QATAR UNIVERSITY

COLLEGE OF ENGINEERING

ENERGY-EFFICIENT USER-EDGE ASSOCIATION AND RESOURCE ALLOCATION IN IOT-BASED HIERARCHICAL FEDERATED LEARNING

BY

HASSAN ABDULRAHMAN SAADAT

A Thesis Submitted to

the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Masters of Science in Computing

June 2022

COMMITTEE PAGE

The members of the Committee approve the Thesis of
Hassan Abdulrahman Saadat defended on 01/01/2022.

Prof. Amr Mahmoud Salem Mohamed
Thesis Supervisor

Dr. Committee One
Committee Member

Dr. Committee Two
Committee Member

Approved:

Khalid Kamal Naji, Dean, College of Engineering

ABSTRACT

Saadat, Hassan, A., Masters : June : 2022, Masters of Science in Computing

Title: Energy-efficient user-edge association and resource allocation in IoT-based hierarchical federated learning

Supervisor of Thesis: Prof. Amr Mahmoud Salem Mohamed.

The proliferation of data as part of the Internet of Things (IoT) systems needs to be efficiently utilized while respecting data privacy and scalability. Edge computing is an emerging paradigm that mandates efficient processing of local data, close to where data is being collected. Such paradigm has motivated enormous research that merges computation and communication resources to explore many trade-offs that address heterogeneity of the IoT devices, while taking care of both scalability and data privacy. Federated learning (FL) is a distributed learning paradigm combining edge computing with artificial intelligence techniques. FL, compared to centralized learning (CL), preserves the data privacy of, and reduces the communication energy consumption by IoT devices, by requiring them to share locally trained machine learning models with the cloud rather than their private raw data. Hierarchical federated learning (HFL) improves FL by deploying a layer of edges that are responsible for multiple intermediate model aggregation rounds before the global aggregation is performed on the cloud. The HFL configuration alongside efficient user-edge association and resource allocation ensure more energy and communication efficient, and skewed-data robust learning scheme compared to FL. In this thesis, we assess the learning performance of the HFL framework while respecting IoT devices' limitations, such as energy budget, computational power, and storage space. First, HFL is evaluated in terms of learning performance and non-identically and independently distributed (non-iid) data handling by implementing an intrusion detection system (IDS) using the NSL-KDD dataset. Then, we formulate and solve a communication energy minimization problem that performs optimal client-edge association and resource allocation. We also implement an alternative less complex solution leveraging reinforcement learning (RL) that provides a fast user-edge association and resource allocation response in highly dynamic HFL networks. The proposed solutions are compared with several state-of-the-art client-edge association techniques, leveraging MNIST dataset. Moreover, we study the trade-off between minimizing the per-round energy consumption and Kullback-Leibler divergence (KLD) of the data distribution, and its effect on the total energy consumption.

DEDICATION

To my mother and the rest of my family for their unlimited support.

ACKNOWLEDGMENTS

This work was made possible by NPRP grant # NPRP13S-0205-200265 from the Qatar National Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

I would like to thank any person that helped making it possible for me to reach this stage of my academic career, whether academically, emotionally, or financially.

Special thanks go to Prof. Amr Mohamed for his guidance and mentorship during my bachelor and masters studies, Eng. MHD Saria Allahham for his continuous academic and technical support, and Eng. Abdulla Abu Madi for his academic partnership during my 6-year span of university education.

TABLE OF CONTENTS

DEDICATION v

ACKNOWLEDGMENTS vi

LIST OF FIGURES ix

LIST OF PUBLICATIONS x

Chapter 1: Introduction 1

Chapter 2: Background & Related Work 7

Centralized, Federated, and Hierarchical Federated Learning 7

Distributed Intrusion Detection System 10

Reinforcement Learning 12

Related Work 13

Federated Learning 13

Hierarchical Federated Learning 15

Chapter 3: FL & HFL COMPARISON: INTRUSION DETECTION SYSTEM 18

System Model 18

Training Loss Estimation 19

Centralized Learning 19

Federated Learning 20

Hierarchical Federated Learning 20

IDS Settings 21

NSL-KDD dataset 21

Dataset Preprocessing and Neural Network 22

Performance Evaluation 23

Study Case A: iid Client-Edge Association 23

Study case B: iid Client-Edge Association with Equal Number of Attacks' Samples 24
Study Case C: Non-iid Client-Edge Association 25
Chapter 4: RL-Assisted Energy-Aware HFL User-Edge Association 27
System model & Problem Formulation 27
Proposed Solutions 29
Relaxation-based Approach 30
Reinforcement Learning-based Approach 30
Performance Evaluation 33
Simulation settings 33
RL Training 34
Energy Consumption and HFL performance 35
Chapter 5: Conclusion 39
Chapter 6: Future Work 41

LIST OF FIGURES

Figure 2.1. High-level view of (a) Centralized Learning, (b) Federated Learning, and (c) Hierarchical Federated Learning. 7

Figure 3.1. System model, and (a) FL and (b) HFL architectures. 19

Figure 3.2. Class distribution of the NSL-KDD (a) training and (b) testing sets 21

Figure 3.3. The used neural network diagram 22

Figure 3.4. (a) Average loss and (b) testing accuracy for study case A. 24

Figure 3.5. (a) Average loss and (b) testing accuracy for study case B. 25

Figure 3.6. (a) Average loss and (b) testing accuracy for study case C. 26

Figure 4.1. System model and HFL architecture. 28

Figure 4.2. RL-based solution overview. 32

Figure 4.3. RL reward convergence for selected $min_{distance}$ values on the non-iid distribution. 35

Figure 4.4. (a) Total communication energy consumption per round, and (b) average edge KLD of the small-scaled environment with non-iid client data. 37

Figure 4.5. (a) Total communication energy consumption per round, and (b) average edge KLD of the medium-scaled environment with non-iid client data. 37

Figure 4.6. Total communication energy consumption per round for the (a) small-scaled and (b) medium-scaled environments with iid client data. 37

Figure 4.7. (a) The effect of average KLD value on the testing accuracy, and (b) the number of communication rounds required to reach target accuracy. 38

Figure 4.8. (a) Communication energy consumption per one round, and (b) the total communication energy consumed to reach target accuracy. 38

LIST OF PUBLICATIONS

[1] H. Saadat, A. Aboumadi, A. Mohamed, A. Erbad and M. Guizani, "Hierarchical Federated Learning for Collaborative IDS in IoT Applications," 2021 10th Mediterranean Conference on Embedded Computing (MECO), 2021, pp. 1-6, doi: 10.1109/MECO52532.2021.9460304.

[2] H. Saadat, M. S. Allahham, A. A. Abdellatif, and A. Mohamed, "RL-Assisted Energy-Aware User-Edge Association for IoT-based Hierarchical Federated Learning," (accepted in) 2022 International Wireless Communications and Mobile Computing (IWCMC), 2022.

CHAPTER 1: INTRODUCTION

The massive advancements of embedded computing power, wireless communication technologies, and wireless sensor networks, have led to the rise of the sun of the Internet-of-Things (IoT) era in the last few decades. The recent tendency towards automating surrounding environment sensing, gathered data processing, device-to-device and device-to-network communication, and the current and future orientation towards building smart houses, cities, grids, etc., have all magnified the need and the deployment of IoT devices, leading to a projection of 500 billion IoT devices to be used worldwide by 2025 [1]. IoT today can be defined as the network of connected heterogeneous devices that interactively collaborate in accomplishing complex tasks with high level of intelligence and minimum human intervention [2]. This definition widens the range of the forms that an IoT device can take. For example, an IoT device can be thought of as a smart house device, such as a smart air conditioner or fridge, a wearable device like a smartwatch, an embedded device in a bigger system, such as a camera in a radar technology in smart vehicles, or a medical device implanted inside a patient human's body. This heterogeneity means that IoT devices are found in highly disparate domains [3][4], such as smart households, smart cities, transportation systems, agriculture, and healthcare. Despite the fanciness behind the term IoT and how IoT networks are meant to reduce humans' efforts, save their time, and raise the quality of their lives, many challenges arise with the huge growth of the reliance on IoT systems. These challenges have grabbed lots of researches' attention who tried to study and propose solutions to the issues related to IoT, such as IoT's limited resources management [2], security concerns [5][6], and big data handling and training [7].

The environments that IoT devices are deployed in are highly demanding and continuously requiring for IoT devices' services. For example, a camera in a surveillance system must be continuously recording and sending the scenes to the deployer's device or monitor, also, an air conditioner in a smart house has to be always sensing the room's temperature and always connected to the network to receive the deployer's commands. Moreover, we still have not mentioned that distributed learning is dominant nowadays, so these devices are expected to be performing machine learning tasks locally. We can come up to the result that IoT devices, in general, need to be highly interactive, interoperable, easily mobile, and always connected to the network [2]. However, that comes with the cost of IoT devices having constraints related to resources like power supply, storage space, communication capabilities, and processing and computational power. To address these limitations, one of the possible solutions is to use the concept of cloud computing, with a central cloud processing and providing any required services by the IoT devices. However, the data privacy of IoT devices may be threatened, and high delays and energy consumption are expected due to the unsecure public internet path and the long distance between the IoT devices and the cloud server. Therefore, the concept of edge computing has significantly emerged [8], where the IoT devices' computations are pushed to a physically near edge node, providing a more end-to-end data privacy-aware, delay-immune, and energy-efficient service center compared to the cloud. The merge of edge computing with artificial intelligence (AI) results in having powerful distributed learning paradigms, such as Federated learning (FL) [9], which will be explained in details in the next chapter.

The vastly emerging growth of IoT systems introduces severe security threats and more complicated challenges compared to normal information technology (IT) devices [10] and to any other computing systems [6]. That hinders the goal of protecting the confidentiality, integrity, authentication, authorization, availability, and non-repudiation in IoT systems [6], and makes the three IoT layers, namely, perception, network, and application layers vulnerable to various types of attacks [11]. For example, in the perception layer, where sensors and actuators are usually deployed in outdoor environments, a physical attack can take place by the attacker gaining unauthorized access and tampering the physical components of the IoT device, which threatens 1) the confidentiality of this layer by the attacker performing a replay attack or a timing attack to extract useful information or the encryption key, 2) the integrity by the attacker introducing an IoT node to the network and using it to send useless or malicious data, and 3) the availability of the IoT device by the attacker continuously sending malicious data that drains out the device's energy, which is known as denial-of-service attack (DoS). In the network layer, where the data transmission between the different components of the IoT network occurs, the privacy and confidentiality can be endangered, and the communication channel can be exposed to the attacker using different types of attacks, such as eavesdropping. Depending on the capabilities of the hardware components built-in inside IoT devices, some of them use Sigfox or cellular networks, which are suitable for long ranges, while others communicate using communication protocols of shorter ranges, such as ZigBee, 6LoWPAN, and RFID, with each protocol having trade-offs between performance metrics, such as energy consumption, communication range, data rate, and security measures [12]. Such inconsistency of the networking protocols used in IoT devices, the heterogeneity of IoT networks' components, alongside the weak infrastructure, and the low computational capabilities introduce broad vulnerability surfaces in the network layer [5]. Finally, in the application layer, the lack of standardized policies in IoT systems results in each application applying its own authorization and authentication techniques, which complicates the identity authentication of the user, and the need to reveal highly sensitive and private data, such as personal, financial, biometric, and medical information of the user to an edge node or to the cloud to acquire more advanced services, or to contribute to the training of a global learning model are all factors that threaten the privacy and confidentiality of the users' data. This brief introduction to the ocean of security issues in IoT systems shows that it is extremely necessary to adopt integrated security measures to overcome these vulnerabilities, however, as discussed above, complex security measures are not compatible with IoT devices' limited resources, therefore, having lightweight and efficient security solutions in IoT networks has been and still is a hot and open research field.

