

LEBANESE AMERICAN UNIVERSITY

**Cooperative Caching Policy in Fog Computing for
Connected Vehicles**

By

Ali Ghazleh

A thesis

submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

School of Arts and Sciences

May 2023

© 2023

Ali Ghazleh

All Rights Reserve

THESIS APPROVAL FORM

Student Name: Ali Ghazleh I.D. #: 201705863

Thesis Title: Cooperative Caching Policy in Fog Computing for Connected Vehicles

Program: Master of Science in Computer Science

Department: Computer Science and Mathematics

School: Arts and Sciences

The undersigned certify that they have examined the final electronic copy of this thesis and approved it in Partial Fulfillment of the requirements for the degree of:

Master of Science in the major of Computer Science

Thesis Advisor's Name: Ramzi A. Haraty

Signature:  Date: 19 / 05 / 2023
Day Month Year

Committee Member's Name: Samer Habre

Signature:  Date: 19 / 05 / 2023
Day Month Year

Committee Member's Name: Sanaa Kaddoura

Signature:  Date: 19 / 05 / 2023
Day Month Year

THESIS COPYRIGHT RELEASE FORM

LEBANESE AMERICAN UNIVERSITY NON-EXCLUSIVE DISTRIBUTION LICENSE

By signing and submitting this license, you (the author(s) or copyright owner) grants the Lebanese American University (LAU) the non-exclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic formats and in any medium, including but not limited to audio or video. You agree that LAU may, without changing the content, translate the submission to any medium or format for the purpose of preservation. You also agree that LAU may keep more than one copy of this submission for purposes of security, backup and preservation. You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant LAU the rights required by this license, and that such third-party owned material is clearly identified and acknowledged within the text or content of the submission. IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN LAU, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT. LAU will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Name: Ali Ghazleh

Signature:



Date: 05 / 05 / 2023

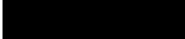
Day Month Year

PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that:

1. I have read and understood LAU's Plagiarism Policy.
2. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
3. This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Ali Ghazleh

Signature: 

Date: 05 / 05 / 2023
Day Month Year

DEDICATION

This work is dedicated to my family and friends, who supported me during the different stages of this accomplishment.

ACKNOWLEDGMENT

I want to express my heartfelt gratitude to Dr. Ramzi Haraty for his advice and assistance throughout this journey. I would also like to thank my committee members, Dr. Samer Haber and Dr. Sanaa Kaddoura, for their support and cooperation.

Cooperative Caching Policy in Fog Computing for Connected Vehicles

Ali Ghazleh

ABSTRACT

In this era, the magnitude of data shared is enormous and raised the bar for the quality of service and maintenance it requires. This paved the road for the integration of Fog Computing, which is an extension of the Cloud. Fog Computing's main advantage is the increased quantity in which it can be deployed while in close vicinity of the end-users, thus enhancing their Quality of Experience (QoE). The connected vehicles domain is one of many domains that can benefit from Fog Computing. Moreover, caching has been an area of study for many years by researchers that aim to increase cache hit rate and decrease request delays affecting Connected Vehicles networks. Many studies implemented Machine Learning models to enhance cache hit rate and request delays. In this thesis, we implemented cooperation between a Deep Reinforcement Learning (DRL) model and Federated Learning to improve caching in Connected Vehicles connected to fog nodes. Furthermore, the results showed the proposed model's effectiveness compared to traditional algorithms.

Keywords: Fog Computing, Caching, Connected Vehicles, Machine Learning, Deep Reinforcement Learning, Federated Learning.

TABLE OF CONTENTS

DEDICATION.....	v
ACKNOWLEDGMENT.....	vi
ABSTRACT.....	vii
TABLE OF CONTENTS.....	viii
LIST OF TABLES.....	ix
LIST OF FIGURES.....	x
Introduction.....	1
1.1 Overview.....	2
1.1.1 IoT Devices.....	2
1.1.2 Cloud Computing.....	4
1.1.3 Fog Computing.....	6
1.1.4 Caching.....	9
1.2 Problem Statement.....	10
1.3 Thesis Contribution.....	11
1.4 Thesis Organization.....	11
Related Work.....	12
2.2 Caching.....	14
The Suggested System Model.....	31
3.1 Model overview.....	32
3.2 Dueling Deep Q-Network (DDQN).....	34
3.3 Horizontal Federated Learning (HFL).....	37
3.4 An Example.....	38
Experimental Results.....	41
4.1 Simulation Environment.....	41
4.2 Data.....	41
4.3 Model Performance.....	42
Conclusion and Future Work.....	48
References.....	49

LIST OF TABLES

Table 1- Cloud cache at $t=0$	38
Table 2- Fog node cache at $t=0$	38
Table 3- RSU cache at $t=0$	39
Table 4- Cloud cache at $t=1$	39
Table 5- Fog node cache at $t=1$	39
Table 6- RSU cache at $t=1$	40

LIST OF FIGURES

Figure 1- Forwarding scheme results [13].....	15
Figure 2- Cache replacement results [13].....	16
Figure 3- Model illustration [14].....	17
Figure 4- Cache hit rate as a function of time [14].....	18
Figure 5- Request delay as a function of time [14].	18
Figure 6- Model Illustration [19].....	19
Figure 7- FLCH scheme flow [19].	20
Figure 8- LSTM and collaborative caching model flow [21].....	22
Figure 9- FADE System model [29].	25
Figure 10- Simulation results for FADE [29].....	26
Figure 11- Example of cache replacement flow adopted in [30].	27
Figure 12- The Suggested Model.	33
Figure 13- Model workflow pseudo code.	34
Figure 14- Architecture comparison between DQN (top) and DDQN (Bottom) as provided in [34]...	35
Figure 15- Experimental results DQN vs. DDQN in [34].....	36
Figure 16- HFL architecture as represented in [35].	37
Figure 17- Cache hit rate versus time, where $s=0.8$, $N=m=1000$	43
Figure 18- Cache hit rate versus time, where $s=1$, $N=m=1000$	44
Figure 19- Cache chit rate versus time, where $s=1$, $N=m=2000$	45
Figure 20- Cache chit rate versus time, where $s=1$, $N=m=5000$	46
Figure 21- Request delay (ms x 1000) versus time T.....	47

CHAPTER ONE

INTRODUCTION

In today's day and age, the reliance on technology increased exponentially, and it is somehow become a dependency in multiple aspects, for example, automation, medicine, banking, gaming, video streaming, etc. One of the key similarities between these is data management, and here is where cloud computing came into play.

Cloud computing offers services and data management. Even if the data is not on the end user's personal computer [1], cloud computing allows the user to access his/her data from anywhere in the world with only an internet connection.

Moreover, fog computing generally extends the services of the cloud but with less power. Unlike the cloud, fog nodes can be deployed in many numbers and at the network's edge. They will be physically close to the end-users, increasing the quality of service, as well as offering high bandwidth and low latency.

Fog computing can enhance several aspects required daily such as QoS or Quality of Service. One of many fields that require high QoS would be connected vehicles or smart cars, and since they are connected to the internet, they are considered IoT devices. They are connected to the internet, which means they are communicating with other devices, and these communications require low latency, which the fog can provide.

Caching can be one of many services that can be integrated into the fog layer to optimize the communications and performance of connected vehicles. In other words, caching will allow these vehicles to perform tasks much faster, given that high-rate content retrieval is provided through the fog layer.

1.1 Overview

This section will address essential information that allows us to proceed with this thesis.

1.1.1 IoT Devices

In the past few decades, technological progress has been exponentially increasing in all fields, and communications are one of the many things that have progressed along the way. The inception of the Internet of Things or IoT devices has allowed for machine-to-machine M2M communications [2][3]. In other words, everything around us connected and communicated through the internet can be considered an IoT device [4].

To better understand the importance of IoT devices in the modern world, the authors of [3] compiled a list of applications that can integrate the functionalities of an IoT device, such as:

- i- Healthcare: the integration of IoT in the healthcare systems has been growing over the years even though it has not yet reached the majority of people, which means it still has growth potential in the coming decades. IoT has encouraged people to use their devices to monitor their health and share some data with authorized personnel such as doctors or lab operators.
- ii- Smart Livings: IoT has also been integrated into numerous fields essential for human beings. For example, IoT is used in agriculture to accelerate several stages of food production, which is essential for the increasing world population. Also, we can find that IoT had its way into our homes in what so-called Smart Home System [2], where almost everything in a house can be automated and connected to the internet. An example of where IoT can be valuable could be fire alarm systems, CCTV cameras, etc.

- iii- Wearables: people can also wear devices such as watches, rings, etc., that can monitor their health. These include heartbeat sensors, sleep trackers, or oximeters, and all these devices help people look closer at their health daily.
- iv- Connected vehicles: the driving industry also took part in integrating IoT in the production of vehicles which allows it to be fully automated with lots of capabilities such as platooning, merging, lane changing, and even self-parking [5]. Furthermore, it is expected that the production of autonomous vehicles will increase drastically in the coming years, and the major brands in the automobile industry are revolutionizing the sector accordingly [3]. In addition, autonomous vehicles require many sensors where their data can be given to an AI system that analyzes and learns to provide an optimized decision in different scenarios. Vehicles can also be connected to communicate and exchange data, meaning these IoT devices will be connected to RSUs or roadside units. RSUs store traffic data and can act as a cache system for connected vehicles. Also, RSUs can optimize the retrieval time of the requested content by these vehicles using modern caching algorithms with the integration of machine learning or artificial intelligence, resulting in a high cache hit ratio and reducing retrieval time.

1.1.2 Cloud Computing

Cloud computing is introduced to the world to facilitate the use of computer systems and data management, merging them into a one-centric system. Eventually, a user can store his/her data, access programs, develop applications, etc., over the web instead of his/her local computer [6].

In addition to that, to understand more about cloud services, the following represent four distinct service models that the cloud provides:

- i- Software as a Service (SaaS): this service allows the user to access software in the cloud with extreme ease as if it were on his/her computer such as using Google sheets [7]. Some of the benefits of this service might be an increase in scalability, an unelevated concern for infrastructure since it is on the internet (from the user's standpoint), and the accessibility of its services from anywhere and anytime [1].
- ii- Platform as a Service (PaaS): this is a more complex service since on SaaS, the user can access already integrated software such as Google sheets, whereas PaaS allows the user to develop his/her cloud services by offering a development platform functioning as Google AppEngine, Microsoft Azure, etc. [6][7]. This type of service reduces the development cost for companies since they are investing less in infrastructure because it is all managed by the cloud provider. In extension, PaaS provides a simplified deployment so companies can focus more on the development process rather than on upgrading their infrastructure which the cloud providers already maintain [1].
- iii- Infrastructure as a Service (IaaS): clients can use this service as a substitute for purchasing servers or acquiring more space to place these servers, data centers, etc., which is all provided by the cloud such as Amazon EC2 [7]. IaaS offers

more flexibility to users by reducing their costs on hardware, land, maintenance, etc. [1].

- iv- Container as a Service (CaaS): this service is based on virtualization. It is introduced to tackle development issues presented in PaaS, which means that CaaS eliminates the development boundaries set by the PaaS environment [7].

Furthermore, the cloud also offers deployment services that divide the cloud as follows [1]:

- i- Public Cloud: these types of clouds are readily available for thousands of people and offered by several cloud providers such as Google, Amazon, Microsoft, etc.
- ii- Private Cloud: this type of cloud is restricted to a specific organization or a business and managed by this organization. A private cloud maximizes resource utilization, achieves high security, and gives organizations complete control over their data [7].
- iii- Community Cloud: this type of cloud offers the same infrastructure and services for organizations with the same interests.
- iv- Hybrid Cloud: this type of cloud is used most of the time by organizations to allow them to have a combination of the previous two types. Hybrid clouds enable organizations to optimize their resources and use the services of both entities [7].

To this day, several cloud service providers (CSPs) offer all kinds of the previously mentioned services to different types of people and organizations. Most of these CSPs are well known such as:

- i- Google: this provider offers a wide variety of services, from data management and databases to online apps such as Google docs/sheets/slides [1], all of which facilitate a user's needs at the present time.
- ii- Amazon Cloud Services: an example of this provider's services is Amazon Prime which is a video streaming platform [1]. Amazon cloud services also offer an infrastructure for deploying and managing websites, apps, databases, storage spaces, etc.
- iii- Microsoft Azure: similar to what Amazon Cloud Services provide, Azure offers a set of services and infrastructure for data management.
- iv- Apple iCloud: this is mainly used for online storage, which means uploading images, videos, and data backups [1].

