

UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

**Federated Deep Learning in Clustered Wireless Ad Hoc
Networks**

Diploma Thesis

Maria Papadopoulou

Supervisor: Dimitrios Katsaros

September 2022



UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

**Federated Deep Learning in Clustered Wireless Ad Hoc
Networks**

Diploma Thesis

Maria Papadopoulou

Supervisor: Dimitrios Katsaros

September 2022



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**Ομόσπονδη Βαθεία Μάθηση σε Ομαδοποιημένα Ασύρματα
Ad-Hoc Δίκτυα**

Διπλωματική Εργασία

Μαρία Παπαδοπούλου

Επιβλέπων: Δημήτριος Κατσαρός

Σεπτέμβριος 2022

Approved by the Examination Committee:

Supervisor **Dimitrios Katsaros**

Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Georgios Thanos**

Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Member **Spyros Lalis**

Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements


Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Δημήτριο Κατσαρό για την καθοδήγηση που μου προσέφερε και το χρόνο που διέθεσε δίνοντάς μου χρήσιμες συμβουλές και οδηγίες για την ολοκλήρωση της πτυχιακής μου εργασίας. Η ολοκλήρωση αυτής της εργασίας θα ήταν αδύνατη χωρίς την πολύτιμη συμβολή της πολύ καλής μου φίλης και συμφοιτήτριας Ελευθερίας Χίνη. Της εκφράζω ένα βαθύ ευχαριστώ για την άριστη συνεργασία που είχαμε στα πλαίσια εκπόνησης αυτής της εργασίας, την συνεχή συμπαράσταση και την όμορφη επικοινωνία όλο αυτό το διάστημα. Τέλος, θέλω να ευχαριστήσω πολύ τους γονείς μου, οι οποίοι υπήρξαν πάντα ένα ανεκτίμητο στήριγμα για μένα και στους οποίους οφείλω όλη τη διαδρομή των σπουδών μου, μέχρι σήμερα.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Maria Papadopoulou



Diploma Thesis

Federated Deep Learning in Clustered Wireless Ad Hoc Networks

Maria Papadopoulou

Abstract

Nowadays, the expansion of Internet-Of-Things (IoT) devices or other cloud based applications over siloed data centers has generated significant interest in training these models to overcome the privacy and overhead challenges of centralized machine learning. Federated learning has been proposed to enable multiple devices connected via an ad-hoc network to train an ML model without letting out raw data. Nearly all prior works are focused on "star" topologies consisting of distributed clients/nodes and a central server, which creates robustness concerns as well as requires a significant amount of communication resources. To overcome these issues, we propose a fully decentralized federated deep learning model in wireless ad-hoc networks. Specifically, we develop a clustered network, where there is a clusterhead node that has the responsibility to act as the local server among the cluster nodes and implement the averaging. Another significant feature of our model is that the participation of nodes is based on data similarity which reduces the communication cost between nodes with similar data. We did a bunch of different experiments on both real-world and synthetic datasets that verify our analytical results and demonstrate the efficiency of our model.

Keywords:

Federated learning, ML model, ad-hoc networks, fully decentralized, averaging, data similarity, node participation, real/synthetic data

Διπλωματική Εργασία

Ομοσπονδη Βαθείά Μάθηση σε Ομαδοποιημένα Ασύρματα Ad-Hoc

Δίκτυα

Μαρία Παπαδοπούλου

Περίληψη

Σήμερα, ο πολλαπλασιασμός των συσκευών του Internet-Of-Things (IoT) ή άλλων cloud εφαρμογών σε "siloes" κέντρα δεδομένων έχει δημιουργήσει σημαντικό ενδιαφέρον για την εκπαίδευση αυτών των μοντέλων ξεπερνώντας την ιδιωτικότητα και τις γενικές προκλήσεις της κεντρικοποιημένης μηχανικής μάθησης. Η Ομοσπονδιακή εκμάθηση έχει προταθεί για να καταστεί δυνατή η εκπαίδευση ενός μοντέλου ML πολλαπλών συσκευών που συνδέονται μέσω ενός δικτύου ad-hoc χωρίς να εκπέμπονται ακατέργαστα δεδομένα. Σχεδόν όλες οι προηγούμενες εργασίες επικεντρώνονται σε τοπολογίες σε σχήμα αστεριού που αποτελούνται από καταναμημένους πελάτες/κόμβους και έναν κεντρικό διακομιστή, πράγμα που δημιουργεί ισχυρές ανησυχίες ευρωστίας καθώς και απαιτεί σημαντικό όγκο επικοινωνιακών πόρων. Για να ξεπεραστούν αυτά τα ζητήματα, προτείνουμε ένα πλήρως αποκεντρωμένο ομοσπονδιακό μοντέλο βαθιάς εκμάθησης σε ασύρματα ad-hoc δίκτυα. Συγκεκριμένα αναπτύσσουμε ένα ομαδοποιημένο (clustered) δίκτυο, όπου υπάρχει ένας clusterhead κόμβος που έχει την ευθύνη να ενεργεί ως τοπικός διακομιστής μεταξύ των κόμβων του cluster και να κάνει το averaging. Ένα άλλο σημαντικό χαρακτηριστικό του μοντέλου μας είναι ότι η συμμετοχή κόμβων βασίζεται στην ομοιότητα των δεδομένων, πράγμα που μειώνει το κόστος επικοινωνίας μεταξύ κόμβων με όμοια δεδομένα. Κάναμε αρκετά διαφορετικά πειράματα σε σύνολα δεδομένων τόσο σε πραγματικά όσο και σε συνθετικά δεδομένα που επαληθεύουν τα αναλυτικά μας αποτελέσματα και αποδεικνύουν την αποτελεσματικότητα του μοντέλου μας.

Λέξεις-κλειδιά:

Ομοσπονδιακή εκμάθηση, μοντέλο ML, ad-hoc δίκτυα, πλήρως αποκεντρωμένο, averaging, ομοιότητα δεδομένων, συμμετοχή κόμβων, πραγματικά/συνθετικά δεδομένα

Table of contents

Acknowledgements	ix
Abstract	xii
Περίληψη	xiii
Table of contents	xv
List of figures	xvii
List of tables	xix
Abbreviations	xxi
1 Introduction	1
1.1 Thesis Structure	2
2 Background	3
2.1 Deep Learning	3
2.1.1 Clustering	5
2.2 Neural Networks	6
2.2.1 Activation functions	6
2.2.2 Loss functions	9
2.2.3 Optimizers	10
2.2.4 Recurrent Neural Networks	11
2.3 Federated Learning	14
2.4 Related Work	15

3	Methodology	17
3.1	Data set details and preprocess	17
3.2	Wireless Network Architecture	18
3.3	Clustering	18
3.3.1	Max-Min D-Cluster Algorithm	18
3.3.2	Betweenness Centrality	20
3.4	Deep Learning Architecture	20
3.4.1	GRU	21
3.5	Federated Learning Process	22
3.5.1	Node Participation Protocols	22
3.5.2	Federated Averaging Algorithm	24
4	Experiments and Results	27
4.1	Experimental setup	27
4.2	Performance Metrics	29
4.2.1	Mean Squared Error(MSE)	29
4.2.2	Accuracy	30
4.3	Results	31
4.3.1	An overview	31
4.3.2	All-Node Participation Experiment (ANP)	31
4.3.3	Random-Node Participation Experiment (RNP)	32
4.3.4	Similarity-based-Node Participation Experiment (SNP)	36
4.3.5	Final Conclusion	38
5	Conclusions	39
5.1	Summary and Conclusion	39
5.2	Future work	39
	Bibliography	41

List of figures

2.1	Relationship between AI, ML and DL	3
2.2	Deep Neural Network structure	4
2.3	Random example of partitional clustering algorithm	5
2.4	Binary step function	7
2.5	Linear function	7
2.6	Sigmoid activation function	8
2.7	Tanh activation function	9
2.8	ReLU activation function	9
2.9	GRU architecture	12
2.10	Structure of GRU-based model	12
2.11	Structure of GRU memory cell	13
2.12	Centralized FL (a) and Decentralized FL (b)	15
3.1	Wireless ad-hoc network architecture	18
3.2	Example of 3-cluster Max-Min Formation in 25-node graph	20
3.3	GRU architecture summary	21
3.4	Federated Learning in clustered network	25
4.1	MSE example	30
4.2	MSE and Accuracy for the representative node (20 nodes)	31
4.3	MSE and Accuracy for the representative node (50 nodes)	32
4.4	Global Averaging Effect (50 nodes)	32
4.5	Loss progress comparison for the representative node (20 nodes)	33
4.6	Accuracy comparison for the representative node (20 nodes)	33
4.7	Global Averaging Effect for 75% participation of nodes (20 nodes)	34
4.8	Global Averaging Effect for 25% participation of nodes (20 nodes)	34

4.9	Loss progress comparison for the representative node (50 nodes)	34
4.10	Accuracy comparison for the representative node (50 nodes)	35
4.11	Global Averaging Effect for 75% participation of nodes (50 nodes)	35
4.12	Global Averaging Effect for 50% participation of nodes	36
4.13	Global Averaging Effect for 25% participation of nodes	36
4.14	Loss progress comparison for the representative node	36
4.15	Accuracy comparison for the representative node	37
4.16	MSE and Accuracy for the representative node	37
4.17	Global Averaging Effect	38

List of tables

4.1	Data set configuration used for testing	27
-----	---	----

Abbreviations

IoT	Internet of Things
AI	Artificial Intelligence
ML	Machine Learning
FL	Federated Learning
i.e.	Id Est
e.g.	exempli gratia
ANNs	Artificial Neural Networks
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
Adam	Adaptive Moment Estimation
RNN	Recurrent Neural Network
BPTT	Backpropagation through Time
GRU	Gated Recurrent Unit
MSE	Mean Squared Error
MAE	Mean Absolute Error
DFL-UN	Decentralized Federated Learning for UAV
ARMA	Autoregressive Moving Average
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
ANP	All Node Participation
RNP	Random Node Participation
SNP	Similarity based Node Participation

Chapter 1

Introduction

The modern era has witnessed an explosion in the number of Internet of Things (IoT) devices to empower different applications like mobile phones, smart health care, wearable devices, autonomous vehicles, augmented reality among others. The growing computational power of these devices and privacy-preserving concerns have led to emerging technologies that keep the devices' datasets local and thrust network computation to the edge. Traditionally, model training has been carried out at a central node (i.e. a server) known as centralized machine learning (ML). However, as these devices generate larger volumes of data and the capacity and energy consumption increase, it is conceivable to improve local resources. Also there are a lot of privacy-sensitive applications where end users may not want to transmit their raw data. For these reasons, Federated Learning (FL) has emerged as a technique for distributed model training across devices.[1]

Traditional FL consists of a central node (server) connected to multiple devices (clients). Each round of the model (e.g. a neural network) training has two steps:

1. **Local training** where each device updates its local model using its collected dataset and the global model with gradient descent methods.
2. **Global aggregation** where the main server collects all the local updates, computes a new global model and then synchronizes all the nodes to begin next round of training with these aggregated version.