Another major challenge that is faced in IoT networks is how to store, process, and communicate the huge amounts of data that are generated from IoT devices, and use them in training artificially intelligent learning models for predicting future events, making efficient decisions, and detecting threatening attacks. Focusing on the AI part, the authors in [7] provide detailed explanation on different techniques for efficient distributed AI in IoT applications. The focus in this thesis will be on the three main collaborative learning schemes, where IoT devices collaborate in training a global machine learning model, namely, centralized learning (CL), federated learning (FL), and edge-assisted FL or hierarchical FL (HFL). Each of which has its own advantages and disadvantages in terms of learning performance, data privacy preservation, communication efficiency, and energy consumption. A detailed discussion is provided in the following chapters.

Based on the discussion given, the introduction can be summarized in three major points: 1) IoT devices are resource-constrained in terms of computation and communication capabilities, power supply, and storage space, 2) major security issues arise with the emergence of IoT networks and they require comprehensive and efficient solutions, and 3) the extremely huge amounts of data collected in IoT environments need to be handled and processed properly to deploy powerful, yet efficient, machine learning models for better future decision making. This leads us to the following research questions:

- How efficient is it to use distributed machine learning in applying security measures such as an intrusion detection system (IDS)?
- How do different distributed learning schemes improve learning performance and reduce the non-identically and independently distributed (non-iid) data effect?
- How to efficiently associate users with edges and use the available network resources, Channel State Information (CSI), and data distribution on the users in HFL, such that the energy consumption by the power-limited IoT devices is minimum?
- What possible techniques can handle the mobility nature of IoT devices and dynamically perform user-edge association and resource allocation?

In light of these research questions, we highlight the objectives of this thesis as follows:

1. Evaluate and compare the learning performance and skewed-data robustness of distributed learning scenarios such as FL, and HFL in implementing a distributed IDS.
2. Model the HFL in IoT systems as an energy minimization framework that takes into consideration the class distribution on the edges and the communication latency on the client-edge level.

3. Propose efficient solutions for the NP-hard mixed-integer nonlinear programming problem (MINLP) by dividing it into two subproblems to facilitate decentralization.

4. Deploy a reinforcement learning (RL) agent as a less complex online alternative solution to the optimization problem, that performs very fast user-edge association and resource allocation in highly dynamic IoT networks.

5. Conduct a comparative study between the relaxation-based, RL-based, and other state-of-the-art client-edge association methods.

CHAPTER 2: BACKGROUND & RELATED WORK

Centralized, Federated, and Hierarchical Federated Learning

Centralized learning (CL), Figure 2.1.(a), is one of the learning schemes in which a main cloud applies a learning algorithm, such as a convolutional neural network (CNN), on the data that are gathered from the IoT devices connected to it i.e., its clients. This method, usually, ensures high learning accuracy and fast convergence because of the large amount and the wide diversity of the data samples that it receives from the clients. However, several disadvantages can be noticed in this process [13]. A crucial drawback of CL is that the private data of the clients, such as the personal, medical, and financial information of their owners must be shared with and exposed to the main cloud. Also, a large amount of communication energy gets consumed by the power-limited IoT devices during the recurrent process of transmitting their data to the main cloud. Moreover, the physical configuration and the long distance between the main cloud and the connected clients may cause high communication latency and connection inconsistency.

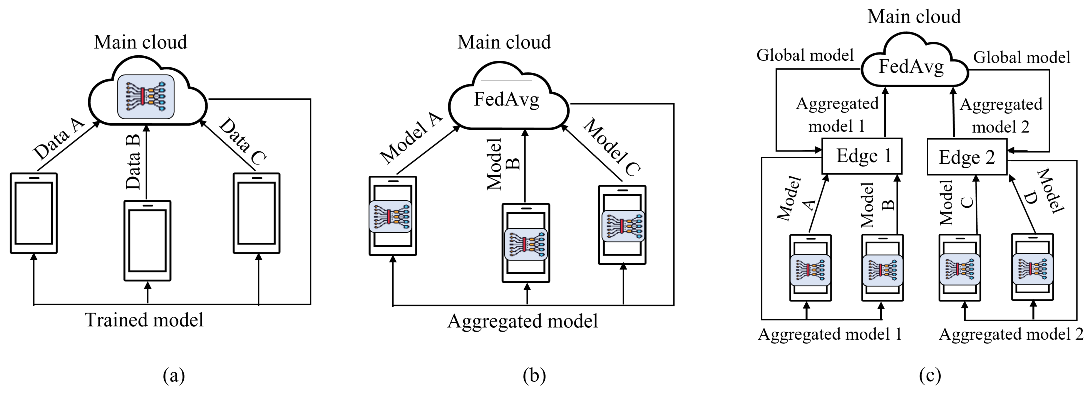


Figure 2.1. High-level view of (a) Centralized Learning, (b) Federated Learning, and (c) Hierarchical Federated Learning.

A communication-efficient learning technique on distributed data known as FL was proposed by Google in 2016 [9]. In FL [13], Figure 2.1.(b), first, the main cloud initiates a learning model and sends its weights to the clients. Then, each client uses its own local data to update its local model i.e., the model it received from the main cloud. Once all clients have finished updating their respective models for a specific number of local training epochs, they send them to the main cloud to get aggregated. Finally, the global/aggregated model is sent back to the clients, and that process is repeated until convergence is reached. FL is thought to be more useful when the data distribution is non-iid compared to iid, where the clients make use of the previously unseen knowledge of other clients, which actually is the dominant case in IoT networks due to the heterogeneity of the IoT devices and the environments they are deployed in. Although, from the learning problem point of view, CL may face an easier time reaching a high accuracy and fast convergence compared to FL because the whole data is present at one place, FL tends to overcome the other issues related to CL. For example, the privacy of clients' data is preserved because they never have to leave the IoT end-device. Also, FL is more communication-efficient compared to CL due to the fact that only the model weights are communicated between the cloud and the clients instead of the large amounts of raw data, which results in less network bandwidth wasted and less communication energy consumed by the IoT devices in FL compared to CL. The communication energy consumption of the IoT devices and the learning convergence speed can be further controlled by varying the communication frequency between the cloud and the clients i.e., varying the number of learning epochs that clients should do before sending their models to the cloud. The study in [14] shows that the more frequently the models are exchanged, the faster the convergence is reached, however, the more communication energy is consumed by the IoT devices.

In FL, there are different optimization algorithms or aggregation methods that can be used. Federated Stochastic Gradient Decent (FedSGD) is the baseline aggregation technique in which the clients send their models to the cloud after one epoch of learning. In [9], Google introduced the concept of Federated Averaging (FedAVG) and compared it with FedSGD on thorough experiments on different visual datasets. FedAVG strongly outperformed FedSGD in terms of training loss, testing accuracy, and speed of convergence. The implementation difference in FedAVG compared to FedSGD is that, in FedAVG, the clients perform a common multiple number of local learning epochs before sending their models to the aggregator, which significantly enhanced the learning process. FedPROX [15] is a general form of FedAVG in which clients are assigned different number of learning epochs and can use a local solver of their choosing based on each client's local resources, such as computation capabilities and battery size. FedPROX provides a better handling of the heterogeneity of IoT device and discrepancy in their characteristics. In our work, FedSGD is implemented in the client-edge level, and FedAVG is applied in the edge-cloud level.

An advancement of FL was recently developed and studied, which is known as edge-assisted or Hierarchical Federated Learning (HFL), which is achieved by adding a layer of edge nodes between the main cloud and the clients. The architecture of HFL, Figure 2.1.(c), consists of multiple smaller FL instances, where each edge node and its connected clients form an FL instance, and the main cloud along with the edge nodes form a bigger FL instance. The collaborative learning process is very similar to FL, where initial models are sent from the main cloud to the clients passing through the edge nodes. After a specific number of learning epochs, edge-level models are aggregated by each edge retrieving the models from its respective clients. After a specific number of client-edge models' exchange rounds, the edge-level models are aggregated on the main cloud to form a global model and this model is sent back to the clients to continue the learning process. Despite its infrastructure overhead and complexity of assigning clients to edges arbitrarily, one of the main advantages of HFL over FL is that the distance between the edge and its clients is always shorter than the distance between the main cloud and the clients, which results in better communication efficiency, unlike FL, where the probabilities of delay and communication disconnection are higher. Also, HFL preserves the other benefits of FL, such as the clients' data privacy. Moreover, the comparison study cases done in our previous work [16] on three different data distributions show that HFL can handle more efficiently and reduce the effect of non-iid data compared to FL in terms of the testing accuracy and speed of convergence.

Distributed Intrusion Detection System

The security discussion provided in the introduction chapter confirms that the IoT network layer is vulnerable to different types of intrusion attacks. Those attacks may compromise the confidentiality and integrity of data, and the availability of devices and services in highly sensitive environments, which can result in disastrous consequences. Therefore, adopting security countermeasures such as IDS in IoT applications is of extreme necessity. However, due to the resource limitations found in IoT devices, implementing an IDS on each IoT device is a challenging task. Therefore, IDS implementation techniques that use machine and deep learning algorithms and take into consideration the issues related to energy consumption, data privacy, and non-iid data have been a major research field in IoT systems recently. Moreover, the deployment of distributed learning schemes, such as FL, in training an IDS, makes each IoT device benefit from the previously unseen attack types that are present on the neighboring IoT devices' datasets.

There are different datasets that can be used for intelligent machine learning-based IDS implementation. Some examples include: 1) CIC-IDS2017 [17], 2) CSE-CIC-IDS2018 [18], and 3) NSL-KDD [19]. The CIC-IDS2017 [20] dataset was developed by researchers in the Canadian Institute of Cybersecurity (CIC) in 2017 to overcome the shortages related to the available IDS datasets at that time, such as artificial injection of attacks, data redundancy, data corruption, and lack of attacks' comprehensiveness. The dataset contains nearly 3 million samples consisting of a normal class and 14 different types of the most common networks attacks, where each sample has 84 features. The CIC in collaboration with the Communications Security Establishment (CSE) proposed CSE-CIC-IDS2018 [21] as an enhancement of CIC-IDS2017. The major advancement in the 2018 version is that they managed to reduce the attacks' imbalance ratio in the dataset by applying Synthetic Minority Oversampling TEchnique (SMOTE). Both datasets proved their efficiency and they have been used by scholars since the day they were published. The reason behind selecting the NSL-KDD dataset in our work is because of the extensive reliance and number of times it has been used in the literature by researchers and experts is unmatched compared to other IDS datasets. The details of the NSL-KDD and discussion about its efficiency and reliability are provided as we move forward in this thesis.

Many researchers have attempted implementing IDS using machine and deep learning techniques. A comparison between the performance of K-Means and fuzzy C-Mean clustering on NSL-KDD dataset was done in [22]. A comprehensive study on the implementation of IDS over different imbalanced datasets, such as KDDCup

99, NSL-KDD, UNSW-NB15, and WSN-DS was conducted in [23]. A variety of other machine and deep learning techniques were applied for IDS on NSL-KDD and other datasets [6], [24], [25]. Most of the previously mentioned approaches tackled the problem from a single node's point of view that sees the whole data, not as FL or HFL. DeepFed [26] implemented an FL approach for IDS for cyber-physical systems (CPS) using industrial CPS datasets. In [27], a comparison between centralized, self, and federated learning on NSL-KDD using real implementation on raspberry pi was conducted. An enhanced FL algorithm for intrusion detection was developed by [28], and it was tested on a dataset that is a combination of KDDCUP 99, CICIDS2017, and WSN-DS datasets. The authors in [16] used the HFL framework in implementing an IDS using the NSL-KDD dataset.

