar or integer, and they are linked to specific states or state-action connections. This association deepens the agent's goal in a given setting (the reward function)[32].
State and action are fairly broad ideas. Actions are any decisions that an agent could need to learn how to make, and states are any factors that the agent might consider when making those decisions. One aspect of the state that serves as the basis for an action could be a model of the environment. This model could serve as a representation of the actor's environment in the past.[32]
The mapping between potential states and potential actions is known as the agent's policy (or policy function). The mapping could be a fairly simple look-up table, sometimes known as a stored policy.[32]

3.6.2 Q-Learning

This algorithm is one of the most used in the reinforcement learning community due to its simplicity, its robustness and the formal proofs of its convergence in PDMs.

The Q-learning system picks up actions that are against the established rules, such acting arbitrarily. The "Q" in Q-learning refers for quality, which denotes how valuable a given activity is in obtaining a reward in the future.[33]

Algorithm 3 Q-Learning Algorithm

0: **/* α is the learning rate */**
0: Initialize(Q_0)
0: **for** $n \leftarrow 0$ to $N_{\text{tot}} - 1$ **do**
0: $s_n \leftarrow$ Choose-State
0: $a_n \leftarrow$ Choose-Action
0: $(s', r) \leftarrow$ Simulate(s_n, a_n)
0: /* Update Q_n */
0: $Q_n \leftarrow Q_{n+1}$
0: $\delta_n \leftarrow \mathrm{r}_n + \gamma \max_b(\mathrm{s}_n^{'}$, b)- $Q_n(\mathrm{s}_n, a_n)$
0: $\mathrm{Q}_{n+1}(\mathrm{s}_n, a_n) \leftarrow \mathrm{Q}_n(\mathrm{s}_n, a_n) + \alpha_n(s_n, a_n)\delta_n$
0: **end for**
0: **return** N_{tot} =0

3.6.3 Algorithm Explanation

The principle of the Q-Learning algorithm, is to update iteratively, following each transition $(s_n, a_n, s_n + 1), r_n)$, the current value function Q_n for the pair (s_n, a_n) , where s_n represents the current state, a_n action selected and performed, $s_n^{'}$ the resulting state and r_n immediate reward [34]
In this algorithm, N_tot is an initial parameter fixing the number of iterations ,The learning rate $a_n(s, a)$ is specific to each pair (s, a) on each pass, drops towards 0. The Simulate function returns a new state and the associated reward according to the dynamics of the system . The choice of the current state and the action to be performed is performed by the functions Choose-State and Choose-Action ,The Initialize function mostly amounts to initializing the components of Q0 to 0. However, there are more effective initializations available.[34]
It is immediate to observe that the Q-Learning algorithm is a stochastic formulation of the value iteration algorithm for MDPs Indeed , this last one may be expressed directly in terms of action value function:

$$V_{n+a}(s) = \max_{a \in A} \overbrace{\{r(s,a) + \gamma \sum_{s' \in s} p(s^{'}|s,a) V_n(s^{'})\}}^{Q_n(s,a)}$$

$$\Longrightarrow Q_{n+1}(s,a) = r(s,a) + \gamma \sum_{s' \in s} p(s^{'}|s,a) V_n(s^{'})$$

$$\Longrightarrow Q_{n+1}(s,a) = r(s,a) + \gamma \sum_{s' \in s} p(s^{'}|s,a) \max_{a^{'} \in A} Q_n(s^{'}, a^{'})$$

The Q-Learning is then obtained by replacing $r(s,a) + \gamma \sum_{s' \in s} p(s^{'}|s,a) \max_{a' \in A} Q_n(s^{'}, a^{'})$ By the estimator constructed from the current transition

the function Qn almost surely converges to Q* , The following presumptions are necessary To demonstrate that this algorithm is convergent:

- finiteness of S and A.
- Every couple (A, S) is visited an infinite number of times.
- $\sum_n \alpha_n(a,s) = \infty$ and $\sum_n \alpha_n^2(a,s) < \infty$
- $\gamma < 1$ or $\gamma = 1$

this almost sure convergence means that $\forall s, a$ the sequence $Q_n(s,a)$ converges to $Q^n(s,a)$ with a probability 1.

3.7 Neural networks

3.7.1 What is a neural network?

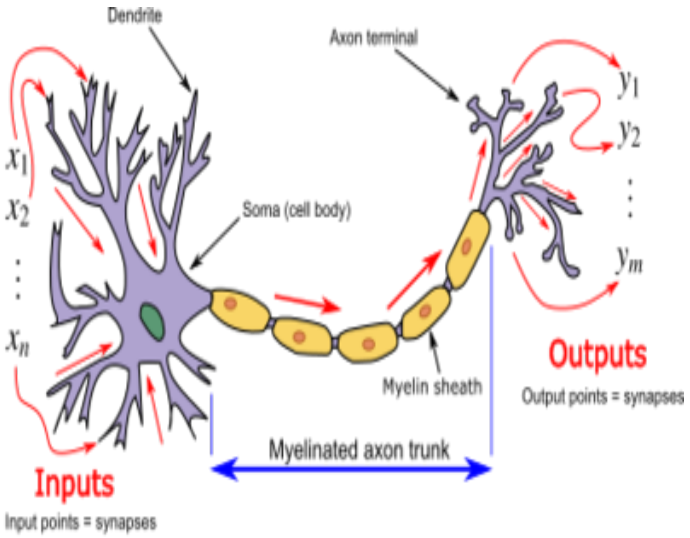

Figure 3.4: Biological neurons

A neural network is a type of reasoning that imitates the human brain, consisting of interconnected neurons that process information. The human brain has close to 10 billion neurons and 60 trillion synapses connecting them.As a result, the brain can carry out activities faster than even the most advanced computers[35].

Even though individual neurons are simple, their collective power is enormous. Each neuron contains a soma (cell body), dendrites (fibers that branch out around the soma), and an axon (a single long fiber that connects to the dendrites and somas of other neurons)[35] . Our brain is a complex, parallel, nonlinear information-processing device. Instead of being kept in discrete areas, information is simultaneously processed and stored across the complete neural network[35]. Neural networks can learn from experience, which is a fundamental and vital feature of biological neural networks. The natural ease with which neural networks learn has inspired efforts to simulate a biological neural network on a computer [35].

3.7.2 Main Architectures of Artificial Neural Networks

The structure of artificial neural networks (ANNs) significantly affects their ability to process information. This encompasses the connections within the network and the functions used for transferring information. Generally speaking, an artificial neural network can be categorized into three components referred to as layers, which are commonly recognized as[36].

- **Input Layer** The Input Layer of a neural network is accountable for accepting data from the external environmen.
- **Hidden Layer** Intermediate layer between input and output layer and consisting of neurons, this layer is tasked with identifying patterns related to the given process.
- **Output Layer** This layer is presenting the final network outputs, which result derived from the processing carried out by the neurons in the earlier levels.

3.7.3 ANN classification

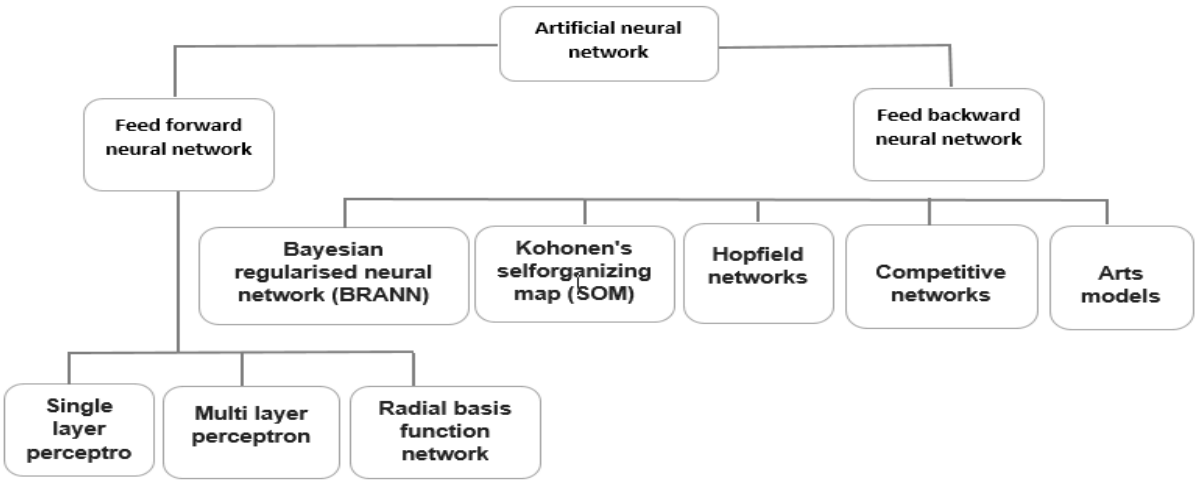

Figure 3.5: ANN classification

A feedforward neural network (FFNN) is a classification algorithm used in machine learning. It consists of layers organized in a manner resembling the processing units of human neurons. In a feedforward neural network (FFNN), every unit within a layer is connected to all other units in the layers. However, the connections between layers are not uniform, as each connection possesses a unique weight or strength. These weights determine the level of influence each connection has within the network. Furthermore, the units of a neural network are referred to as nodes. The network's information processing begins with data input from the input units, which then travels through the network, passing from one layer to the next until it reaches the output unit[37]. In(FFNN), information is only sent in one direction: from the input nodes to the output nodes, optionally traveling through hidden nodes if present.[38].