A main advantage of federated learning is that the data itself is never exposed between the clients and the main server, which solves privacy issues as mentioned above. In conventional FL a server is required for the communication and parameters' aggregation to occur. However

this is limiting and prohibitive to contemporary networks where the devices are expected to work without a server and can only communicate directly with the adjacent ones.[2]

In this Thesis, we propose a fully decentralized federated deep learning model in clustered ad-hoc networks to overcome communication and computation constraints. In other words, our network is divided into clusters where we select the node with the biggest betweenness centrality to become the clusterhead. This node acts as the local server among the nodes of the cluster and it is responsible for the local averaging. There's also a global averaging, where there is data exchange between the different clusters. We also focused in reducing communication cost and energy consumption from data transmission. To achieve this, the nodes' participation in averaging is based on data similarity. Meaning that between neighbor nodes with similar data is necessary only one of them to participate in the training and send its data to the clusterhead (local server). Three different approaches were used to test our model: fully participation of all nodes in the network, random participation based on a selected threshold and data similarity participation.

1.1 Thesis Structure

This Thesis structure is organized as it follows. In Chapter 2 we make an extended introduction to Deep and Federated Learning as well as related work. Chapter 3 discusses our proposed model for federated learning in ad-hoc wireless networks. In this chapter, we also present the neural network used in our model and the innovations used in the federation part. Chapter 4 presents our experiments and the results. Finally, in chapter 5 we summarize our work and also discuss about other directions for future work.

Chapter 2

Background

2.1 Deep Learning

Deep learning has emerged from research in artificial intelligence and machine learning. Figure 2.1 illustrates the relationship between AI, deep and machine learning. The term Artificial Intelligence refers to the science and engineering of making intelligent machines, especially computers that can think and act like humans.

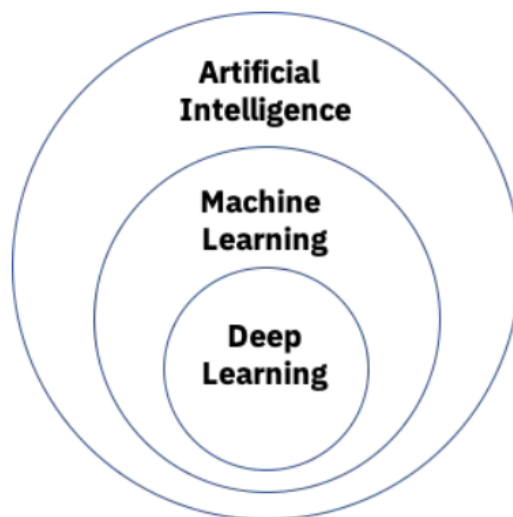


Figure 2.1: Relationship between AI, ML and DL

The difference between deep learning and classical machine learning is the type of data used and the way in which they learn. Deep learning is a subset of machine learning that focuses on deep neural network models. The word "deep" refers to the depth of the networks' layers which are usually three or more. Figure 2.2 illustrates the structure of a deep neural net-

work. Multiple layers of interconnected nodes exchanging data, weights and biases, working together and trying to predict or classify objects with accuracy. Deep learning algorithms can work with both structured and unstructured data. That means that they can process data like text and images without the dependency on human experts to remove unnecessary features and organize them into structured format.[3]

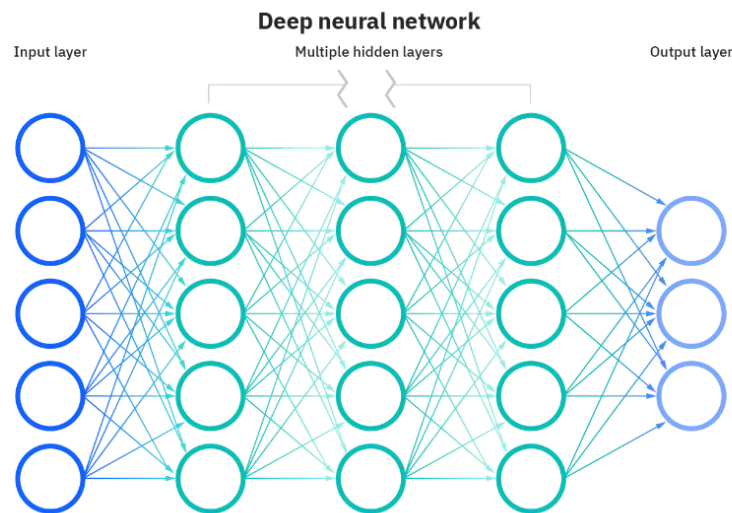


Figure 2.2: Deep Neural Network structure

In machine and deep learning there are four commonly used learning models usually categorized as supervised learning, semi-supervised learning, unsupervised learning and reinforcement learning.

1. **Supervised learning** utilizes patterns in the training dataset to map features to the target. The desired learning task is to teach the model for every input to predict the expected output. The accuracy of the algorithm is measured through the loss function aiming to sufficiently minimize the error between the inputs and the targets of the model. The most common supervised learning tasks are regression and classification. Classification uses an algorithm to predict in which category an example belongs to. By recognizing specific features within the dataset attempts to conclude on how those entities should be defined. On the other hand regression involves predicting numeric data by understanding the relationship between dependent and independent variables.
2. **Unsupervised learning** uses unlabeled data. It attempts to detect patterns in the dataset based on similarities or differences and categorize individual instances to groups. Common unsupervised learning tasks are clustering, filtering and anomaly detection.

3. **Semi-supervised learning** as its name suggests is the medium between supervised and unsupervised learning. It is mainly used when the dataset contains both labeled and unlabeled data (i.e. all features are present, but not all features have associated targets).
4. **Reinforcement learning** is arguably the closest attempt at modeling the human learning experience. This technique is based on rewarding desired behaviors and/or punishing undesired ones. A reinforcement learning agent takes actions and learns through trial and error. No single correct answer is desired, but a long-term overall outcome to the optimal solution.[4]

2.1.1 Clustering

Clustering[5] is considered the most crucial aspect of unsupervised learning challenge, finding a structure in a collection of data that isn't labeled. Consequently, a cluster is a group of items that are "similar" and in some ways "different" from the ones in other clusters.

There are two types of clustering. Data clustering might be partitional or hierarchical. The use of hierarchical algorithms is to identify subsequent clusters that have been existing from the beginning, whereas partitional algorithms are used for all clusters to be found simultaneously. Hierarchical Algorithms can be agglomerative (from the bottom up) or divisive (top-down). Agglomerative algorithms start with each element as a distinct cluster, then combine them successively into larger clusters. Partitional algorithms start with the entire collection and partition it into successive parts, smaller clusters.

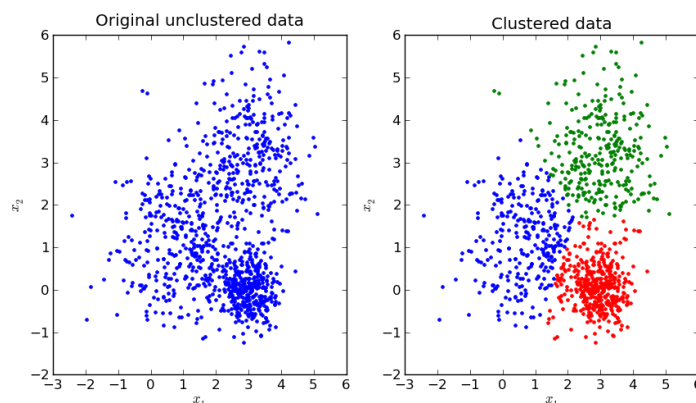


Figure 2.3: Random example of partitional clustering algorithm

2.2 Neural Networks

Neural networks also known as artificial neural networks (ANNs) are a subset of machine learning and at the heart of deep learning algorithms.[6] Neural networks were first proposed in 1944 by Warren McCulloch and Walter Pitts at University of Chicago. [7] Their idea was based on how neurons in the human brain work and create a model using electrical circuits that could mimic these actions and solve problems. Over time researchers moved to another direction and started using neural networks to perform particular tasks and not focus only on neuroscience and biology. Till today , they support a variety of tasks like computer vision, machine translation etc.[8] ANN consists of interconnecting artificial neurons processing information and trying to solve specific problems like real biological neurons would do.

In principle, all artificial neural networks have a similar topology. There are neurons that interface with the real world receiving the input data as well as other neurons in charge of providing the real world with the network's output. These are called the input and output layers respectively. Between these two layers can be many other hidden layers that also consist of interconnected neurons.[9] The process is simple. Data are fed to the network through input neurons, also known as nodes, which are small units where computation happens. They combine data received from the above layer with a set of coefficients/weights that either amplify or dampen that data before they pass them to the next layer until they end up to the output layer. At the hidden layers, the data-weights product is then summed and passed through node's function called activation function that determines to what extent the signal should continue propagating through the network and how much it will affect the outcome. [10] Our aim is to have dynamic networks and the ability to extract complicated information from the data and recognizing mappings between input and output, that's why the presence of an activation function is necessary.[11]

2.2.1 Activation functions

Activation functions play a key role in proper functioning of the network. They're responsible to decide whether the neuron's input is important to the network for the process of the prediction or it is irrelevant.

Activation functions fall into three categories:

1. **Binary step function**

Binary step function depends on a threshold that determines whether a neuron should be activated or not. Input data insert the node and compare to a particular threshold. If the input is greater than it, the neuron is activated, else is deactivated, which means that the output is not fed to the next layer.

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

Figure 2.4: Binary step function

2. Linear activation function

Linear activation function is proportional to the input. It doesn't change the input and just passes the value as it was given.

$$f(x) = x$$

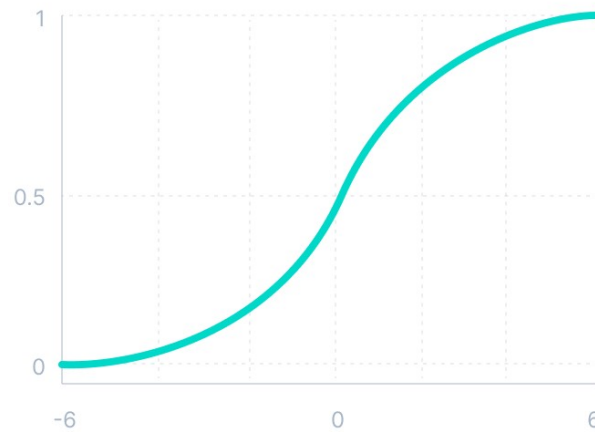
Figure 2.5: Linear function

3. Non-linear activation function

Non linear activation functions are the most useful as they give to the network the ability to learn complex tasks and create more difficult mappings between inputs and outputs. The most common used non-linear activation functions are listed below:

- Sigmoid activation function

Sigmoid function has an S-shape as shown in the figure 2.6. It takes any real value as input but the output is in range 0 and 1. As the probability of anything is always 1 or 0 (if it exists or not), this function is suitable when we have to predict a probability as an output. Another advantage of sigmoid function is that it is differentiable, avoiding with this way jumps in the output values.

