ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κατανεμημένη Ομόσπονδη Βαθιά Μάθηση σε Ad Hoc Δίκτυα Οχημάτων

Διπλωματική Εργασία

Διονύσιος-Βασίλειος Κρίτσας

Επιβλέπων: Κατσαρός Δημήτριος

Ιούλιος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κατανεμημένη Ομόσπονδη Βαθιά Μάθηση σε Ad Hoc Δίκτυα Οχημάτων

Διπλωματική Εργασία

Διονύσιος-Βασίλειος Κρίτσας

Επιβλέπων: Κατσαρός Δημήτριος

Ιούλιος 2025

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Distributed Federated Deep Learning in Vehicular Ad Hoc Networks

Diploma Thesis

Dionisios-Vasileios Kritsas

Supervisor: Dimitrios Katsaros

July 2025

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Κατσαρός Δημήτριος**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Μέλος **Θάνος Γεώργιος**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Μέλος **Ραφαηλίδης Δημήτριος**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο/Η Δηλών/ούσα

Διονύσιος-Βασίλειος Κρίτσας

Διπλωματική Εργασία

Κατανεμημένη Ομόσπονδη Βαθιά Μάθηση σε Ad Hoc Δίκτυα Οχημάτων

Διονύσιος-Βασίλειος Κρίτσας

Abstract

Στη σύγχρονη εποχή, με την αυξανόμενη κυκλοφορία οχημάτων, τα Vehicular Ad-hoc Networks (VANETs) αναδύονται ως μια σημαντική τεχνολογία που επιτρέπει την επικοινωνία και τη συνεργασία σε πραγματικό χρόνο. Ωστόσο, η δυναμική φύση των VANETs, η οποία χαρακτηρίζεται από υψηλή κινητικότητα και ασταθή συνδεσιμότητα, καθιστά τις παραδοσιακές μεθόδους ομαδοποίησης (clustering) δύσκολα εφαρμόσιμες. Για να ξεπεράσουμε αυτά τα ζητήματα, προτείνουμε ένα κατανεμημένο πλαίσιο Ομοσπονδιακής Μάθησης (Federated Learning) βασισμένο σε ομαδοποίηση για τα VANETs, όπου οι κόμβοι ομαδοποιούνται βάσει της ταχύτητας, της γεωγραφικής θέσης και του degree τους ώστε να διασφαλιστεί ο σχηματισμός μεγάλων και σταθερών clusters. Κάθε κόμβος εκπαιδεύει τοπικά ένα νευρωνικό δίκτυο και αποστέλλει τα βάρη του στον επιλεγμένο clusterhead. Οι clusterheads συγκεντρώνουν τα βάρη και υπολογίζουν τον μέσο όρο τους, τον οποίο στη συνέχεια διανέμουν εκ νέου σε όλα τα μέλη. Η προσέγγισή μας επικεντρώνεται στην επεκτασιμότητα του δικτύου και στη μείωση του απαιτούμενου επικοινωνιακού φόρτου. Μέσα από προσομοιώσεις, αποδεικνύουμε την αποτελεσματικότητα της μεθόδου ομαδοποίησης μας στη διατήρηση σταθερών ομάδων και στην ακρίβεια του μοντέλου υπό διάφορα σενάρια.

Keywords:

VANET, clustering, ML model, ad-hoc networks, federated learning, averaging, data similarity, node participation

Diploma Thesis

Distributed Federated Deep Learning in Vehicular Ad Hoc Networks

Dionisios-Vasileios Kritsas

Abstract

Nowadays, with the increased number of vehicles, vehicular ad-hoc networks (VANETs) are emerging as a critical technology enabling communication and cooperation in real-time. However, the dynamic nature of VANETs, characterized by high mobility and intermittent connectivity, makes the traditional clustering approaches difficult. In this thesis, we propose a distributed cluster-based Federated Learning framework for VANETs, where nodes are clustered based on velocity, geographical position and their degrees to ensure large and stable clusters. Each node, within the cluster, trains a neural network locally and sends its weights to the assigned cluster head. The cluster heads aggregate the weights and compute a global average that then is transmitted back to all the members. Our approach focuses on the scalability of the network and reduces the communication overhead that is required. Through simulation-based evaluations, we demonstrate the effectiveness of our clustering method in maintaining stable clusters and model accuracy under varying scenarios.