Reinforcement Learning

Besides supervised learning in which machine learning models are trained on labeled datasets, and unsupervised learning paradigms that try to find the implicit correlation between the data samples in an unlabeled dataset, a third AI technique called reinforcement learning (RL) arise [29]. RL is a machine learning framework that tries to learn the pattern of the sequential states in highly dynamic environments. RL sequential training does not rely on labeled or unlabeled collected datasets, but, it observes the states of the environment it is deployed in, and tries to make optimal decisions (actions) that maximize the environment's benefit (reward). For example, as in chapter 4, an RL agent can be deployed in a HFL environment to find the mobility pattern of IoT devices by observing their locations and CSI (state), associate IoT devices with edge nodes and allocate bandwidths (action), such that the energy consumed by IoT devices during HFL training is minimized (maximum reward). During RL training, when the state of the environment changes, the RL agent has two options to consider while attempting taking the action that yields the maximum reward, it can either exploit its previous experience and act upon it, or it can explore new decision-making policies which might lead to even higher rewards compared to its previous policy. Therefore, the trade-off between exploitation and exploration steps must be considered. Another benefit of RL is that, after a good enough number of episodes, i.e. iterations, it can be deployed for decision-making and it can keep learning from the environment at the same time. By learning the sequence pattern, RL can predict the behavior of the environment in the future several states and take actions that maximize the reward of a number of future states, rather than taking a greedy action that only maximizes the reward of the very next state.

Related Work

Many researchers have studied, deployed, and dealt with the variations of FL and HFL, whether in the applications of security, healthcare, transportation, or any other fields. Our focus in this section will not be on the type of application, rather it will be on the more abstract objectives of the works, such as energy optimization, resource allocation, and learning performance enhancement.

Federated Learning

Working on FL, the authors in [30] applied a partial client selection policy with higher participation probability given to clients with bigger amount of data, also, a flexible global model aggregation with communication time limit was performed to avoid the effect of straggler clients, which all led to faster convergence and less communication cost. Energy-constrained client selection with model aggregation happening in analog manner (signal averaging) was done in [31]. Decentralized FL was implemented by the authors in [32] and FedMes [33], where FL took place in clusters with a selected client or edge node as a cluster head aggregator without the need for a central cloud. In [32], clients were clustered based on their social interaction with each other, and the aim of the work was to minimize the computation and communication latency by performing resource allocation while considering the computation energy consumption by the IoT devices during learning. In FedMes, each client was connected to the physically nearest edge, and the core idea of their work was that the clients residing in the overlapping region between two edges should share their local model with both edges. That simple idea led to knowledge exchange between two clusters without the need for a central cloud. Faster learning convergence in terms of time, not communication rounds, was achieved by FedMes compared to HFL because no long edge-cloud communication was needed, however, no energy consumption evaluation or resource allocation was pointed. FogFL [34] deployed a layer of fogs between the cloud and end devices [35], where in each global aggregation round the cloud chooses the fog node with the least delay and workload to act as an aggregator. That resulted in similar IoT energy consumption compared to HFL, however, the communication delay was significantly reduced. In [36], after a certain number of FL rounds, clients get clustered based on the similarity of their local models, and extensive evaluation cases show that the learning performance was significantly enhanced in terms of convergence speed and final accuracy compared to FL, however, no energy or latency analysis or comparisons were provided.

Hierarchical Federated Learning

In the HFL architecture, several works have come up with different solutions to improve and speed up the learning experience by focusing on data distribution and model characteristics while paying less attention to the physical aspects, such as the energy, latency, bandwidth, and CPU resources. Heuristic user-edge assignment based on statistical class distribution and network topology constraints successfully managed to reduce the communication rounds required in HFL and resulted in a performance that is very close to optimal in small-scaled HFL environments [37]. Mobility-Aware Cluster FL (MACFL) [38] provided a solution for the mobility nature of IoT devices, such that a device moving from the range of one edge to another participates in the training of its new edge's model with the level of contribution limited to the level of similarity between its current model and the new edge's previous model. In [39], 5% of the global dataset is resident in all edge nodes, and that portion of data is used to further train the aggregated edge model before sending it to the cloud, which enhanced the learning performance compared to FL and conventional HFL. SHARE [40] proposed a communication efficient user-edge association while considering the trade-off between the communication cost, defined in terms of physical distance and model size, and the Kullback-Leibler Divergence (KLD) which defines how far the class distribution of the edge is from the desired (usually uniform) distribution.

Other authors concentrated on efficient resource allocation, such as bandwidth, power, and CPU frequency, and minimization of the communication and computation energy and latency as in [41]. The authors in [42] formulated and solved an optimization problem that aims to minimize the cost of one global communication round by allocating CPU frequency and transmission power that give the highest local accuracy. The cost in their work is defined in terms of the computation energy and latency of the IoT device and edge, and the communication energy and latency of the IoT device, edge, and cloud. In [43], each client's model is given an importance indicator based on its training loss, and in each communication round, a fraction of clients are selected and CPU frequencies are allocated such that the positive difference between the selected devices' models' importance and their communication and computation latencies is maximum while considering the energy budget of the selected clients. An optimization problem that minimizes the weighted sum of the communication and computation energy and delay of one global round was formulated in [44]. They proposed a heuristic algorithm to estimate the solution of this problem by firstly assigning IoT devices to their nearest respective edges, then allocating bandwidths and CPU frequencies, and finally exchanging and transferring the clients between the edges until finding the desired solution.

There are other works that got taste from both approaches, like in [45], where they perform user-edge association that minimizes the KLD of the data on the edges to handle the problem of imbalanced data while performing bandwidth allocation and meeting the physical constraints of the IoT devices, such as the communication energy and latency of the clients. In [46], the learning model is partitioned into submodels, and before every communication round, clients get allocated a specific submodel, CPU frequency, and transmission rate based on their channel states, such that the ratio between the sum of gradients information of the clients and the product of the communication and computation energy and latency of the clients is maximum. The authors in [47] worked on minimizing the weighted sum of the training loss and computation latency, with the optimization parameters being the number of local iterations, allocated bandwidth, and the allowed maximum training time for each client. They also provide convergence analysis to show how the number of local iterations is determined at each communication round. An interesting work in [14] provides a convergence analysis of the HFL scheme for both convex and non-convex loss functions. They also experiment the trade-off between the client-edge and edge-cloud communication rounds frequencies in terms of their effect on the learning accuracy, speed of convergence, and communication and computation energy and latency on the architectures of FL, edge (clustered) FL, and HFL. There are other works that are also worth mentioning, such as the communication-efficient and privacy-preserved client-edge association policy implemented in [48], the context aware client-selection and client-edge association [49], and the communication-efficient semi-asynchronous model aggregation with gradient information maximization in [50].

As will be shown in chapter 4, the main goal of this thesis is to perform user-edge association and resource allocation that yields minimum communication energy consumption by IoT devices during HFL iterations while maintaining good learning performance. As was seen in the literature, there are three main paths to tackle this problem: 1) enhance the training experience such that the learning converges faster, so less communication rounds are needed and less total communication energy is consumed by the IoT devices, 2) take a greedy approach by minimizing the communication energy consumed per one communication round, however, that may lead to more communication rounds required to reach convergence and more consumed energy, or 3) find a good trade-off between the previous two solutions and merge them into one hybrid solution.

CHAPTER 3: FL & HFL COMPARISON: INTRUSION DETECTION SYSTEM

In this chapter, a comparison between the performances of FL and HFL will be conducted at different iid and non-iid data in terms of training loss, testing accuracy, and their speed of convergence. A distributed IDS will be implemented using the NSL-KDD dataset as a use case for this comparison. The work in this chapter was successfully submitted and accepted as a conference paper [16].

System Model

The system model seen in Figure 3.1 shows the configurations of FL and HFL that will be used to implement a distributed IDS. In the FL configuration, there are M mobile (IoT) devices with each device containing a subset of a specific attack's samples. These IoT devices use their local data to train local models and send them to the cloud after a specific number of training epochs. The cloud then aggregates the received models and sends them back to the mobile users. As for the HFL architecture, a layer of N edge nodes is added between the clients and the cloud, where these edges perform intermediate edge-level aggregation for a certain number of client-edge communication rounds before sending the edge-aggregated models to the cloud. As stated in the beginning of this chapter, the goal here is to study how FL and HFL handle the effect of non-iid data to reach high intrusion detection accuracy and fast convergence. Therefore, we assign clients with edges based on their data distribution only and do not consider any physical constraints such as the channel state, client-edge and client-cloud distances, and available bandwidth. That is why we can see in Figure 3.1 that an IoT device from far left can be associated with edge 2 and an IoT device from far right can be associated with edge 1.

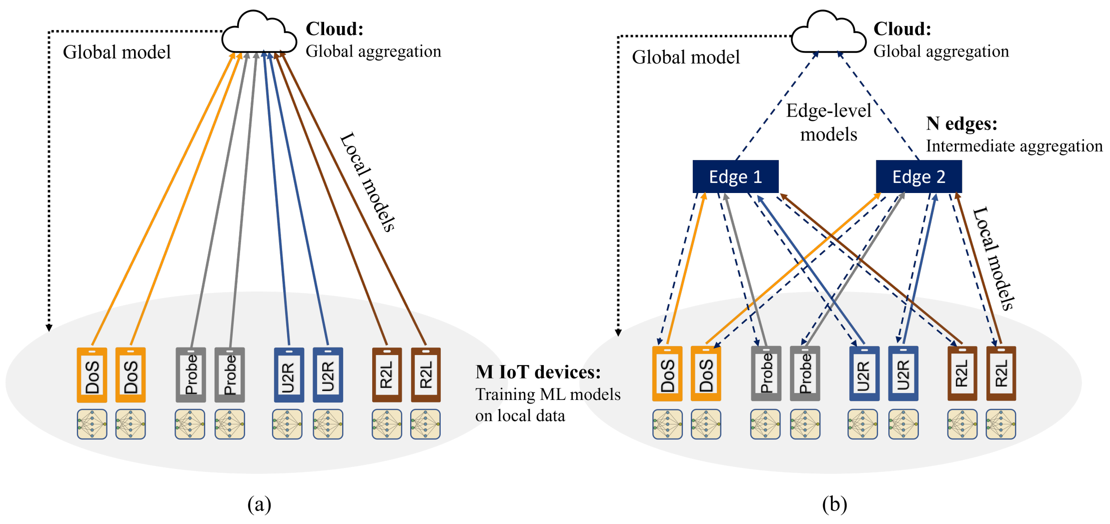


Figure 3.1. System model, and (a) FL and (b) HFL architectures.

Training Loss Estimation

During the learning process, the objective will be to minimize the training loss which shall result in maximizing the testing or intrusion detection accuracy. To estimate the training loss, we will be using the widely adopted cross entropy loss function. Since the NSL-KDD dataset consists of 5 classes, therefore, the loss function is calculated as a categorical cross entropy loss function.

Centralized Learning

In the case of CL, the clients share their data samples with and the training takes place at the main cloud. Therefore, the total loss is calculated as the summation of the losses at each sample [9]:

$$f_{CL}(w) = \frac{1}{|D|} \sum_{s=1}^{|D|} f_s(w) \tag{3.1}$$

$$f_s(w) = -\sum_{c=1}^{C} y_c \cdot \log(\hat{y_c}) + (1 - y_c) \cdot \log(1 - \hat{y_c}) \tag{3.2}$$

Where w represents the model weights, $|D|$ is the total number of samples, C is the number of classes, y_c is the true probability of class c for sample s (0 or 1), $\hat{y_c}$ is the predicted probability of class c for sample s, and $f_s(w)$ is the loss at data sample s.