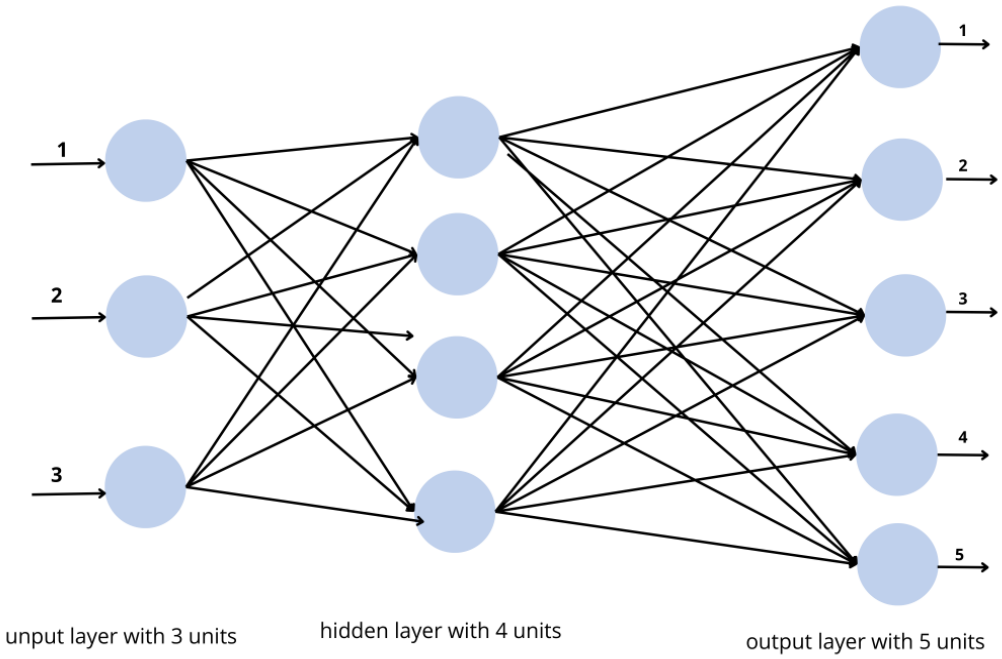

Figure 3.6: Feed-forward Neural Networks

Instances of feedforward neural networks (FFNNs) include the single-layer perceptron and the multilayer perceptron. An illustration of a two-layer network is depicted in Figure 3.6, where there are 3 input units, 4 units in the hidden layer, and 5 units in the output layer, represented by circles. Figure 3.6 illustrates the presence of 3 input units represented by circles. However, these input units do not belong to any specific layer within the network system. In some cases, the input units are considered a virtual layer, often referred to as having 0 layers. There is a hidden layer in Figure 3.6 that is separate from the input layer and the output layer. A neural network with one hidden layer and one output layer is shown in the diagram, along with all of the connections between the units in these layers. It is obvious that each layer only connects to the one before it. Dynamical system control is one of two categories for applications of feedforward neural networks (FFNNs)[39][40].

Deep networks refer to neural networks (NNs) that contain two or more hidden layers, indicating a higher level of complexity compared to NNs with only one hidden layer. In contrast to feedforward neural networks (FFNNs), feed-backward neural networks (FBNNs) are capable of processing sequences of data inputs by utilizing internal state or "memory" to store information. This implies that FBNNs can handle tasks that require logical processing of inputs based on their order. Un-segmentation and pattern recognition tasks, notably linked handwriting recognition, are well suited to FBNNs. The applications of feed-back neural networks include classification, seismic data fitting, and mathematical proofs, seismic data fitting, medicine, science, engineering, classification, function estimation, and time-series prediction, and so on.

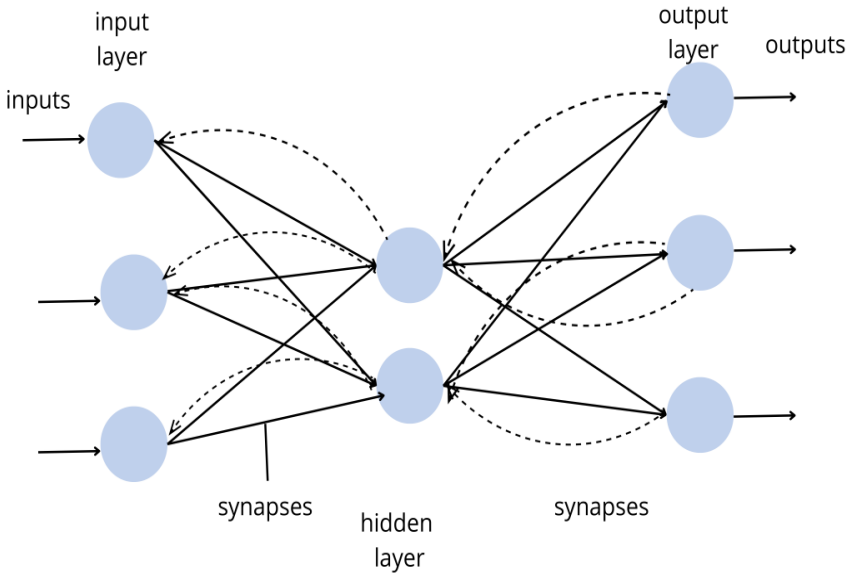

Figure 3.7: Feed-backward Neural Networks

The Figure 3.7 presents the architecture of a feed-backward neural network (FBNN) In feedback neural networks (NNs) or backpropagation networks, the connections between nodes form a sequential coordinated graph. This allows feedback NNs to exhibit dynamic behavior over a period of time due to the sequential coordination in the graph. Two examples of such networks are Kohonen's self-organizing map and recurrent neural networks (RNNs). RNNs are a specific type of neural network that evolves over time, where

the edges feed into the subsequent time step rather than the next layer simultaneously. RNNs are designed for recognizing sequences, such as textual or speech signals. They contain cycles, which indicate the presence of short-term memory within the network. In contrast to a recurrent neural network, an RNN can be viewed as a hierarchical network, where the input needs to be processed hierarchically in a tree-like structure since there is no inherent notion of time in the input sequence

3.7.4 Types of ANNs

- **Multilayer Perceptron network (MLP)**

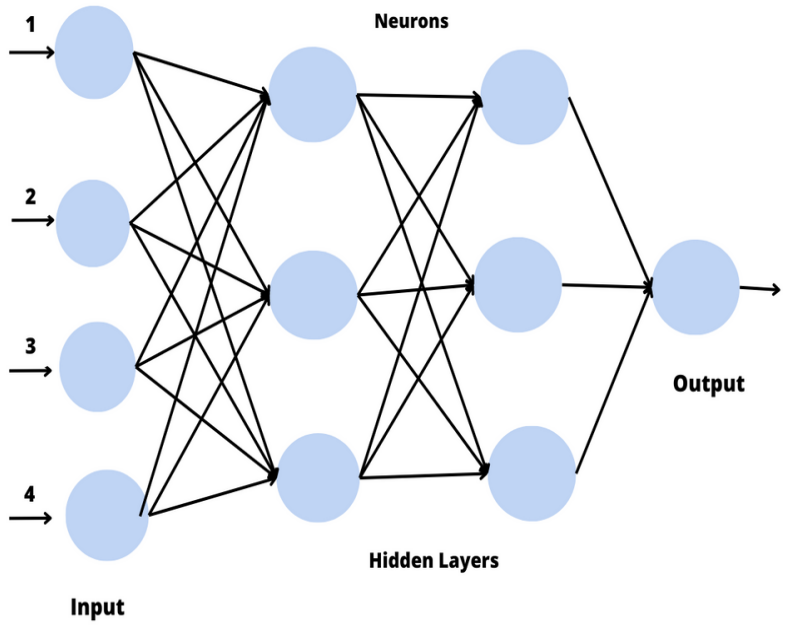

Figure 3.8: Multi layer perceptons (MLPs)

MLP consists of three or more layers and is particularly useful for classifying data that cannot be separated linearly.This network is completely connected, so that each node is connected to every other node in the subsequent layer[41].
The inputs received by a neuron are aggregated and then passed through an activation function. The connections between neurons are established, and each connection possesses individual weights. The output of each neuron is transmitted through a connection and scaled by the weight assigned to that connection. The resulting product is then forwarded to the neurons in the subsequent layer. Additionally, biases, which are fixed values (typically one), are incorporated within the architecture of the Artificial Neural Network (ANN). The product of the bias and the weight of its corresponding connection is then introduced to the neurons in the following layer.Considering the ith neuron in a layer of an Artificial Neural Network (ANN), where p1,..., pR represent the outputs of the neurons in the previous layer, Wij denotes the weight associated with the connection between the neuron and the jth

neuron of the preceding layer, b represents the bias of the previous layer, and f denotes the activation function of the layer; the output of the layer (a) is expressed in equation [42]:

$$a = f(sum_{j=1}^{R} w_{ij} pj + b)$$

In Multi-Layer Perceptrons (MLPs), activation functions remain constant while weights can vary during the training process. Perceptrons have found application in various control domains, such as printing devices, heat exchangers, and spacecraft maneuvering. Extensive research has been conducted on Multi-Layer Perceptrons for control purposes[42].