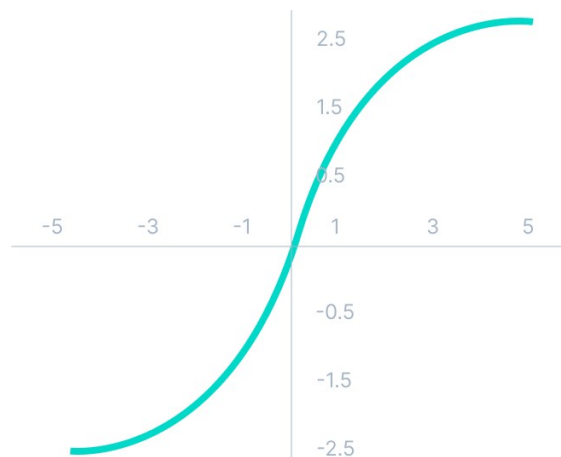


$$f(x) = \frac{1}{1 + e^{-x}}$$

Figure 2.6: Sigmoid activation function

- Tanh activation function

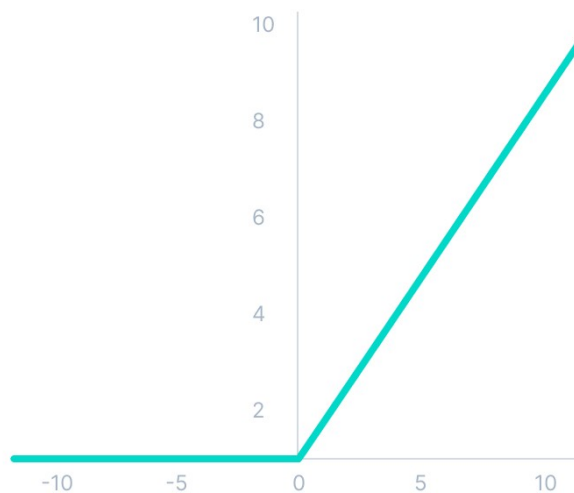
The tanh function has also an S-shape but the output range is in -1 and 1. It is centered to zero which means that the more positive the input is the closer to 1 will be the output and on the other hand the more negative the input is the closer to -1 will be the output. Tanh function is commonly used in hidden layers.



$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

Figure 2.7: Tanh activation function

- ReLU activation function ReLU (Rectified Linear Unit) might look similar to the linear activation function, but it is actually different as it has a derivative. It doesn't allow all the neurons to be activated in the same time. The neurons will be activated only if the output of the transformation is greater than 0, that's why it is considered more computationally efficient as well as it converges faster than the other activation functions mentioned above.[11]



$$f(x) = \max(0, x)$$

Figure 2.8: ReLU activation function

2.2.2 Loss functions

The main task of a deep learning neural network is to map a set of inputs to a set of outputs. However this process cannot be done perfectly. For this reason there are loss functions

used to calculate the accuracy of the training and measure how well the algorithm fits the data. Typically, in neural networks our aim is to minimize the error of the model during the optimization process which means choosing the best loss/cost function.[12]

Loss functions can be divided into two main categories depending upon the type of our learning goals: classification and regression losses. In classification we're trying to map input variables to a class label. Most common classification loss functions are max likelihood and cross entropy. In regression, on the other side, we are trying to predict a real value quantity. Some loss functions used in regression at deep neural networks are Mean Square Error (MSE) and Mean Absolute Error (MAE). [13]

2.2.3 Optimizers

In order to make our model perform better, we need to minimize the loss, as mentioned before. This process is called optimization and the algorithms used to reduce these losses optimizers. Especially in deep neural networks, optimizers define how we should change model's parameters like weights and learning rate to achieve the most accurate results possible.

There are a lot of different optimizers and the most commonly used are mentioned below:

- **Gradient Descent** Gradient Descent is an optimization algorithm trying to find a local minimum of a differentiable function through an iterative process. The learning rate defines how large (or small) the iterative steps gradient descent takes in the direction of the local minimum will be.
- **Stochastic Gradient Descent** Stochastic Gradient descent is almost the same with gradient descent but the changes of the parameters are made more often. Particularly, the SGD algorithm performs a parameter update for 'each' training example which provides faster convergence and also can handle extremely large datasets that don't fit in memory.
- **Adam (Adaptive Moment Estimation)** The main difference with the other optimizers mentioned above, is that Adam doesn't maintain a single learning rate for all weight updates, but it computes adaptive learning rates for each parameter. [14]

2.2.4 Recurrent Neural Networks

Recurrent neural networks are a type of artificial neural networks commonly used in time series problems. Their main characteristic is that they have 'memory'. That means that they use information from previous input data to predict the current inputs and outputs. Another element that make them stand out is that each node doesn't have its own parameter but the weights are shared within each layer of the network.

For training a recurrent neural network we use the backpropagation through time (BPTT) algorithm to calculate the gradients. As the parameters are shared across each layer it is necessary to sum the errors at each time step. One problem that arises in RNN is the vanishing gradients. It may occur when the network is very deep, which means having too many hidden layers. Then, the gradients used to compute the weight update may get very close to zero, preventing the algorithm from learning. [15]

GRU

An overall

GRU (Gated Recurrent Networks) are an improved version of standard recurrent networks, trying to solve the vanishing gradient problem mentioned above. They use two gates: the update gate and the reset gate. These gates are responsible for choosing which information should be carried forward to the output. Their distinguishing feature is that they can keep information from long ago, which make them really effective.

The update gate's role is to decide how much of the past information is necessary to be passed along to the future. On the other hand, reset gate determines how much of the past information the model needs to forget. In this way GRUs are capable of storing and filtering the information, performing very well even in complex problems. [16]

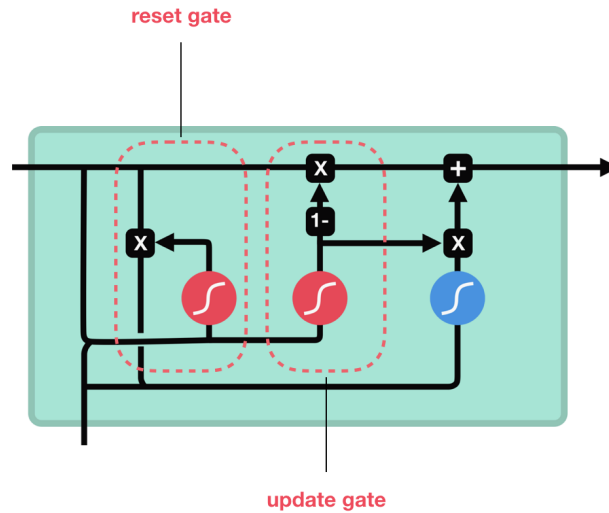


Figure 2.9: GRU architecture

The digital design behind the model in more details

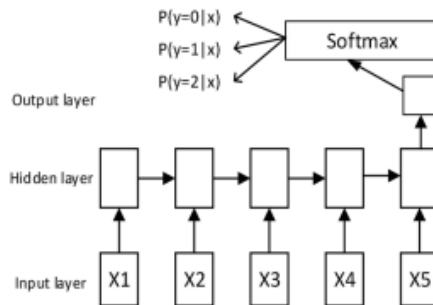


Figure 2.10: Structure of GRU-based model

As seen in Fig. 2.9[17], the input layer is made up of a number of neurons, with the number of neurons being dependent on the size of the feature space. A similar relationship exists between the output space and the number of neurons in the output layer. The GRU networks' primary functions are covered by the hidden layer(s) that comprise memory cells. Two gates in the cell—a reset gate (r_t) and an update gate (u_t)—are responsible for maintaining and changing the status of the cell. Figure 2.11 shows a circuit schematic representing the structure of a memory cell.

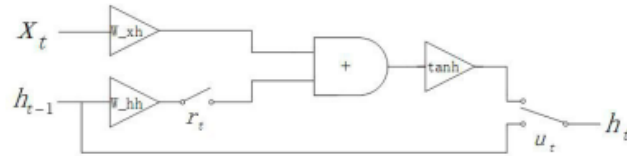


Figure 2.11: Structure of GRU memory cell

Two elements, x_t and h_{t-1} , are connected to both gates. The first comes from the input sequence, while the second represents the value of the memory cells' output from the preceding instant in time. In order to achieve the goal of filtering, each gate accordingly completes a variety of tasks:

The reset gate is used to limit how much information from a unit's previous timestep, or h_{t-1} , will influence the current information, or x_t . Making h_{t-1} has no effect on x_t if it is not crucial to x_t , hence r_t can be opened.

The update gate defines if the current information x_t should be disregarded. We will disregard the present information x_t when u_t is switched on the under branch and create a "short-circuit link" from h_{t-1} to h_t .

This successfully resolves the gradient vanishing problem by causing the gradient to propagate in reverse.

2.3 Federated Learning

Federated learning (also known as collaborative learning) is a machine learning technique that aims to train an algorithm over numerous local datasets contained in decentralized edge nodes or servers without explicitly exchanging parameters and other data information. This approach differs from conventional centralized machine learning methods where local data need to be transferred over to a central device (server). [18]

Federated learning already provides significant benefits over traditional machine learning approaches. It is mainly proposed in order to solve privacy-preserving problems. In typical FL, there is a central server connected to numerous end devices. While training occurs, each device updates its local model using its own local dataset and exchanging with the server only the updated parameters. Then, the server gathers and computes a new global model and sends the up-to-date parameters back to the local devices to start next round of training. [19]

Federated learning systems are categorized based on their topology:

- **Centralized federated learning**

In centralized federated learning systems, there is a central server in charge of organizing the training process and coordinating all the nodes participating. However, since all the local devices need to send updates to a single unit, there is the potential of failure as the server may end up a bottleneck of the system.

- **Decentralized federated learning**

In decentralized federated learning (also known as peer-to-peer distributed learning) there is no central server, but only peer-to-peer communication between the local nodes which are exchanging updates. The topology here in contrast to the star graph of the server-client architecture is represented as a connected graph of nodes that are the clients and an edge indicates a communication channel between the clients, as seen in Figure 2.12(b). [20]

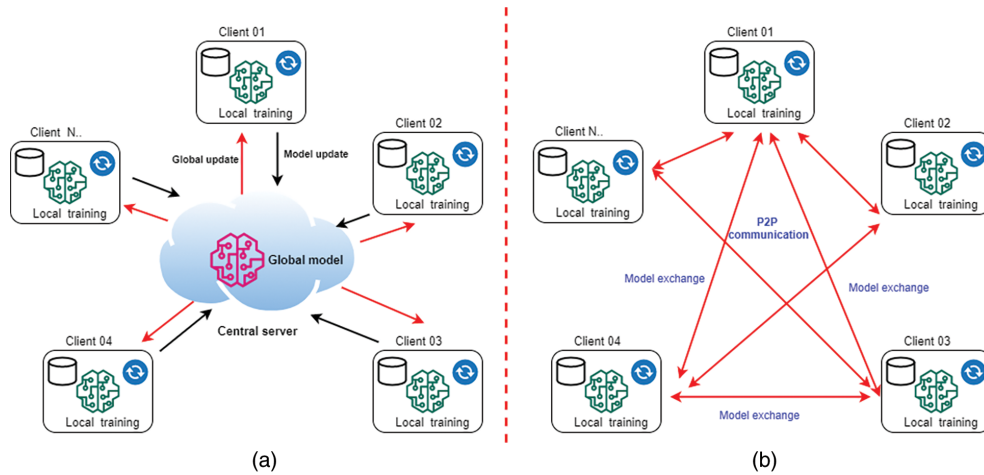


Figure 2.12: Centralized FL (a) and Decentralized FL (b)

Even though federated learning is emerging, there are some challenges remaining:

- Communication overheads

When there are a lot of clients, the system might run many rounds before convergence which may overload the network.