Federated Learning

In the case of FL, the loss at each client is calculated as in (3.1), and just like the weights, the clients' losses are summed and weight-averaged at the main cloud. The loss function of FL can be mathematically represented as [9]:

At the cloud:

$$f_{FL}(w) = \sum_{i=1}^{M} \frac{|D_i|}{|D|} F_i(w) \tag{3.3}$$

At client i:

$$F_i(w) = \frac{1}{|D_i|} \sum_{s \in D_i} f_s(w) \tag{3.4}$$

Where M is the number of clients, $|D_i|$ is the number of samples given to client i, D_i is the set of the indexes of the samples given to client i, F_i is the loss at client i, and f_s is the same as (3.2).

Hierarchical Federated Learning

In the case of HFL, the loss at the cloud is just the weighted sum of the edges' losses. The HFL training loss is represented as:

$$f_{HFL}(w) = \sum_{j=1}^{N} \frac{|D_j|}{|D|} F_j(w) \tag{3.5}$$

Where N is the number of edge nodes, $|D_j|$ is the number of samples given to edge j, and $F_j(w)$ is the loss calculated at edge j in exactly the same way as (3).

IDS Settings

In this section, the characteristics of the dataset and neural network used are presented.

NSL-KDD dataset

The NSL-KDD has been very widely used in IDS researches previously [24]. That is due to its sufficiency in terms of number of data records and attack types diversity. It contains 125,973 training samples and 22,544 testing samples. Each of the 148,517 samples consist of 41 features. The dataset contains normal and abnormal classes. The training dataset contains 39 attacks categorized to Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L) and Probe attacks. The testing set, besides the training set attacks, contains 17 extra attacks to assess the flexibility of the model in detecting unseen-before attacks. In order to understand the upcoming performance evaluation study cases, it is important to highlight the percentage distribution of the classes in the training and testing sets, which are shown in Figure 3.2.

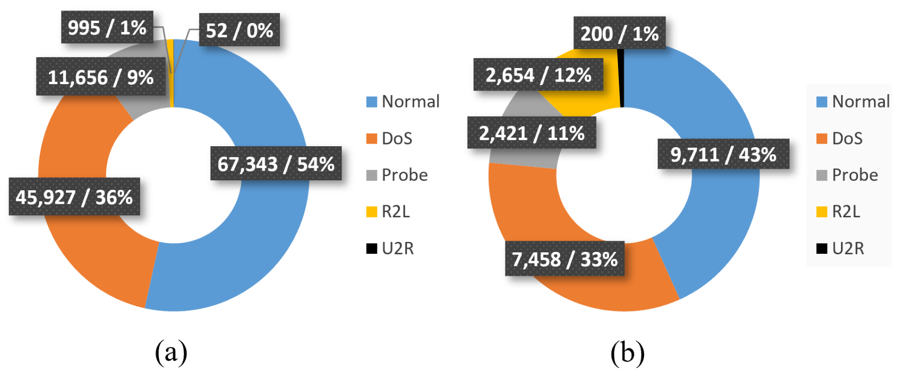


Figure 3.2. Class distribution of the NSL-KDD (a) training and (b) testing sets

Since the neural network (NN) is a mathematical model, it can only be fed with numbers. Therefore, one-hot encoding [51] is used to convert the non-numeric features, which are protocol type, service, and flags, into numeric ones. The one-hot encoding results in each data sample having 122 features. Then, the values of features are normalized to be between 0 and 1 to avoid any inconsistency. Based on the number of features, our NN, Figure 3.3, is composed of 122-neuron input layer, and two hidden layers of 80 and 40 neurons, respectively, and 5-neuron output layer. It also performs neuron-dropout of 30% after each of the hidden layers, and it uses ReLU as an activation function.

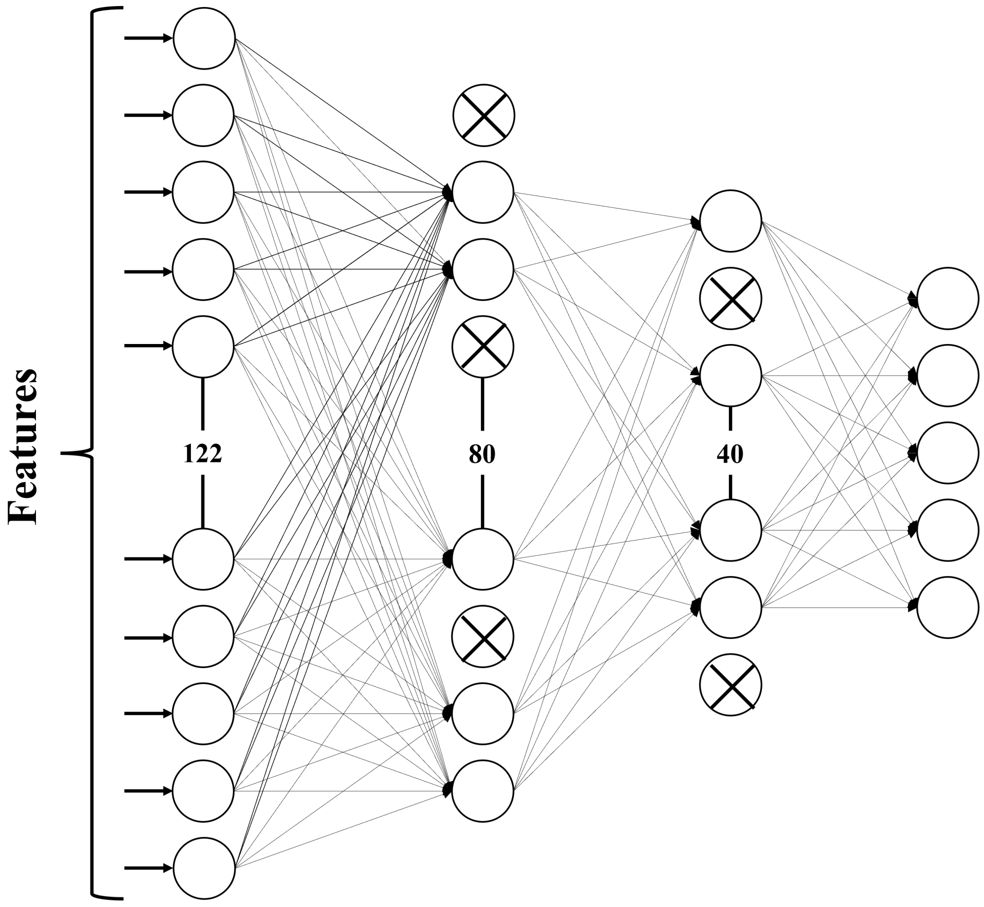


Figure 3.3. The used neural network diagram

Performance Evaluation

In this section, the FL and HFL systems will be evaluated. The number of learning epochs per communication round is constant and set to 5 for the client-cloud layer in FL and edge-cloud layer in HFL. However, in the client-edge layer in HFL, FedSGD is implemented, which is equivalent to 1 learning epoch per communication round, i.e., the clients send their local models to the edge following every single training epoch. Different study cases of training data distribution will be evaluated as follows:

Study Case A: iid Client-Edge Association

In this case, the FL scenario consists of 8 clients, where every 2 clients share one attack class samples in half, i.e., each one of client 1 and client 2 has one half of the DoS attack samples, each one of client 3 and client 4 has one half of the Probe attack samples, and so on. The normal samples are divided among the clients based on the portion of attack samples that they have, e.g., client 1 has one half of the DoS attack samples, which is around 39% of all the attack samples, therefore it gets 39% of the normal samples. This strategy of the normal samples' distribution applies to cases B. and C.

In the HFL scenario, two edges are put in the middle, and each edge gets four clients that have completely non-iid data. For example, edge1 gets client 1 (Dos attack), client 3 (Probe attack), client 5 (R2L attack), and client 7 (U2R attack), and edge 2 gets the other four clients. As a result, the two edges have iid data and can see all types of attacks.

The FL scenario suffers from the non-iid distribution and the inconsistency in clients' training data. That fact, as seen in Figure 3.4, affected the training loss, testing accuracy, and the speed of convergence. Whereas in the HFL scenario, the iid client-edge assignment enhanced the performance in terms of all of the three metrices. This is highlighted through showing the loss which is almost negligible, the detection accuracy which is better than FL with 3-4% difference, and the convergence state which is reached around 19 rounds faster than FL as illustrated by the black line.

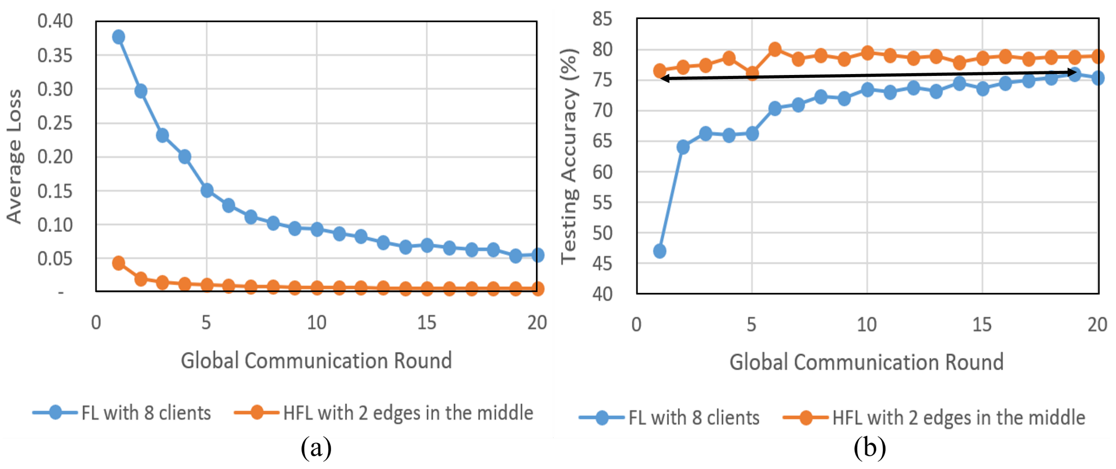


Figure 3.4. (a) Average loss and (b) testing accuracy for study case A.

Study case B: iid Client-Edge Association with Equal Number of Attacks' Samples

In this case, the FL scenario also consists of 8 clients. Each one of client 1 and client 2 has 500 randomly chosen DoS samples and 500 randomly chosen normal samples (each client has its own distinct set of samples). Each one of client 3 and client 4 has 500 randomly chosen Probe samples and 500 randomly chosen normal samples, and so on for the other two attacks and four clients. Of course, as seen in Figure 3.2, U2R has only 52 samples, therefore, its samples were naively duplicated until they reached a total of 1000 samples. Although this approach is not efficient, but it was done only to facilitate the FL comparison in this study case.

For the HFL scenario, same as study case A, two edges are put in the middle, and each edge gets four clients that have completely non-iid data. As a result, each edge aggregates the model for all classes, which helps speed up the convergence.

Figure 3.5 shows that the FL scenario performed poorly in terms of maximum testing accuracy with 6% drop compared to HFL, and in terms of the speed of convergence which is around 14 rounds slower than HFL as illustrated by the black line. On the other hand, the HFL scenario started with high testing accuracy and reached near the convergence state from the first communication round. One thing to notice is that the loss is almost equal in both scenarios.