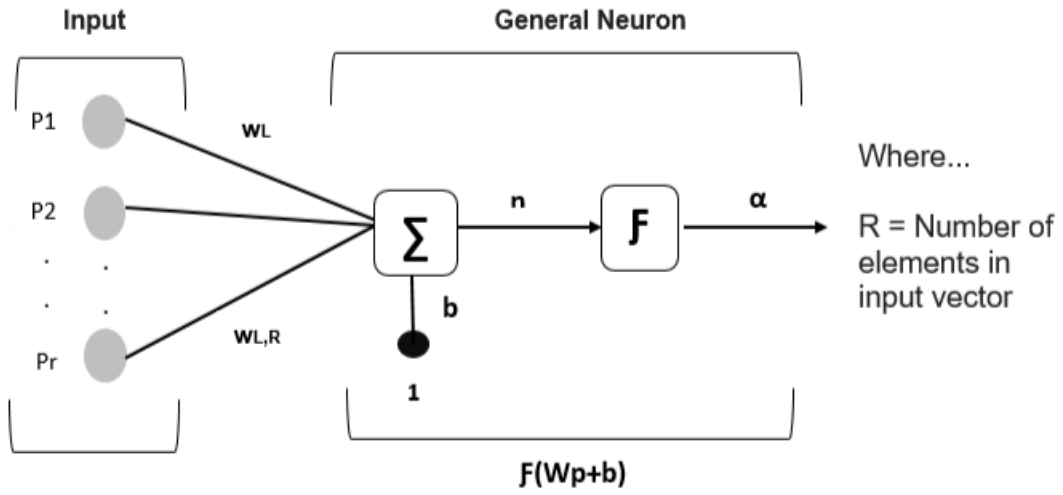

Figure 3.9: A typical neuron of an ANN

- **Radial basis function network (RBF)**

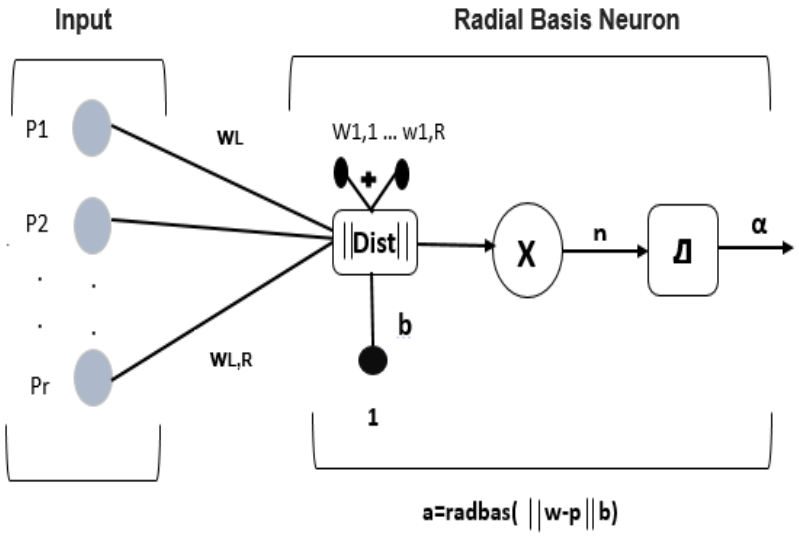

Figure 3.10: First layer of a (RBFN)

Radial Basis Function Networks (RBFNs) are widely used in control applications and are the second most popular type of neural network for this purpose. These networks typically have two layers of neurons, the first of which differs from perceptron layers and the second of which has characteristics in common with them. Applications for RBFNs are found throughout several industries, demonstrating their adaptability[42].
The input vector p and weight vector IW are fed into the dist box in Figure 3.10 , where their dot product is calculated. The final step is to multiply the resultant value by the bias b and serves as the input to the radial basis transfer function.

$$radbas(x) = e^{x^2}$$

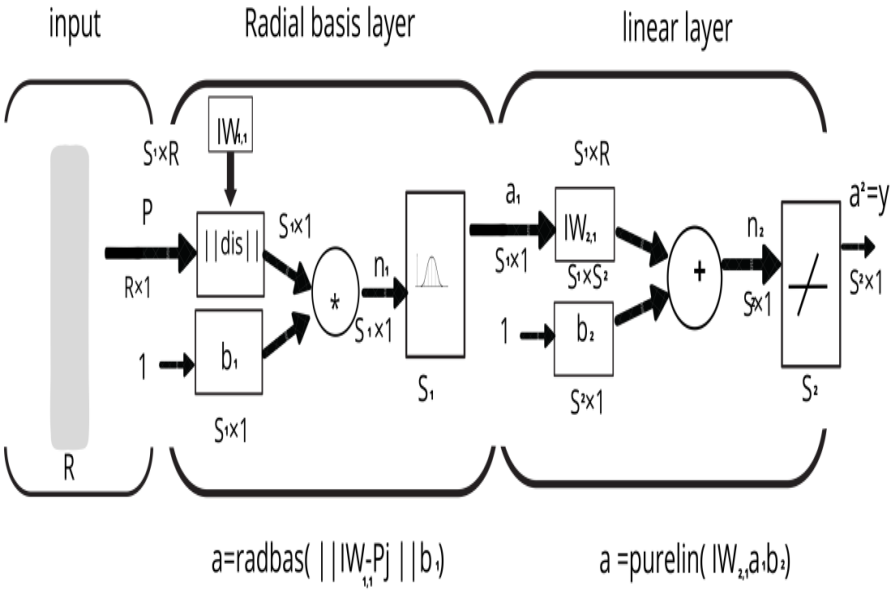

Figure 3.11: A typical RBFNN

As shown in Figure 3.11, if the input layer's weight vector IW contains S1 elements and the second (linear) layer's weight vector LW contains S1 S2 elements, the input vector p has R elements, the output vector y has S2 elements, and the input layer's and the second layer's biases are represented by b1 and b2,then:

$$y_k = \sum_{i=1}^{S^1} radbas(\sum_{i=1}^{R} IW_{ij}P_j + b_{1j})LW_{ik} + b_{2k}$$

- **Convolutional Neural Network** A Convolutional Neural Network (CNN) plays a crucial role in tasks related to image classification and picture recognition, making it the primary choice for such applications. CNNs are commonly employed in various domains, including face recognition and object identification. Like Feedforward Neural Networks (FNNs), CNNs consist of neurons that possess biases and weights

that can be learned. CNNs accept input images that have been labeled and preprocessed to belong to specific categories, such as dog, cat, lion, tiger, and so on. It is widely known that the resolution of an image impacts how the computer perceives it in terms of pixels. Depending on the image resolution, the computer will interpret it as having dimensions of h * w * d, where h represents height, w represents width, and d represents the image's depth or number of channels In CNN, every input image undergoes a series of convolution layers, pooling, fully connected layers, and filters (also referred to as kernels). The Softmax function is then applied to classify an object, assigning probabilistic values between 0 and 1 [41].

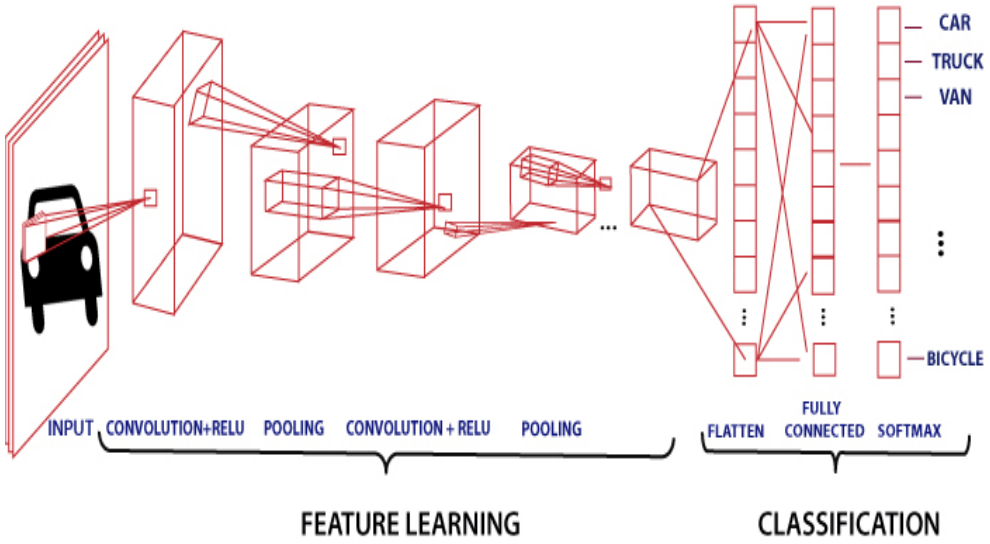

Figure 3.12: Convolutional Neural Network

- **Recurrent Neural Network** The foundation of Recurrent Neural Networks (RNNs) lies in prediction. In this type of neural network, the output of a specific layer is stored and looped back to the input, enabling the prediction of the layer's behavior. The initial layer of an RNN is constructed similarly to that of a Feedforward Neural Network (FFNN), and the recurrent neural network process commences from the subsequent layer[41]. While inputs and outputs are typically treated as independent in most cases, there are situations where it becomes necessary to forecast the subsequent word in a sentence. Subsequently, the prediction will rely on the preceding word within the sentence. The distinguishing and critical characteristic of Recurrent Neural Networks (RNNs) is their Hidden State, which retains information about a sequence[41].