Keywords:

VANET, clustering, ML model, ad-hoc networks, federated learning, averaging, data similarity, node participation

Contents

Συντομογραφίες

VANET	Vehicular Ad-Hoc Netowrk
ML	Machine Learning
FL	Federated Learning
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
MSE	Mean Squared Error
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
ANP	All Node Participation
RNP	Random Node Participation
SNP	Similarity based Node Participation
CNN	Chosen Neighbor Node
CN	Current Node
PCH	Potential Cluster Head
DHCV	D-hop clustering algorithm
MLP	Multiplelayer Perceptron
V2V	Vehicular-To-Vehicual
MEC	Multi-Access Edge Cluster

Chapter 1

Introduction

The rapid increase in the number of vehicles on the road has led to the development of technologies that are based on Vehicular Ad-Hoc Networks (VANETs). With these technologies, vehicles can cooperate with each other in real-time and exchange data that enable applications such as traffic management, safety alerts, and autonomous driving. Despite the promising solutions, VANETs face significant challenges due to their dynamic nature, where high mobility and intermittent connectivity can cause frequent topology changes, making traditional machine-learning approaches non-optimal. Traditionally, each model is trained locally, known as machine leaning (ML). However, since most vehicles have limited computational power and handle a large amount of privacy-sensitive data, Federated Learning (FL) has emerged as a technique for distributed model training across devices.

FL is a machine learning technique where multiple entities (clients) collaboratively train a local model without the sharing of raw data. Each round of the model consists of two main steps: 1. **Local training** where each node trains its local model using its own data set. 2. **Global aggregation** where the main server collects all the local parameters, computes a global average and broadcasts the new aggregated version to all nodes to begin the next round of training.

FL is the best approach to our problem, considering its distributed nature and the fact that the raw data is never exposed to the main server. However, in traditional FL, a server is required to collect and aggregate the data, which is limited and prohibitive in networks where the nodes are expected to work in a decentralized manner and can only communicate with the adjacent ones.

In this thesis, we propose a fully decentralized cluster-based FL framework suitable for vehicular ad-hoc networks that function without a main server. More specifically, our network is

divided into clusters, electing cluster-heads based on the geographical position, velocity, and degree of each node. Each clusterhead acts as a local server and is responsible for the local averaging of the models and the data exchange between the clusters. Clusterheads in turn communicate with each other to perform a federated averaging of the received parameters and produce a global model that is transmitted back to all nodes. To further reduce communication overhead, our framework uses data similarity analysis among cluster members. In each round if two nodes of the same cluster have similar data, the clusterhead is responsible for choosing one of them to send model parameters based on conditions such as velocity or position.

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 Neural Network

A neural network is a type of machine learning model inspired by the structure and function of the human brain. It consists of layers of interconnected units called neurons, which process and pass information through the network. Each neural network has an input and an output layers as well as a number of hidden layers in between. The input layer is where we provide the data to the neural network. The data is then passed through the network via connections that have weights that adjust as the network learns. Finally, the data are sent to the output layer, where the final output is computed.

Neural networks are particularly good at identifying patterns and making predictions, even from complex or unstructured data such as images, audio, or time series. They are widely used in fields such as image recognition, speech processing, and forecasting.

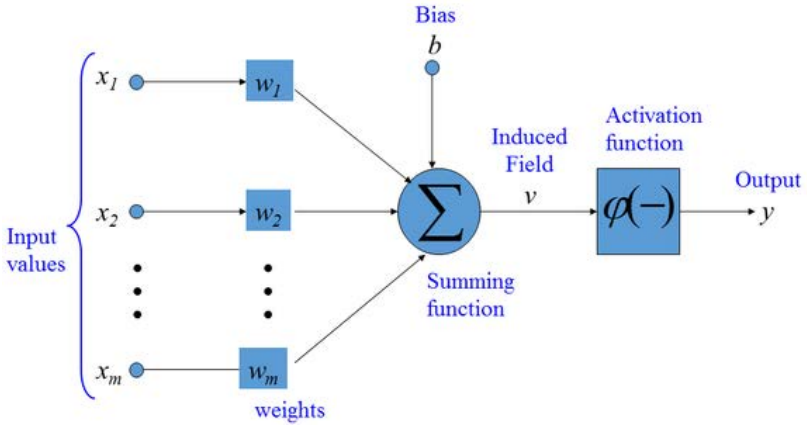


Figure 2.1: A Single Neuron[1]