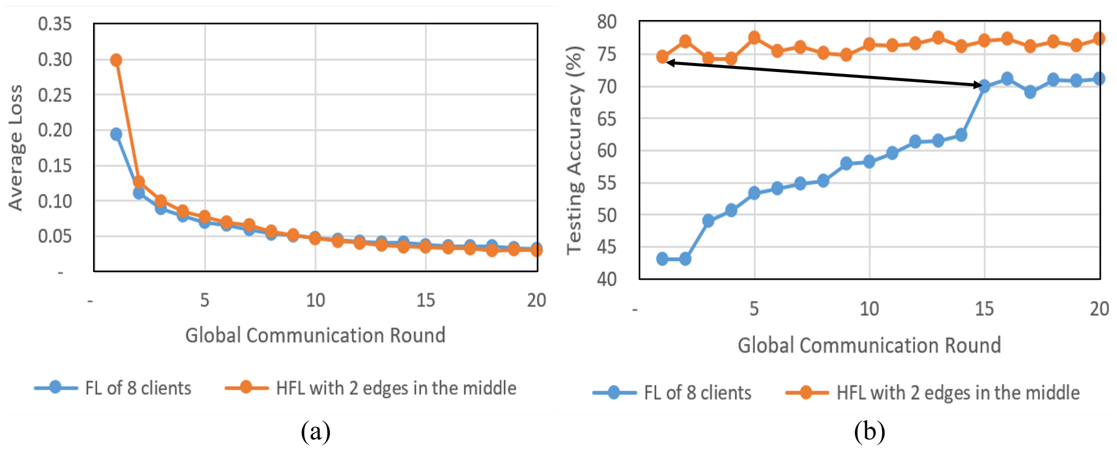


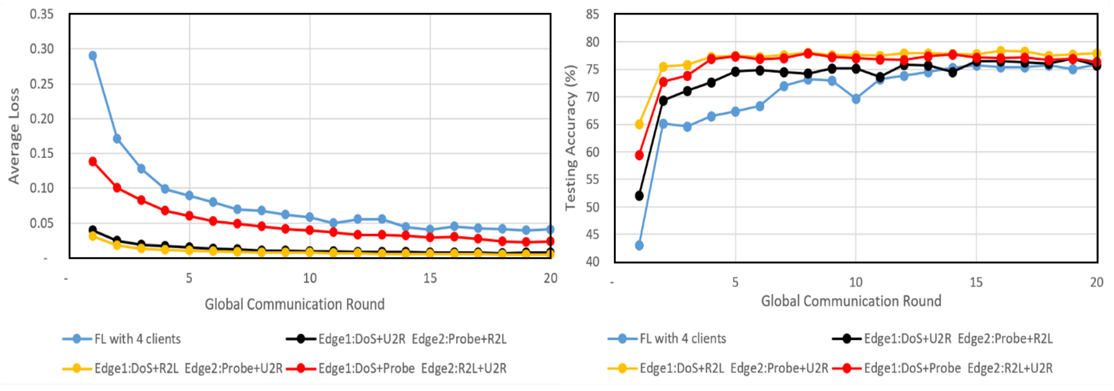
Figure 3.5. (a) Average loss and (b) testing accuracy for study case B.

Study Case C: Non-iid Client-Edge Association

In this study case, the FL scenario consists of 4 clients, each having all the samples corresponding to one specific attack and the corresponding portion of normal samples. For example, client 1 has all the 45,927 samples of the DoS attack (78% of all the attacks' samples), therefore it has 78% of the normal samples.

Regarding the HFL scenario, two edges are added in the middle between the clients and the cloud. The HFL scenario is repeated three times: i) where edge 1 has DoS + Probe attacks and edge 2 has the others, ii) where edge 1 has DoS + R2L attacks and edge 2 has the others, and iii) where edge 1 has DoS + U2R attacks and edge 2 has the other classes of attacks.

From Figure 3.6, it can be seen that even if the edges end up aggregating models from clients with non-iid data, they can still perform better than FL with non-iid data, whether in terms of training loss, testing accuracy (slightly better), or speed of convergence. It seems like the edge layer tends to absorb some of the effect of the non-iid distribution before it sends the aggregated model to the main cloud.



(a) (b)

Figure 3.6. (a) Average loss and (b) testing accuracy for study case C.

CHAPTER 4: RL-ASSISTED ENERGY-AWARE HFL USER-EDGE ASSOCIATION

In this chapter, different user-edge association policies in HFL are implemented and compared. Also, the effect of the trade-off between minimizing the per-round communication energy consumption and KLD of the data distribution on the total energy consumption is studied. The work in this chapter was submitted and is got accepted as a conference paper [52].

System model & Problem Formulation

In the system model highlighted in Figure 4.1, there are M mobile (IoT) devices distributed in an area containing N edge nodes with a main cloud controlling the network. To deploy HFL scheme in this case, before the learning process starts, the cloud associates each edge j with $|M_j|$ clients and distributes the available bandwidth B_j^{max} on each edge j on its corresponding clients. Then, the KLD of the virtual dataset D_j on each edge j, which indicates how close the class distribution on each edge is from the uniform distribution Q, is calculated as $KLD(P_j||Q) = \sum_{c=1}^{C} P_j(c) \log \frac{P_j(c)}{Q(c)}$, where $P_j(c)$ is the probability distribution of class c samples in D_j, and C is the total number of classes in the dataset. After a pre-determined number of local training epochs, each client i uploads its local model weights to the associated edge j with communication latency of $t_{ij}^{comm} = W/\rho_{ij}$, where W is the size of the model weights which is constant for all clients, and ρ_{ij} is the upload transmission rate from client i to edge j. The communication energy consumed during model uploading by each client i in one communication round is calculated as [45]:

$$e_{ij}^{comm} = \frac{W \cdot N_0 \cdot B_{ij}}{\rho_{ij} \cdot g_{ij}} (2^{\rho_{ij}/B_{ij}} - 1) \tag{4.1}$$

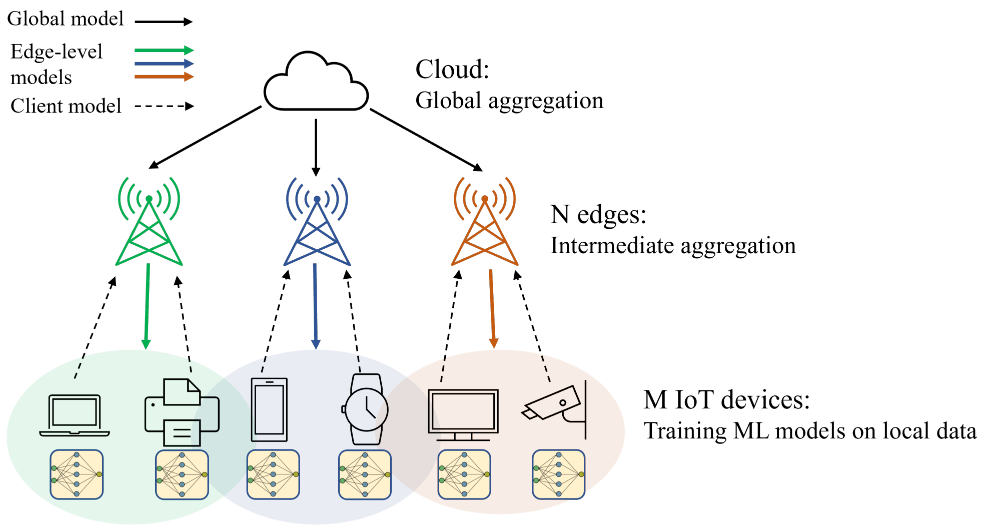


Figure 4.1. System model and HFL architecture.

With N_0 being the noise spectral density, B_{ij} the bandwidth allocated from edge j to client i, and g_{ij} the channel gain between client i and edge j. Reaching to this end, we formulate an optimization problem that tries to minimize the total communication energy consumed by the IoT devices during model uploading, while respecting the transmission time limit, and the KLD limit on all edges:

$$\textbf{P1:} \quad \min_{\lambda_{ij}, B_{ij}} \quad k_1 k_2 \sum_{i=1}^{M} \sum_{j=1}^{N} \lambda_{ij} e_{ij} \tag{4.2}$$

$$\textbf{s.t.} \quad KLD(P_j||Q) < KLD_{max}, \quad \forall j \in N \tag{4.3}$$

$$\sum_{j=1}^{N} \lambda_{ij} t_{ij}^{comm} \leq t^{max}, \quad \forall i \in M \tag{4.4}$$

$$\sum_{i=1}^{M} \lambda_{ij} B_{ij} \leq B_j^{max}, \quad \forall j \in N \tag{4.5}$$

$$\sum_{j=1}^{N} \lambda_{ij} = 1, \quad \forall i \in M \tag{4.6}$$

$$\lambda_{ij} = \{0, 1\}, \quad \forall i \in M \,\&\, \forall j \in N \tag{4.7}$$

Where λ_{ij} being 1 means client i is associated with edge j and 0 otherwise, and k_1 and k_2 are the number of client-edge communication rounds per one global round, and the number of edge-cloud rounds before reaching convergence, respectively.

In **P1**, k_1 and k_2 are directly related to the speed of the learning convergence, hence, they are not deterministic and their values cannot be known precisely before or during the training. However, they can be reduced by balancing the distribution of data classes over the edge nodes [53]. Therefore, constraint (4.3) makes sure that the client-edge association will result in no edge having a KLD greater than KLD_{max}. Constraint (4.4) ensures eliminating the effect of straggler clients i.e., guarantees that all clients will transmit their models in a time interval not longer than t^{max}. Constraints (4.5) and (4.6) make sure that the allocated bandwidths do not exceed the available bandwidths on the edges, and each client is associated with one edge only, respectively.

Proposed Solutions

In this section, we try to solve the optimization problem in **P1** using two approaches, namely, 1) relaxation-based approach, and 2) reinforcement learning-based approach.

Relaxation-based Approach

As seen in (4.7) and in the objective function of the optimization problem, **P1** is a mixed-integer nonlinear programming problem (MINLP) which is NP-hard [54]. To reduce its complexity, we relax the association variable λ_{ij} as in (4.9) and split **P1** into two subproblems to facilitate the decentralization of the solution:

$$\textbf{SP1:} \quad \min_{\lambda_{ij}} \quad k_1 k_2 \sum_{i=1}^{M} \sum_{j=1}^{N} \lambda_{ij} e_{ij} \tag{4.8}$$

s.t. (4.3), (4.6), and

$$\lambda_{ij} = [0, 1], \quad \forall i \in M \,\&\, \forall j \in N \tag{4.9}$$

$$\textbf{SP2:} \quad \min_{B_{ij}} \quad k_1 k_2 \sum_{i=1}^{M} \sum_{j=1}^{N} \lambda_{ij} e_{ij} \tag{4.10}$$

s.t. (4.4) and (4.5).

Where **SP1** is solved on the cloud, which is responsible for client-edge association where the bandwidth is treated as a constant. Once **SP1** is solved and λ_{ij} is de-relaxed i.e., the optimal associations are found, **SP2** performs the bandwidth allocation on the edge node.

Reinforcement Learning-based Approach

As pointed out earlier, **P1** is a NP-hard MINLP problem with combinatorial complexity. Although, **SP1** and **SP2** in the previous section can be solved using different optimization tools, however, if any change in the network happened during training, such as if a client moved from the range of one edge to another, **SP1** and **SP2** will have to be re-solved. Resolving **SP1** and **SP2** with every change in the IoT devices' locations alongside the NP-hardness and the combinatorial complexity of the optimization problem would be highly non-practical, especially in highly dynamic environments. Therefore, to propose a more complexity-efficient solution, we deploy an RL agent to solve **P1** from another perspective. The RL agent in our case will try to learn the mobility patterns of the IoT devices systematically. Hence, in case of any movement of IoT devices, the RL agent, using the pre-trained model, will instantly make the new client-edge association and bandwidth allocation decisions, whilst the relaxation-based would have to resolve the highly-complex **SP1** and **SP2** all over again.

State: In the RL environment, as seen in Figure 4.2, we assume that the channel state information (CSI) between all the edges and clients are available all the time. Hence, the state that is fed to the RL agent is a matrix representing the channel gain values between all edges and clients.

Action: The RL agent returns the values of λ_{ij} and B_{ij} in the range [0,1] i.e., performs client-edge association and bandwidth allocation at the same instance.