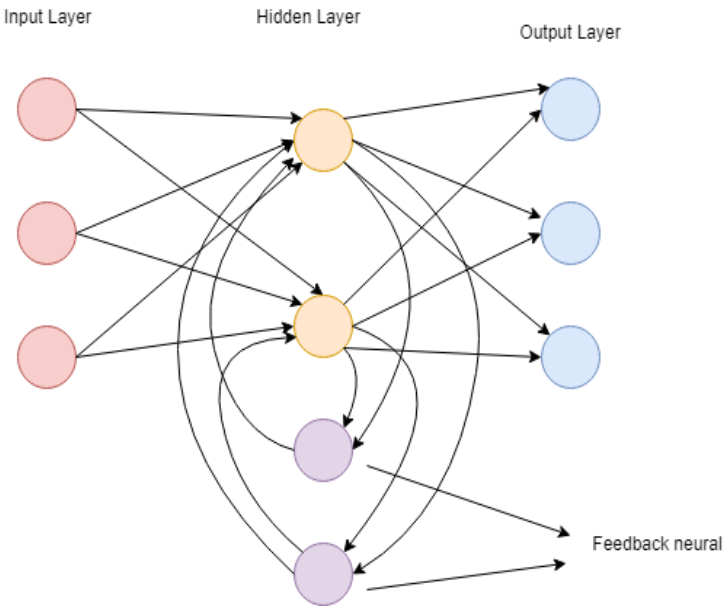

Figure 3.13: Recurrent Neural Network

3.8 Conclusion

In this chapter, we covered the topic of neural networks and various machine learning algorithms, including supervised, unsupervised, semi-supervised, and reinforcement learning. Neural networks are inspired by the human brain and consist of interconnected neurons that learn from data by adjusting weights and biases. Supervised learning involves training models using labeled data, while unsupervised learning discovers patterns in unlabeled data. Semi-supervised learning combines labeled and unlabeled data to improve model performance, and reinforcement learning focuses on training agents to make optimal decisions in dynamic environments. Each learning algorithm has its strengths and applications, helping us choose the appropriate approach for different machine learning tasks. In the upcoming chapter, we will delve into Federated Learning for Cache Management

Chapter 4

Federated Learning for Cache Management

4.1 Introduction

An essential component of federated learning systems' effective operation is cache management. It is possible to train models on distributed datasets using federated learning, a decentralized machine learning method, without the requirement for data centralization. In this model, the training process is shared across several edge devices or local servers, each of which contributes its own local data while retaining local storage for it.
The tactics and methods used to efficiently manage the caches existing on the edge devices or local servers taking part in the federated learning system are referred to as "cache management for federated learning." In federated learning contexts, caching is essential for lowering communication costs and boosting training effectiveness.

4.2 Related work

- **Work 1** [43] **:** In this article by Yang et al., a comprehensive overview of federated machine learning is provided, encompassing both theoretical foundations and practical applications. The article explores the motivations behind federated learning, the challenges involved in its implementation, and the potential advantages in situations where data cannot be stored centrally. Various federated learning algorithms and architectures are presented, along with their application areas in domains such as healthcare, Internet of Things (IoT), and mobile devices. Additionally, the article addresses the security and privacy concerns associated with federated learning and outlines future research directions in the field.

- **work 2** [44]**:** In this article, the focus is on training distributed machine learning models for resource-constrained IoT devices using federated learning (FL). The article examines existing studies on FL, discusses the assumptions for implementing FL with IoT devices, and identifies their drawbacks. It also delves into the challenges and issues encountered when applying FL in an IoT environment. The article provides a comprehensive overview of FL, surveys problem statements, and highlights emerging challenges, with a particular emphasis on heterogeneous IoT

environments. Furthermore, it suggests future research directions that intersect FL with resource-constrained IoT environments.

4.3 Federated Learning

4.3.1 Introduction

The rapid development of Internet of Things (IoT) and social media applications is leading to a significant growth in data generated at the network, which is experiencing an exponential increase. fully It was predicted that the rate of data generation would soon be exceeded Bandwidth of the current Internet. Frequent sending of all data to a remote cloud unnecessary and impossible due to network traffic and privacy issues.
By leveraging federated learning, the need for storing training data in the cloud is eliminated, as mobile devices collaboratively develop a shared prediction model while retaining all the training data on the device it self[45].
In the beginning, the central server distributes the current global model to the clients or workers.The group of clients then carry out local optimization and send the central server the learnt model weights. By carefully combining the client updates, the central server builds a worldwide shared model. To reduce network traffic, it is preferable to do the client training in batches[46].
Data owners often face limitations in sharing their raw data for cloud-based model training due to privacy, security, or legal restrictions. To train the global model, they might profit from a collaborative learning process. In this particular scenario, federated learning proves to be an appropriate solution. Federated supervised learning incorporates an inherent privacy-preserving method as it only uploads the model weights, which are a statistical summary of multiple raw data samples, to the cloud. Additionally, for federated supervised learning to work, tagged data needs to be available at the edge[46].

4.3.2 Federated Learning Steps

Four main steps make up the FL training process:

- Transfer of the model parameter from the server to the clients[47]
  In order to change the parameter sent back to the edge nodes for the subsequent iteration, the aggregator aggregates these parameters.[1]
- The client's local training[47]
- The transfer of model parameters from clients to servers.[47]
- The server's aggregation of model parameters.[47]
  The aggregator combines these parameters and sends an updated parameter back to the edge nodes for the next iteration. It is possible to alter the repetition rate of global aggregation, whether it occurs after one or more local updates [1].

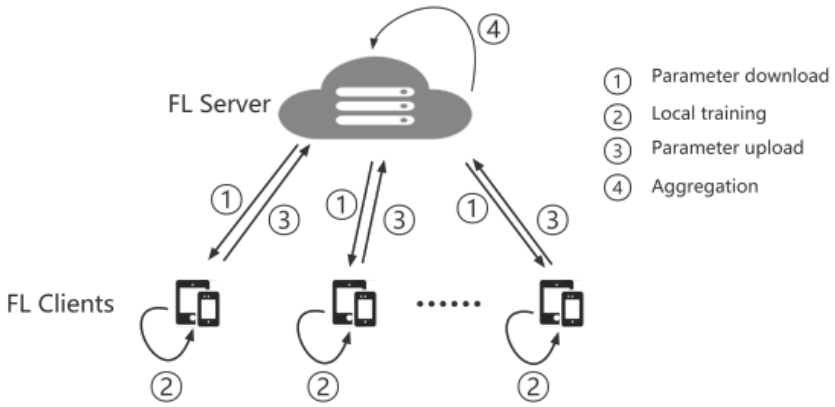

Figure 4.1: steps of FL training process

4.3.3 Federated Learning Algorithms

Loss Function

A set of parameters are part of machine learning models, and they are discovered using training data. Typically, a training data sample j has two components. One is a scalar yj, which is the desired output of the model, and the other is a vector xj, which is considered the input of the machine learning model (such as the label of the image). Each model has a loss function defined on its parameter vector w for each data sample j to aid in learning. The learning procedure for the model involves minimizing the loss function on a set of training data samples. The loss function represents the error of the model on the training data[1].

Stochastic Gradient Descent

An optimization approach called Stochastic Gradient Descent (SGD) is frequently used in deep learning and machine learning to train models. It is a variation on the gradient descent approach that is more computationally effective for large datasets since it changes model parameters based on a small selection of training data rather than the complete dataset at once. A model's loss function is minimized using SGD by iteratively updating a model's parameters in the direction of the loss function's negative gradient with respect to the parameters[48].

Federated Averaging

The popular federated learning approach known as "federated averaging" includes averaging model updates from various clients or entities. According to this method, each client updates the model weights and sends them to a central server after training the model on its own local data. The weighted average of the model weights from various clients, which are calculated based on variables like the quantity of local data samples or the processing capability of each client, is then combined by the central server. The process is repeated iteratively to improve the overall model before the aggregated model

is provided back to the customers[1].

Algorithm: Federated Averaging

Input:

- Set of N clients: $C_1, C_2, \ldots, C_N$
- Shared global model parameters: θ_0
- Number of communication rounds: T
- Learning rate: η
- Client selection method: ClientSelect()
- Weight assignment method: WeightAssign(C_i)
- Encryption function: Encrypt(x)
- Decryption function: Decrypt(x)

Output: Aggregated global model parameters: θ_T
Initialization:

- Initialize θ_0 randomly

For $t = 1$ **to** T **do**:

1. **Select** a subset of clients $S_t \subseteq \{C_1, C_2, \ldots, C_N\}$ using ClientSelect()
2. **For** each client $C_i \in S_t$ **do** in parallel:
   - Receive current global model parameters: θ_{t-1}
   - Update local model parameters using local data: $\theta_i \leftarrow \theta_{t-1} - \eta \nabla f_i(\theta_{t-1})$
   - Encrypt local model parameters: $\theta_i^{enc} \leftarrow \text{Encrypt}(\theta_i)$
   - Send encrypted local model parameters to central server
3. **At** central server:
   - Aggregate encrypted local model parameters: $\Theta^{enc} \leftarrow \sum_{i=1}^{N} \text{WeightAssign}(C_i) \cdot \theta_i^{enc}$, where WeightAssign($C_i$) assigns a weight to each client based on its importance or contribution
   - Decrypt aggregated model parameters: $\Theta \leftarrow \text{Decrypt}(\Theta^{enc})$
   - Update global model parameters: $\theta_t \leftarrow \Theta$
4. **Broadcast** updated global model parameters to all clients

Return: Aggregated global model parameters: θ_T

4.4 Cache Decision Based on Federated Learning

Due to the rapid advancement of smart devices and the rising demand for video streaming services, user latency is increasing. This is mostly due to the substantial growth in mobile traffic, putting a pressure on the networks that connect local base stations to the Internet. To overcome this issue and boost performance while lowering costs,

Content caching has emerged as a potential method in edge computing due to user latency and backhaul network congestion. Content caching tries to mitigate these issues by storing frequently requested files at neighboring base stations. However, due to the cache entity's limited storage capacity, it is critical to estimate future popularity of things and proactively store the most popular files. In recent years, several caching systems have focused on addressing the primary difficulty of proactive caching, which requires learning and interpreting content popularity trends [11].
Using the multi-armed bandit (MAB) approach to investigate the popularity distribution of the material for content caching, a collaborative filtering-based caching algorithm for small cell networks is proposed. Nonetheless, existing proactive caching solutions are designed for highly regulated settings where users must upload their local data to the central server, which may pose privacy and security problems, and scalability is a difficulty for those systems. as the number of users and the amount of data collected by users grows. We suggest a hierarchical architecture termed Federated learning based Proactive Content Caching (FPCC) approach to overcome the aforementioned difficulties. This method is divided into two layers: the bottom layer is made up of users that request material, and the top layer is made up of a central server outfitted with a cache entity [11].
Each user in the FPCC scheme downloads a stacked autoencoder model from the server. The model is then trained using local data, and the updated model parameters are uploaded to the server on each communication loop. Furthermore, users propose a set of N files to the server. The recommendation method employs hybrid filtering, which takes use of the similarity between people and files based on latent characteristics retrieved using the stacked autoencoder[11].