2.1.1 Multi-Layer Perceptrons MLPs

A multi-layer perceptron (MLP) is a type of artificial neural network consisting of multiple layers of neurons. The neurons in the MLP typically use nonlinear activation functions, allowing the network to learn complex patterns in data. MLPs are significant in machine learning because they can learn nonlinear relationships in data, making them powerful models for tasks such as classification, regression, and pattern recognition.[2]

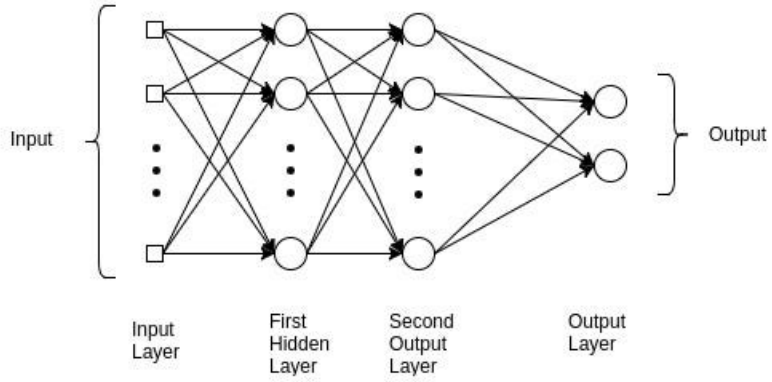


Figure 2.2: MLP Architecture[3]

2.1.2 Recurrent Neural Networks RNNs

Recurrent Neural Networks are a specific type of neural networks that are used for time series problems. Their characteristic is that they have ”memory”. This means that they use information from previous data to predict the current output. For RNN training, we use the backpropagation through time algorithm.

Problems that RNNs have are the vanishing and exploding gradients and the difficulty of learning long-term dependences. Vanishing gradients means that the gradients may get too close to zero, preventing the model from learning. Similar is the exploding gradients problem but with the weights becoming too big. GRU (Gated Recurrent Units) and LSTM (Long-Short Term Memory) solve this problem by introducing gates to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.

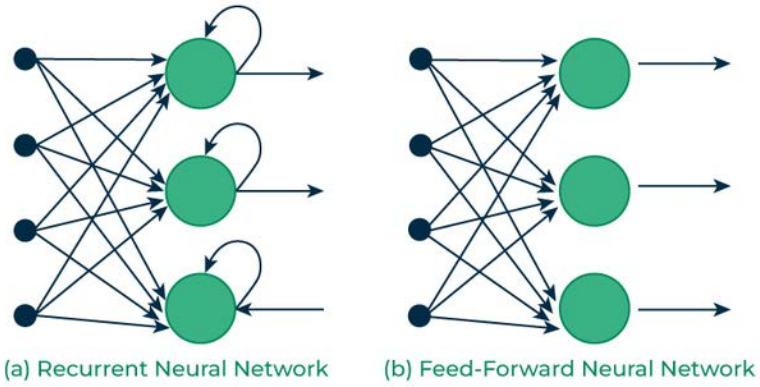


Figure 2.3: RNN Architecture[4]

2.1.3 GRU

Gated Recurrent Units (GRUs) are an improved version of RNN introduced by Cho et al. in 2014. The core idea behind GRUs is to use gating mechanisms to selectively update the hidden state at each time step allowing them to remember important information while discarding irrelevant details. GRUs aim to simplify the LSTM architecture by merging some of its components and focusing on just two main gates: the update gate and the reset gate.[5]

2.1.4 GRU Architecture

1. **Input layer:** The input layer takes in sequential data, such as a sequence of words or a time series of values, and feeds it into the GRU.

2. **Hidden layer:** The hidden layer is where the recurrent computation occurs. At each time step, the hidden state is updated based on the current input and the previous hidden state. The hidden state is a vector of numbers that represents the network's "memory" of the previous inputs.