- Systems and Data Heterogeneity

The system is heterogeneous and local devices have different computational and communication capabilities. Also, the data may differ in features and samples (non-identically distributed data) which also may affect the execution of the model.

- Privacy and Security

Federated learning is already preserving privacy by sharing only models' parameters. However it is always possible that sensitive information may be revealed by a malicious central server or participant. [21]

2.4 Related Work

The idea of Federated Learning (FL) was first proposed by Google in 2015[22]. Under the control of a central server, numerous devices work together to develop a machine learning model without disclosing any of their personal information. This presents numerous opportunities in crucial industries like healthcare, finance, and others where it is dangerous to exchange sensitive customer data with other businesses or technology. Although FL seems

like a potential Machine Learning (ML) method to protect the confidentiality of local data, it is also susceptible to attacks like other ML models.

Today, there are various papers [23] that have explored theoretically the opportunities and difficulties in federated learning in light of the rising interest in the FL sector as well as giving a comprehensive overview of existing methodologies.

Later articles introduce us to the idea of distributed, “serverless”, machine learning over wireless networks. In [24], for instance, the authors explain the local networks’ topological structures coordinating the various diverse nodes at each network layer for cooperative device-to-device collaboration for learning communications. Moving away from star network topologies used in federated learning to transfer parameters to more distributed topologies .

Beyond purely theoretical papers, FL researchers move on to more practical solutions. The [1] supports that FL can be achieved via an architecture called Decentralized Federated Learning for UAV Networks (DFL-UN) without a central authority. Furthermore, it carries out an exploratory simulation research to confirm the viability and efficiency of the DFL-UN architecture. Additionally, numerous FL algorithms have been developed to better fit to the features of wireless edge networks including multihop concepts like multihop FL, collaborative FL and Fog learning [25].

A relevant work to ours is that of, which suggests utilizing [19] FL principles to networks that are completely decentralized and do not rely on a central server to manage learning activities, introducing us to federated averaging (FA) and exchange of both model updates and gradients to improve convergence.

Another related study [26] suggests a method for choosing the most suitable candidate clients in federated learning based on test accuracy as well as a clients’ alarm application that notifies users when unauthorized IoT devices are present in their area.

Although until this day various solutions have been published, our study inventively combines a purely distributed federated learning on ad hoc network, a network specifically clustered to fit optimally to the FL algorithms and an effective elimination of unwanted communication by proposing two algorithms for node participation in the learning process.

Chapter 3

Methodology

3.1 Data set details and preprocess

The selection of data plays a significant role in the efficiency of our proposed model. Since we are working on federated learning which is mainly used in IoT devices (e.g. smartphones, wearable devices, autonomous vehicles) we decided to use time series data for prediction. Particularly, we used a real data set obtained by Kaggle [27] that refers to temperatures (in Celcius degree) of five Chinese cities for a time period of five years (2010-2015). Each 'city' file contains 52585 temperature measurements.

We also used a synthetic data set. With the help of Autoregressive Moving Average model (also known as ARMA) we generated time series data. Each synthetic data file consists of 10000 samples. The raw data sets were processed before training. As the data sets were huge, we divided them into smaller ones, to serve our purpose. In this way, every node has a unique data set to perform training.

As a part of cleaning and processing data, we identified the rows with Null/NaN values considered as missing and drop them. To implement that we used the `dropna()` method in python.

Standardization of the data is another common requirement before ML training. All individual features should look like standard normally distributed data. To standardize the data we used the `StandardScaler()` function from scikit-learn library in python which removes the mean from the features and scales them to unit variance.

To finally perform the ML training, we split the data into two parts: 1) the training data set (90 % of the original) and the test data set (10 % of the original).

3.2 Wireless Network Architecture

In this paper, we consider the network architecture as depicted in Fig. 3.1. It is an ad-hoc network consisting of n nodes/devices connected to each other through wireless links. For the purposes of our work, the network has a single layer and there is no central server as we are working in a fully decentralized mode.

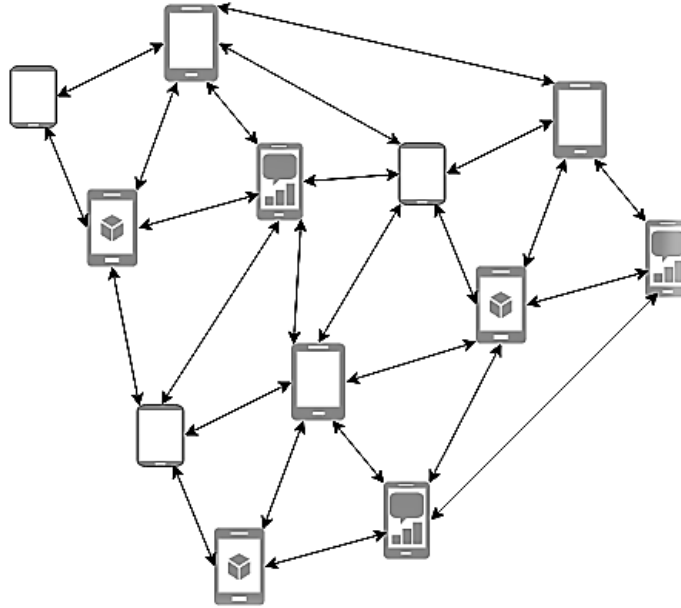


Figure 3.1: Wireless ad-hoc network architecture

We constructed our network using a network generator implemented by Mr. Papakostas Dimitrios. By determining the number of layers, the number of nodes and the mean degree of each node we created a random network whose connectivity is characterized by an undirected graph $G(V, E)$ with $V = \{1, 2, \dots, n\}$ denoting the set of nodes and $E \subseteq \{(i, j) \in V \times V | i \neq j\}$ denoting the set of edges. Each node i can only communicate (i.e. exchange parameters/ weights) with its neighbors. The set of neighbors of node i is denoted as $N_i = \{j \subseteq V | (i, j) \subseteq E\}$ [28].

3.3 Clustering

3.3.1 Max-Min D-Cluster Algorithm

Our goal in this section is to represent the ad hoc network we working on as a set of clusters suitable for reducing communication among nodes. The algorithm of choice , Max-

Min D-Cluster[29], introduced by Alan D. Amis, Ravi Prakash, Thai H.P. Vuong and Dung T. Huynh is specifically designed for large, wireless, ad hoc networks and is explained briefly below.

The Algorithm

Nodes in ad hoc networks can move around freely and connect to one another wirelessly. To facilitate effective communication between nodes, a wireless backbone architecture can be created. All nodes have the same capabilities, but only a selected few are chosen to create the backbone. These are the winner nodes in a cluster, or clusterheads. Clusterheads are nodes that are entrusted with the duty of message routing every node in their cluster. This wireless backbone can be used either to distribute routing information or to route packets, or both.

Further Definitions needed:

- Floodmax

Locally, each node broadcasts its winner id To all of its 1-hop neighbors,. After one cycle of hearing from each neighboring node, the node chooses as the new winner the id with the highest value from the round. This method keeps going for d rounds (specified by user).

- Floodmin

Similarly to Floodmax, Floodmin runs the same algorithm for d extra rounds only that in the end it elects as winner the id with the smallest value.

There are five logical stages of the heuristic as follows:

- Each node initializes its winner id to its own,
- larger node ids selected as winners via floodmax,
- followed by the spread of smaller node ids as winners using floodmin,
- final clusterhead determination and lastly,
- the link among clusterheads.

Finally, the ultimate clusters are formed from nodes with the same final clusterhead.

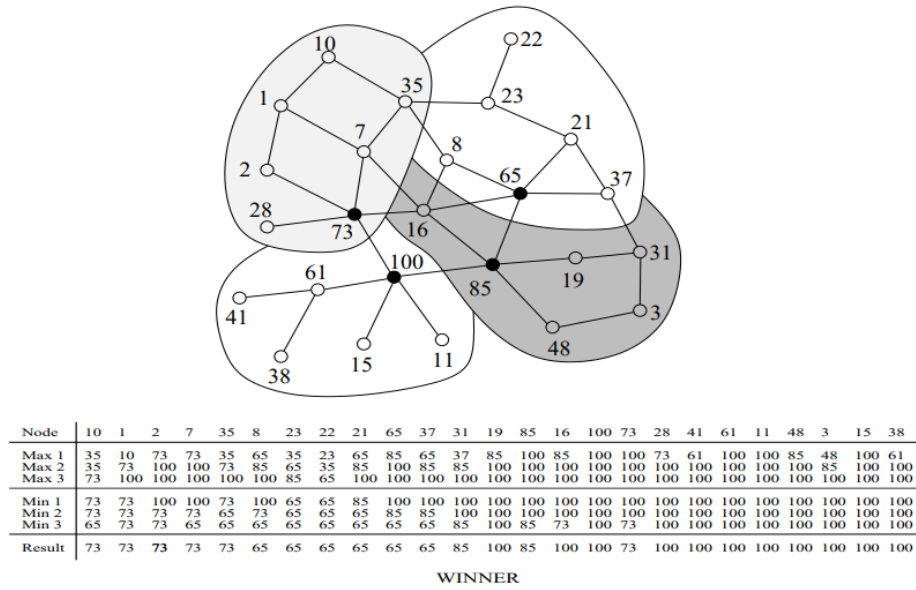


Figure 3.2: Example of 3-cluster Max-Min Formation in 25-node graph

3.3.2 Betweenness Centrality

Although the proposed heuristic performs quite well, it is denoted that it provides an optimal solution only when the largest node ids are spaced d distance apart. Therefore, in our code we suggest an extra step at the end of the given algorithm.

The elected final clusterheads give up their winning state and name clusterhead the node with the highest betweenness centrality[30] in their cluster.

High betweenness vertices may have significant influence inside a network due to their control over how information is passed between others. Additionally, because they are located on the greatest number of message paths, they are the ones whose removal from the network will have the most impact on communication between other vertices.

Consequently, the choice of this centrality measure has a massive impact on the low cost in both time and messages around the network we want to achieve.

3.4 Deep Learning Architecture

For our thesis in which we focus on time series prediction we have decided to work with one model, a GRU, which will be presented below in detail. Although GRU is effective in

our case taking into consideration the type of the dataset, not only our code but our entire idea is flexible enough to accommodate other models, with a variety of layers and parameters, as long as they are trained using weight exchange.