Reward: The reward indicates how close the current action is from the optimal action. To elaborate on how the reward is calculated, using the CSI and initial values of the allocated bandwidths, a matrix of communication energies between each pair of client i and edge j is generated. Then, we perform the association that yields the best (least) possible total energy and that which yields the worst (highest) possible total energy. Once the action is taken by the agent, we assess how good it is based on the

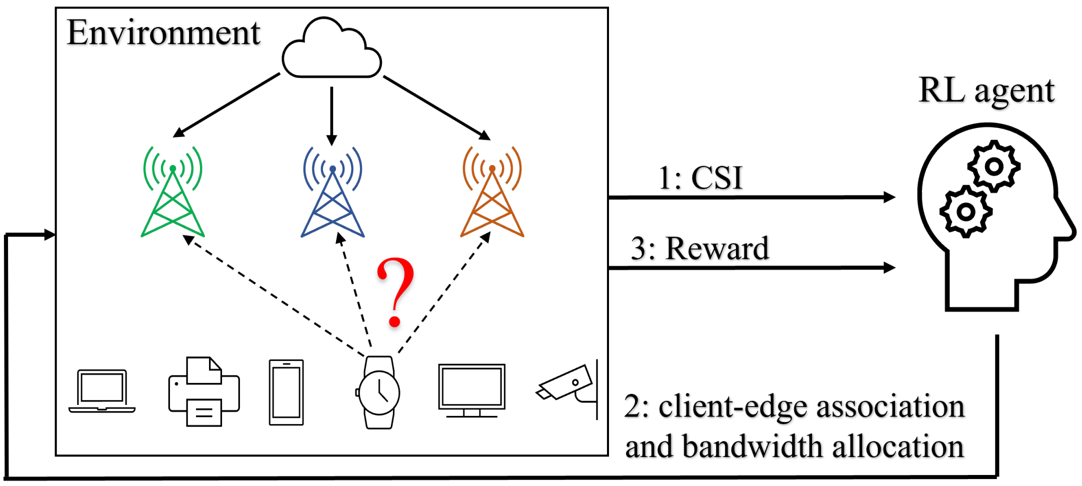


Figure 4.2. RL-based solution overview.

following reward function:

$$r(action) = \begin{cases} -1 & (4.3)\ or\ (4.4)\ is\ violated \\ -2 & (4.3)\ and\ (4.4)\ are\ violated \\ 1 - \dfrac{action\ energy - best\ energy}{worst\ energy - best\ energy} & no\ constraint\ is\ violated \end{cases} \tag{4.11}$$

Where constraint (4.5) is always met by treating the bandwidth action as the bandwidth fraction of each client and multiplying it by the maximum bandwidth of the associated edge, constraint (4.6) is managed by the relaxation and de-relaxation steps, and (4.9) is always true due to the fact that the RL agent always returns the action in the range [0,1].

After clients are associated with edges and bandwidths are allocated, the HFL process starts with the objective of minimizing (3.5).

Performance Evaluation

In this section, we compare the performances of the relaxation-based, RL-based, least-energy, distance-based, and best-KLD-below-distance-threshold (best-KLD) associations. In the least-energy association, we generate the energy matrix and perform the association that leads to the minimum communication energy per one round. The distance-based policy associates each client with its nearest edge. Some clients may exist in the region of two edge nodes, therefore, the best-KLD policy tries all the different association possibilities of these clients and chooses the association that yields the minimum average KLDs of the edge nodes. Assuming a client can be in the region of one or two edges only, the complexity of best-KLD algorithm is $O(2^{No.\ of\ overlapping\ clients})$.

Simulation settings

We study two HFL environments, a small-scaled one with 10 clients and 3 edges, and a medium-scaled environment with 40 clients and 5 edges. The clients are distributed such that for each client the closest edge to it is $min_{distance} + 2 \times min_{distance} \times uniform(0, 1)$ meters away, the second nearest edge is **2**$\times min_{distance} + 2 \times min_{distance} \times uniform(0, 1)$ meters away, the third edge is **3** $\times\ min_{distance} + 2 \times min_{distance} \times uniform(0, 1)$ meters away, and so on, where $min_{distance}$ is a hyperparameter that we vary throughout the experiments to control the size of the HFL environment. By this distribution, if the distance threshold in the best-KLD algorithm is set as $2 \times min_{distance}$, which is actually the case in our experiments, then each client will have a probability of 0.5 being in the range of one edge only and 0.5 of being in the range of two different edges.

The widely experimented handwritten digits classification dataset MNIST [55] is used to evaluate the HFL performance. It consists of 60000 training samples and 10000 testing samples distributed on 10 handwritten digits (labels). The data samples are distributed among the clients in two different strategies, the first one is a non-iid distribution strategy where each client is randomly assigned samples from two, three, or four classes, and the second is iid distribution where all clients have the same portions from all classes.

RL Training

The RL-based solution is only applied in this paper to the small-scaled problem with the state matrix being of size 3x10, and the action matrix of size 2x3x10. If we try to train the RL agent on the medium-scaled environment, the state matrix would become of size 5x40 and the action matrix would contain 2x5x40 elements, which is a relatively large problem that needs a very long time to converge, or it may never converge unless deep actor and critic networks are used. One possible solution that can reduce the size of the problem and make it feasible, is to perform the edge association and bandwidth allocation row by row i.e., each client at time, as was done in [56]. The RL agent was trained on the small-scaled problem on different $min_{distance}$ values and on both iid and non-iid data distributions. The actor neural network used has an input layer of size 30 neurons, hidden layer of 512 neurons, and output layer of size 60 units. The critic neural network has 30 neurons as input layer, three hidden layers of size 128, 128, and 64, respectively, and 1 output neuron. RL training was run for 1500 episodes with 100 iteration per episode. The first 100 episodes were dedicated for full policy exploration. Figure 4.3 shows the convergence of the average reward per episode on selected $min_{distance}$ values on the non-iid distribution case. The reward

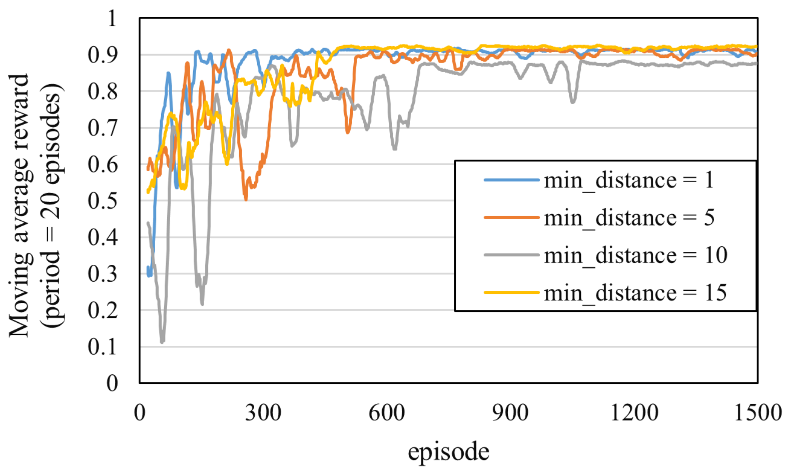


Figure 4.3. RL reward convergence for selected $min_{distance}$ values on the non-iid distribution.

functions are converging in the range of 0.85-0.93, so we expect from the RL agent to give client-edge association that is near-optimal in terms of one round communication energy consumption in most of the cases.

Energy Consumption and HFL performance

In this subsection, we compare the performance of the five client-edge association policies in terms of communication energy consumption per one round, the average KLD of the edges, and the total communication energy consumption after convergence is reached. Figure 4.4 and Figure 4.5 show that the relaxation-based and the least-energy policies have very similar performance and yield the lowest energy consumption per communication round, however, that comes with an average KLD that is relatively high compared to the other policies. As its name states, the best-KLD policy has the lowest average KLD, but it results in having the highest energy per round alongside the RL-based. The distance-based seems to be the most balanced policy with a good trade-off between the energy consumption and the average KLD, especially in the small-scaled environment. On the contrary, the RL-based is the most unstable policy with its energy consumption per round and average KLD keeping fluctuating as the size of the HFL environment increases. Figure 4.6 compares the five policies in the case of iid clients' data distribution, where no matter what client-edge association we have, the average KLD will always be 0, which will lead to very similar testing accuracy and convergence speed. Therefore, in the case of iid, it is best to deploy the relaxation-based or the least-energy association policy. Also, we can notice that the best-KLD policy acts exactly the same as the distance-based policy in the case of iid distribution.

To study the effect of the average KLD value, we fix $min_{distance}$ to be 15 meters and run the policies multiple times on different channel states. The evaluation is done on the small-scaled environment with non-iid distribution. Figure 4.7 shows that the lower the average KLD, the higher the final testing accuracy is and the faster the convergence is reached. This can be clearly seen as the best-KLD with average KLD of 0.22 is dominating the learning performance with a great competition against the distance-based policy that has average KLD of 0.37. We can also see that the other policies that have higher average KLD consumed more communication rounds to converge, and they never exceeded 95% testing accuracy. Finally, Figure 4.8 illustrates the trade-off between the communication energy consumption per round and the average KLD in determining the amount of total communication energy consumed by the IoT devices to reach convergence. The RL-based yielded an energy per round that is close to 21 μJ, which is slightly less than the best-KLD and the distance-based policies, however, its 0.54 average KLD caused it to converge very slowly which made it consume significantly more total energy compared to the other policies. The relaxation-based and the least-energy policies have a per-round-energy consumption that is almost half of those in

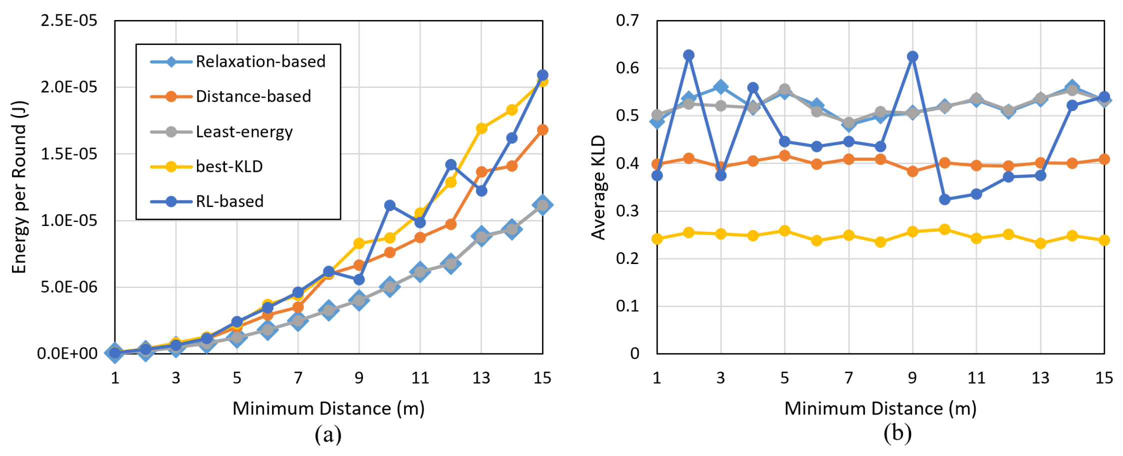


Figure 4.4. (a) Total communication energy consumption per round, and (b) average edge KLD of the small-scaled environment with non-iid client data.