1.1.3 Fog Computing

Fog computing is introduced to extend the capabilities of the cloud, and it is placed at the edge of the network, making it available and physically near the end users [8][9].

This fog layer thus provides the services that the cloud has, but at a reduced scale to deploy more fog nodes that can serve more users considering the quality of service (QoS), availability, connectivity, reliability, etc.

There are several reasons behind introducing the fog layer, which also serves as the difference between cloud computing and fog computing. An intuitive example might be latency-sensitive applications such as video streaming. As mentioned earlier, fog nodes are placed at a close distance from the end user, making QoS for such applications highly satisfactory [10]. Also, in terms of hardware and power utilization, fog nodes require less hardware and can provide services that require low power consumption, whereas the cloud can be used for large data centers, for instance [10].

Since it is in close vicinity to the end user, fog nodes can be integrated with mobile technologies to form a RAN or radio access network [9]. A fog-RAN can be used for data retrieval at the edge of networks, which ensures a faster retrieval rate and low power consumption in the process.

Furthermore, many applications for fog computing can be witnessed on a day-to-day basis such as:

- i- Autonomous vehicles: fog computing can be used as a medium of communication between autonomous or self-driving vehicles that require data from their surroundings and in a short amount of time to perform on a high level [8][9][10].
- ii- Augmented reality: AR services need real-time video processing, which in turn requires high computational power leading to the introduction of fog computing to support AR systems to maximize the user experience [9][11].
- iii- Health data: since health data are considered essential and should not be altered, fog computing offers a secure space that allows the user to upload or update his/her records and share them with authorized people such as a doctor or test lab operator [9].
- iv- Content delivery and caching: fog computing offers high QoS, which means the content retrieval rate is dynamically optimized [11] to meet the end user's needs.
- v- Detection algorithms: fog computing enhances the ability to design and integrate applications that can aid in day-to-day routines with increased security. For instance, face detection algorithms can run on fog nodes instead of locally on the client side. The original photo will be compared with the one

provided by the user and then allows the user's device to be unlocked instantly, given the low latency nature given by the fog [9].

Along with its many advantages, fog computing has several challenges. The following are some of the challenges that face fog computing and can be considered a future research area:

- i- Mobile fog computing: according to [10], fog computing is considered fixed not mobile, whereas mobility is applied for its connected end devices. Although mobilizing fog nodes can be challenging, it could enhance its services by achieving improved QoS, reduced costs, and optimized energy consumption. The dynamicity of this mobilization can allow the fog nodes to form new networks [10].
- ii- Green fog computing: the authors in [10] argued that energy consumption relating to fog computing has yet to be fully explored. In addition to that, energy harvesters can be implemented on IoT devices generally and fog nodes specifically to generate their power. This can be a new challenge that can benefit and enhance the fog computing paradigm in the future.
- iii- Security and privacy: providing security and privacy has always been challenging in this era. To mitigate attacks on fog nodes, the addition of intrusion detection systems along with authentication and access control can significantly benefit fog nodes' security [10]. But this does not eliminate the need to explore more ways to strengthen the security of the fog layer. Privacy is always a client's concern, and the authors in [11] acknowledged this risk and proposed the need for privacy-preserving algorithms on the links between the cloud and the fog, and between the fog and the end user to prevent data leakage.

- iv- Fog node site selection: the authors in [10] argued that site selection for deploying fog nodes can be an exciting area of research, taking into consideration several factors such as storage space, communications, power consumption, and computing.

1.1.4 Caching

Caching is used to store data for future requests. In other words, if a user requests a webpage from a particular website, for example, it will be stored in his/her cache. Whenever the user re-requests the same webpage, it can be fetched from the cache and returned directly in case the webpage is not updated. However, if the webpage is updated, it has to be requested from its originated server, then the updated webpage will be stored in the user's cache.

In addition to that, the authors in [12] compiled a list of caching algorithms that have been traditionally used over the years, for instance:

- i- Least Recently Used (LRU): in this algorithm, the content that was requested the least in a predetermined set time interval is evicted to make space for another content since the cache memory is generally finite. This, in turn, might increase the cache hit rate, which also increases the users' QoS.
- ii- LRU-Threshold: it has the same algorithm as LRU, but the difference is that the algorithm predefines a threshold to the content size in the cache memory where the content should not be exceeded for it to be stored.
- iii- Least Frequently Used (LFU): this algorithm checks the frequency of the requested content, and if it is least frequently requested, it should be removed. However, the difference between this algorithm and the LRU is that if a

content, for example, has a high request frequency but did not get requested in the time interval that is set by the LRU algorithm, it cannot be discarded.

- iv- Lowest Latency First (LLF): to minimize the request latency, this algorithm discards the content whose download latency is the lowest.

Given the above, caching algorithms are being updated more often to satisfy the requirements that are set by today's technological advancement. Caching algorithms are being implemented in fog nodes as well as in the cloud to enhance the QoS for the users. Several research aims in the direction of integrating machine learning and artificial intelligence in the implementation and enhancement of caching algorithms.

1.2 Problem Statement

With the growing need for better QoS and requirements to achieve such a high benchmark, the field of connected vehicles requires low response time and high content retrieval rate to achieve optimal communications between the vehicles and their corresponding RSUs (i.e., Road Side Units).

Consequently, the traditional caching algorithms can be limited and adaptive to what is required to achieve this better QoS in general. For instance, the authors in [13] suggested a simple update on the LCE or Leave Copy Everywhere caching algorithm by making RSUs send content to every connected vehicle according to the probability of this vehicle being near the RSU.

Based on the above, this thesis will suggest a model by applying a cooperative federated deep reinforcement learning algorithm [14] for content retrieval. The algorithm will be composed of two parts: the fog layer will use a Q-network algorithm to achieve optimal cache management, and the cloud will run HFL or Horizontal Federated Learning algorithm that updates the global model according to the fog nodes' local models.

Moreover, this thesis will be comparing the suggested model with what is available in the literature [13][14] concerning connected vehicles on different metrics, such as cache hit/miss ratio, response time, etc.

1.3 Thesis Contribution

The power of fog computing cannot be unhinged, and it can offer limitless advantages just by being on the edge of the network, in close vicinity to the end-users. Applying a caching policy algorithm would increase QoS for the end-users and provide a global model that can be referred to by different fog nodes.

Having said that, the contribution of the thesis will be:

- Adding a fog layer between RSUs and the cloud.
- Applying a cooperative caching algorithm at the fog level to achieve a high cache hit ratio accompanied by a low response time, which generally improves QoS.
- Offering a comparative study between existing work and the proposed model in the thesis.

1.4 Thesis Organization

The thesis will proceed as follows: Chapter 2 will discuss the literature review that is related to the topic. Then, Chapter 3 will suggest the system model for a cooperative caching policy in applied in fog nodes for connected vehicles. In Chapter 4, experimentation results will be shown and analyzed. Ultimately, Chapter 5 will include the conclusion of the thesis and suggest future work.

CHAPTER TWO

RELATED WORK

In this chapter, the thesis will tackle relevant studies that focus firstly on the challenges of vehicular networks and their proposed solutions—secondly on implementing state-of-the-art caching strategies where IoT devices, generally, and connected vehicles specifically, where the connected vehicles are communicating with the server running the caching strategies.

2.1 Vehicular networks

This section will dive more into the challenges and possible solutions for vehicular networks. Vehicular networks are complex, thus, making their requirements different such as reduced delay for communication, stable connection with an RSU (Road Side Unit) or a BS (Base Station), high security and increased privacy due to the volume of vehicles involved.

To tackle the challenges that face vehicular networks, different solutions were presented. One of these solutions is AI generally and Machine Learning (ML) specifically. But first, we will discuss some of the challenges that are presented in [15]:

- i- Resource allocation: in order to deploy a wireless network for vehicular networks, several aspects are to be considered first, such as quick response, precise environment modeling, and self-adaptivity. But the high mobility of vehicles makes these aspects challenging to achieve.
- ii- Network traffic control: this includes routing, congestion avoidance, and offloading. And in vehicular networks, the challenges for these aspects can be future state prediction and correlation recovery, where gathering global

information of the whole network is hard in real time. Also, there is proactive exploration and self-learning due to the interference of human touch in order to update strategies.

- iii- Network security: security has always been a challenge in the network's paradigm, which also extended this challenge to vehicular networks. Vehicular networks' challenges have different branches such as mobility and heterogenous structure, where several aspects (e.g., computational power, storage, etc.) need to have a balanced resource allocation to increase security in case of breaches. These networks should also be large-scale, provide privacy, and meet real-time requirements.

However, machine learning has been an integral part and a step forward in solving most of the challenges mentioned before, targeting the following challenges respectively as mentioned in [15]:

- i- Resource allocation: ML aided by introducing a reinforcement learning model. This model helped by focusing on multiple criteria in order to optimize resource allocation instead of focusing on a single criterion. Thus, allowing the model to focus on traffic distributions, whether homogeneous or inhomogeneous, time-varying traffic patterns, and channel failures.
- ii- Traffic control: Different ML-based solutions for network traffic control such as predicting network traffic control and information recovering from incomplete data. In other words, vehicles' connections are always susceptible to being broken due to the high mobility of vehicles. As a result, machine learning allows us to predict vehicles' locations, which helps predict network traffic and minimize broken connections. Moreover, by implementing a deep

learning approach, data can be retrieved from disrupted connections by recovering the routing information from previous network traffic.

- iii- Network Security: ML-based approaches can also be integrated to address security issues that vehicular networks face. Some techniques include misuse detection; misuse attacks are classical attacks where an attacker detects previously known attacks using their signatures and exploits them. Another approach is anomaly and hybrid detection algorithms; the types of attacks that fall under the governance of anomaly detection algorithms do not have records similar to misuse-based attacks. Thus, anomaly detection algorithms identify benchmark network traffic, and if anything is out of the ordinary, it gets recognized as an anomaly.

Moreover, the introduction of federated learning can tackle the security and privacy issues that vehicular networks pose. The authors in [16] suggested a new vehicular network concept called Federated Vehicular Network (FVN), leading to a more stable connection that enables the integration of federated learning. And by allowing the integration of FL, model training will occur at the node level without making the data known to the whole network, thus increasing privacy. And to improve security, each transaction will be recorded and validated by the blockchain model.

2.2 Caching

This section will tackle the studies that focus on optimizing caching strategies used in different paradigms such as fog computing, edge computing, cloud, etc. Moreover, some of these studies are also conducted aiming to enhance vehicular networks.

An efficient in-network caching scheme for connected vehicles-based networks is suggested by the authors of [13]. This proposed strategy consists of a forwarding scheme and

a cache replacement scheme. In other words, the authors aimed to optimize cache utilization while considering the mobility of vehicles. The forwarding scheme proposed is Enhanced-LCE (Leave Copy Everywhere) which is based on the traditional LCE. This forwarding scheme allows a specific RSU to broadcast messages to every connected vehicle. The enhanced version of LCE will depend on the vehicle's direction on the highway and to be near an RSU that will be broadcasting the message. TBE (Time Based Eviction) is implemented as a cache replacement strategy where each content available in the cache of an RSU will have a time limit. When the time limit is exceeded, the content stored will be evicted, allowing other content to replace it. Simulation results for the forwarding scheme and the cache replacement scheme showed an increased cache hit ratio and reduced cache redundancy, as shown in Fig. 1 and 2, respectively.

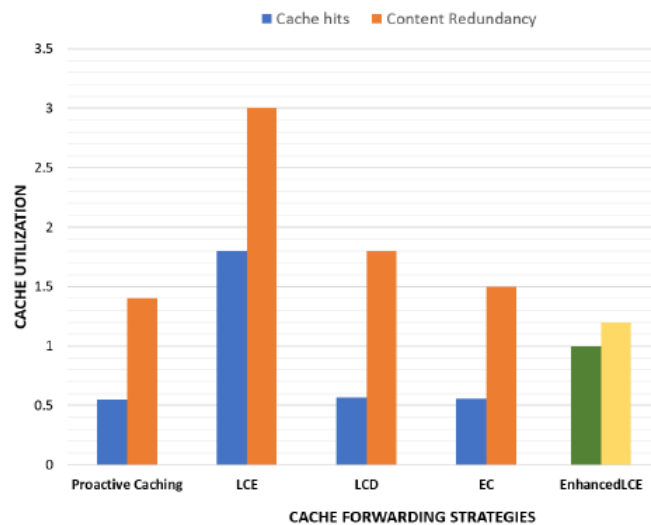


Figure 1- Forwarding scheme results [13].