The federated averaging procedure is used by the server to aggregate the submitted model parameters. It also chooses the most popular files from among those suggested by all users. This strategy mitigates security and privacy problems by keeping training data local and transmitting only model parameter changes to the central server[49].

Several content caching strategies that do not rely on prior knowledge of content popularity have been created. These approaches use machine learning techniques such as reinforcement learning and collaborative filtering to evaluate the popularity of files. The caching method, for example, is created particularly for tiny cell networks. To measure the popularity of material, this method employs collaborative filtering (CF). This is accomplished by utilizing sparse training data throughout the training phase [11].
A collaborative stacking autoencoder based on hybrid filtering is used by both the model in each user's equipment and the central server. This signifies that the model employs a collaborative approach in which user recommendations and preferences are considered. Furthermore, the hybrid filtering approach is employed to compute the similarity between people and files based on the latent characteristics retrieved by the stacked autoencoder[11].

4.4.1 The FPCC Scheme

This study's proactive content caching technique is made up of three key procedures: encoding, hybrid filtering, and federated learning. The primary principle behind content caching is to leverage user requests and contextual information to learn how to make better caching decisions in the future. As a result, in order to cache the most popular files for each user, a caching entity must understand the popularity of information particular to their context [11].

The key problem is determining the similarities between distinct sets of material for each user, as well as the similarities between files, using a complicated dataset obtained from individuals' contextual information. This problem lends itself readily to a neural network model, thus we will modify and expand a specific neural network model called stacked autoencoder based on hybrid filtering to properly solve it [11].

Stacked Autoencoder

An unsupervised learning model is stacked autoencoder. The stacked autoencoder successfully trains a neural network with one hidden layer to recover input data from its latent representation problem[50].

In recent years, neural network models have shown great potential in learning hidden representations[51]. The stacked autoencoder takes a set of data instances, denoted as x(1), x(2), x(i), ..., x(m), where i ranges from 1 to m. Each instance x(i) belongs to a d-dimensional space (Rd)[52].

The stacked autoencoder is made up of two basic parts: an encoder and a decoder. Using an activation function h(x), the encoder transfers the input data to a hidden representation, denoted as y(i).The original is then mapped to the latent representation.From the input, the decoder generates a reconstructed version of x[52].

The goal of the stacked autoencoder is to learn a function hW,b(x) that approximates the input x, where W and b are weight matrices and bias vectors, respectively. By training the autoencoder, the model aims to find the optimal values for W and b that minimize the reconstruction error, thereby learning meaningful latent representations of the input data[52].

Hybrid Filtering

The similarity of users and files is assessed by using stacked autoencoders, which extract characteristics from the users and files. We assess the similarity of the files based on the current user's watch history and the watch history of their K nearest neighbor users to provide a suggestion list of popular files for caching. The active user reflects the user's requests in a given scene, whereas the watch history of the K nearest neighbor users represents comparable scenes connected to that unique scene. Hybrid filtering is used, which combines content-based, demographic, and collaborative filtering. This method primarily use similarity techniques to determine the distance between two files or two individuals based on their file ratings and personal profiles [11].

1. Data pre-processing: entails constructing a rating matrix based on each user's request history. In addition, a user information matrix is created by combining personal information such as location and time of day[11].

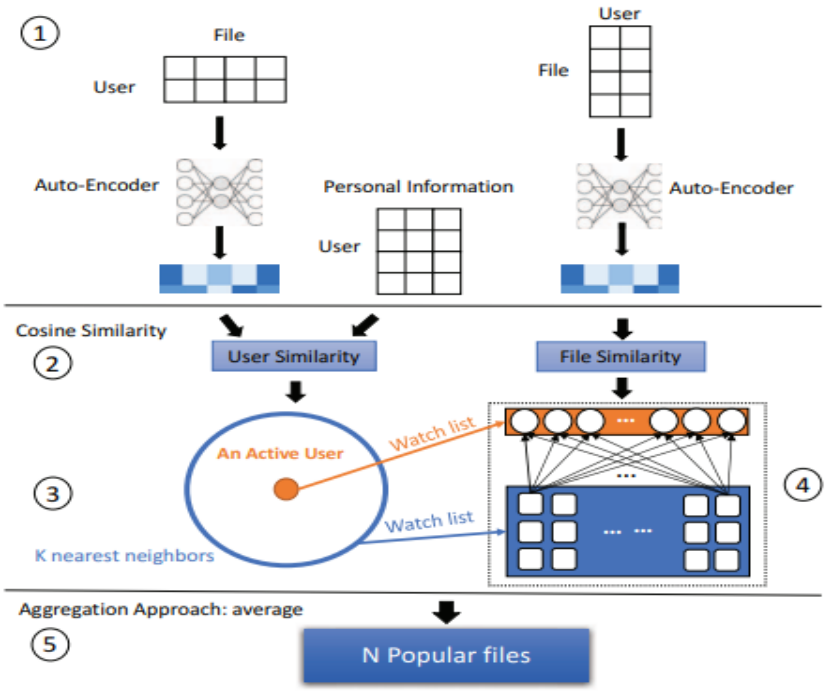

Figure 4.2: Hybrid Filtering Model

2. Exploration of latent representation: Using the rating matrix as input, the autoencoder discovers hidden characteristics and connections between users and files. These newly found characteristics are then combined with the user information matrix to produce similarity matrices for both users and files. This approach makes use of cosine similarity, which is well-known for its efficacy with sparse matrices. The entries in the user's and file's similarity matrices reflect the distances between each user and file, respectively[11].

3. The history matrix is constructed as follows: Assuming that the current user is the active user, we utilize the user's similarity matrix to find the K nearest neighbor users. We create a matrix of historical watchlists (K*) from these K chosen neighbor users' prior watching histories[11].

4. Obtaining the similarity: The matrix of the active user's history requests is designated as A*. The similarity matrix of files is used to compute the average value of the similarity between each element in A* and K*. This entails comparing each element in A* to the equivalent elements in K* and calculating their average similarity value using the file similarity matrix [11].

5. Aggregation: The popularity of files is predicted using an aggregation strategy, which entails picking the files with the greatest to nth greatest similarity. This selection procedure creates a list of popular files for caching. Each user submits their own list of recommendations to the server. As shown in Fig 4.2 , the server collects all estimated results from users before selecting the top N most popular files. These files are considered cache material for the cache entity [11].

Federated Learning

To aggregate the results, the parameters of the stacked autoencoder must be transferred from the user side to the central server. This method has a big benefit because it allowing the model to be trained using local user data without the need to upload the actual data to the central server. By avoiding data transmission, it greatly reduces security and privacy risks. To implement this approach, users first download the global model W from the central server. They then compute their respective updated models, denoted as W1t, W2t, W3t, ..., Wct, based on their local data. Here, t represents the round number, and 1, 2, 3, ..., c corresponds to the index number of each participating user. The updates are represented as Hct := Wct - Wt, which captures the difference between the participant's local model and the global model. Finally, the updated parameters, along with a recommendation list of popular files estimated through hybrid filtering, are sent to the central server. This enables the central server to receive the necessary model updates and the suggested list of popular files, facilitating further analysis and decision-making based on the aggregated information[11].

Algorithm 4 Content caching algorithm: User

Require: c, w, b: Parameters

B: Split training data into batches of size B
E: Number of local epochs
η: Learning rate
for $i = 1$ to E **do**
  for each batch $b \in B$ **do**
    $w \leftarrow w - \eta \cdot \nabla(w; b)$
  end for
end for
$UserFeatures \leftarrow AUTOENCODERUSER$(user-file matrix)
$UserSim \leftarrow SIMILARITY(UserFeatures)$
$FileFeatures \leftarrow AUTOENCODERFILE$(file-user matrix)
$FileSim \leftarrow SIMILARITY(FileFeatures)$
$AUTOENCODERUSER(x)$
$autoencoder(x, x, E, B)$
$AUTOENCODERFILE(y)$
$autoencoder(y, y, E, B)$
$SIMILARITY(x)$
$\text{sim}(\text{It}^i, It^j) = \frac{It^i It^j}{|It^i||It^j|}$
Select $K_1, K_2, K_3, ..., K_n$ from $UserSim \rightarrow K^*$
Average: $\frac{1}{n}\sum_{n=1}^{n}$ for the similarity between requested content of target user K^* and $A^* \rightarrow N^*$
Select $N_1, N_2, N_3, ..., N_n$ from N^*
return $c, w, b, N = 0$

On the server side, the central server combines all the updated models from the user side in order to enhance its global model using Federated Averaging, as shown in Equation (1).

$$\eta_t$$

Algorithm 5 Content caching algorithm: Server

Require: C: Users are indexed by c
Require: t: Number of communication rounds
Require: w, b: Model parameters
Require: N_c: Predict popular files from each user
Initialize w, b
for $t = 1, 2, ..., t$ **do**
  for each user $c \in S_t$ in parallel **do**
    $w_{t+1}^c, b_{t+1}^c, N_{t+1}^c \leftarrow$ User Updates$(c, w_{c,t}, b_{c,t})$
  end for
  $w_{t+1} \leftarrow \sum_{c=1}^{C} \frac{1}{C} w_{t+1}^c$
  $b_{t+1} \leftarrow \sum_{c=1}^{C} \frac{1}{C} b_{t+1}^c$
end for
Count A_c
Select top-N $N_1, N_2, N_3, ...,$N$_n = 0$

represents the learning rate[11].

$$W_{t+1} = W_t + \eta_t H_t, \quad H_t = \frac{1}{n_t} \sum_{i \in S_t} H_i^t \quad (1)$$

Federated Averaging employs a weighted average sum to combine all updates, taking into account the size of each selected user's training dataset. Ultimately, the server produces a list of recommended popular files for caching[11].