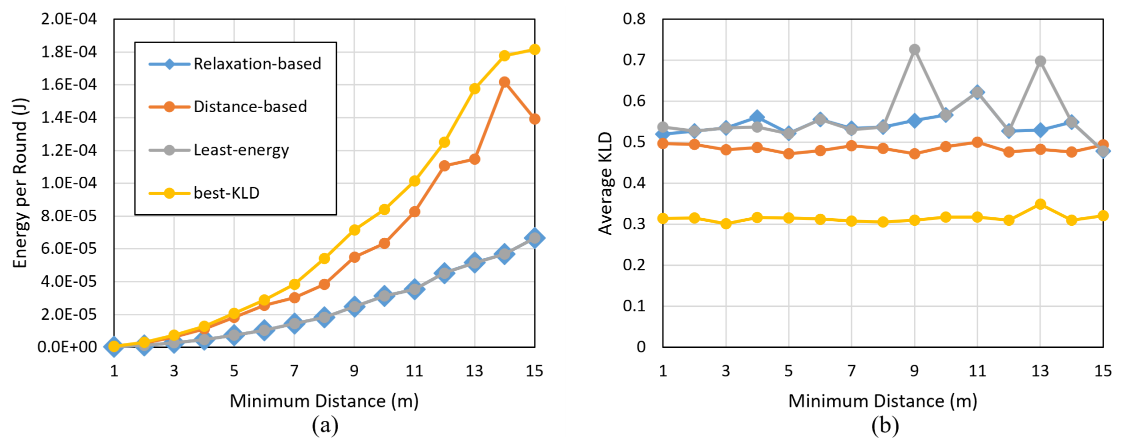


Figure 4.5. (a) Total communication energy consumption per round, and (b) average edge KLD of the medium-scaled environment with non-iid client data.

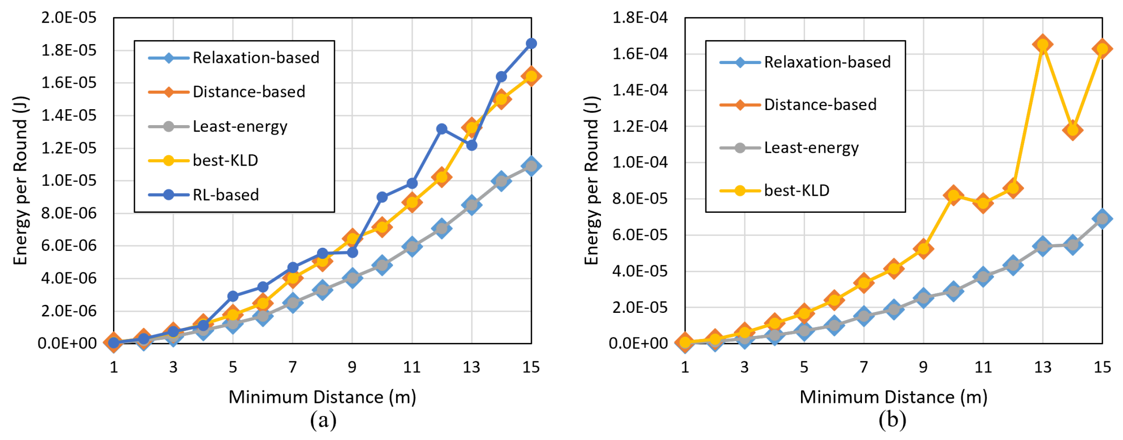


Figure 4.6. Total communication energy consumption per round for the (a) small-scaled and (b) medium-scaled environments with iid client data.

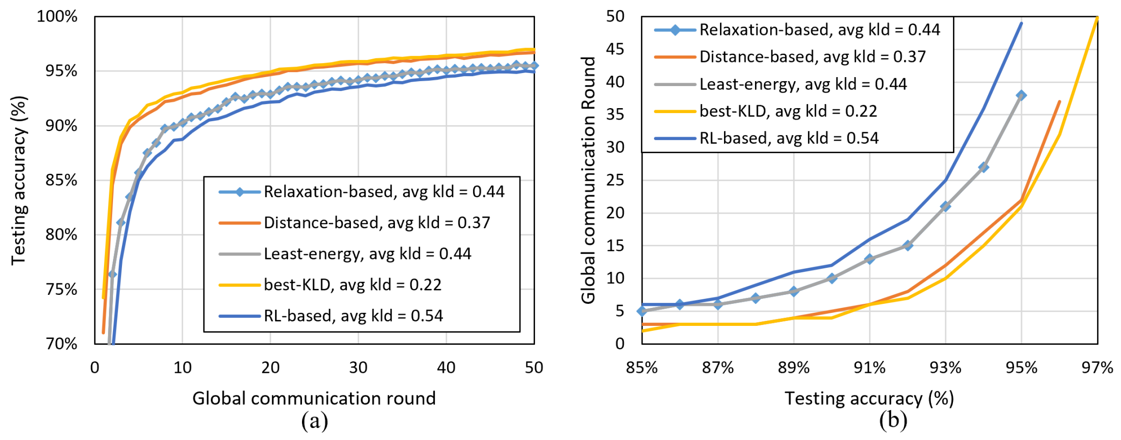


Figure 4.7. (a) The effect of average KLD value on the testing accuracy, and (b) the number of communication rounds required to reach target accuracy.

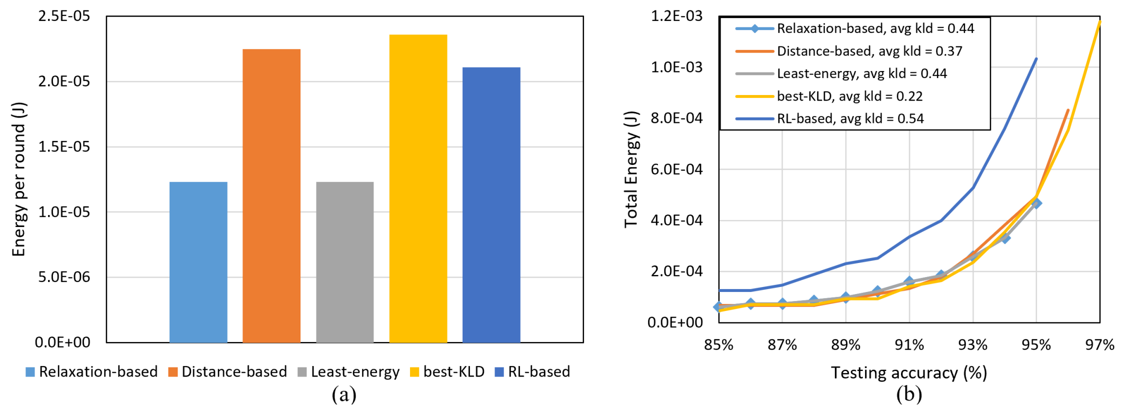


Figure 4.8. (a) Communication energy consumption per one round, and (b) the total communication energy consumed to reach target accuracy.

the best-KLD and distance-based policies, however, the low average KLDs of the latter two guaranteed a very competitive total energy consumption against the former ones until reaching 95% accuracy. The distance-based and best-KLD managed to reach an accuracy of 96% and 97%, respectively, but, that came with burden of doubling the total energy consumption, which is not worthy unless the application that they are deployed in gives way higher importance to the testing accuracy over the energy consumption.

CHAPTER 5: CONCLUSION

In this thesis, a thorough investigation of the characteristics of the HFL scheme in the IoT networks was conducted. First, in chapter 1, an overview on the current and future expected state of the IoT devices in terms of their general properties, applications, deployment rates, and major challenges was provided. Then, lights were shed on the big amounts of data being gathered and processed in IoT systems and how they can be handled and used in training and building artificially intelligent event-detection and decision-making models. For that, in chapter 2, a detailed theoretical study on CL, FL, and HFL schemes and the discrepancies in their data privacy preservation ability, communication and computation efficiency, energy consumption, and machine learning performance was conducted.

In chapter 3, we point at the necessity of paying attention to and proactively taking countermeasures towards the security vulnerabilities in IoT environments by implementing an IDS using the NSL-KDD dataset. Also, a comparison based on performance evaluation between FL and HFL in terms of training loss, testing accuracy, and speed of convergence is conducted. HFL proved its efficiency over FL in two iid client-edge association study cases and one non-iid client-edge study case.

In chapter 4, energy-efficient client-edge association and resource allocation in HFL environment were performed. An IoT communication energy minimization problem that takes into consideration the data distribution and model transmission latency was formulated and solved using a relaxation technique. Another complexity-efficient solution that uses reinforcement learning was implemented to provide a fast client-edge association and bandwidth allocation response to the continuous movement of the IoT devices. These solutions were compared to different state-of-the-art client-edge association policies. Moreover, the trade-off between minimizing the per-round energy consumption and KLD of the data distribution on the total energy consumption was studied.

CHAPTER 6: FUTURE WORK

Several aspects in this thesis can be further improved in the future. Some of the possible enhancements include:

- It can be clearly seen that the most challenging part in the IDS implementation in chapter 3 was how to deal with highly-skewed non-iid NSL-KDD dataset. Therefore, instead of naively duplicating the data samples to balance the dataset, using generative adversarial networks (GAN) would be more convenient. Also, it would be interesting to combine GAN models with the models at the node/edge levels, and aggregate the GAN effect using federated averaging.

- The optimization problem in chapter 4 only considers the energy consumption and latency caused by the model communication. However, the amount of processing and computations performed by the IoT devices during model training is not negligible and can consume long time and high energy. Therefore, adding the computational latency and energy consumption to the optimization problem is necessary for more efficient IoT devices' energy minimization.

- Our optimization problem only performs user-edge association and bandwidth allocation. However, there are other resources that can be efficiently allocated. For example, the transmission power, and the CPU frequency (if the computational energy is added to the optimization problem).

- Find efficient solutions that can leverage our RL framework to adapt with and work in medium- and large-scaled HFL environments.

- In this work, we assumed that a client can be associated with only one edge at a time. However, this is not the case in today's advanced wireless technologies and 5G networks. Therefore, the concept of multi-homing can be studied and deployed in future versions of this work, where a client can be associated with more than one edge. Clients sharing their models with multiple edges can result in better learning and faster convergence.

REFERENCES

[1] Cisco, “Prepare to succeed with the Internet of Things,” Tech. Rep., 2017.

[2] P. Sethi and S. R. Sarangi, “Internet of Things: Architectures, Protocols, and Applications,” *Journal of Electrical and Computer Engineering*, vol. 2017, 2017, ISSN: 20900155. DOI: 10.1155/2017/9324035. [Online]. Available: https://www.hindawi.com/journals/jece/2017/9324035/.

[3] J. Ding, M. Nemati, C. Ranaweera, and J. Choi, “IoT connectivity technologies and applications: A survey,” *IEEE Access*, vol. 8, pp. 67 646–67 673, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2985932. arXiv: 2002.12646.

[4] S. Nižetić, P. Šolić, D. L.-I. González-de-Artaza, and L. Patrono, “Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future,” *Journal of Cleaner Production*, vol. 274, p. 122 877, Nov. 2020, ISSN: 0959-6526. DOI: 10.1016/J.JCLEPRO.2020.122877.

[5] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, “IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019, ISSN: 23274662. DOI: 10.1109/JIOT.2019.2935189.

[6] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, “A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020. DOI: 10.1109/COMST.2020.2988293.

[7] E. Baccour, N. Mhaisen, A. A. Abdellatif, *et al.*, "Pervasive AI for IoT Applications: Resource-efficient Distributed Artificial Intelligence," May 2021. arXiv: `2105.01798`. [Online]. Available: `http://arxiv.org/abs/2105.01798`.

[8] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017, ISSN: 00189162. DOI: `10.1109/MC.2017.9`.

[9] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, PMLR, Feb. 2016, ISBN: 1602.05629v3. arXiv: `1602.05629`. [Online]. Available: `https://arxiv.org/abs/1602.05629v3`.

[10] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019, ISSN: 21693536. DOI: `10.1109/ACCESS.2019.2924045`.

[11] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (IoT) security: Current status, challenges and prospective measures," in *2015 10th International Conference for Internet Technology and Secured Transactions, ICITST*, Institute of Electrical and Electronics Engineers Inc., Feb. 2015, pp. 336–341, ISBN: 9781908320520. DOI: `10.1109/ICITST.2015.7412116`.