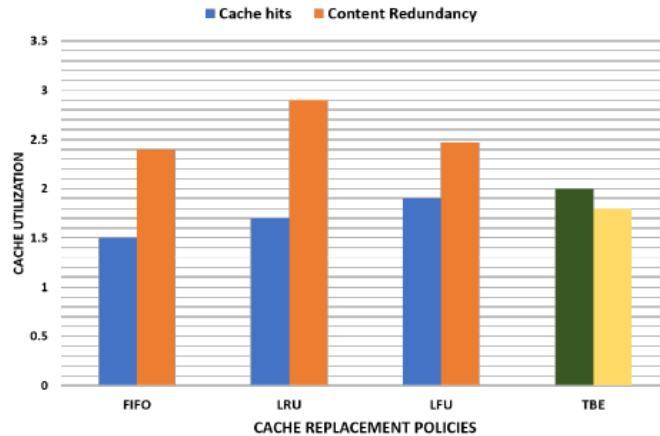


Figure 2- Cache replacement results [13].

The authors in [14] proposed a caching strategy that is different from the conventional algorithms such as FIFO (First In First Out), LRU (Least Recently Used), LFU (Least Frequently Used), etc. The proposed strategy is a cooperative FDRL (Federated Deep Reinforcement Learning) algorithm combining reinforcement and federated learning. Reinforcement learning is part of machine learning, which is based on a reward system, where a learning agent gets a reward for choosing an action depending on the state of the environment it's running on, which eventually alters this environment [17]. This reward can be of positive or negative value, allowing the agent to learn from the state of the environment and the reward depending on its actions. On the other hand, federated learning incorporates several IoT devices that are running a machine learning model, which is being trained on different datasets from each other. However, these devices collaborate with a central server that aggregates the models from each connected device and re-distributes a new global model to be used by all devices [18].

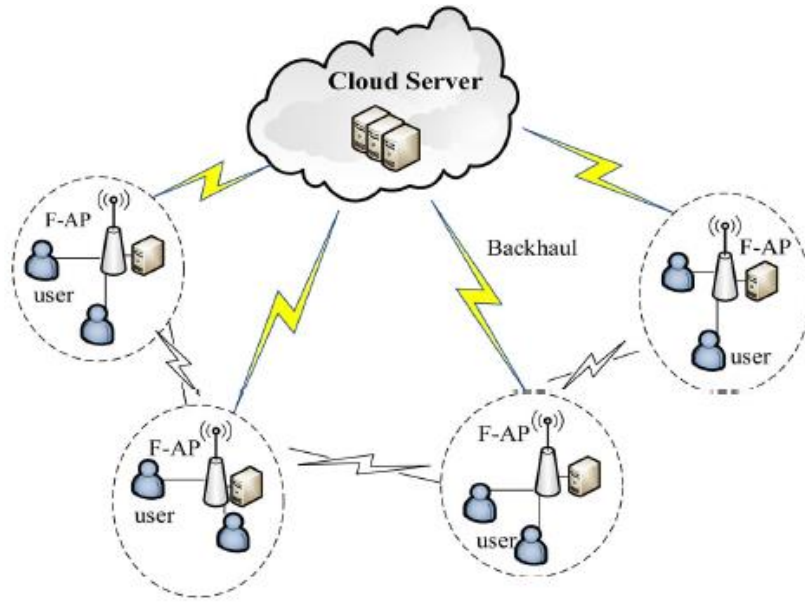


Figure 3- Model illustration [14].

The proposed FDRL caching strategy works as shown in Fig. 3; each F-AP (i.e., Fog Access Point) runs a DDQN or Deep Dueling Q-Network, a reinforcement learning algorithm. Each F-AP agent learns according to the data provided by its environment. When a user requests content, which in turn can be returned in three ways [14]:

- i- F-AP returns from its cache.
- ii- F-AP asks neighboring F-APs for the requested content.
- iii- F-AP asks the cloud server for the requested content.

The learning agent has three actions to choose from in order to maximize the reward, which is based on content availability and request delay. In case the F-AP elects to request the content from the server, the cloud server then aggregates all the connected F-APs models using HFL or Horizontal Federated Learning into one global model and sends it back to the F-APs to use [14].

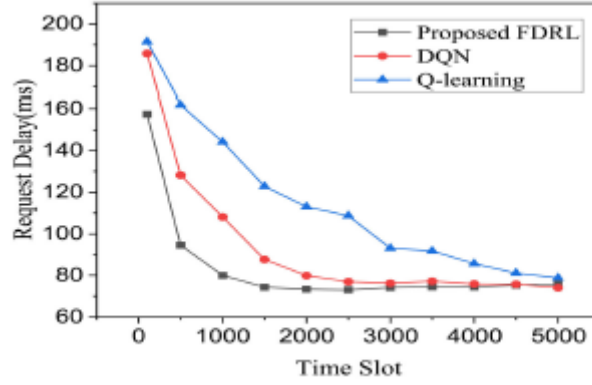


Figure 4- Cache hit rate as a function of time [14].

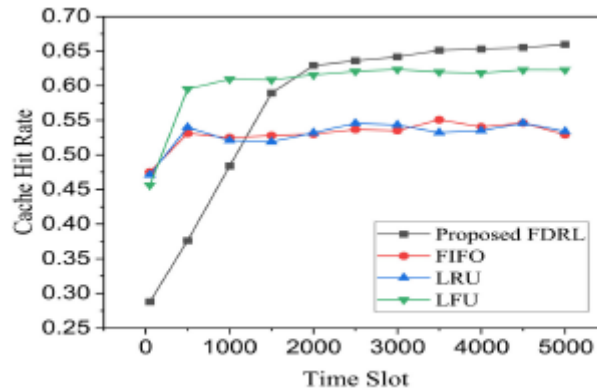


Figure 5- Request delay as a function of time [14].

As a result, the proposed FDRL by the authors of [14] maximizes cache hit rate and minimizes request delay compared to traditional caching algorithms, as shown in Fig. 4 and 5.

Another caching approach proposed by the authors in [19] included a cooperation scheme between federated learning and a machine learning model to increase privacy for the end users. The proposed scheme is called FLCH or Federated Learning-based Cooperative Hierarchical caching scheme. The scheme incorporates both horizontal and vertical federated learning. According to [19], the horizontal federated learning part H-FLCH is applied between neighboring F-APs, whereas the vertical part V-FLCH occurs between F-APs and BBU pool (i.e., Base Band Unit), which is used to connect the F-APs to the cloud layer as shown in Fig. 6.

Furthermore, the authors in [19] applied OCC-VAE (i.e., One-Class Collaborative Variational Autoencoder) prediction model as a machine learning model. An AE or Autoencoder is an algorithm that takes an input and tries to reconstruct this input data to output data.

The proposed model displayed in Fig. 6 will be applied to each connected device and trained on its private data, thus the concept of federated learning. After each model trains on separate data, the models are then horizontally aggregated on the F-AP level using H-FLCH to generate a new global model. When the results of the model training become stable, vertical aggregation of the data (V-

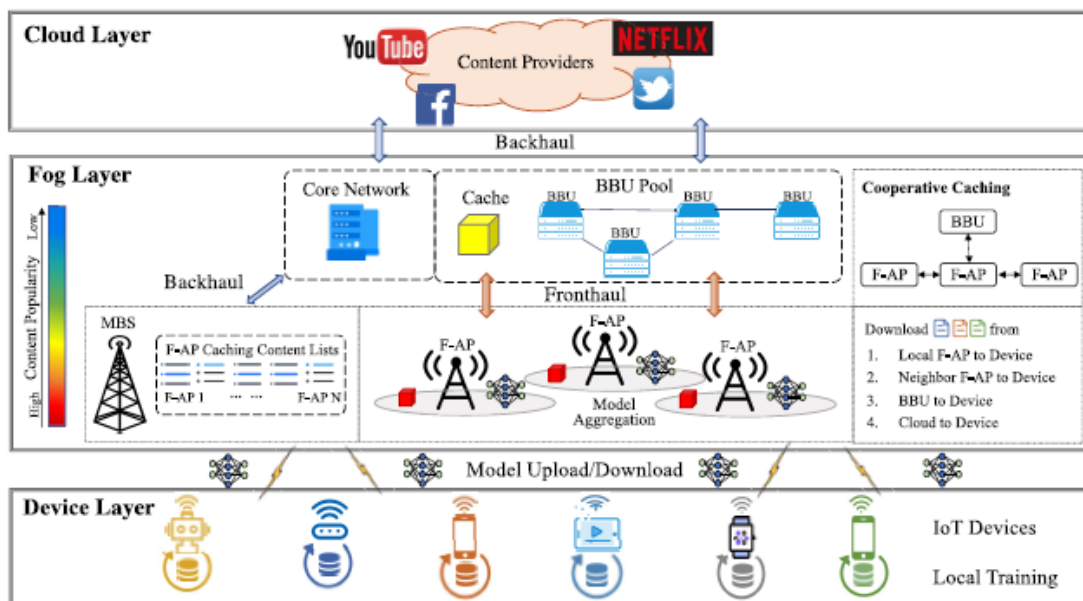


Figure 6- Model Illustration [19].

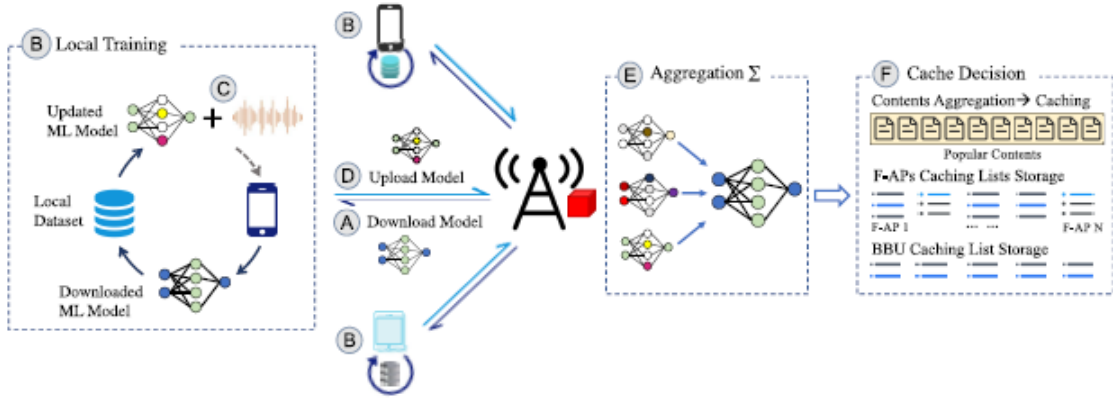


Figure 7- FLCH scheme flow [19].

FLCH) can be executed [19] as shown in Fig. 7. The content that is now cached on the cloud layer using V-FLCH is considered the popular requested content from each F-AP.

The results provided by the study show that the proposed scheme performed better than the conventional caching algorithms such as LRU, LFU, Random, etc.

In addition to that, the authors in [20] proposed a CAEC or Capacity Aware Edge Caching framework in order to achieve low ADT (i.e., Average Download Time) from the fog layer. The model provided by the authors is an ADMM-based algorithm or Alternating Direction Method Multipliers. The purpose of this algorithm is to generate an optimized global solution that aids in determining the best CAEC strategy while taking into consideration fog cache capacity and last-mile connectivity capacities of BSs (Base Stations) [20]. These BSs will be placed as an intermediary between the users and the fog nodes, acting as a relay.

Due to the applicability of the proposed algorithm for convex optimization problems in distributed deployment scenarios [20], the authors then used this algorithm to converge faster. Moreover, the proposed algorithm is compared with two other methods; the first method is Interior Point Method or IPM, and the second one is the Subgradient method. These methods (IPM and Subgradient method) specialize in solving convex optimization problems.

The results showed that the proposed algorithm converges faster compared to IPM and Subgradient method, leading to a lower ADT. However, after several iterations, all three methods reach a similar time.