Each node in the network splits its unique data in chunks and so the training is performed per chunk for all of them. We have selected chunks of 500 data points each, and as a result approximately 20 rounds of training for each node.

$$\text{Rounds} = \frac{\text{len}(\text{data})/\text{node}}{\text{len}(\text{chunk})} \approx \frac{10000}{500} = 20$$

3.4.1 GRU

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 15)	810
dropout (Dropout)	(None, 15)	0
dense (Dense)	(None, 1)	16

=====
 Total params: 826
 Trainable params: 826
 Non-trainable params: 0
 =====

Figure 3.3: GRU architecture summary

The GRU RNN is a Sequential Keras model. After initializing our Sequential model, we'll need to add in the layers. The first layer we'll add is the Gated Recurrent Unit layer. Accordingly to the dataset, we have make this a 15-cell layer to have the optimal results. Adding this layer is what makes our model a Gated Recurrent Unit model.

After adding the GRU layer, we'll add a Dropout one. In order to avoid overfitting, the Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training. Furthermore, the sum of all inputs is maintained by scaling up non-zero inputs by $1/(1 - \text{rate})$.[]

Finally, we'll add a dense layer as output. The dense layer will have 1 unit. We have 1 unit in our output layer since we are dealing with a regression task.

3.5 Federated Learning Process

The federated learning algorithm can be briefly presented by the following methodical steps:

1. During the first round of the procedure the entirety of the nodes of our network implements local training to the first chunk of data.

2. In the second and every other run, before the training begins each node send the trainable parameters of the previous round to their clusterhead, which will be operating as a local server.

3. In each cluster the local servers having gathered their clients' trainable parameters, will then create and send back to the clients a new set of weights to continue the training, by averaging the delivered packages.

Only the nodes who participated in the specific round will receive the updated weights. The rest of them will begin the next training round with their own trainable parameters.

4. Once the federated learning hits a specific number of adequate iterations, global averaging takes place. Meaning that clusterheads interrupt the previous explained process to communicate with each other. Using the backbone method, servers exchange their weights, average them and distribute them to their cluster members. Therefore, in the next round of learning, every node will set exactly the same weights to its ML model.

5. The algorithm is decided to terminate as soon as the accuracy of the majority of the models have been established to a descent amount, meaning that we have reached convergence.

3.5.1 Node Participation Protocols

Under the heuristic of FL , the selective set of nodes begin to train the machine learning model along with the help of clusterheads, which in our case play the role of local servers, exchanging training variables with each other as well as their cluster members. Dealing with massive networks, big datasets and complex ML models, every opportunity to reduce unwanted and unfitting communication between nodes is valuable. Thus, we came across with

two possible solutions that can help us eliminate the cost as well as improving the accuracy of the final results.

Random participation of nodes

The first, and quite simple, way to implement such thing is to randomly distribute weights to each node and continue by selecting only a percentage of nodes in each cluster, based on their weights, to proceed to a round of training. This way, the learning procedure not only is done quickly but also with lower cost of message exchange.

Although with this technique we can expect some of the above benefits, it is clear that in the long run we will come face to face with quality problems, since we are unable to know which node and in which round can be helpful and contribute the most to the training process.

Data similarity-based participation of nodes

Moving on to the second data participation protocol, we are now focusing on the quality of data that its node has to offer. Therefore, the package of trainable parameters (weights) each node wants to sent to its clusterhead, if not unique, must be at least not very similar to other packages within the cluster. The measure of uniqueness, or in this case the “similarity” among the nodes’ data is not an easy task to deal with, since they are time series. After a thorough research, we decided upon the paper of Rakesh Agrawal, Christos Faloutsos, Arun Swami[31], which proposes processing similarity challenges by taking advantage of the Discrete Fourier Transform (DFT) and mapping time sequences to the domain of frequency, knowing that only the earlier frequencies are strong, meaning that they are adequate to behold the information of the whole sequence. Thus, relying their work on the Parseval’s theorem, they provide us guidance on how from comparing 500 data points for each node per round, or even worse 826 trainable parameters per node, we will end up with only a few (5-10) integers.

More specific, in each training round, data of each node pass through the Fast Fourier Transform function (FFT) from the scipy library, providing us with a list of integers representing the DFT coefficients of the data. Subsequently, we keep the first few integers to implement the comparison. This implementation is achieved with the help of the most known similarity measurement, Euclidean distance. The comparison is done among all the nodes in

every cluster, with the expectation to drop some nodes out of the training process if they are similar to others in the same cluster.

Theorem 3.1 (Parseval's theorem). *Let \bar{X} be the DFT of the sequence \bar{x} . Then we have*

$$\sum_{t=0}^{n-1} |x_t|^2 = \sum_{f=0}^{n-1} |X_f|^2$$

That is, the energy in the time domain is the same as the energy in the frequency domain.

Definition 3.1 (Euclidean Distance). *The Euclidean distance formula, as its name suggests, gives the distance between two points (or) the straight line distance. Let us assume that (x_1, y_1) and (x_2, y_2) are two points in a two-dimensional plane. Here is the Euclidean distance formula:*

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

, where d is the distance between the two points.

Either way, it is decided that clusterheads will participate in every training round despite of the algorithms' results since they have to fulfill a much more meaningful task.

3.5.2 Federated Averaging Algorithm

Each node i only has access to its own local data set D_i . All nodes share a common machine learning model class and our goal is to minimize the objective function $F(x)$ that is defined as follows:

$$F(x) = \frac{1}{n} \sum_{i=1}^n F_i(x)$$

where x denotes the model parameters (the weights in our case) and $F_i(x)$ is the local objective function. Federated Averaging process is executed into two steps:

- **Federation and averaging within each cluster**

At each round, at each cluster, the nodes (that participate to federation according to the node participation protocols described above) train their local model and share their parameters (weights) with the selected clusterhead. The clusterhead collects all the

local updates including its own and implements the averaging. Afterwards, it synchronizes the nodes of the cluster with the new parameters and they continue training with the updated model. In this way, clusterhead basically plays the role of a local server within the cluster.

- **Global federation and averaging**

Global averaging is implemented between clusters. Particularly every k rounds, the clusterheads of every cluster communicate with each other exchanging their parameters. Global aggregation is executed, the new global parameters are computed and broadcasted back to every clusterhead to start the next round of training.

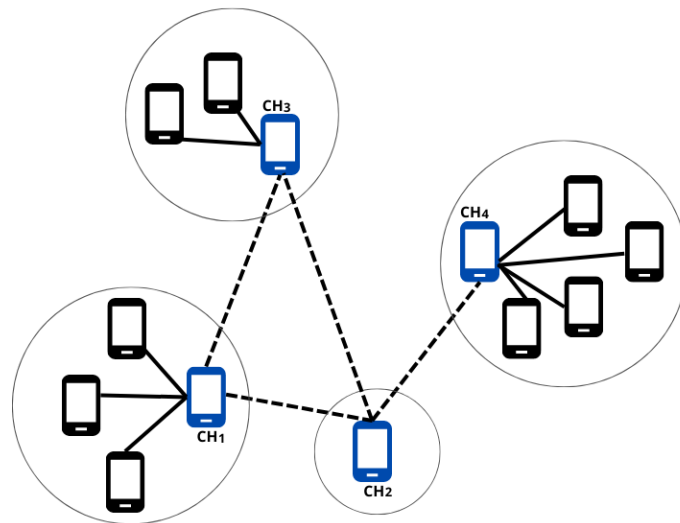


Figure 3.4: Federated Learning in clustered network

In Fig. 3.4 there is an illustration of our proposed model. The communication links between the devices of each cluster are represented with the black line. Within each cluster devices don't need to communicate with each other, but they must necessarily communicate with the clusterhead. The selected clusterheads are depicted with blue color. For the global averaging there is a backbone network that connects the clusterheads with each other which is represented with the dashed line.

Chapter 4

Experiments and Results

4.1 Experimental setup

We evaluated our proposed model conducting extensive numerical experiments. We assume a simulated network consisting of nodes organized in clusters. As we mentioned before, we test our approaches with both real and synthetic data. For the real data experiments, each node contains a unique data set of 13.147 samples and for the synthetic, it contains a data set of 10.000 samples.

The data set is then split into chunks and so the training is performed per chunk for all of them. We have selected chunks of 500 data points and as result they arise the following:

Data sets	# of samples	# of chunks
Real data	13.147	27
Synthetic data	10.000	20

Table 4.1: Data set configuration used for testing

We performed different experiments using the above data for networks with 20 and 50 nodes respectively.

1. Firstly we consider a simple scenario in which all nodes participate to federated learning within each cluster and all clusterheads participate to global averaging as described above.

2. Next we conduct an experiment where the nodes participating to federation at every round are selected randomly. Particularly we test our model with three different random participation thresholds: 0.25, 0.5 and 0.75 that implies that the 75 %, 50% and 25% of the nodes participate respectively.
3. We also study the case where the nodes participating to federation are selected based on their data similarity. Here, utilizing the protocol described in 3.5.1 , we experiment keeping the first either 5 or 10 integers of the DFT coefficients to implement the comparison and select the nodes participating.

All 6 scenarios mentioned above are repeated but this time with a global delay. Meaning that when global averaging occurs, we simulate a time delay so that not all clusterheads participate together. The participation is random and for this purpose we set a threshold (0.5). Clusterheads that don't participate in global averaging at the particular round continue training and implement the averaging locally (into the cluster).

4.2 Performance Metrics

The task to be completed in this paper can be considered a regression problem since we want to predict the upcoming value of a time-series dataset.

In several fields, regression machine learning techniques have been shown to be useful for making predictions. Evaluation of the model is one of the most important stages in machine learning studies. Comparing the trained model predictions with the actual (observed) data is the goal of the evaluation.

There is a variety of performance metrics used for this purpose such as error measures, accuracy measures, distance, similarity, dissimilarity etc. Although the main focus of this study is not as related to the performance as it is to the innovative ideas of constructing the network and implementing the federating learning, we have used two metrics to evaluate our model in each iteration.

4.2.1 Mean Squared Error(MSE)

Loss Functions

A loss function in machine learning is a metric for determining how well your ML model can forecast the actual result, or the ground truth.

The output value of our model and the predicted value based on the ground truth will serve as the inputs for the loss function. The loss, which is a measurement of how well our model predicted the result, is the result of the loss function.

When the loss is high, our model did not perform well. A low loss number indicates that our model performed really effectively.

The right loss function must be used in order to train an accurate model. For our GRU model, the loss function that is selected is MSE.

MSE

Perhaps the most basic and widely used loss function is the Mean Squared Error (MSE). The MSE is calculated by taking the difference between the predictions made by your model and the actual data, squaring it, and averaging it over the entire dataset.

Since we are always squaring the mistakes, the MSE can never be negative. The following

equation provides the official definition of the MSE:

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

Where N is the number of samples we are testing on.

The MSE is excellent for ensuring that our trained model does not contain any outlier predictions with significant mistakes since it gives more weight to these errors thanks to the squaring component of the function.

Nonetheless, the squared portion of the function amplifies errors made by our model when it makes a single really poor prediction. However, in most real-world situations, we don't really worry about these outliers and instead strive for a well-rounded model that works well enough for the majority of the data.