[12] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, "Internet of Things (IoT) Communication Protocols : Review," in *2017 8th International Conference on Information Technology (ICIT)*, 2017, pp. 685–690, ISBN: 9781509063321. DOI: `10.1109/ICITECH.2017.8079928`.

[13] S. Abdul Rahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A Survey on Federated Learning: The Journey from Centralized to Distributed On-Site Learning and Beyond," *IEEE Internet of Things Journal*, 2020, ISSN: 2327-4662. DOI: 10.1109/jiot.2020.3030072.

[14] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-Edge-Cloud Hierarchical Federated Learning," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, ISBN: 9781728150895. DOI: 10.1109/ICC40277.2020.9148862. arXiv: 1905.06641.

[15] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," Dec. 2018. DOI: 10.48550/arxiv.1812.06127. arXiv: 1812.06127. [Online]. Available: https://arxiv.org/abs/1812.06127v5.

[16] H. Saadat, A. Aboumadi, A. Mohamed, A. Erbad, and M. Guizani, "Hierarchical Federated Learning for Collaborative IDS in IoT Applications," in *2021 10th Mediterranean Conference on Embedded Computing, MECO 2021*, Institute of Electrical and Electronics Engineers Inc., Jun. 2021, ISBN: 9780738133614. DOI: 10.1109/MECO52532.2021.9460304.

[17] UNB, *Intrusion Detection Evaluation Dataset (CIC-IDS2017)*. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html (visited on 05/18/2022).

[18] ——, *CSE-CIC-IDS2018 on AWS*. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html (visited on 05/18/2022).

[19] ——, *NSL-KDD dataset*. [Online]. Available: `https://www.unb.ca/cic/datasets/nsl.html` (visited on 02/26/2021).

[20] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*, vol. 2018-January, pp. 108–116, 2018. DOI: `10.5220/0006639801080116`.

[21] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset," *IEEE Access*, vol. 8, pp. 32 150–32 162, 2020, ISSN: 21693536. DOI: `10.1109/ACCESS.2020.2973219`.

[22] P. S. Bhattacharjee, A. K. M. Fujail, and S. A. Begum, "A Comparison of Intrusion Detection by K-Means and Fuzzy C-Means Clustering Algorithm Over the NSL-KDD Dataset," in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, IEEE, 2017, ISBN: 9781509066209. DOI: `10.1109/ICCIC.2017.8524401`.

[23] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019, ISSN: 21693536. DOI: `10.1109/ACCESS.2019.2895334`.

[24] R. Thomas and D. Pavithran, "A Survey of Intrusion Detection Models based on NSL-KDD Data Set," in *2018 Fifth HCT Information Technology Trends (ITT)*, IEEE, 2018, pp. 286–291, ISBN: 9781538671467. DOI: `10.1109/CTIT.2018.8649498`.

[25] A. M. Mahfouz, D. Venugopal, and S. G. Shiva, "Comparative Analysis of ML Classifiers for Network Intrusion Detection," in *Fourth International Congress on Information and Communication Technology. Advances in Intelligent Systems and Computing*, vol. 1027, 2020, ISBN: 9789813293427. DOI: 10.1007/978-981-32-9343-4_16.

[26] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber-Physical Systems," *IEEE Transactions on Industrial Informatics*, 2020, ISSN: 1551-3203. DOI: 10.1109/tii.2020.3023430.

[27] S. Abdul Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning?" *IEEE Network*, pp. 1–8, 2020, ISSN: 1558156X. DOI: 10.1109/MNET.011.2000286.

[28] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, "Intrusion Detection for Wireless Edge Networks Based on Federated Learning," *IEEE Access*, vol. 8, pp. 217 463–217 472, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3041793.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. 2018, pp. 1–3, ISBN: 9780262039246. [Online]. Available: https://mitpress.mit.edu/books/reinforcement-learning-second-edition%20http://www.incompleteideas.net/book/the-book-2nd.html.

[30] S. Liu, J. Yu, X. Deng, and S. Wan, "FedCPF: An Efficient-Communication Federated Learning Approach for Vehicular Edge Computing in 6G Communication Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23,

no. 2, pp. 1616–1629, Feb. 2021, ISSN: 15580016. DOI: 10.1109/TITS.2021.3099368.

[31] Y. Sun, S. Zhou, and D. Gunduz, “Energy-Aware Analog Aggregation for Federated Learning with Redundant Data,” in *IEEE International Conference on Communications*, vol. 2020-June, Institute of Electrical and Electronics Engineers Inc., Jun. 2020, ISBN: 9781728150895. DOI: 10.1109/ICC40277.2020.9148853. arXiv: 1911.00188.

[32] L. U. Khan, M. Alsenwi, Z. Han, and C. S. Hong, “Self Organizing Federated Learning Over Wireless Networks: A Socially Aware Clustering Approach,” in *International Conference on Information Networking*, vol. 2020-Janua, IEEE Computer Society, Jan. 2020, pp. 453–458, ISBN: 9781728141985. DOI: 10.1109/ICOIN48656.2020.9016505.

[33] D. J. Han, M. Choi, J. Park, and J. Moon, “FedMes: Speeding up Federated Learning with Multiple Edge Servers,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3870–3885, Dec. 2021, ISSN: 15580008. DOI: 10.1109/JSAC.2021.3118422.

[34] R. Saha, S. Misra, and P. K. Deb, “FogFL: Fog-Assisted Federated Learning for Resource-Constrained IoT Devices,” *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8456–8463, May 2021, ISSN: 23274662. DOI: 10.1109/JIOT.2020.3046509.

[35] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, “From Federated to Fog Learning: Distributed Machine Learning over Heterogeneous Wireless Networks,” *IEEE Communications Magazine*, vol. 58, no. 12, pp. 41–

47, Dec. 2020, ISSN: 15581896. DOI: 10.1109/MCOM.001.2000410. arXiv: 2006.03594.

[36] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, ISBN: 10.1109/IJCNN48605.2020.9207469.

[37] N. Mhaisen, A. Awad, A. Mohamed, A. Erbad, and M. Guizani, "Optimal User-Edge Assignment in Hierarchical Federated Learning Based on Statistical Properties and Network Topology Constraints," *IEEE Transactions on Network Science and Engineering*, 2021, ISSN: 23274697. DOI: 10.1109/tnse.2021.3053588.

[38] C. Feng, H. H. Yang, D. Hu, *et al.*, *Mobility-Aware Cluster Federated Learning in Hierarchical Wireless Networks*, Aug. 2021. arXiv: 2108.09103. [Online]. Available: https://arxiv.org/abs/2108.09103v1.

[39] J. Tursunboev, Y. S. Kang, S. B. Huh, D. W. Lim, J. M. Kang, and H. Jung, "Hierarchical Federated Learning for Edge-Aided Unmanned Aerial Vehicle Networks," *Applied Sciences*, vol. 12, no. 2, pp. 670–682, Jan. 2022, ISSN: 2076-3417. DOI: 10.3390/APP12020670. [Online]. Available: https://www.mdpi.com/2076-3417/12/2/670/htm%20https://www.mdpi.com/2076-3417/12/2/670.

[40] Y. Deng, F. Lyu, J. Ren, *et al.*, "SHARE: Shaping data distribution at edge for communication-efficient hierarchical federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, vol. 2021-July, Institute of Electrical and Electronics Engineers Inc., Jul. 2021, pp. 24–34, ISBN: 9781665445139. DOI: 10.1109/ICDCS51616.2021.00012.

[41] M. S. Abad, E. Ozfatura, D. GUndUz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2020-May, Institute of Electrical and Electronics Engineers Inc., May 2020, pp. 8866–8870, ISBN: 9781509066315. DOI: 10.1109/ICASSP40776.2020.9054634. arXiv: 1909.02362.

[42] J. Feng, L. Liu, Q. Pei, and K. Li, "Min-Max Cost Optimization for Efficient Hierarchical Federated Learning in Wireless Edge Networks," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, Nov. 2021, ISSN: 1045-9219. DOI: 10.1109/TPDS.2021.3131654.

[43] B. Xu, W. Xia, J. Zhang, X. Sun, and H. Zhu, "Dynamic client association for energy-aware hierarchical federated learning," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 2021-March, Institute of Electrical and Electronics Engineers Inc., 2021, ISBN: 9781728195056. DOI: 10.1109/WCNC49053.2021.9417267.

[44] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint Edge Association and Resource Allocation for Cost-Efficient Hierarchical Federated Edge Learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020, ISSN: 15582248. DOI: 10.1109/TWC.2020.3003744. arXiv: 2002.11343.

[45] A. A. Abdellatif, N. Mhaisen, A. Mohamed, *et al.*, "Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data," *Future Generation Computer Systems*, vol. 128, pp. 406–419, Mar. 2022,

ISSN: 0167-739X. DOI: 10.1016/J.FUTURE.2021.10.016. arXiv: 2107.06548.

[46] R. Yu and P. Li, “Toward Resource-Efficient Federated Learning in Mobile Edge Computing,” *IEEE Network*, vol. 35, no. 1, pp. 148–155, Mar. 2021, ISSN: 1558156X. DOI: 10.1109/MNET.011.2000295.

[47] B. Xu, W. Xia, W. Wen, P. Liu, H. Zhao, and H. Zhu, “Adaptive Hierarchical Federated Learning over Wireless Networks,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 2070–2083, 2021, ISSN: 19399359. DOI: 10.1109/TVT.2021.3135541.

[48] H. Yang, “H-FL: A Hierarchical Communication-Efficient and Privacy-Protected Architecture for Federated Learning,” in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence, Aug. 2021, pp. 479–485. DOI: 10.24963/IJCAI.2021/67. arXiv: 2106.00275.

[49] Z. Qu, R. Duan, L. Chen, J. Xu, Z. Lu, and Y. Liu, *Context-Aware Online Client Selection for Hierarchical Federated Learning*, Dec. 2021. arXiv: 2112.00925v2. [Online]. Available: https://arxiv.org/abs/2112.00925v2.

[50] Q. Chen, Z. You, and H. Jiang, *Semi-asynchronous Hierarchical Federated Learning for Cooperative Intelligent Transportation Systems*, Oct. 2021. arXiv: 2110.09073. [Online]. Available: https://arxiv.org/abs/2110.09073v1.

[51] D. Yadav, *Categorical encoding using Label-Encoding and One-Hot-Encoder*, 2019. [Online]. Available: https://towardsdatascience.com/categorical-

encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd (visited on 02/26/2021).

[52] H. Saadat, M. S. Allahham, A. A. Abdellatif, and A. Mohamed, "RL-Assisted Energy-Aware User-Edge Association for IoT-based Hierarchical Federated Learning," in *(under review) 2022 International Wireless Communications and Mobile Computing (IWCMC)*, 2022.

[53] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-Balancing Federated Learning with Global Imbalanced Data in Mobile Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, Jan. 2021, ISSN: 15582183. DOI: 10.1109/TPDS.2020.3009406.

[54] J. Lee and S. Leyffer, *Mixed Integer Nonlinear Programming*, J. Lee and S. Leyffer, Eds., ser. The IMA Volumes in Mathematics and its Applications. New York, NY: Springer New York, 2012, ISBN: 978-1-4614-1926-6. DOI: 10.1007/978-1-4614-1927-3. [Online]. Available: http://link.springer.com/10.1007/978-1-4614-1927-3.

[55] Y. LeCun, C. Cortes, and C. J. Burges, *THE MNIST DATABASE of handwritten digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/ (visited on 03/09/2022).

[56] Z. Chkirbene, A. A. Abdellatif, A. Mohamed, A. Erbad, and M. Guizani, "Deep Reinforcement Learning for Network Selection over Heterogeneous Health Systems," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 258–270, 2022, ISSN: 23274697. DOI: 10.1109/TNSE.2021.3058037.