Furthermore, other studies explored the area of enhancing fog-RANs or Random-Access-Networks by integrating AI [21]. This integration included a traffic prediction system alongside a cognitive caching algorithm to improve QoE and request delay time. According to the authors in [21], the caching policy is based on LSTM (i.e., Long Short-Term Memory) and a collaborative filtering scheme. LSTM is a branch of recurrent neural networks but does not suffer from problems in optimization that face SRNs or Simple Recurrent Networks [22]. The idea behind this concept is a memory that stores its state and can regulate the input flow in and out of the memory state. LSTM has been used in many advanced problems such as language modeling, protein secondary structure prediction, audio/video analysis, etc. [22]. On the other hand, the collaborative filtering model used first in [23] is part of 3 models combined in the Smart-Edge-CoCaCo algorithm that aims to enhance communications, cache hit ratio, and computational offloading. The collaborative filtering model considers features from both the user and the content, leading to more accurate and systematic caching [23].

Thus, the authors in [21] combined LSTM and collaborative caching by making LSTM predict the requested content type. In other words, it sorts out the content types first, and then the collaborative filtering algorithm can match each content with the requesting user, as shown in Fig. 8.

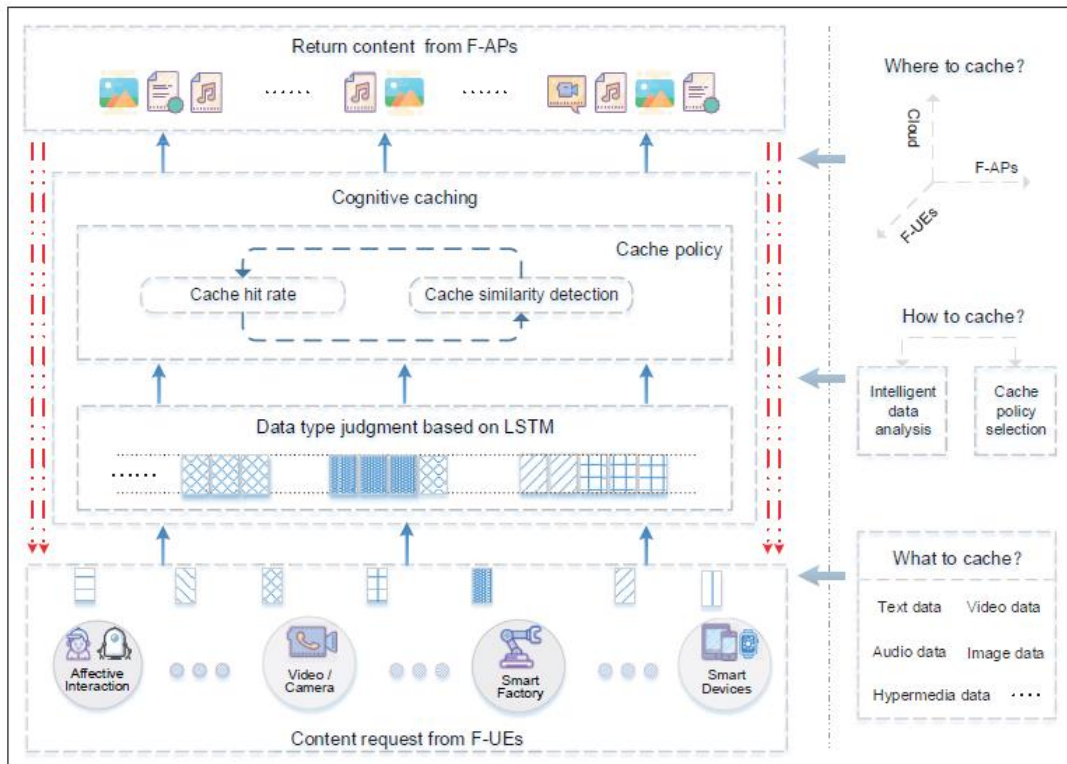


Figure 8- LSTM and collaborative caching model flow [21].

In [24], the authors explored machine learning schemes that can be applied to caching to enhance the cache hit ratio and decrease request delay. These schemes benefited the entity running the caching strategy in different ways depending on the features of each scheme. For example, classification algorithms are used for predicting traffic levels and content popularity, clustering is used to identify patterns in data and sort them out accordingly, and reinforcement learning's reward feedback feature allows the agent to learn with each iteration and choose actions that are rewarding according to the environment such as Q-learning which what the proposed algorithm in this thesis is based upon, also there is deep learning which aids in content delivery while reducing the complexity of the operation due to its nature in extracting features in the data provided.

New research has explored the integration of machine learning schemes for caching. The authors in [25] proposed a machine-learning scheme for one-time image caching based on decision trees. In other words, the authors wanted to implement a one-time-access-

exclusion policy, meaning that if an image is not to be accessed for a specific amount of time, it should not be cached [25]. Since the authors wanted to have a balance between high precision and low complexity algorithms, they went with the decision trees model. Compared to their initial results without the proposed classifier, the proposed model showed a slight enhancement in cache hit rate in algorithms such as LRU, FIFO, S3LRU, etc. [25].

Furthermore, in vehicular networks, the mobility of vehicles poses a problem with data transmission since these vehicles are connecting and disconnecting leading to data retransmission which affects the user's QoE. Thus, the authors in [26] proposed a cluster-based cooperative caching approach with mobility prediction (COMP). The clustering factors include the vehicle's mobility, the distance between a specific vehicle and its neighbors, and the total number of vehicles [26]. Each cluster of vehicles will be connected to a caching node, where this node, which in this study is an RSU, runs the cache placement according to Most Popular Data (MPD) and Least Popular Data (LPD). COMP will allow MPD to be stored in different nodes due to high demand, while LPD will be stored and distributed among different RSUs. The experimental results showed lower request delay and higher cache hit ratio.

More research focused on caching related to multimedia traffic dominated by video streaming, for example. For this reason, the researchers in [27] proposed D2D caching using reinforcement learning. Each agent can learn independently from its data or jointly from its data alongside the learning of other devices. Moreover, the RL model that is being used is the Multi-Armed Bandit problem (MAB), which is a classical problem in probability theory and machine learning where an agent has several arms, each with an unknown distribution of rewards and an unknown mean, the agent explores accordingly in order to maximize the reward system [27]. Both proposed approaches JAL (Joint Action Learners) and IL (Independent learners), are tested against classical algorithms IUB (Informed Upper Bound),

LFU, and LRU. IUB was superior in cache hit rate and average download latency than all four algorithms, but the proposed approaches performed better than LFU and LRU. However, it should be noted that JAL was better than IL because its agent can learn more from global values than local ones.

Additionally, the authors in [28] proposed a content caching scheme based on permissioned blockchain and deep reinforcement learning. Because vehicular edge computing requires the data to be cached in proximity to vehicles, the integration of permissioned blockchain helps secure the overall communications in the vehicular network. In other words, if a vehicle can't cache locally, it can request a caching resource from a neighboring vehicle [28]. However, this transaction should be secure, thus blockchain technology, where the content would be encrypted using a public key of the owner and can't be decrypted unless with the private key of the same owner, which leads to an increase in privacy and security. Additionally, each vehicle will send the caching request to the nearest BS, which will manage the caching placement. And to record the caching event, a transaction should be issued, and the BS verifies it, encrypts it, and sends it to the whole blockchain network to be validated [28]. But the challenge of vehicle mobility would still be solved with DRL. The DRL algorithm adopted in this study is Deep Deterministic Policy Gradient (DDPG). The main difference compared to other classical DRL algorithms such as Q-learning and Deep Q-Network is that they can learn policies in high-dimensional observational and action spaces [28]. Simulation results showed that the proposed DRL algorithm had more average reward when compared to 2 benchmark algorithms; the first one is Greedy Content Caching, where a cache requester delivers its content to the cache provider having a high wireless connection, while the second algorithm is Random Content Caching where a cache provider is chosen randomly but taking into consideration that the distance between the requester and provider should not pass a certain threshold [28]. In addition, the proposed DRL showed superiority in

other comparison criteria, such as the percentage of successful content caching, with a rate nearing 90%.

Similar to the study in [14], the authors in [29] tackled the issue of highly dynamic environments posed by IoT services and applications. This led to proposing a federated deep-reinforcement-learning-based cooperative edge caching (FADE) to optimize edge caching of such environments. The framework aims to reduce performance loss, improve hit rate, and decrease average delay time. The model, as shown in Fig. 9, consists of three layers; (1) IoT devices, (2) edge layer, and (3) cloud layer.

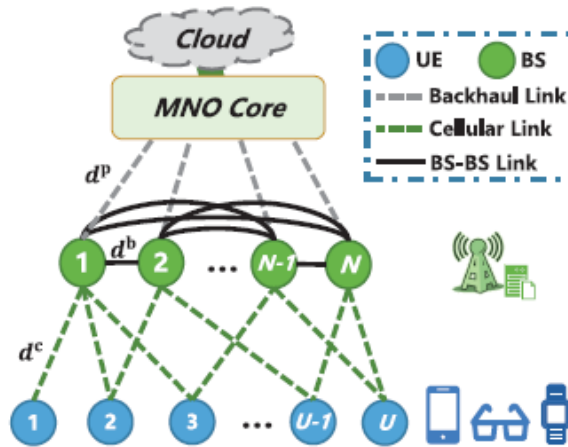


Figure 9- FADE System model [29].

Double Deep Q-Network (DDQN) is adopted as the reinforcement learning algorithm to address the issue of huge data sampling. Furthermore, the algorithm has three actions to perform and each with a specific reward:

- i- Local action: the requested content can be processed locally.
- ii- Cooperative action: if the requested content is not cached in the local BS of the requesting UE, then the request should be redirected to a neighboring BS.
- iii- Remote action: if the requested content is unavailable locally or in a neighboring BS, the request should be redirected to the cloud.

Moreover, in each iteration, the algorithm's weights of each BS should be aggregated to get a more optimal global model. The global model is then sent to every BS [29].

The proposed decentralized framework is then compared with several benchmark algorithms such as FIFO, LRU, LFU, Oracle, and Centralized, on hit rate and average delay. As shown in Fig. 10, the proposed framework showed better results when compared to classical methods.

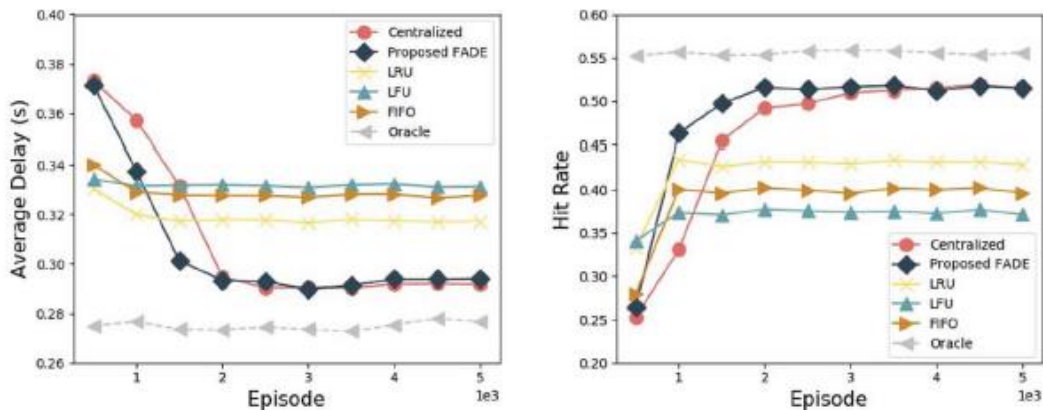


Figure 10- Simulation results for FADE [29].

Another machine learning model proposed by [30] for content caching in vehicular networks is based on federated learning. The proposed model is called Mobility-aware Proactive Edge Caching (MPCF) and is adopted to tackle the known issues that caching in vehicular networks faces. As discussed previously in other studies, these challenges orbit around vehicles' mobility and the content being quickly out-of-date due to their high mobility [30]. The MPCF scheme has five stages:

- i- Vehicle selection: to perform the FL training, the vehicle to be selected depends on it being available in the coverage area of the RSU, having enough training data, and having a good connection.
- ii- Model download: if the vehicle is selected, they should download the global model from the RSU to train their local model.

- iii- Model training: after the download, the vehicle will use the global model to train its local model with the available data it possesses.
- iv- Upload local model: once the training is finished, the vehicle uploads its trained local model to the RSU.
- v- Model weight aggregation: the RSU gets the weights of all models from its connected vehicles and aggregates them to generate a new global model.