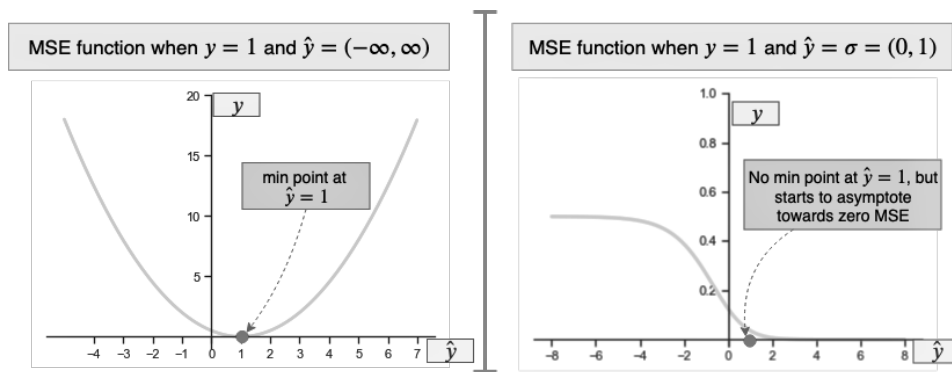


Figure 4.1: MSE example

4.2.2 Accuracy

The percentage of accurate predictions is expressed using the metric of accuracy. We determine it by dividing the total number of forecasts by the number of correct guesses.

$$Accuracy = \frac{NumOfCorrectPredictions}{TotalNumOfPredictions}$$

To find the number of correct prediction we might as well know which predictions are correct and why.

For each and every prediction of the model we calculate the following percentage:

$$Error = \frac{|predY - testY|}{|testY|}$$

,where $\text{pred}Y$ is our prediction and $\text{test}Y$ is the actual value.

If the error is lower than the class of 30 % the prediction is considered correct.

Therefore, having the number of predictions that have a low error we can divide it by our test size to find the model's final accuracy.

4.3 Results

4.3.1 An overview

In this section, we show the results obtained from all the experiments discussed in 4.1 and evaluate our proposed model. This thesis doesn't focus on achieving the highest accuracy possible as far as predictions are concerned, but our main goal is to explore the ability of the proposed heuristic in the distributed federated learning part compared against other related models that we mentioned before.

Since we cannot present all experiments and results for all the 12 different cases described earlier, we are focusing on the progress of one node in the synchronized network, considering it adequate representative for the whole procedure.

4.3.2 All-Node Participation Experiment (ANP)

For the 20 node network the progress of MSE and accuracy during the whole federation process is shown bellow:

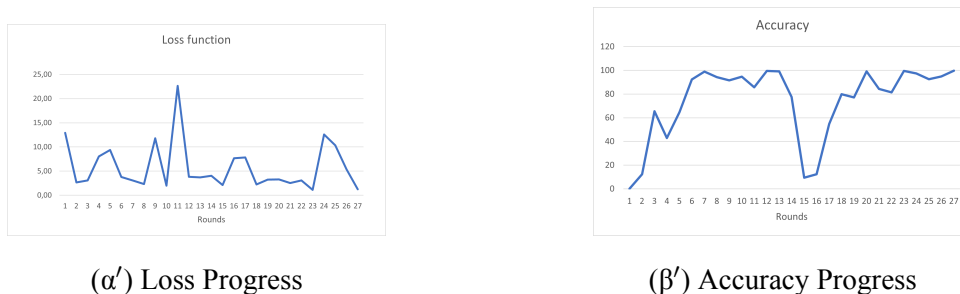


Figure 4.2: MSE and Accuracy for the representative node (20 nodes)

For the first experiment where all nodes participate in the federation process we run a 10-epoch model, but as we can see from the diagrams above our model performs pretty good, converging and achieving 1,24 loss and 99,78% accuracy in the final round.

For the 50 node network the progress of MSE and accuracy during the whole federation process is shown bellow:

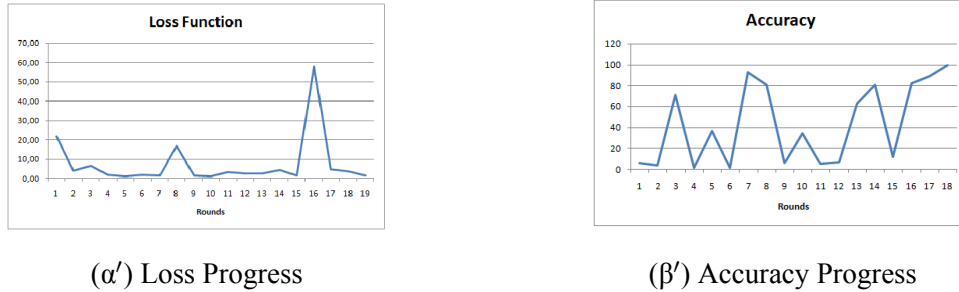


Figure 4.3: MSE and Accuracy for the representative node (50 nodes)

In the final round of training we note a loss of 1,4 and accuracy of 97%. Although the end values are perfect, to consider the model converged the course must be uniform and therefore the model does not converge to the 10-epochs model but to the 50-epochs instead.

The plots of validation loss/loss (above) and of predictive/actual values (below) for our representative node in the rounds of global averaging are presented as following :

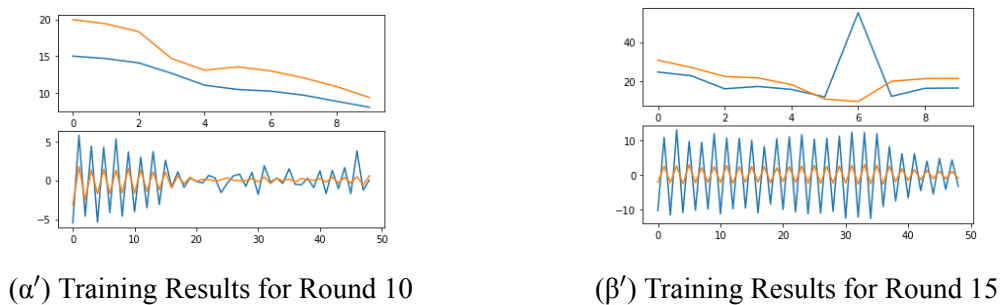


Figure 4.4: Global Averaging Effect (50 nodes)

4.3.3 Random-Node Participation Experiment (RNP)

For the 20 node network the progress of MSE and accuracy during the whole federation process is shown bellow:

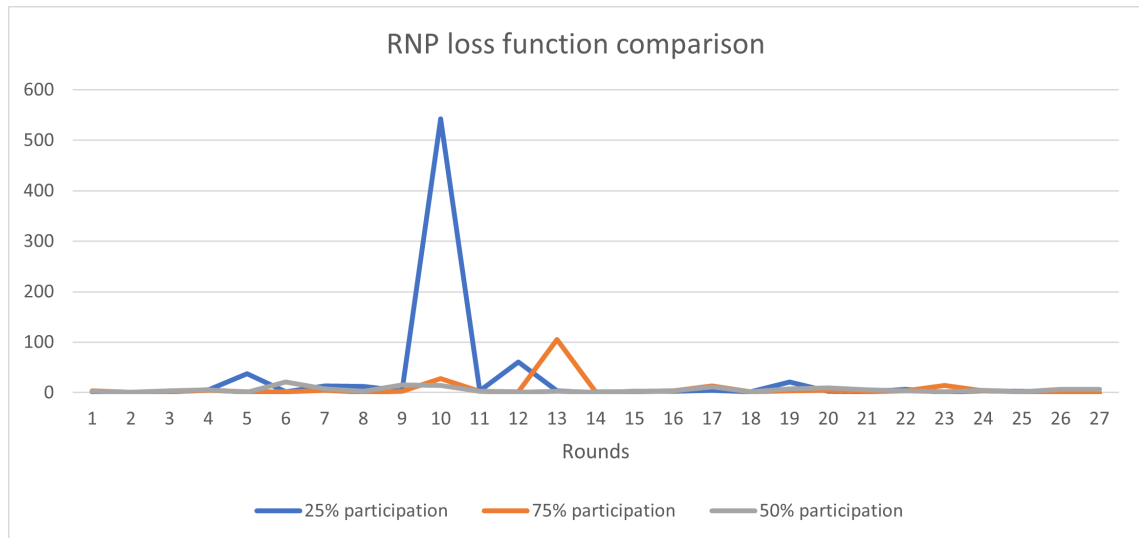


Figure 4.5: Loss progress comparison for the representative node (20 nodes)

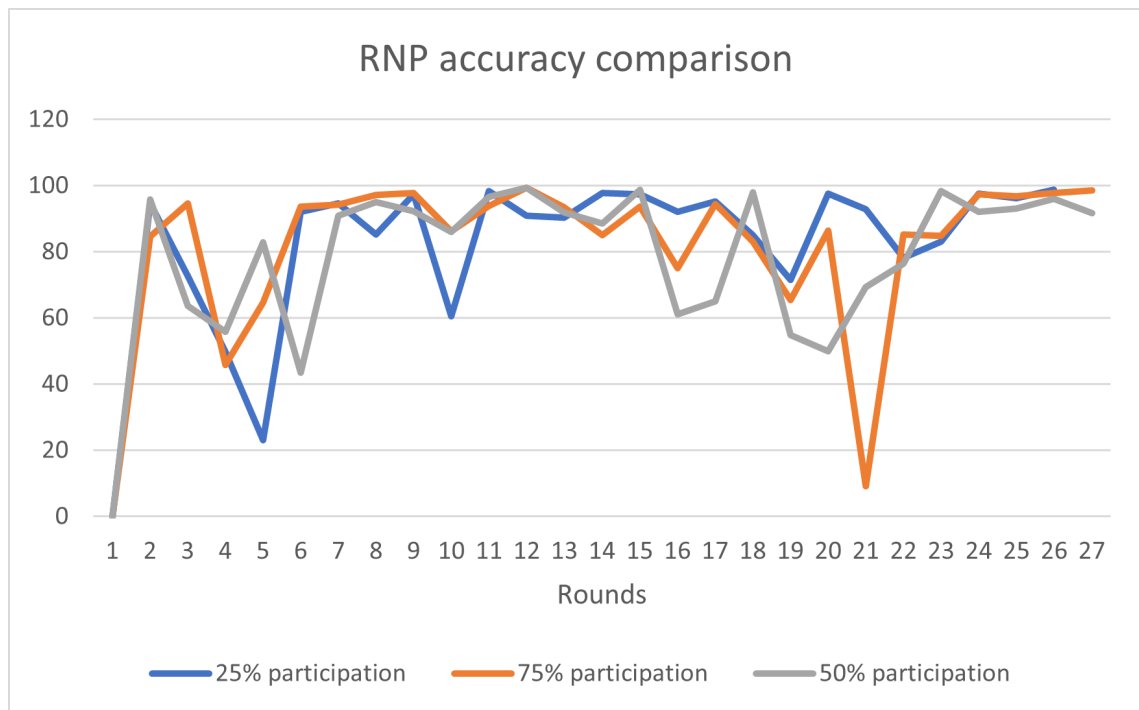
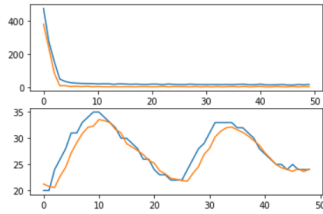


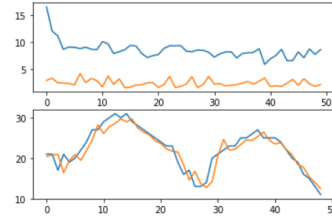
Figure 4.6: Accuracy comparison for the representative node (20 nodes)

In RNP case, our model also converges as we can notice from the diagrams above. In the final round of training the 25% node participation in the federated process model achieves 0,4 loss and 98,73% accuracy, having the best performance among the three models depicted.