3. **Reset gate:** The reset gate determines how much of the previous hidden state to forget. It takes as input the previous hidden state and the current input, and produces a vector of numbers between 0 and 1 that controls the degree to which the previous hidden state is "reset" at the current time step.

4. **Update gate:** The update gate determines how much of the candidate activation vector to incorporate into the new hidden state. It takes as input the previous hidden state and the

current input, and produces a vector of numbers between 0 and 1 that controls the degree to which the candidate activation vector is incorporated into the new hidden state.

5. **Candidate activation vector:** The candidate activation vector is a modified version of the previous hidden state that is "reset" by the reset gate and combined with the current input. It is computed using a tanh activation function that squashes its output between -1 and 1.

6. **Output layer:** The output layer takes the final hidden state as input and produces the network's output. This could be a single number, a sequence of numbers, or a probability distribution over classes, depending on the task at hand.[6]

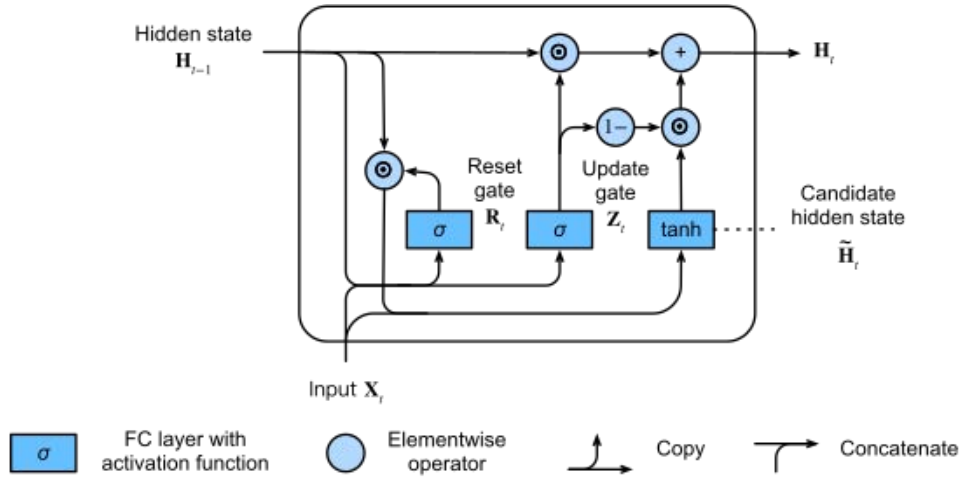


Figure 2.4: GRU[7]

The reset gate $\mathbf{R}_t$ and update gate $\mathbf{Z}_t$ are computed as follows:

$$\mathbf{R}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xr} + \mathbf{H}_{t-1}\mathbf{W}_{hr} + \mathbf{b}_r)$$
$$\mathbf{Z}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xz} + \mathbf{H}_{t-1}\mathbf{W}_{hz} + \mathbf{b}_z)$$

In a conventional RNN, we would have a hidden state update of the form:

$$\mathbf{H}_t = \tanh\left(\mathbf{X}_t\mathbf{W}_{xh} + \mathbf{H}_{t-1}\mathbf{W}_{hh} + \mathbf{b}_h\right)$$

We use tanh to ensure that the values of the hidden states remain in the interval (-1, 1). If we want to be able to reduce the influence of the previous states we can multiply $\mathbf{H}_{t-1}$ with $\mathbf{R}_t$ elementwise. Whenever the entries in the reset gate Rt are close to 1, we recover a conventional RNN. For all entries of the reset gate Rt that are close to 0, the hidden state is the result of an

MLP with Xt as input. Any pre-existing hidden state is thus reset to defaults. This leads to the following candidate hidden state (it is a candidate since we still need to incorporate the action of the update gate)

$$\tilde{\mathbf{H}}_t = \tanh\left(\mathbf{X}_t \mathbf{W}_{xh} + \left(\mathbf{R}_t \circ \mathbf{H}_{t-1}\right) \mathbf{W}_{hh} + \mathbf{b}_h\right)$$