Furthermore, the FL model trained is called Contextual-Aware Adversarial Autoencoder (C-AAE) and is used to predict content popularity [30]. And for optimizing the caching strategy, the authors integrated a cache replacement method to replace the cached data for a disconnected vehicle from an RSU, as shown in Fig. 11.

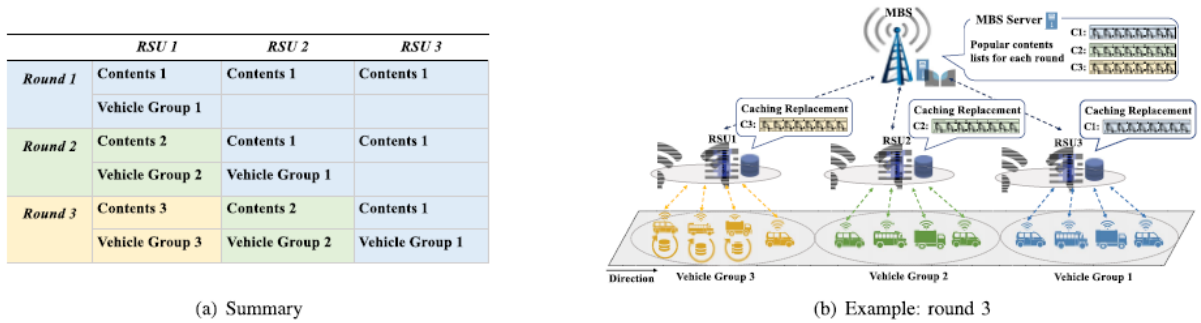


Figure 11- Example of cache replacement flow adopted in [30].

Different benchmark algorithms such as LRU, LFU, Oracle, etc., are used to view the viability of the proposed framework on cache hit rate. Only the Oracle algorithm performed better than MPCF, proving that the authors' approach is better than the different existing methods for content-caching.

In addition, a federated learning scheme using a blockchain-assisted compression algorithm for content caching in edge computing called CREAT is proposed by the authors of [31] to address security and privacy problems posed by data transmission. According to [31], the blockchain's model uses four contracts, each required whenever a transaction is made:

- i- Identity authentication: this contract's responsibility is to validate each entity, whether it is an IoT device or an edge node.
- ii- Submission contract: this contract allows the edge nodes to submit their training model's weights and save them in the blockchain.
- iii- Verification contract: this contract is responsible for electing supervisory consortiums and transaction verification.
- iv- Credit contract: this contract takes care of the networks' participants' token reward and punishment.

Moreover, the FL-based caching algorithm is an autoencoder known as FPCC [31]. This autoencoder will be used to determine the features of the input it receives. The outputs of two of these autoencoders are used to determine the characteristics of the users and files. Then, each edge node will upload its own trained model to the cloud, satisfying the core principles of federated learning where data is not to be shared and training only happens locally. Since minimizing communications costs is a problem to be tackled in IoT, the authors in [31] implemented and integrated a compression algorithm that speeds up the model's training process. The compression algorithm considers gradients that affect the model heavily as necessary, and their values should stay the same compared to gradients with less importance.

Furthermore, the essential gradients are then clustered using K-means clustering, and each cluster is given a centroid value that is then used to estimate the exact values of the gradients. Simulation results showed that FPCC and CREAT have similar cache efficiency and are higher than the two other algorithms that are being compared. The first benchmark algorithm is the Random algorithm, where caching does not have any consideration to cache upon; content is cached randomly without considering content popularity. Whereas the second benchmark algorithm is called Thompson Sampling, which is based on beta distribution, and

whenever the cache file is updated, the beta distribution is updated accordingly. Furthermore, CREAT showed reduced update time compared to FPCC, which does not run the compression algorithm [31].

The researchers in [32] formulated a caching framework deployed in MBS (Macro Base Station). The framework is based on ICRP or Individual Content Request Probability estimation, which reflects the personal preference of the requesting user. Moreover, the authors in [32] designed a Bayesian learning model to accurately predict users' personal preferences. The proposed Bayesian learning model is based on CBPMF (Constraint Bayesian Probabilistic Matrix Factorization), which is based on BPMF. BPMF is derived from PMF, a model heavily used in collaborative filtering [33].

With the usage of CBPMF, the authors in [32] can now evaluate the ICRP estimation that will be used in a DCA (Deterministic Caching Algorithm). The DCA algorithm is a DLA or Discrete Learning Automata reinforcement learning model. DLA aims to optimize action selection from a set of finite actions, with each action having a specific reward [32]. In addition, the authors in [32] implemented a D2D (Device 2 Device) method that minimizes the request delay.

Simulation results showed that the proposed CBPMF, when compared with another two prediction methods (BPMF and MF-SGD) on RMSE Root Mean Square Error, both CBPMF and BPMF outperformed MF-SGD. However, the proposed prediction method scored better than the traditional BPMF. Furthermore, the proposed framework DCA-ICRP is compared with the following methods: DCA-GCRP, where GCRP stands for Global Content Request Probability, PC-ICRP (Probabilistic Caching-ICRP), PC-GCRP, and RC or Random Caching strategy on cache hit rate and outperforming them. Also, enabling D2D connections alongside DCA-BPMF lowered the request delay for the proposed caching method [32].

In the studies mentioned above, the researchers aimed to suggest strategies to enhance vehicular networks and improve caching algorithms. Vehicular networks have several requirements: low response time, continuous connection, high privacy, and security. Thus, the studies discussed in this section targeted most issues that vehicular networks suffer from. On the other hand, most of the caching strategies presented integrated Machine Learning to target several aspects that limit the traditional caching strategy (i.e., FIFO, LRU, LFU, etc.).

This thesis will continue addressing the issues that most of the discussed studies aimed at addressing. Our work will present a cooperative federated learning scheme that seeks to increase the cache hit rate and reduce response time in vehicular networks.

CHAPTER THREE

THE SUGGESTED SYSTEM MODEL

In this chapter, the thesis will dive into the model that is adopted in this work. As previously mentioned, the vehicular network is an emerging paradigm that is posing several challenges on different levels (i.e., communications, resource allocation, security, etc.). Moreover, several studies addressed the vehicular networks' challenges using various methods, generally, specifically machine learning. In addition, this thesis addresses caching strategies related to vehicular networks to enhance QoE by increasing the hit rate and reducing delay time. Furthermore, integrating machine learning strategies in caching allows the exploration of new methods to strengthen caching generally, compared to the classical techniques (i.e., FIFO, LRU, LFU, etc.).

The integration of machine learning in this work is divided into two fronts: the first front is where the fog layer would run a deep reinforcement learning strategy in a federated architecture, and on the second front, the cloud layer would be aggregating the several models provided by different fog nodes to produce a new optimized global model. The fog nodes will use this new global model later, and these strategies will be discussed thoroughly in this chapter.

3.1 Model overview

The suggested model is divided into three layers, as shown in Fig. 12:

- i- RSU layer: the Road-Side Unit (RSU) layer is the direct communication link with the connected vehicles on the road. The RSU relays the request of the vehicles to the fog layer.
- ii- Fog layer: the fog layer receives the relayed request from the RSU and chooses an action according to the deep reinforcement learning algorithm used to handle vehicles' requests; DDQN (Dueling Deep Q-Network).
- iii- Cloud layer: the cloud layer is a central server for all fog nodes. The cloud has two main jobs:
 - a- Model aggregation: the suggested model will be using an aggregator according to HFL or Horizontal Federated learning in order to generate a new global model to be used by the fog nodes, thus, ensuring optimized decision-making algorithms for all of the nodes.
 - b- Cache: besides acting as a model aggregator for the fog nodes, the cloud also caches the requested content from connected vehicles. In other words, if the fog node chooses to retrieve the requested content from the cloud, then the cloud fulfills the request and caches the content if it is not available. Once the request is fulfilled, the cloud triggers model aggregation to globalize the DRL (Deep Reinforcement Learning) model and sends it back to the connected fog nodes.

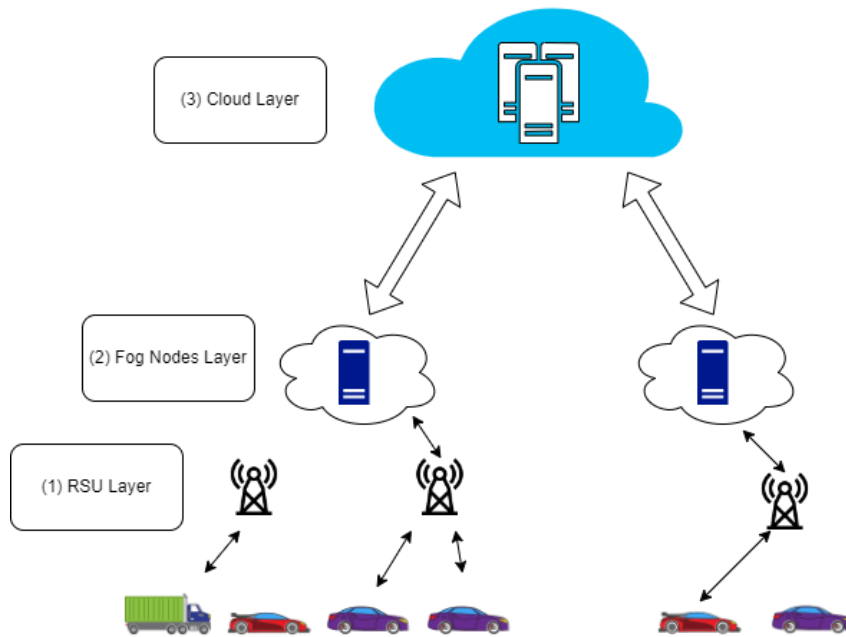


Figure 12- The Suggested Model.

In this work, we created a custom caching environment that the DDQN agent uses to learn according to the criteria we imposed on the model. Also, we added an RSU layer between the end-users, in this case, the connected vehicles and the fog layer to help better communicate with the fog nodes, thus decreasing the overall response time.

The pseudo-code below represents the workflow of the suggested model, as shown in Fig. 13. When a user requests a particular content, the agent will choose from what layer the request should be served. If the RSU should serve the request, thus, the RSU checks the cache to see if the requested content is available, and if the content is available, then the content is retrieved. Otherwise, the fog layer would be checked for the same content and retrieved if the content is available in the fog layer's cache. Otherwise, the cloud layer's cache should be checked. When checking the cloud for the content if the content is available or not, the aggregation of all fog nodes' models will occur and be sent back to train according to the new model.

```

for time slot  $t = 1, 2, \dots, T$  do
  for user  $u = 1, 2, \dots, U$  do
     $\rightarrow$   $u$  requests content  $f$ 
    if  $f$  cached in connected RSU
       $\rightarrow$  fetch  $f$  and send it to user  $u$ 
    else
      if  $f$  cached in the fog node that connects the requesting RSU
         $\rightarrow$  fetch  $f$  and send it back to the RSU
      else
         $\rightarrow$  fetch  $f$  from the cloud server and send it to the requesting fog
          node and in turn to RSU then to user  $u$ 
         $\rightarrow$  update fog node cache using Q-network
         $\rightarrow$  update cloud server using HFL
      end
    end
  end
end

```

Figure 13- Model workflow pseudo code.

In addition, all three layers act as a cache environment for the requested content from the vehicles, which means that some of the content stored will be evicted to create a space for new content. But there is a criterion to evict content; for example, if the content has frequent requests, meaning it's popular and stored at the end of the cache, it should not be evicted. As a result, the content with the least popularity will be evicted to create a space for new content.

3.2 Dueling Deep Q-Network (DDQN)

First, to understand how the DDQN works, we will need to look at the algorithm from which it is derived; DQN or Deep Q-Network. Secondly, we must understand how DDQN provides more optimization and accuracy compared to its parent version DQN.

DQN (top) and DDQN (bottom), as shown in Fig. 14, have the same convolutional neural networks, which are two. However, DQN has only one output stream for the Q-values, representing the actions in a specific environment. On the other hand, DDQN has two streams before having the output stream (Q-values). The first stream represents the current state of the environment which is a single value, and the second stream represents the advantage stream for each of the actions available in the provided environment. In the end, the Q-values are

calculated by the values of the state and the advantages stream according to the following formula:

$$q = v + a - a.mean()$$

The formula above represents the Q-values according to the state (v) and advantage of each action (a). The addition of those values is then deducted by the mean values of each action represented in the advantage stream.