The plots of validation loss/loss (above) and of predictive/actual values (below) for our representative node in the rounds of global averaging are presented as following :

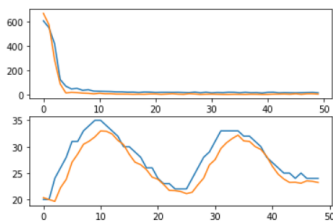


(α') Training Results for Round 10

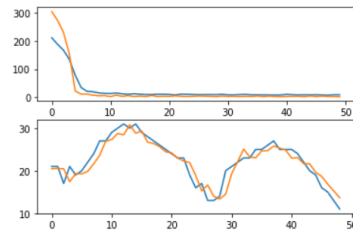


(β') Training Results for Round 23

Figure 4.7: Global Averaging Effect for 75% participation of nodes (20 nodes)



(α') Training Results for Round 10



(β') Training Results for Round 23

Figure 4.8: Global Averaging Effect for 25% participation of nodes (20 nodes)

For the 50 node network the progress of MSE and accuracy during the whole federation process is shown below:

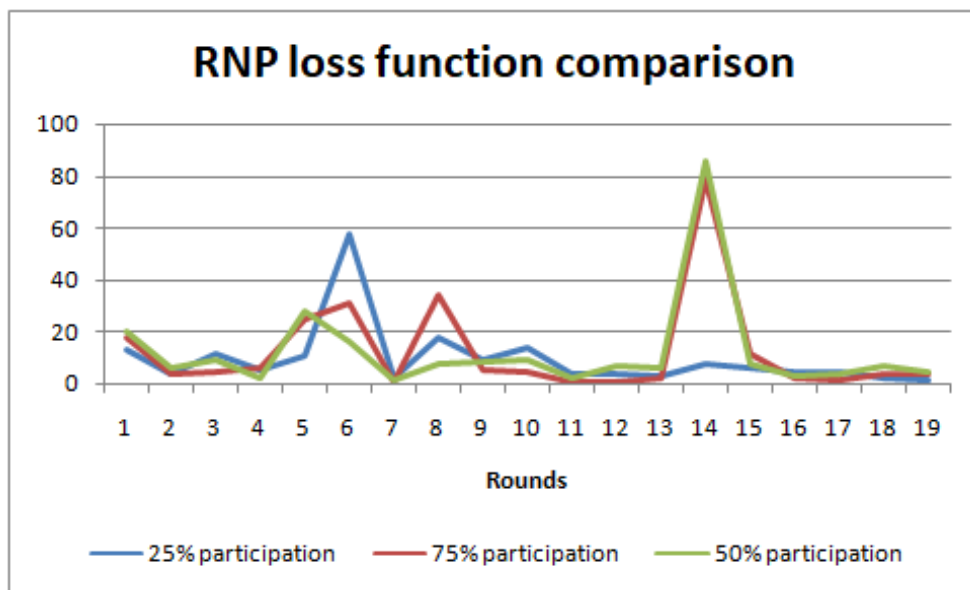


Figure 4.9: Loss progress comparison for the representative node (50 nodes)

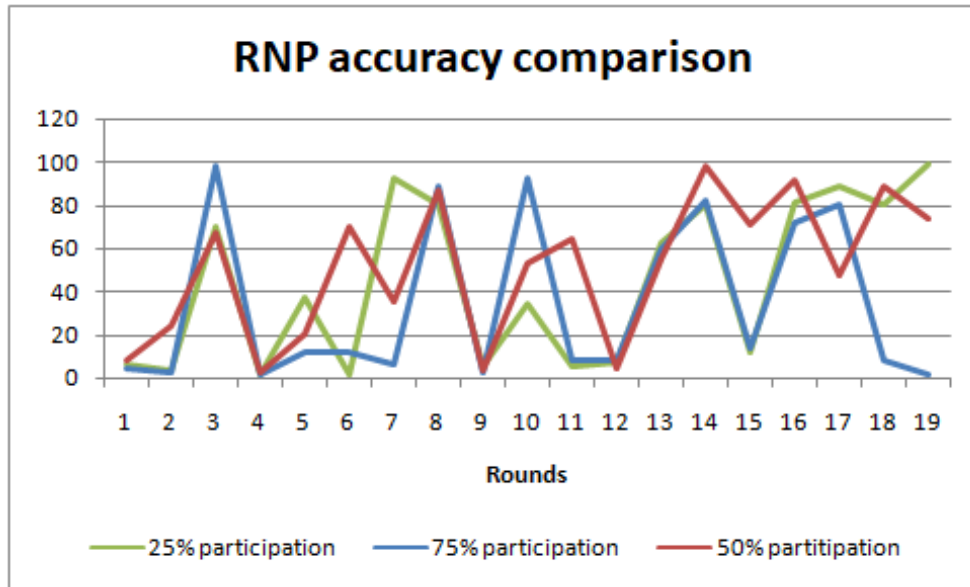
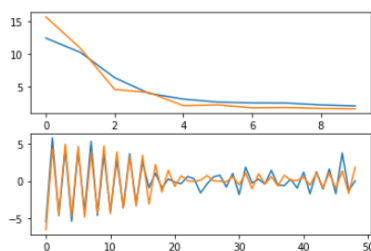


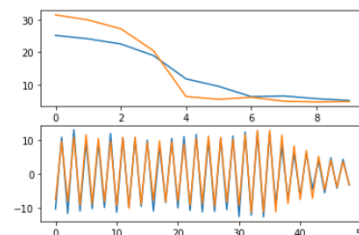
Figure 4.10: Accuracy comparison for the representative node (50 nodes)

In the final round of training we note a loss of 3,92 and accuracy of 2% for 75% node participation in the federated process, 5 and 73% for 50% of participation and lastly 1,7 and 99,2% for 25% of participation. Therefore, it is clear that in the first case we need 50-epoch model to reach convergence when in the remaining two we have remarkable results already from the 10-epoch model.

The plots of validation loss/loss (above) and of predictive/actual values (below) for our representative node in the rounds of global averaging are presented as following :

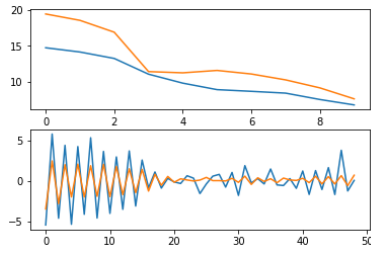


(α') Training Results for Round 10

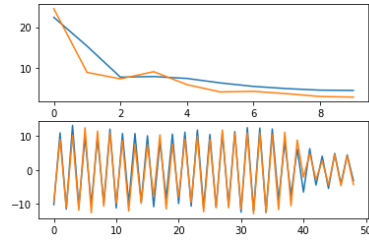


(β') Training Results for Round 15

Figure 4.11: Global Averaging Effect for 75% participation of nodes (50 nodes)

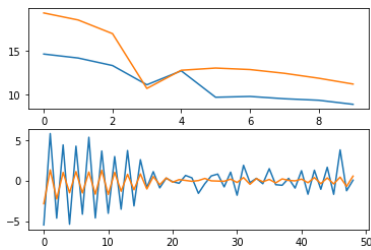


(α') Training Results for Round 10

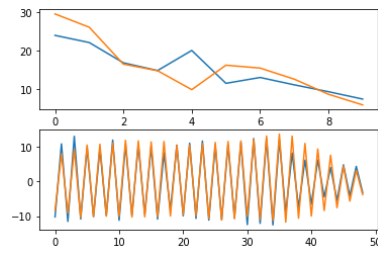


(β') Training Results for Round 15

Figure 4.12: Global Averaging Effect for 50% participation of nodes



(α') Training Results for Round 10



(β') Training Results for Round 15

Figure 4.13: Global Averaging Effect for 25% participation of nodes

4.3.4 Similarity-based-Node Participation Experiment (SNP)

For the 20 node network the progress of MSE and accuracy during the whole federation process using 5 and 10 coefficients respectively is shown bellow :