Next we need to incorporate the effect of the update gate $\mathbf{Z}_t$. This determines the extent to which the new state $\mathbf{H}_t$ is just the old state $\mathbf{H}_{t-1}$ and by how much the new candidate state $\tilde{\mathbf{H}}_t$ is used. The gating variable $\mathbf{Z}_t$ can be used for this purpose, simply by taking elementwise convex combinations between both candidates. This leads to the final update equation for the GRU

$$\mathbf{H}_t = \mathbf{Z}_t \circ \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \circ \tilde{\mathbf{H}}_t$$

Whenever the update gate $\mathbf{Z}_t$ is close to 1, we simply retain the old state. In this case the information from $\mathbf{X}_t$ is essentially ignored, effectively skipping time step t in the dependency chain. In contrast, whenever $\mathbf{Z}_t$ is close to 0, the new latent state Ht approaches the candidate latent state $\tilde{\mathbf{H}}_t$. These designs can help us cope with the vanishing gradient problem in RNNs and better capture dependencies for time series with large time step distances. In summary, GRUs have the following two distinguishing features:

- **Reset gates help capture short-term dependencies in time series**
- **Update gates help capture long-term dependencies in time series**

2.2 Federated Learning

Federated learning is a type of machine learning in which the model is trained across multiple devices or servers (clients) that each have local data, without ever sending data to a central server. The main idea is that each device trains the model locally on its own data and only sends the weights to a central server. The central server then aggregates the weights and creates a global model which it sends to all clients. Federated Learning achieves privacy and bandwidth efficiency.

2.2.1 Training Process

1. Local machine learning (ML) models are trained on local heterogeneous datasets.

2. The parameters of the models are exchanged between the devices periodically. In many models, these parameters are encrypted before exchanging. Local data samples are not shared. This improves data protection and cyber security.

3. A shared global model is built.

4. The characteristics of the global model are shared with local data centers to integrate the global model into their ML local models.[8]

2.2.2 Chalenges of Federated Learning

- **Communication overhead** is a major bottleneck. Federated learning requires frequent exchanges of model updates between devices and a central server. For example, training a large neural network (e.g., ResNet-50) across thousands of devices could generate terabytes of data traffic, straining bandwidth and increasing costs. Devices with unstable connections (e.g., smartphones in areas with poor coverage) may drop out, delaying training. Techniques like model compression or reducing update frequency help, but they risk losing precision or slowing convergence. Developers must balance efficiency and model quality, often requiring custom protocols tailored to their hardware constraints.

- **Data heterogeneity** complicates model convergence. In federated settings, data distributions vary widely across devices. For instance, a keyboard app might collect user-specific typing patterns, leading to non-IID (non-independent and identically distributed) data. This can cause the global model to perform poorly on individual devices. A health app trained on data from diverse demographics might fail to generalize. Solutions like Federated Averaging (FedAvg) struggle with skewed data, and advanced methods (e.g., adaptive optimization or personalized models) add complexity. Developers must test robustness across diverse data splits and mitigate bias through careful sampling or regularization.

- **Security and privacy risks** persist despite data remaining on-device. Model updates can leak sensitive information; for example, gradient updates in image classification might

reveal identifiable features through inversion attacks. Malicious actors could also submit poisoned updates to manipulate the global model (e.g., spam filters being tricked to allow harmful content). While techniques like differential privacy or secure aggregation (e.g., encrypting aggregated updates) help, they introduce trade-offs. Adding noise for privacy degrades model accuracy, and cryptographic protocols increase computation time. Developers must implement safeguards without undermining performance, often requiring rigorous adversarial testing and layered defenses.[9]

2.2.3 Types of Federated Learning

There are 3 different types of Federated Learning based on model architecture:

- **Centralized Federated Learning:** A central server aggregates updates from client models.

- **Distributed Federated Learning:** No central server, clients directly communicate with each other to exchange model updates.

- **Heterogeneous Federated Learning:** Clients can use different model architectures.