However, the reason behind DDQN being more optimized and accurate than DQN is not created by only splitting the final stream into two streams. In other words, the two implementations provide the optimal Q-values at the end but using different methods. The real improvement can be seen in the training of the agents that are running each algorithm. In other words, introducing the state value in the DDQN approach allows the learning of the state value efficiently without a pre-update of the Q-values. Moreover, the advantage stream makes the neural network robust to action re-ordering. Thus, the values won't be biased with each training sample.

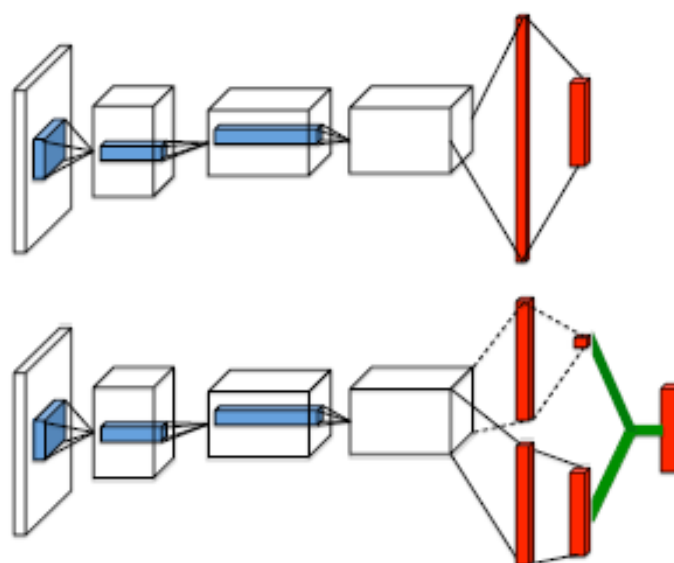


Figure 14- Architecture comparison between DQN (top) and DDQN (Bottom) as provided in [34].

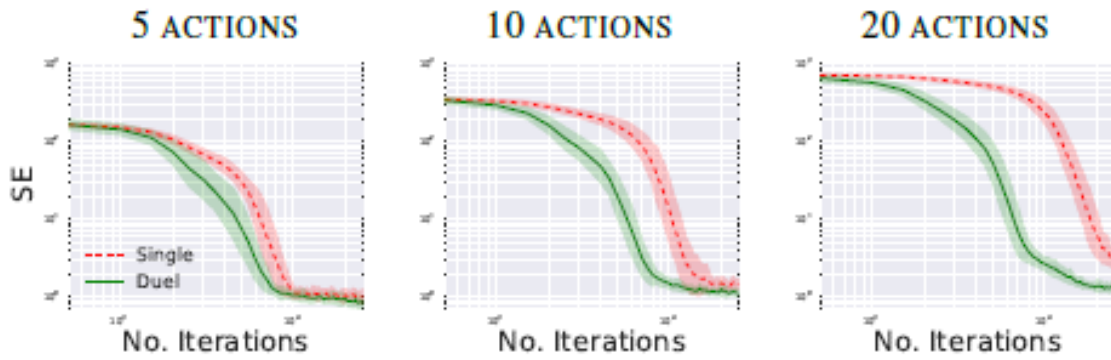


Figure 15- Experimental results DQN vs. DDQN in [34].

The comparison between DQN and DDQN, as represented in Fig. 15, showed that DDQN outperformed DQN in three instances (5, 10, and 20 actions) regarding Squared Error (SE), thus, proving the superiority of DDQN.

Furthermore, we must add a reward system allowing the agent to learn and choose actions accordingly for the model to work and learn. Thus, the system we implemented has three rewards, as represented in the following:

- i- RSU: the agent chooses to check the RSU for the requested content, and if it is available in the RSU's cache, then there will be a positive reward equaling 3; otherwise negative.
- ii- Fog node: the agent chooses to retrieve the content from the fog node; if it is available, a positive reward equals 2; otherwise negative.
- iii- Cloud: the agent chooses to retrieve the requested content from the cloud layer, and the reward is equal to 1.

However, the three types of rewards indeed have the same logic, but the value of the reward is different. In other words, there are several aspects to consider while choosing to retrieve the content from the RSU can be highly rewarding since the delay would be minimal due to the close distance between the RSU and the connected vehicles. Still, the RSU's cache size is finite and can't store too much content. As a result, this reward system enables the

agent to learn what action to choose to maximize the cache hit rate and reduce the request delay to enhance QoE for the end-users. As a result, the request delay between the user and the cloud, fog node, or RSU would be set to 10ms, 20ms, and 200ms, respectively.

3.3 Horizontal Federated Learning (HFL)

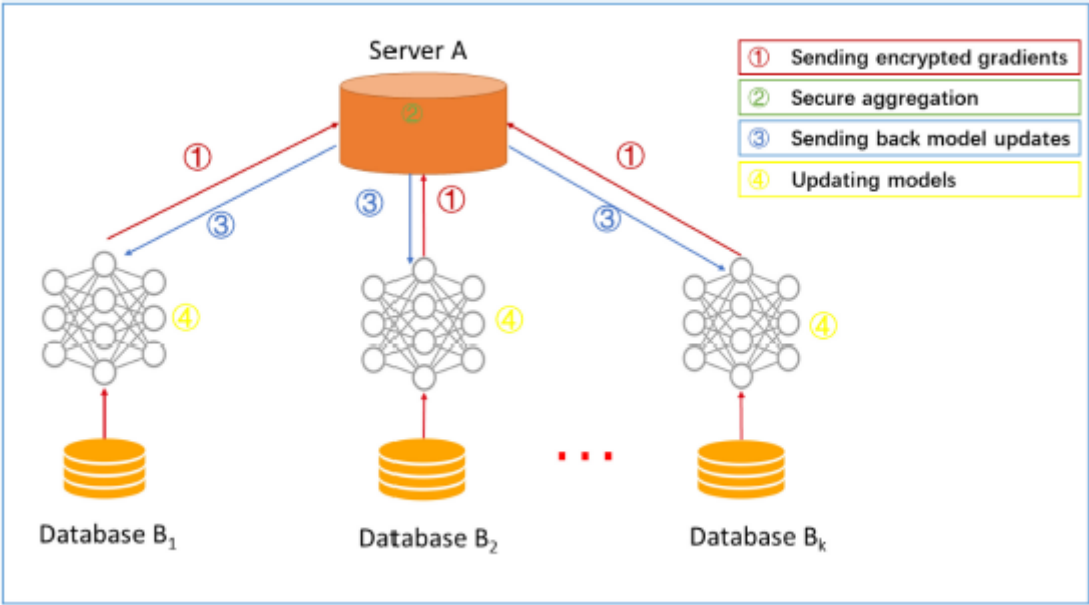


Figure 16- HFL architecture as represented in [35].

The introduction of federated learning enabled IoT networks to be more secure and private. Thus, we used this architecture in our model for the mentioned advantages that FL introduces to increase security and privacy. In order to understand how FL works, Fig. 16 shows the steps as follows:

- i- Step 1: each fog node sends the encrypted gradients for the ML algorithm working in each fog node; DDQN in our case, to the central server; the cloud.
- ii- Step 2: the cloud then aggregates the gradients received from the connected fog nodes.
- iii- Step 3: the cloud re-sends the new global model to the fog nodes.

- iv- Step 4: each fog node uses the new global model as the base model to learn using its data.

However, in our suggested model, the cloud aggregates the model in case the fog node only chooses to retrieve the data from the cloud.

As a result, the blend of these two methods, DDQN and HFL, will create a cooperative strategy that aids in enhancing QoE for end users (i.e., connected vehicles) by increasing the cache hit rate and reducing the response delay.

3.4 An Example

In this section, we will show how the suggested model would work. The example will have content to be requested, and we will show what would happen if the suggested algorithm chose all three actions adopted.

The cache stream and the popularity of each content cached for each layer in the suggested model at $t=0$ (i.e., cloud, fog node, and RSU) are represented in Tables 1, 2, and 3. However, popularity is defined by how much a certain content is requested, and the number of columns available represents the cache size.

Table 1- Cloud cache at $t=0$.

Contents	8	2	4	1	4	77	47		
Popularity	2	1	1	1	3	5	1		

Table 2- Fog node cache at $t=0$.

Contents	7	9	3	55	12	
Popularity	2	3	2	1	3	

Table 3- RSU cache at t=0.

Contents	11	99	67
Popularity	5	2	4

At t=1, content having a value of “47” is being requested, and the workflow will go as follows:

- i- Cloud: the algorithm chooses the cloud to serve this request. Since the requested content is already in the cloud’s cache, the reward is positive, and because the cloud has the longest response time, which is 200ms, the reward will equal 1. As a result, the popularity of the requested content will increase now.

Table 4- Cloud cache at t=1.

Contents	8	2	4	1	4	77	47		
Popularity	2	1	1	1	3	5	2		

- ii- Fog node: the action chosen by the DDQN agent makes the fog node serve the request. Since the content is unavailable in the cached stream, the reward will be negative, equaling -1. However, if the content were available in the cache, the reward would be 2 since the response time from the fog node equals 20ms, which is much lower than the cloud.

Table 5- Fog node cache at t=1.

Contents	7	9	3	55	12	47
Popularity	2	3	2	1	3	1

- iii- RSU: the action now is to choose from the RSU, and since the RSU’s cache is full, the reward is -1. But now we must evict content to place the newly requested

content in the cache, and the dropped content will be the one with the least popular.

Table 6- RSU cache at $t=1$.

Contents	11	47	67
Popularity	5	1	4

CHAPTER FOUR

EXPERIMENTAL RESULTS

In this chapter, we will discuss the experimental results of our suggested model. First, it will introduce the testing environment (i.e., software and hardware) in which we got the results. Second, it will review the data used to produce the results and how it was generated. Third, this chapter will present the performance of the suggested model.

4.1 Simulation Environment

We adopted Python programming language to implement the algorithms mentioned in the suggested model. Python is widely used to program Machine Learning models. Moreover, we used the Tensorflow library by Google to implement our Reinforcement Learning model (DDQN). Also, to create the virtual environment used for the model, we adopted the Gym library created by OpenAi. The Gym library allows us to specify custom observation and action spaces to train the model efficiently. We used the Anaconda platform to integrate these libraries, allowing us to add the respective libraries, thus hosting our environment.

On the other hand, to produce the results, we used a 2.20 GHz computer with 16 GB of RAM, a core i7 processor, 1 TB HDD and 256 GB SSD, and a 64-bit Windows 11 operating system.

4.2 Data

The data used for the simulation is generated using M-Zipf distribution [14]. M-Zipf distribution allows us to create data with popularity, where some data will be generated continuously according to the popularity assigned to it by the method. And throughout the testing phase, we updated the parameter that allows the data to be sparser to understand and

validate the efficiency of the proposed model. However, we will be manipulating specific parameters of the data generator, such as the sparsity of the data (s), how much content is requested (N), and the range of data (m), in each iteration for us to evaluate the suggested model.

4.3 Model Performance

In this subsection, we will evaluate the suggested model using the simulation environment mentioned earlier. The metrics in which we are evaluating our model are “Cache Hit-Rate” and “Response Time.” In addition, a discussion of the results obtained will be presented with an analysis of what has been achieved.

For our first two tests, we compare our proposed algorithm with three traditional algorithms FIFO, LRU, and LFU. We have two reasons for choosing these algorithms. First, these algorithms are popular in the industry, and second, they perform differently. Furthermore, one of the algorithms might outperform the other depending on the environment, thus providing dynamicity to our testing phase.

In the first set of tests, we manipulated the sparsity of the content requested. In other words, we altered the probability of the data that is to be requested more often, which allows us to simulate real-world data where in some cases, there will be content that won't be requested many often. In addition, as shown in Fig. 17, the sparsity parameter is set to 0.8, meaning less likelihood of a certain content being requested more often. Also, the number of contents requested here is set to 1000, and the range of content is set to 1000, meaning there will be up to 1000 different content eligible to be requested in an iteration. We notice that the proposed FDRL is superior in terms of cache hit rate compared to other traditional algorithms such as FIFO, LRU, and LFU.