4.5 Dataset

id-user	id-movie	rating	timestamps
58365	31	3	836391314
58365	32	5	836391153
58365	34	4	836391124
58365	44	4	836391253
58365	44	3	836391292
58365	62	5	836391332

Table 4.1: Dataset example

The dataset used in this study consists of four columns: id-user , id-movie ,rating and timestamps. Each row in the dataset represents a unique instance, and there are a total of 1600 rows .
The id-user column contains identifiers for individual users who provided ratings for the movies . The id-movie column contains identifiers for the movies that were rated . The rating column represents the ratings given by the users for the corresponding movies, which can range from a minimum value to a maximum value,where 1 indicates the lowest rating and 5 indicates the highest rating. Lastly,the timestamps column records the time when each rating was given.

The dataset has been divided into four subsets: data1, data2, data3, and data4. The division is based on the line numbers in the dataset. Data1 contains the entries from line

0 to 399, data2 contains the entries from line 400 to 799, data3 contains the entries from line 800 to 1199, and data4 contains the entries from line 1200 to 1600

In many cases, privacy concerns arise when dealing with sensitive data in federated learning settings. To address these concerns, Cache Management techniques are employed. In the given scenario, the dataset has been divided into four subsets: client 1, client 2, client 3, and client 4.

By dividing the dataset into clients, each client retains control over its respective subset of data. This decentralized approach helps protect the privacy of individual users and their sensitive information. Cache Management techniques can be applied to efficiently manage the caching and retrieval of data in each client, ensuring that only necessary and relevant data is stored locally.

Federated Learning leverages this distributed architecture, allowing models to be trained collaboratively across multiple clients while preserving data privacy. The use of Cache Management techniques helps optimize the communication and computation overhead, enabling efficient and secure federated learning processes.

By utilizing Cache Management in Federated Learning, the privacy concerns associated with sharing sensitive data are mitigated, allowing for collaborative model training while ensuring the confidentiality and privacy of individual clients' data.

4.6 Implementation of Federated Learning for Cache Management

The employment of federated learning within a caching system necessitates the crucial involvement of the aggregator in the overall procedure. The central server assumes the role of coordinating the Federated Learning (FL) process, which involves the distribution of the initial global model to the clients, collection and aggregation of the model updates from the clients (averaging the received weights), and the generation of an enhanced global model. The aggregator functions to ensure model synchronisation across various devices and supports collaborative learning, all while upholding data privacy and security. In our implementation of federated learning for cache management, we have 4 clients, each with a dataset fragment to train a local artificial neural network model to predict movies with a rating greater than 3, which are considered the most popular and are therefore added to the cache memory. After the learning process, we obtained hyperparameters that encompass the number of neurons and the error percentage during each epoch of training. In the local model, we incorporate three inputs for the Artificial Neural Network (ANN): user ID, movie ID,timestamp, and film. These inputs are utilized to predict the output ratings.
The dataset fragment was divided into 200 elements for testing and 200 elements for training. Specifically, the neural network took 133 elements for testing purposes.Learning stopped after 200 epoch due to convergence of learning error values MES.

In the server-side part of the global model, we aggregate the hyperparameters from each client into a single document. This document serves as the new dataset for the global model.In the global model, we incorporate tow inputs for the Artificial Neural Network (ANN): Mean Squared Error(MES). These inputs are utilized to predict the output ratings.
In the centralized part, we utilize a single client that possesses the global dataset. This

client is equipped with an artificial neural network comprising three inputs: user ID, film ID, timestamp, and the ratings of films as the output.