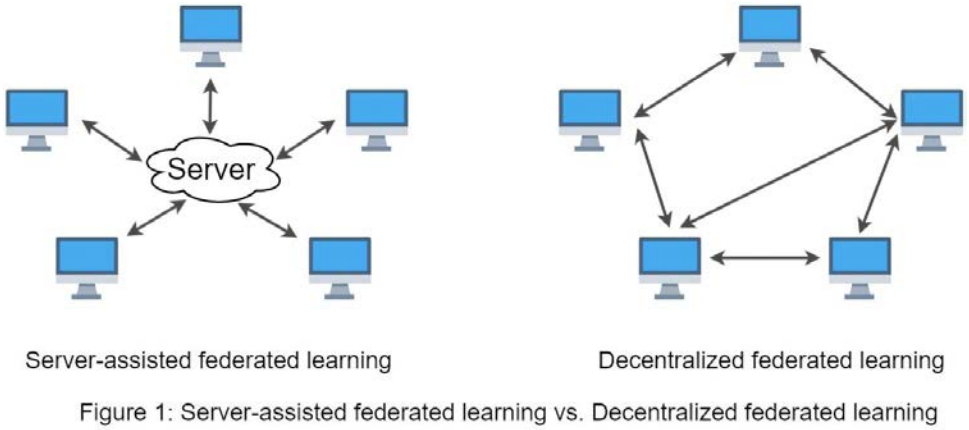


Figure 1: Server-assisted federated learning vs. Decentralized federated learning

Figure 2.5: [10]

2.3 Vehicular Ad-hoc Network (VANET)

2.3.1 Ad-hoc Network

An Ad Hoc Network is a decentralized type of wireless network that doesn't rely on pre-existing infrastructure, such as routers or access points. Instead, each node in the network participates in routing by forwarding data for other nodes. The nodes communicate directly with each other, which makes the network highly flexible and able to be set up quickly, particularly useful in situations where establishing a traditional network infrastructure is not feasible.

Ad Hoc Networks are important because they provide a flexible, scalable, and cost-effective solution for wireless communication. They are particularly useful in emergency situations, military operations, and temporary setups where traditional infrastructure is unavailable or impractical. The ability to self-configure and self-heal makes Ad Hoc Networks resilient and adaptable to changing network conditions and topologies. They also enable peer-to-peer communication, facilitating direct data transfer between devices without the need for central coordination.[11]

2.3.2 Ad-hoc Network Examples and Use Cases

There are several real-world examples and use cases of Ad Hoc Networks, such as:

- **Disaster Recovery:** In the aftermath of natural disasters, Ad Hoc Networks can be quickly deployed to facilitate communication among rescue teams when traditional infrastructure is damaged or unavailable.

- **Military Operations:** Military units use Ad Hoc Networks to maintain communication in the field without relying on fixed infrastructure, which enhances mobility and flexibility.

- **Conferences and Events:** Temporary Ad Hoc Networks can be set up at large events or conferences to provide attendees with Wi-Fi access without needing a permanent network setup.

- **Vehicular Networks:** Ad Hoc Networks enable communication between vehicles (VANETs), supporting applications like traffic management and accident avoidance.[11]

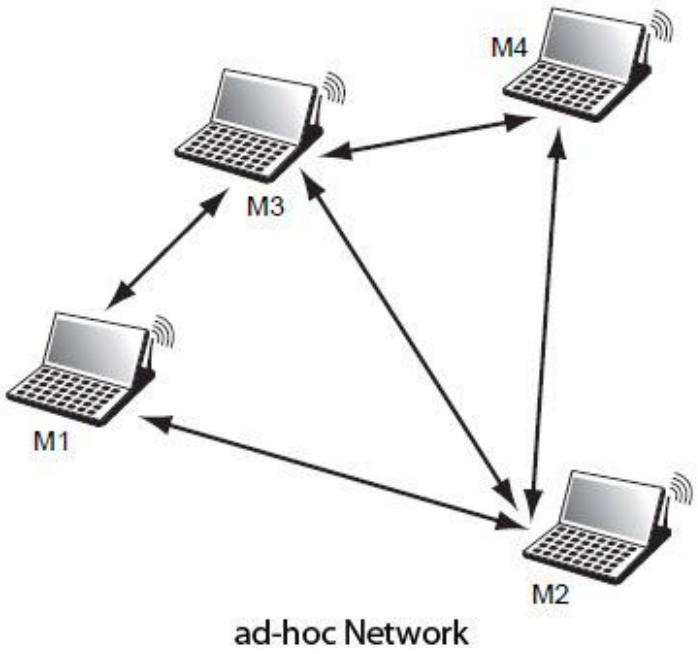


Figure 2.6: Ad-hoc Network[12]