Furthermore, in Fig. 18, the traditional algorithms close the gap to the proposed FDRL approach. The reason behind this visualization is that decreasing the sparsity of the data allowed the traditional algorithms to enhance their performance. In other words, the probability of having the same content requested again increased, allowing the conventional algorithms to serve the requests accordingly. However, our proposed algorithm is still superior to those traditional ones since its hit rate remained higher than the other algorithms, and its performance is consistent. It is also worth mentioning that out of three traditional algorithms, only the LFU achieved similar results compared to our FDRL approach.

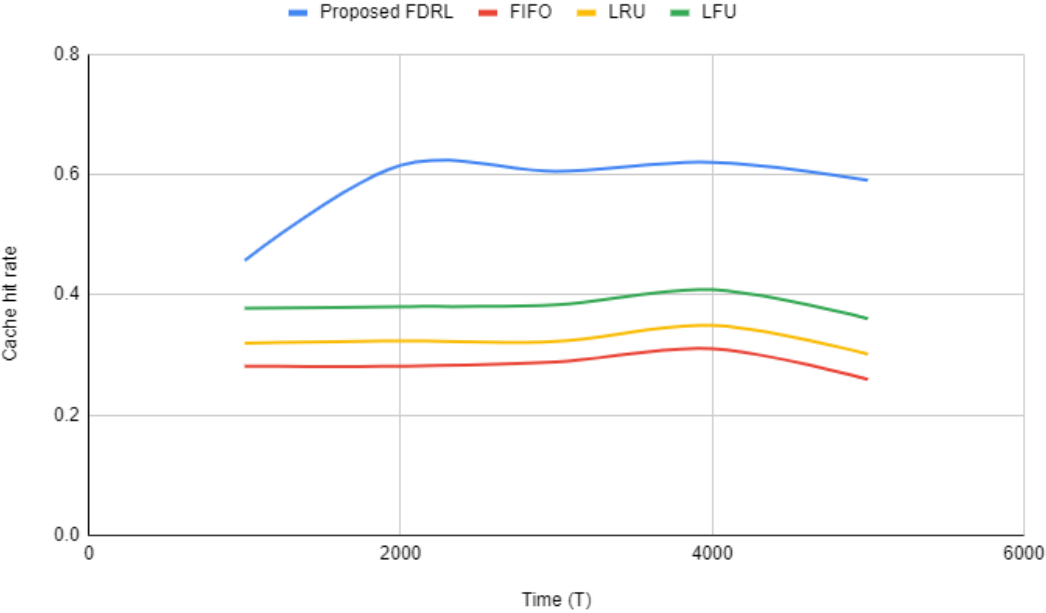


Figure 17- Cache hit rate versus time, where $s=0.8$, $N=m=1000$.

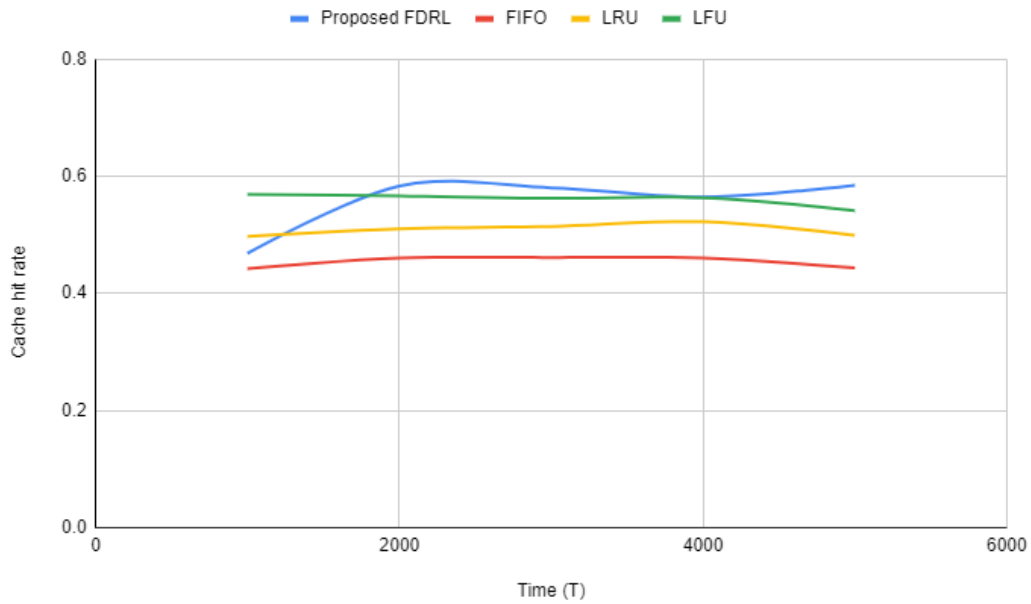


Figure 18- Cache hit rate versus time, where $s=1$, $N=m=1000$.

The first set of results showed that the proposed FDRL approach performs better than the traditional algorithms regarding cache hit rate. This is because the proposed algorithm had a higher cache hit rate in both tests, even with the manipulation of the sparsity parameter, showing consistency. Moreover, as we decrease the sparsity of the data, which means increasing the probability of having the same requested content more often, the traditional algorithms have an increase in their performance compared to when the data is sparse.

Moreover, in the second set of tests, we fixed the sparsity of the data and altered the number of requests in each iteration, in addition to the range of content. This type of test would allow us to test the model with different scenarios for content range, which sometimes would be more than other times. Furthermore, as shown in Fig. 19, the sparsity parameter s is set to 1, and the range of data is set to 2000 compared to 1000 in the first set of tests. Even with the fixation of (s) and altering (N) and (m), the proposed FDRL approach performs better compared to the traditional algorithms. Also, in Fig. 20, we notice the same results where we set $s=1$ and $N=m=5000$, thus increasing the range of data even more. But in the second test, the cache hit rate decreases slightly, and this decrease is relative due to the increased number

of different contents requested. In other words, when expanding the range of content available, the probability of having the same content getting requested decreases; thus, the likelihood of a requested content being available would decrease.

The second set of tests showed that even by fixing the sparsity parameter and altering the range of data available, the proposed FDRL approach is consistent with the results and performs better than traditional algorithms. This is shown when manipulating the range of content requested in both tests. The proposed approach had a higher cache hit rate than the conventional algorithms.

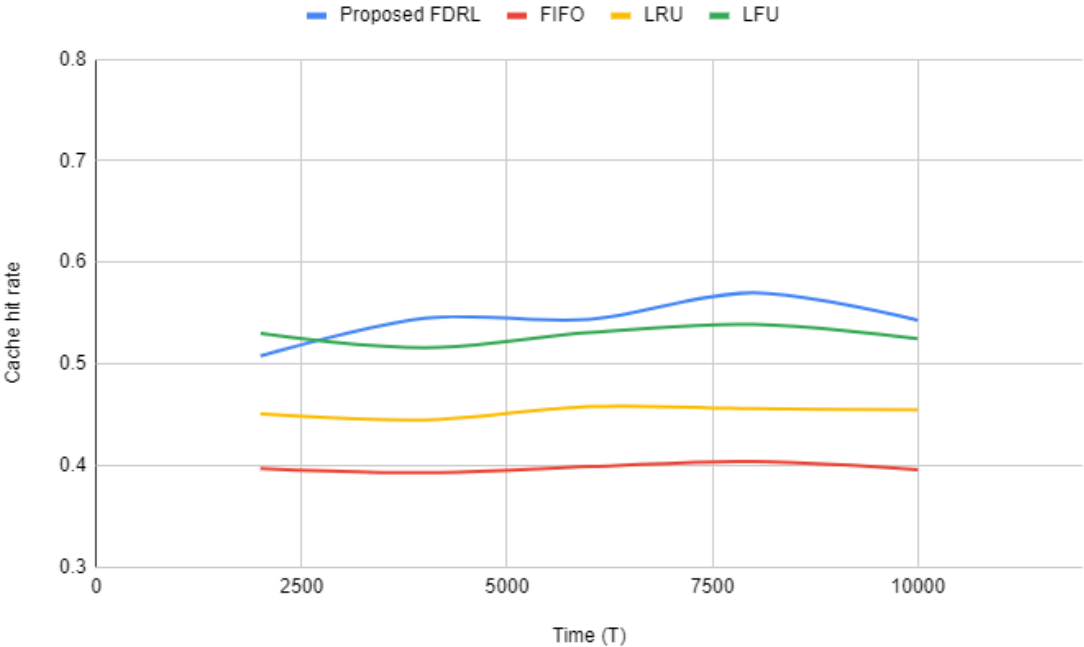


Figure 19- Cache chit rate versus time, where $s=1$, $N=m=2000$.

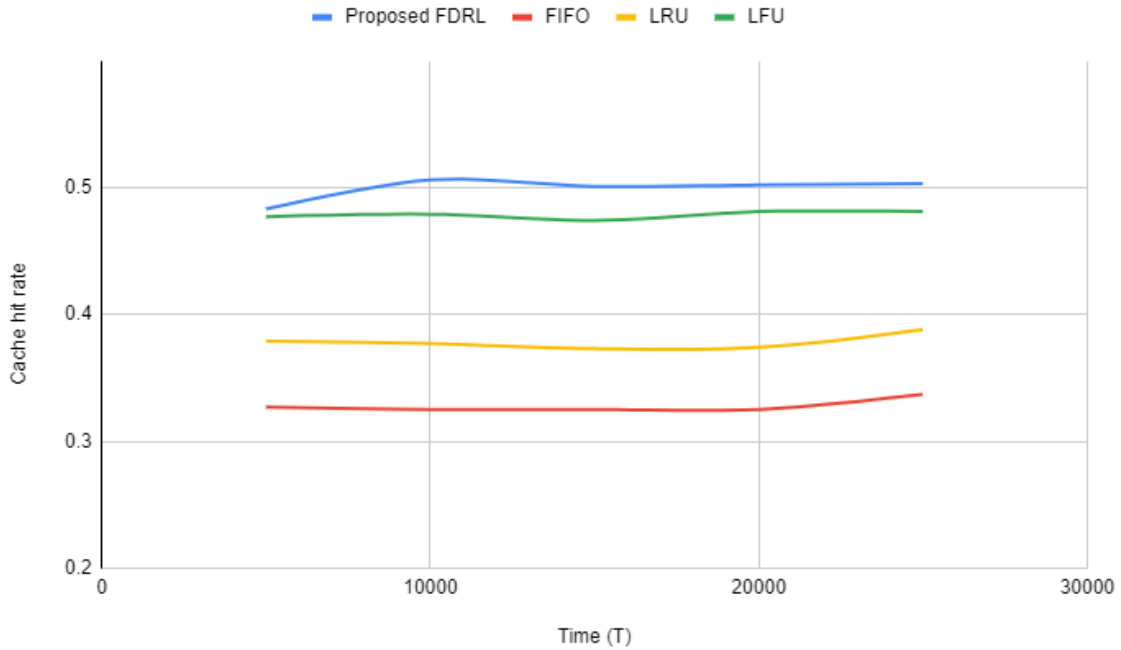


Figure 20- Cache hit rate versus time, where $s=1$, $N=m=5000$.

We also tested the proposed approach on a different metric: the request delay. We chose to compare the suggested model against other DL methods since comparing it against traditional algorithms such as FIFO, LRU, and LFU would not provide accurate visualization of the request delay because the value would be fixed. Moreover, the reason behind the request delay would be fixed for the traditional algorithms because when using these algorithms, we have only one way of communication: with the fog node. However, testing against other DRL (Deep Reinforcement Learning) models, especially models derived from each other with the same environment provided for our proposed model, would give a more accurate visualization of the performance of our proposed model. As shown in Fig. 21, our proposed method performs faster than DQN and Q-learning. Also, it is noticed that at the beginning of the training, all algorithms had relatively high request delay. Still, as training proceeded, only DQN and our proposed approach performed better with a significantly decreased request delay due to the similar nature of both algorithms since DDQN is our proposed algorithm derived from DQN.

In the second part of testing, we compared our proposed approach to other DL models to check whether our system had an enhancement in request delay. The results showed that the FDRL approach had decreased request delay compared to the other models. Furthermore, at the beginning of the training, the FDRL approach was faster, while the proposed FDRL and DQN had similar request delays as training proceeded. On the other hand, Q-learning slightly decreased the request delay. However, at the end of the training, all three models converged at the same value.