4.6.1 Local Model

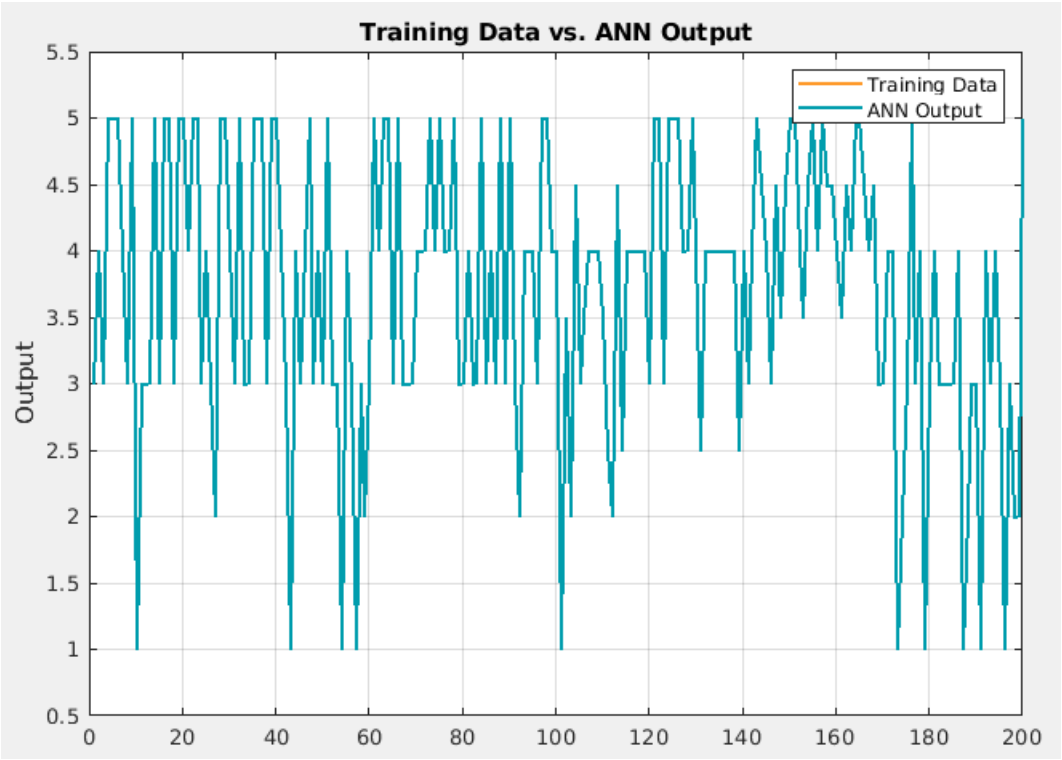

Figure 4.3: Traning Data

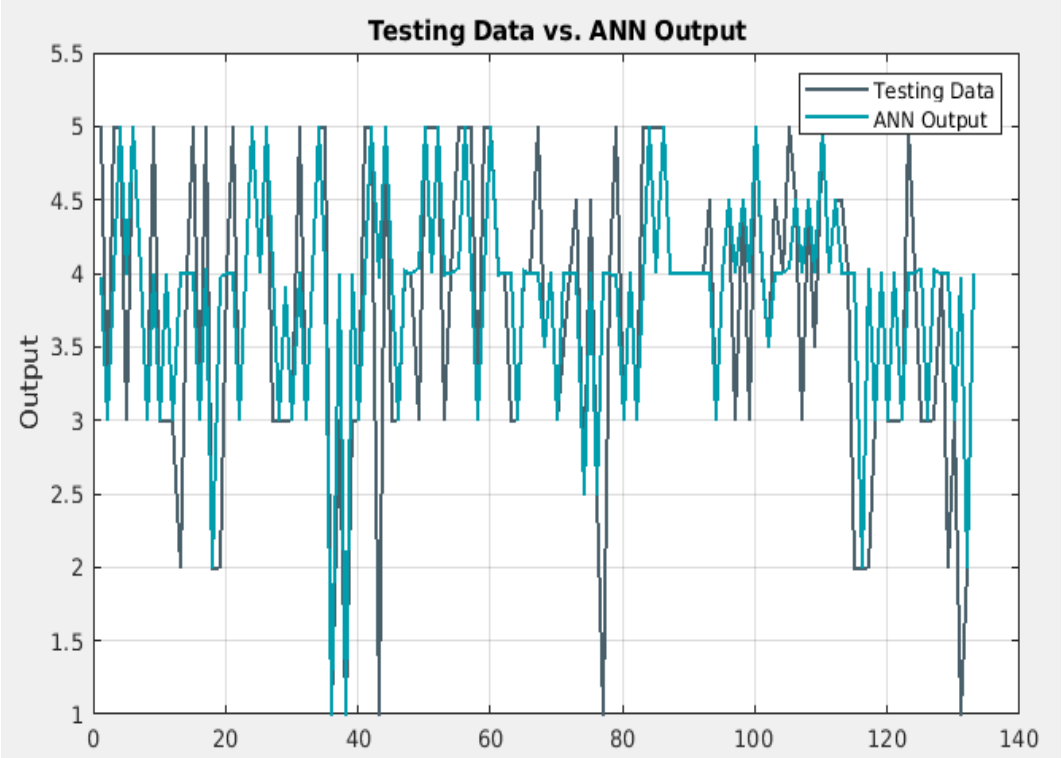

Figure 4.4: Testing Data

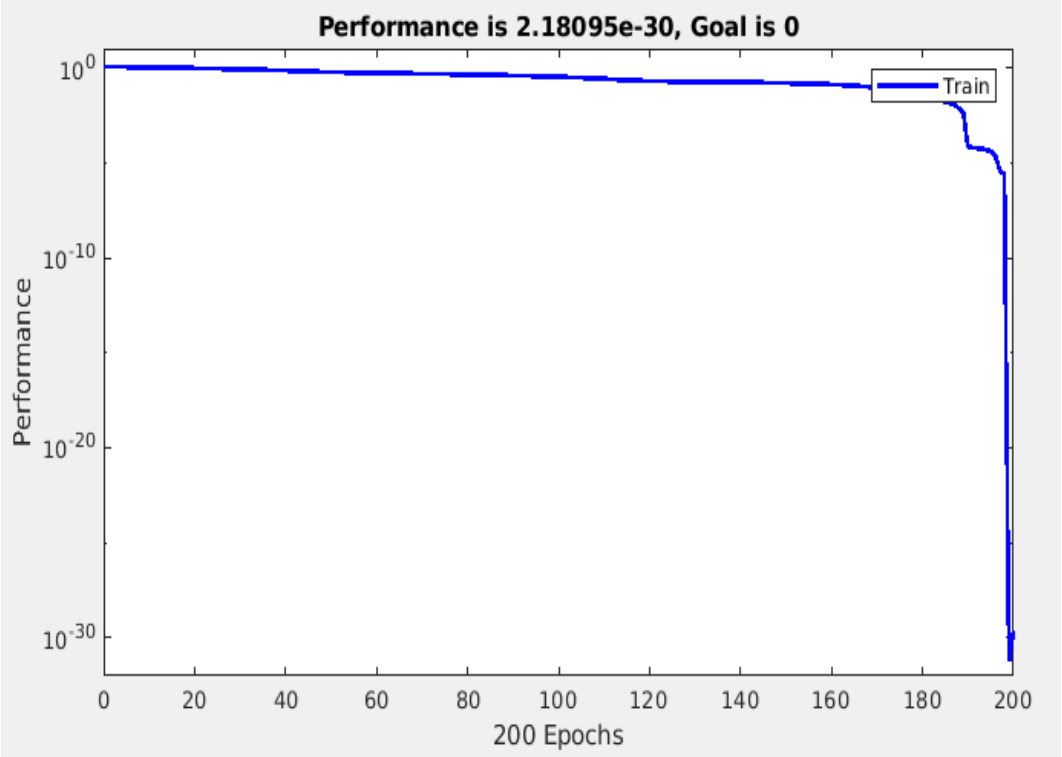

Figure 4.5: Perfomance

4.6.2 Discussion of Graphs

A machine learns by taking input data and processing it to generate an output. The input curve provides the machine's original data, whereas the output curve shows the machine's predictions or created outcomes based on that input.When the input and output curves are described as "somewhat similar", this suggests that the machine's predictions are relatively accurate compared to the original data. This indicates that the computer understands the incoming data and learns from it.

In figure 4.3 An ideal situation when the learning process completely matches the input curve is shown in the initial learning curve. This shows that the model is accurately capturing the relationships and patterns seen in the data, producing reliable predictions. The model's output values are constrained to the range of 1 to 5, indicating some kind of threshold or constraint in the system. In addition, it says that movies with a rating of 3 or above are popular and given preference for storage in the cache memory, whereas movies with a rating of 3 or lower are not stored.
In figure 4.4 A careful observation reveals good similarities between the input and output curves. This similarity indicates that machines can begin to capture underlying patterns and relationships in the data and produce output that closely matches the input provided. This means that the machine learning process is on track as it effectively captures and reproduces the intrinsic properties of the input data. This promising development means machines are gradually improving their ability to learn and make accurate predictions.
In figure 4.5 The model's performance is remarkable, as shown by the performance graph, which has a performance values of 2.18095×10^{-30} and 2.18095×10^{-30} Given that this value is so close to the ideal goal value of 0, the model has performed almost perfectly. The model has successfully reduced the loss or mistake in its predictions, indicating a high level of precision and dependability.
The machine stopped learning after 200 cycles because the error values reached a point where they couldn‘t get any better. This means that the machine has reached its highest level of learning and doesn't need to improve anymore.

4.6.3 Global Model (averaging aggregator)

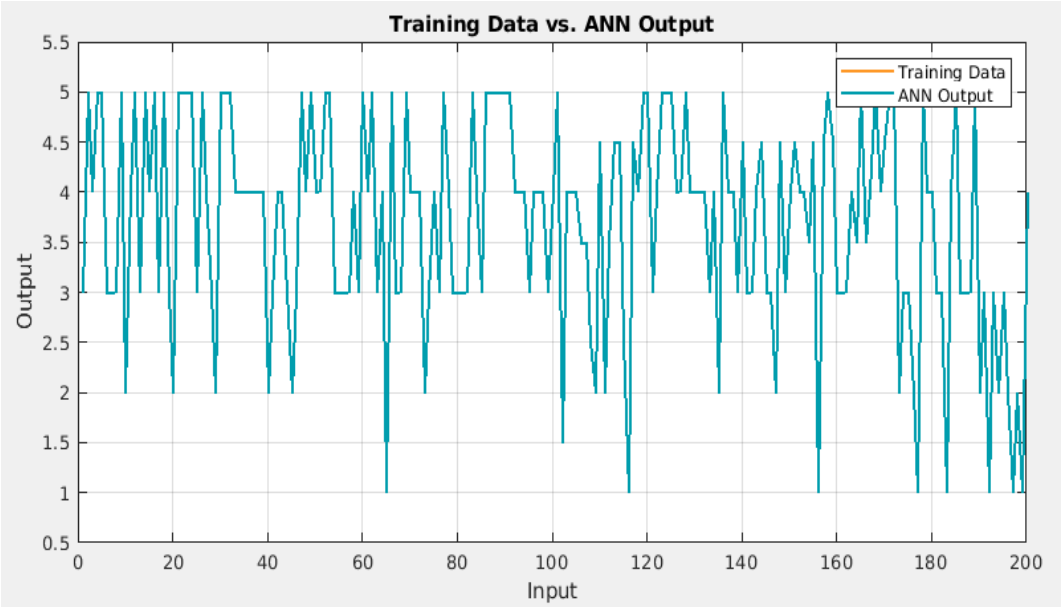

Figure 4.6: Training Data

Figure 4.7: Testing Data

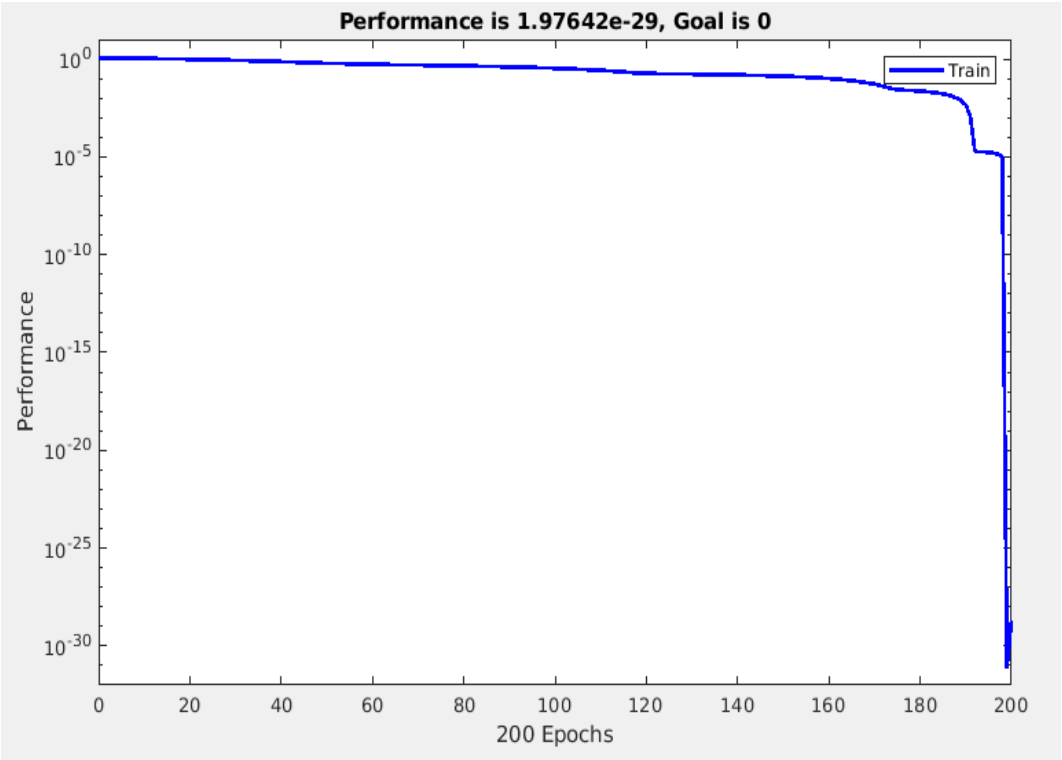

Figure 4.8: Perfomance

4.6.4 Discussion of Graph

The first Figure 4.6 curve depicts an ideal case in which the learning process fully matches the input curve. This implies that the model is accurately capturing data correlations and patterns, allowing it to make trustworthy predictions.
In figure 4.7 the discussion identical with the local models testing curve.
In figure 4.8 The model's performance is outstanding, as shown by the performance graph. The performance value is incredibly close to zero(1.97642×10^{-29}.), which is the ideal goal value. This means that the model has almost achieved perfection. It has successfully reduced mistakes in its predictions, showing that it is very accurate and dependable.
For this performance curve, training stopped after 400 epochs for the same reason.