2.3.3 Mobile Ad hoc Network (MANET)

A mobile ad hoc network (MANET) is a network of wireless mobile devices. A node can be a laptop, a PDA, a mobile phone, or any mobile device with the ability to communicate with other devices. The topology of the network keeps on changing over time as nodes may move about, some new nodes join the network, or some other nodes disengage themselves from the network. The network is created, managed, and organized purely by the nodes themselves without the assistance of any centralized third party or fixed infrastructure. Consequently, co-operation of the nodes among themselves is the platform on which this network is built. A node not only uses the network for communicating with other nodes, but also supports the network by performing routing functions. A node that wants to communicate with another node which is not within its communication range takes the help of the intermediate nodes to relay its message. MANETs have distinct advantages over traditional networks in that they can easily be set up and dismantled, apart from providing flexibility as the nodes are not tethered.[13]

2.3.4 VANET

Vehicular Ad-Hoc Network (VANET) is a sub-class of Mobile Ad-Hoc Network (MANET) that provides communication among nearby vehicles and roadside infrastructure. This type of

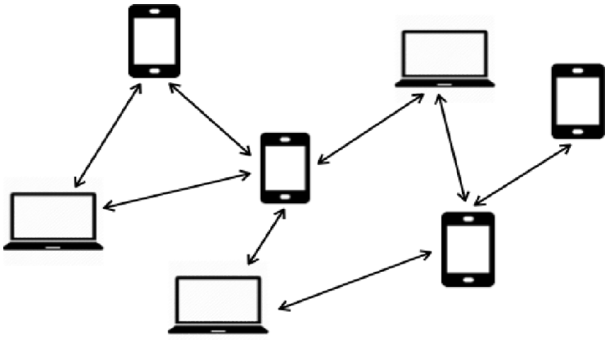


Figure 2.7: Mobile Ad hoc Network (MANET)[14]

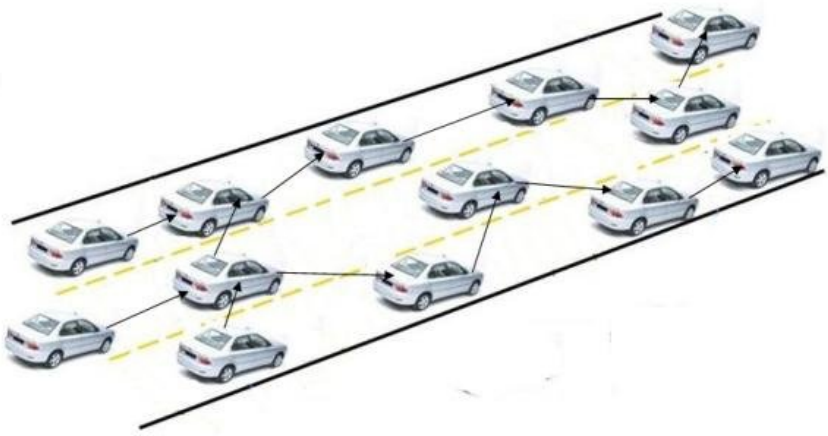


Figure 2.8: Vehicular Ad-Hoc Network (VANET)[16]

network uses vehicles as mobile nodes that belong to a self-organizing network without prior screening or knowledge of each other's presence. The network turns every participating vehicle into a wireless router or node, allowing vehicles in a distance of approximately 100 to 300 m from each other in urban scenario to connect and create a network with a wide range. Nodes may intermittently fall out of the signal range and can join in, thereby dynamically establishing connections between the vehicles such that an inter network is created.

Some of its unique characteristics like geographically constrained topology, predictable mobility and vehicle density, varying channel capacity, etc. constitute VANET as a distinct research field in MANET.[15]

2.4 Related Work

In [17] the authors propose a new architecture for vehicular FL and corresponding learning and scheduling processes. The architecture utilizes vehicular-to-vehicular(V2V) resources to

bypass the communication bottleneck where clusters of vehicles train models simultaneously and only the aggregate of each cluster is sent to the multi-access edge (MEC) server. The cluster formation is adapted for single and multi-task learning, and takes into account both communication and learning aspects.