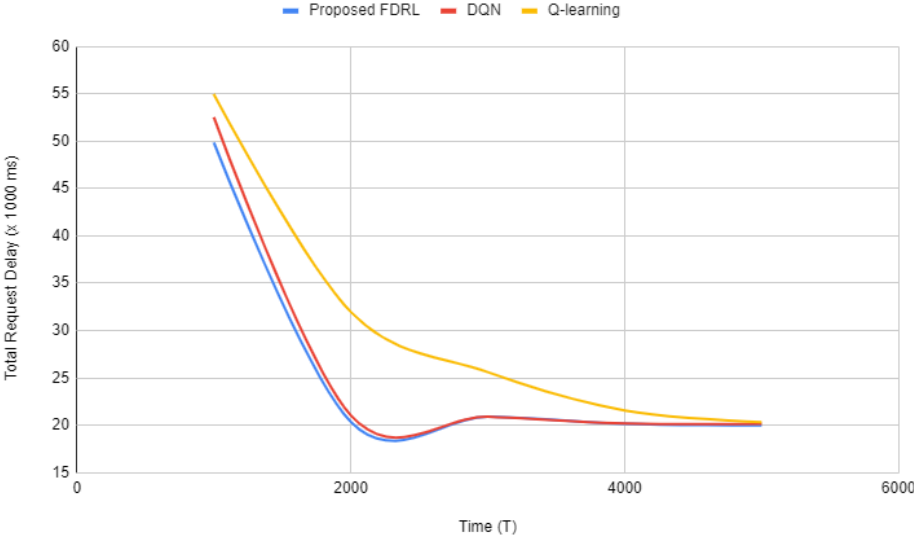


Figure 21- Request delay (ms x 1000) versus time T.

All three sets of testing that we did, where we manipulated sparsity and content range to deduce the model's performance on cache hit rate and compared the proposed model against other DL models, showed enhanced performance and enhanced performance consistency. Moreover, these tests allowed us to put our model through real-world simulation, where sometimes content requested would have less or more probability of being requested again. Also, it will enable us to see how the model would perform in scenarios where the requested content volume can be low or high.

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

Fog computing enhanced the performance of IoT networks. The capabilities of the fog, which is an extension of the cloud's services, combined with being in close vicinity to end-users, paved the road for presenting solutions to existing problems. Vehicular networks' emergence posed new requirements on the performance of the traditional network, such as increased privacy and security, better resource allocation, faster response time, etc. As a result, fog computing can be used to tackle this open problem.

In this thesis, we proposed a cooperative FDRL (Federated Deep Reinforcement Learning) scheme to tackle the response time and privacy/security issues accompanying vehicular networks. The introduction of Federated Learning in this work also allowed us to tackle the security/privacy issue by making each fog node train its Deep Reinforcement Learning model using its data. In addition, this DRL approach allowed dynamic cache management for the vehicular network.

The results of the tests showed the consistency and effectiveness of our proposed model in cache hit rate metric when compared with traditional algorithms such as FIFO, LRU, and LFU. Also, the proposed model performed better on the request delay metric when compared to other Deep Learning algorithms, such as Q-learning and DQN.

For future work, real data could be used to get more accurate, allowing the proposed model to be integrated into real-world applications. Real data may help uncover a problem that might not appear using synthetic data. In addition to that, we could implement different DL models and compare them with the suggested model.

REFERENCES

- [1] Rashid, A., & Chaturvedi, A. (2019). Cloud Computing Characteristics and Services A Brief Review. *International Journal of Computer Sciences and Engineering*, 7(2), 421–426. <https://doi.org/10.26438/ijcse/v7i2.421426>
- [2] Farooq, M., Waseem, M., Mazhar, S., Khairi, A., & Kamal, T. (2015). A Review on Internet of Things (IoT). *International Journal of Computer Applications*, 113(1), 1–7. <https://doi.org/10.5120/19787-1571>
- [3] Laghari, A. A., Wu, K., Laghari, R. A., Ali, M., & Khan, A. A. (2021). A Review and State of Art of Internet of Things (IoT). *Archives of Computational Methods in Engineering*, 29(3), 1395–1413. <https://doi.org/10.1007/s11831-021-09622-6>
- [4] Madakam, S., Ramaswamy, R., & Tripathi, S. (2015). Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications*, 03(05), 164–173. <https://doi.org/10.4236/jcc.2015.35021>
- [5] Faisal, A., Yigitcanlar, T., Kamruzzaman, M., & Currie, G. (2019). Understanding autonomous vehicles: A systematic literature review on capability, impact, planning and policy. *Journal of Transport and Land Use*, 12(1). <https://doi.org/10.5198/jtlu.2019.1405>
- [6] Dillon, T., Wu, C., & Chang, E. (2010). Cloud Computing: Issues and Challenges. *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. <https://doi.org/10.1109/aina.2010.187>
- [7] Alouffi, B., Hasnain, M., Alharbi, A., Alosaimi, W., Alyami, H., & Ayaz, M. (2021). A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies. *IEEE Access*, 9, 57792–57807. <https://doi.org/10.1109/access.2021.3073203>

- [8] Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing – MCC '12*. <https://doi.org/10.1145/2342509.2342513>
- [9] Sanketh, R., MohanaRoopa, Y., & Reddy, P. N. (2019). A Survey of Fog Computing: Fundamental, Architecture, Applications and Challenges. *2019 Third International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. <https://doi.org/10.1109/i-smac47947.2019.9032645>
- [10] Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., & Jue, J. P. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98, 289–330. <https://doi.org/10.1016/j.sysarc.2019.02.009>
- [11] Yi, S., Li, C., & Li, Q. (2015). A Survey of Fog Computing. *Proceedings of the 2015 Workshop on Mobile Big Data*. <https://doi.org/10.1145/2757384.2757397>
- [12] Cao, P., & Irani, S. (1997). Cost-aware WWW proxy caching algorithms. *USENIX Symposium on Internet Technologies and Systems*, 18. https://static.usenix.org/publications/library/proceedings/usits97/full_papers/cao/cao.pdf
- [13] Khattak, H. A., Raja, F. Z., Aloqaily, M., & Bouachir, O. (2021). Efficient In-Network Caching in NDN-based Connected Vehicles. *2021 IEEE Global Communications Conference (GLOBECOM)*. <https://doi.org/10.1109/globecom46510.2021.9685200>
- [14] Zhang, M., Jiang, Y., Zheng, F. C., Bennis, M., & You, X. (2021). Cooperative Edge Caching via Federated Deep Reinforcement Learning in Fog-RANs. *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. <https://doi.org/10.1109/iccworkshops50388.2021.9473609>

- [15] Tang, F., Kawamoto, Y., Kato, N., & Liu, J. (2020). Future Intelligent and Secure Vehicular Network Toward 6G: Machine-Learning Approaches. *Proceedings of the IEEE*, 108(2), 292–307. <https://doi.org/10.1109/jproc.2019.2954595>
- [16] Posner, J., Tseng, L., Aloqaily, M., & Jararweh, Y. (2021). Federated Learning in Vehicular Networks: Opportunities and Solutions. *IEEE Network*, 35(2), 152–159. <https://doi.org/10.1109/mnet.011.2000430>
- [17] Matsuo, Y., LeCun, Y., Sahani, M., Precup, D., Silver, D., Sugiyama, M., Uchibe, E., & Morimoto, J. (2022). Deep learning, reinforcement learning, and world models. *Neural Networks*, 152, 267–275. <https://doi.org/10.1016/j.neunet.2022.03.037>
- [18] Nguyen, D. H., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., & Poor, H. V. (2021). Federated Learning for Internet of Things: A Comprehensive Survey. *IEEE Communications Surveys and Tutorials*, 23(3), 1622–1658. <https://doi.org/10.1109/comst.2021.3075439>
- [19] Yu, Z., Hu, J., Min, G., Wang, Z., Miao, W., & Li, S. (2021). Privacy-Preserving Federated Deep Learning for Cooperative Hierarchical Caching in Fog Computing. *IEEE Internet of Things Journal*, 9(22), 22246–22255. <https://doi.org/10.1109/jiot.2021.3081480>
- [20] Li, Q., Zhang, Y., Li, Y., Xiao, Y., & Ge, X. (2020b). Capacity-Aware Edge Caching in Fog Computing Networks. *IEEE Transactions on Vehicular Technology*, 69(8), 9244–9248. <https://doi.org/10.1109/tvt.2020.3001301>
- [21] Hu, L., Miao, Y., Yang, J., Ghoneim, A., Hossain, M. S., & Alrashoud, M. (2020). IF-RANs: Intelligent Traffic Prediction and Cognitive Caching toward Fog-Computing-Based Radio Access Networks. *IEEE Wireless Communications*, 27(2), 29–35. <https://doi.org/10.1109/mwc.001.1900368>

- [22] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10), 2222–2232. <https://doi.org/10.1109/tnnls.2016.2582924>
- [23] Hao, Y., Miao, Y., Hu, L., Hossain, M. S., & Muhammad, G. (2019). Smart-Edge-CoCaCo: AI-Enabled Smart Edge with Joint Computation, Caching, and Communication in Heterogeneous IoT. *IEEE Network*, 33(2), 58–64. <https://doi.org/10.1109/mnet.2019.1800235>
- [24] Chang, Z., Lei, L., Zhou, Z., Mao, S., & Ristaniemi, T. (2018). Learn to Cache: Machine Learning for Network Edge Caching in the Big Data Era. *IEEE Wireless Communications*, 25(3), 28–35. <https://doi.org/10.1109/mwc.2018.1700317>
- [25] Wang, H., Yi, X., Huang, P., Cheng, B., & Zhou, K. (2018). Efficient SSD Caching by Avoiding Unnecessary Writes using Machine Learning. *International Conference on Parallel Processing*. <https://doi.org/10.1145/3225058.3225126>
- [26] Huang, W., Song, T., Yang, Y. C., & Zhang, Y. (2019). Cluster-Based Cooperative Caching With Mobility Prediction in Vehicular Named Data Networking. *IEEE Access*, 7, 23442–23458. <https://doi.org/10.1109/access.2019.2897747>
- [27] Jiang, W., Feng, G., Qin, S., Yum, T. P., & Cao, G. (2019). Multi-Agent Reinforcement Learning for Efficient Content Caching in Mobile D2D Networks. *IEEE Transactions on Wireless Communications*, 18(3), 1610–1622. <https://doi.org/10.1109/twc.2019.2894403>
- [28] Dai, Y., Xu, D., Zhang, K., Maharjan, S., & Zhang, Y. (2020). Deep Reinforcement Learning and Permissioned Blockchain for Content Caching in Vehicular Edge Computing and Networks. *IEEE Transactions on Vehicular Technology*, 69(4), 4312–4324. <https://doi.org/10.1109/tvt.2020.2973705>

- [29] Wang, X., Wang, C., Li, X., Leung, V. C. M., & Taleb, T. (2020). Federated Deep Reinforcement Learning for Internet of Things With Decentralized Cooperative Edge Caching. *IEEE Internet of Things Journal*, 7(10), 9441–9455.
<https://doi.org/10.1109/jiot.2020.2986803>
- [30] Yu, Z., Hu, J., Min, G., Zhao, Z., Miao, W., & Hossain, M. S. (2021). Mobility-Aware Proactive Edge Caching for Connected Vehicles Using Federated Learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(8), 5341–5351.
<https://doi.org/10.1109/tits.2020.3017474>
- [31] Yu, Z., Hu, J., Min, G., Zhao, Z., Miao, W., & Hossain, M. S. (2021). Mobility-Aware Proactive Edge Caching for Connected Vehicles Using Federated Learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(8), 5341–5351.
<https://doi.org/10.1109/tits.2020.3017474>
- [32] Cheng, P., Ma, C., Ding, M., Hu, Y., Lin, Z., Li, Y., & Vucetic, B. (2019). Localized Small Cell Caching: A Machine Learning Approach Based on Rating Data. *IEEE Transactions on Communications*, 67(2), 1663–1676.
<https://doi.org/10.1109/tcomm.2018.2878231>
- [33] Salakhutdinov, R., & Mnih, A. (2008b). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. *International Conference on Machine Learning*.
<https://doi.org/10.1145/1390156.1390267>
- [34] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning* (pp. 1995–2003).
<http://proceedings.mlr.press/v48/wangf16.pdf>

[35] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated Machine Learning. *ACM Transactions on Intelligent Systems and Technology*, 10(2), 1–19.

<https://doi.org/10.1145/3298981>