4.7 Conclusion

In this chapter, the concept of Federated Learning (FL) is introduced as a solution to address the challenges posed by the exponential growth of data generated at the network edge. FL allows mobile devices to collaboratively develop a shared prediction model while retaining the training data on the device itself, eliminating the need for storing training data in the cloud.
The FL training process involves four main steps: transferring the model parameters from the server to the clients, performing local training on the clients' devices, transferring the updated model parameters back to the server, and aggregating the model parameters on the server to build a global shared model.

Chapter 5

Conclusion

With the rapid technological advancements and the widespread adoption of the Internet worldwide, a tremendous amount of information has been distributed across networks. While this has brought numerous benefits, it has also introduced several challenges. One of the problems that emerged is the issue of response time for requests and replies. As the volume of information increased, it became more challenging to retrieve and process data promptly, leading to delays in communication between users and servers,Another problem stemming from the vast amount of information is the impact on network speed. The sheer volume of data being transmitted and accessed can strain network bandwidth, resulting in slower connection speeds for users. Network congestion is yet another challenge caused by the distribution of extensive data. As more users access the network simultaneously, the traffic load increases, leading to congestion and bottlenecks. Moreover, the burden on server loads has intensified due to the distribution of a large amount of information. Additionally, the transmission of vast amounts of data can increase the possibility of data packet loss. Another issue related to the distribution of vast amounts of information is the security and privacy of user data. To address the challenges mentioned earlier, a concept was introduced to alleviate the burden on databases and improve response times by bringing frequently accessed content closer to the users. This approach involves storing popular or frequently requested data in memory caches that are located closer to the users, rather than relying solely on centralized databases. To ensure the privacy of user information during this caching process, federated learning techniques is employed.

As future work, we plan to work on dynamic cache management strategies that respond to changing network circumstances and user demands can greatly increase resource efficiency and caching system performance. These strategies can enhance the caching process in a variety of ways by dynamically modifying cache sizes, taking into account network congestion levels, and adding real-time user feedback. The ability to alter cache sizes based on current network circumstances and user demands is one feature of dynamic cache management. The cache can be adjusted to fit the most relevant and frequently visited material by monitoring parameters such as network capacity, latency, and congestion levels. During moments of heavy network traffic or congestion, for example, the cache size can be expanded to store more popular or in-demand models, providing faster access .

Bibliography

[1] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE journal on selected areas in communications*, 37(6):1205–1221, 2019.

[2] Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, and Tie Qiu. Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of network and computer applications*, 98:27–42, 2017.

[3] Xueqing Zhang, Yanwei Liu, Jinxia Liu, Antonios Argyriou, and Yanni Han. D2d-assisted federated learning in mobile edge computing networks. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7. IEEE, 2021.

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[5] Zhe Zhang, Chung-Horng Lung, Marc St-Hilaire, and Ioannis Lambadaris. An sdn-based caching decision policy for video caching in information-centric networking. *IEEE Transactions on Multimedia*, 22(4):1069–1083, 2019.

[6] Kumari Nidhi Lal and Anoj Kumar. A cache content replacement scheme for information centric network. *Procedia Computer Science*, 89:73–81, 2016.

[7] Muhammad Bilal and Shin-Gak Kang. A cache management scheme for efficient content eviction and replication in cache networks. *IEEE Access*, 5:1692–1701, 2017.

[8] Isaac Odun-Ayo, M Ananya, Frank Agono, and Rowland Goddy-Worlu. Cloud computing architecture: A critical analysis. In *2018 18th international conference on computational science and applications (ICCSA)*, pages 1–7. IEEE, 2018.

[9] Yongqiang Zhang, Hongchang Yu, Wanzhen Zhou, and Menghua Man. Application and research of iot architecture for end-net-cloud edge computing. *Electronics*, 12(1):1, 2023.

[10] Yongqiang Zhang, Hongchang Yu, Wanzhen Zhou, and Menghua Man. Application and research of iot architecture for end-net-cloud edge computing. *Electronics*, 12(1):1, 2022.

[11] Zhengxin Yu, Jia Hu, Geyong Min, Haochuan Lu, Zhiwei Zhao, Haozhe Wang, and Nektarios Georgalas. Federated learning based proactive content caching in edge computing. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[12] Jiacheng Hou, Haoye Lu, and Amiya Nayak. A gnn-based proactive caching strategy in ndn networks. 2022.

[13] Nikolaos Laoutaris, Sofia Syntila, and Ioannis Stavrakakis. Meta algorithms for hierarchical web caches. *IEEE International Conference on Performance, Computing, and Communications, 2004*, pages 445–452, 2004.

[14] Yiqi Gui and Yongkang Chen. A cache placement strategy based on compound popularity in named data networking. *IEEE Access*, 8:196002–196012, 2020.

[15] Ikram Ud Din, Suhaidi Hassan, Muhammad Khurram Khan, Mohsen Guizani, Osman Ghazali, and Adib Habbal. Caching in information-centric networking: Strategies, challenges, and future research directions. *IEEE Communications Surveys & Tutorials*, 20(2):1443–1474, 2017.

[16] Maroua Meddeb, Amine Dhraief, Abdelfettah Belghith, Thierry Monteil, and Khalil Drira. Cache coherence in machine-to-machine information centric networks. In *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, pages 430–433. IEEE, 2015.

[17] Najla Alzakari, Alanoud Bin Dris, and Saad Alahmadi. Randomized least frequently used cache replacement strategy for named data networking. In *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–6. IEEE, 2020.

[18] Issam El Naqa and Martin J Murphy. *What is machine learning?* Springer, 2015.

[19] Qifang Bi, Katherine E Goodman, Joshua Kaminsky, and Justin Lessler. What is machine learning? a primer for the epidemiologist. *American journal of epidemiology*, 188(12):2222–2239, 2019.

[20] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9, 2020.

[21] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. *Supervised Learning*, pages 21–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[22] Erik G Learned-Miller. Introduction to supervised learning. *I: Department of Computer Science, University of Massachusetts*, page 3, 2014.

[23] Denis Corroyer and Marion Wolff. *L'analyse statistique des données en Psychologie: concepts et méthodes de base.* Armand Colin, 2003.

[24] Maysam Abedi, Gholam-Hossain Norouzi, and Abbas Bahroudi. Support vector machine for multi-classification of mineral prospectivity areas. *Computers & Geosciences*, 46:272–283, 2012.

[25] Michael W Berry, Azlinah Mohamed, and Bee Wah Yap. *Supervised and unsupervised learning for data science.* Springer, 2019.

[26] Alhamza Alalousi, Rozmie Razif, Mosleh AbuAlhaj, Mohammed Anbar, and Shahrul Nizam. A preliminary performance evaluation of k-means, knn and em unsupervised machine learning methods for network flow classification. *International Journal of Electrical and Computer Engineering*, 6(2):778, 2016.

[27] Youguo Li and Haiyan Wu. A clustering method based on k-means algorithm. *Physics Procedia*, 25:1104–1109, 2012.

[28] Shehroz S Khan and Amir Ahmad. Cluster center initialization algorithm for k-means clustering. *Pattern recognition letters*, 25(11):1293–1302, 2004.

[29] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 109(2):373–440, 2020.

[30] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

[31] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93, 2000.

[32] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.

[33] datascientest. Le machine learning avec apprentissage par renforcement. https://datascientest.com/q-learning-le-machine-learning-avec-apprentissage-par-renforcement, 2021.

[34] Laëtitia Matignon. *Synthèse d'agents adaptatifs et coopératifs par apprentissage par renforcement. Application à la commande d'un système distribué de micromanipulation.* PhD thesis, Université de Franche-Comté, 2008.

[35] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[36] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, 1996.

[37] Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

[38] Richard Palmer. Optimization on rugged landscapes. *Molecular Evolution on Rugged Landscapes: Proteins, RNA and the Immune System*, pages 3–25, 2018.

[39] Kui Xiang, Bing Nan Li, Liyan Zhang, Muye Pang, Meng Wang, and Xuelong Li. Regularized taylor echo state networks for predictive control of partially observed systems. *IEEE Access*, 4:3300–3309, 2016.

[40] John G Kuschewski, Stefen Hui, and Stanislaw H Zak. Application of feedforward neural networks to dynamical system identification and control. *IEEE transactions on control systems technology*, 1(1):37–49, 1993.

[41] Java t point. Artificial neural network in tensorflow. https://www.javatpoint.com/artificial-neural-network-in-tensorflow, 2013.

[42] Morteza Mohammadzaheri, Lei Chen, and Steven Grainger. A critical review of the most popular types of neuro control. *Asian Journal of Control*, 14(1):1–11, 2012.

[43] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[44] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M Hadi Amini. A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9(1):1–24, 2021.

[45] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[46] Anirban Das and Thomas Brunschwiler. Privacy is what we care about: Experimental investigation of federated learning on edge devices. In *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, pages 39–42, 2019.

[47] Lei Fu, Huanle Zhang, Ge Gao, Huajie Wang, Mi Zhang, and Xin Liu. Client selection in federated learning: Principles, challenges, and opportunities. *arXiv preprint arXiv:2211.01549*, 2022.

[48] Kallista Bonawitz, Peter Kairouz, Brendan McMahan, and Daniel Ramage. Federated learning and privacy: Building privacy-preserving systems for machine learning and data science on decentralized data. *Queue*, 19(5):87–114, 2021.

[49] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[50] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining*, pages 153–162, 2016.

[51] Xingrui Yu, Xiaomin Wu, Chunbo Luo, and Peng Ren. Deep learning in remote sensing scene classification: a data augmentation enhanced convolutional neural network framework. *GIScience & Remote Sensing*, 54(5):741–758, 2017.

[52] Qi Zhang, Jianhang Zhou, and Bob Zhang. A noninvasive method to detect diabetes mellitus and lung cancer using the stacked sparse autoencoder. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1409–1413. IEEE, 2020.