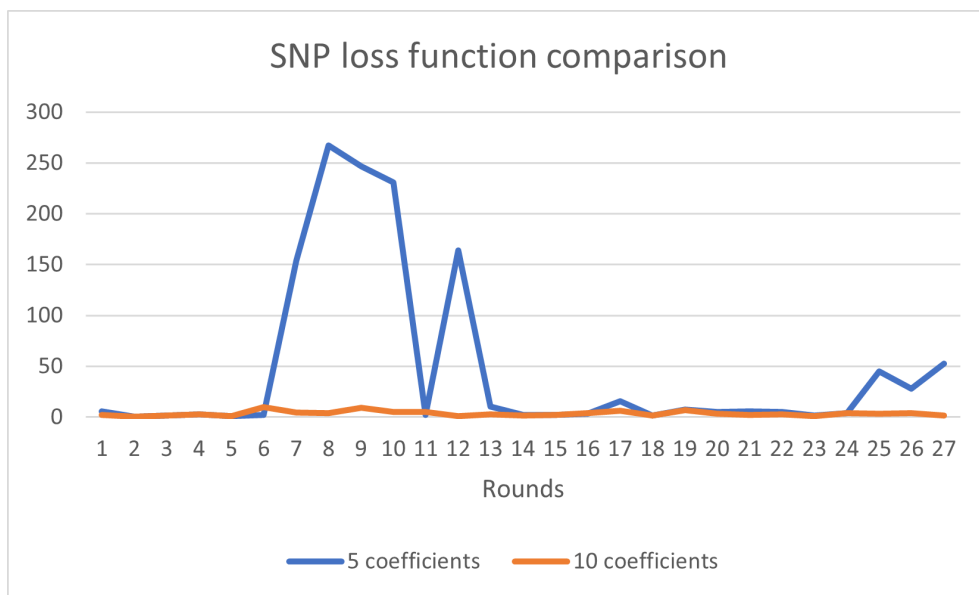


Figure 4.14: Loss progress comparison for the representative node

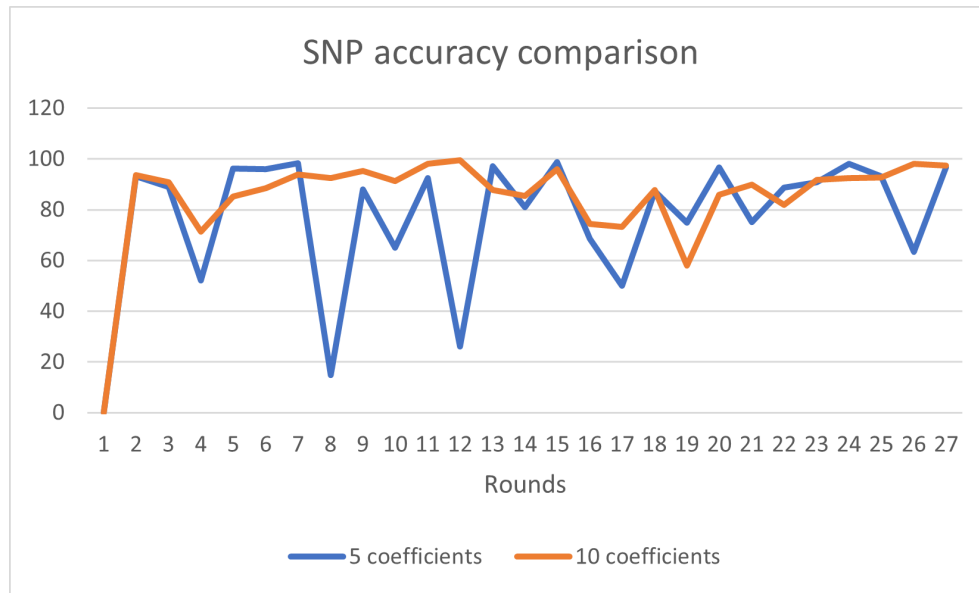
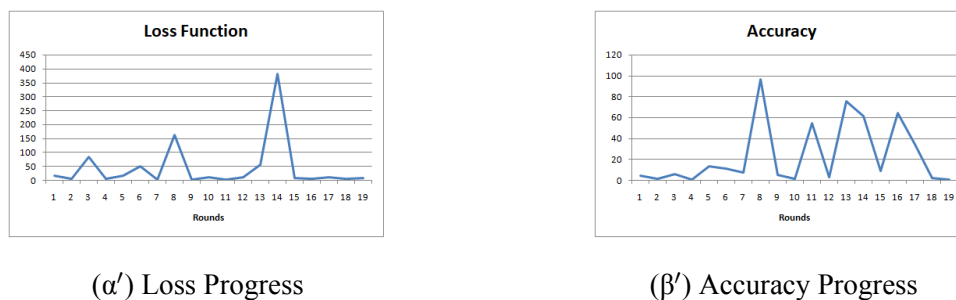


Figure 4.15: Accuracy comparison for the representative node

In the final round of training we notice a loss of 52,3 for the 5- coefficient SNP and 1,5 for the 10-coefficient SNP. Both models converge to 50-epochs model, but from the diagrams above we can clearly conclude that the 10- coefficient model performs better, achieving 99% accuracy.

For the 50 node network the progress of MSE and accuracy during the whole federation process is shown bellow:



(α') Loss Progress

(β') Accuracy Progress

Figure 4.16: MSE and Accuracy for the representative node

The experiment does not converge to 10-epochs model. While the loss ranges among low values (7,01), the accuracy is also low.

The plots of validation loss/loss (above) and of predictive/actual values (below) for our representative node in the rounds of global averaging are presented as following :

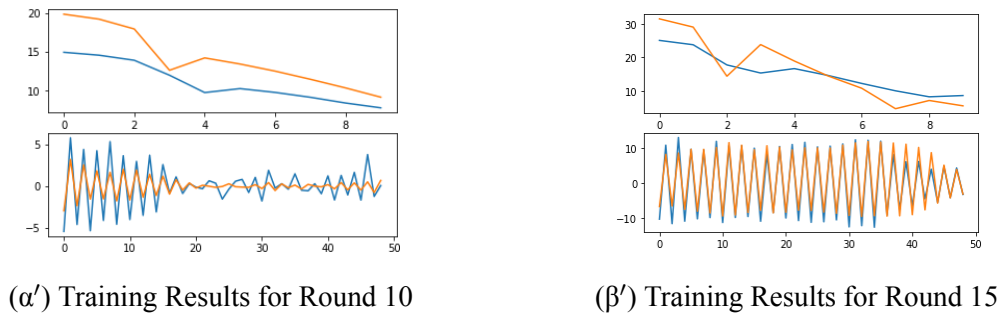


Figure 4.17: Global Averaging Effect

4.3.5 Final Conclusion

During the whole experimental process we notice that some experiments converge with a model trained with 10 epochs while others need 50 or more to achieve convergence.

The impact of global averaging in most situations can be seen uncooperative at first sight, since we are changing the calculated weights by force, but in the long run its effect is very optimistic.

All 3 models performed in the 20 nodes network converge successfully. We can clearly conclude that in the cases where fewer nodes participate in the federated learning (RNP and SNP) we achieve better scores which we expected theoretically to happen.

As an overall conclusion for the implementation for the 50-node network, we could say that the experiment with the best results is the RNP experiment with the lower percentage of node participation (RNP with 25% node participation), while the slowest convergence belongs to the RNP experiment with the highest percentage of node participation (75%).

The impact of the real time implementation, in which we run the same experiments but with an internalized delay, is not that different than the synchronized one. The experiment with the highest scores is once again the SNP and thus we can end up to the following reliable conclusion.

Concentrating our thoughts to the bigger picture, we conclude that the experiment with the most potential is the SNP one, since both the training results and the communication cost are optimal, combining quality and efficiency.

Chapter 5

Conclusions

5.1 Summary and Conclusion

In this thesis, we proposed a fully decentralized deep learning model for devices connected via wireless ad hoc networks. Our motivation was to overcome privacy-preserving problems and also reduce communication costs. For these purposes, we developed a model where there is no central server to coordinate the aggregation, but the devices are divided into clusters where the selected clusterhead is in charge of aggregating the model's updates into each cluster. Then, clusterheads (as representatives of the whole cluster) communicate with each other and exchange parameters to implement global aggregation.

The data that we used were both real and synthetic and we experimented in two different network topologies consisting of 20 and 50 nodes/devices respectively. We proposed a set of policies for the nodes' participation in federated learning: full participation, random participation and data-similarity based participation running many experiments. Results show that the proposed heuristic performs quite satisfactorily. Additionally we notice that when increasing the number of nodes, the model becomes less competitive.

5.2 Future work

This thesis motivates several directions for future work. It would be interesting to extend the results in this heuristic to more general learning settings and experiment with different deep neural networks like convolutional for image classification, LSTMs, etc. We also intend to implement our model in real world networks using sensors, where other issues may arise

relating to device synchronization, congestion aware data transfer and load balancing.

Bibliography

- [1] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.
- [2] Seyyedali Hosseinalipour, Christopher G. Brinton, Vaneet Aggarwal, Huaiyu Dai, and Mung Chiang. From federated to fog learning: Distributed machine learning over heterogeneous wireless networks. *IEEE Communications Magazine*, 58(12):41–47, 2020.
- [3] By:IBM Cloud Education. What is deep learning? <https://www.ibm.com/cloud/learn/deep-learning>.
- [4] Rene Choi, Aaron Coyner, Jayashree Kalpathy-Cramer, Michael Chiang, and John Campbell. Introduction to machine learning, neural networks, and deep learning. *Translational vision science technology*, 9:14, 02 2020.
- [5] T. Soni Madhulatha. An overview on clustering methods. 2012.
- [6] By:IBM Cloud Education. What are neural networks? <https://www.ibm.com/cloud/learn/neural-networks>.
- [7] Larry Hardesty | MIT News Office. Explained: Neural networks. <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>.
- [8] Neural networks: What they are and why they matter. https://www.sas.com/el_gr/insights/analytics/neural-networks.html#technical.
- [9] Ms. Sonali, B. Maind, and Ms. Priyanka Wankar. Research paper on basic of artificial neural network. *International Journal on Recent and Innovation Trends in Computing and Communication*, 2(1):96–100, Jan 2014.

- [10] A beginner's guide to neural networks and deep learning. <https://wiki.pathmind.com/neural-network>.
- [11] Activation functions in neural networks [12 types amp; use cases]. <https://www.v7labs.com/blog/neural-networks-activation-functions>.
- [12] Jason Brownlee. Loss and loss functions for training deep learning neural networks, Oct 2019. <https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/>.
- [13] Ravindra Parmar. Common loss functions in machine learning, Sep 2018. <https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23>.
- [14] Optimization algorithms in neural networks, Dec 2020. <https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html>.
- [15] By: IBM Cloud Education. What are recurrent neural networks?, Sep 2020. <https://www.ibm.com/cloud/learn/recurrent-neural-networks>.
- [16] Simeon Kostadinov. Understanding gru networks, Nov 2019. <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>.
- [17] Guizhu Shen, Qingping Tan, Haoyu Zhang, Ping Zeng, and Jianjun Xu. Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia Computer Science*, 131:895–903, 2018. Recent Advancement in Information and Communication Technology:.
- [18] Federated learning, Aug 2022. https://en.wikipedia.org/wiki/Federated_learning.
- [19] Akihito Taya, Takayuki Nishio, Masahiro Morikura, and Koji Yamamoto. Decentralized and model-free federated learning: Consensus-based distillation in function space, 2021. <https://arxiv.org/abs/2104.00352>.

- [20] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrede Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2019. <https://arxiv.org/abs/1912.04977>.
- [21] Priyanka Mary Mammen. Federated learning: Opportunities and challenges, 2021. <https://arxiv.org/abs/2101.05428>.
- [22] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15*, page 1310–1321, New York, NY, USA, 2015. Association for Computing Machinery.
- [23] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, may 2020.
- [24] Latif U. Khan, Walid Saad, Zhu Han, and Choong Seon Hong. Dispersed federated learning: Vision, taxonomy, and future directions. *IEEE Wireless Communications*, 28(5):192–198, 2021.
- [25] Seyyedali Hosseinalipour, Christopher Brinton, Vaneet Aggarwal, Huaiyu Dai, and Mung Chiang. From federated to fog learning: Distributed machine learning over heterogeneous wireless networks. *IEEE Communications Magazine*, 58:41–47, 12 2020.
- [26] Ihab Mohammed, Shadha Tabatabai, Ala Al-Fuqaha, Faissal El Bouanani, Junaid Qadir, Basheer Qolomany, and Mohsen Guizani. Budgeted online selection of candidate iot

- clients to participate in federated learning. *IEEE Internet of Things Journal*, PP, 11 2020.
- [27] UCI Machine Learning. Pm2.5 data of five chinese cities, Aug 2017. <https://www.kaggle.com/datasets/uciml/pm25-data-for-five-chinese-cities>.
- [28] Hong Xing, Osvaldo Simeone, and Suzhi Bi. Federated learning over wireless device-to-device networks: Algorithms and convergence analysis. *IEEE Journal on Selected Areas in Communications*, 39(12):3723–3741, dec 2021. = <https://doi.org/10.1109>
- [29] A.D. Amis, R. Prakash, T.H.P. Vuong, and D.T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 1, pages 32–41 vol.1, 2000.
- [30] Meghana Nasre, Matteo Pontecorvi, and Vijaya Ramachandran. Betweenness centrality – incremental and faster. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014*, pages 577–588, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- [31] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In David B. Lomet, editor, *Foundations of Data Organization and Algorithms*, pages 69–84, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. https://link.springer.com/chapter/10.1007/3-540-57301-1_5#citeas.