In [18] they develop a D-hop clustering algorithm, called DHCV is presented which organizes vehicles into non overlapping clusters which have adaptive sizes according to their respective mobility. The D-hop clustering algorithm creates clusters in such a way that each vehicle is at most D hops away from a cluster head. To construct multi-hop clusters, each vehicle chooses its cluster head based on relative mobility calculations within its D-hop neighbours.

In [19] an algorithm is proposed based on a hierarchical organization of the underlying static ad-hoc network along with a generic principled way to select the clients that will participate in the federation based on the similarity of the data they produce.

Article [20] proposes an energy-efficient distributed clustering protocol for wireless sensor networks, based on a metric for characterizing the significance of a node, w.r.t. its contribution in relaying messages. The protocol achieves small communication complexity and linear computation complexity.

This paper [21] in context of clustering in vehicles, develops a pair of algorithms, namely, sociological pattern clustering (SPC) and route stability clustering (RSC), the latter being a specialization of the former that exploits, for the first time in the relevant literature, the “social behavior” of vehicles, i.e., their tendency to share the same/similar routes. Both methods exploit the historic trajectories of vehicles gathered by roadside units located in each subnetwork of a city and use the recently introduced clustering primitive of virtual forces. The mobility, i.e., mobile patterns of each vehicle, is modeled as semi-Markov processes.

Another related study [22] models an IoBT network as a network with multiple layers and employs the concept of domination for multilayer networks. Explaining why it is necessary to depart from old works designed for single layer networks they establish the computational complexity of the problem, and design a distributed algorithm for computing connected dominating sets with small cardinality.

Our study extends the federated learning on Vehicular Ad-Hoc Networks (VANETs) of [17] to be completely distributed by removing the central base stations and using a hierarchical clustering algorithm based on [18] to select cluster heads that act as the base stations to aggregate the FL model. [22]

Chapter 3

Clustering and Federated Learning Approach

3.1 Clustering

3.1.1 D-hop clustering algorithm

Our goal in this section is to represent the ad hoc network as a set of clusters suitable for reducing communication among nodes.

The algorithm

Vehicles periodically advertise their information using beacon messages. After the formation of the clusters, each node should either be a CH or at most D-hop away from its chosen CH. This is what we call a D-hop cluster. Our proposed algorithm will produce stable and large clusters depending on the vehicles' mobility.

The algorithm is executed for D rounds. In each round, every node selects as its potential cluster head the node within its corresponding multi-hop neighborhood that has the lowest relative mobility.

More specifically, to further illustrate our algorithm, we define the following terms:

1. **Current Node (CN):** A node that tries to find and select a CH up to D-hops away.

2. **Chosen Neighbor Node i (CNN_i):** It is a direct neighbor selected by CNN_{i-1}. CNNi is selected as the direct neighbor with least relative mobility to CNN$_{i-1}$. CNN_0 is CN.

3. **Degree:** degree of connectivity of a CNN_i.

4. **Potential Cluster Head i (PCH_i):** it is the CNN_j with the maximum degree among the set of the previous CNNs of the node i.

Each CN constructs two arrays of information, CNN and PCH of D size containing the CNN_i and the PCH_i of each round i.For example, CNN_1 denotes the chosen node from the 1st round with the least relative mobility.The algorithm consists of 3 phases, the **Initialization**, the **RelativeMax** and the **Merging**.

Initialization:

At the beginning, each node sets its PCH_0 and CNN_0 as its own id.

RelativeMax:

Each vehicle broadcasts a beacon message including its speed, location and degree information to all of its 1-hop neighbors. In the first round, once a node has received beacons from all neighboring nodes, it calculates its relative mobility with them and chooses the vehicle with the least relative mobility. If the chosen node has a better degree than the PCH of the previous round, or if they have equal degrees but lower velocity, then PCH_1 is set as the CNN_1 else PCH_1 is set as PCH_0. For the rest of the rounds, each node expects to receive a message from PCH_{i-1}, which includes the node that PCH_{i-1} selected in the $i-1$ round and follows the same logic as above to set its PCH_i. For example, node X in round 2, expects a message from PCH_1 that includes its PCH_1. PCH_1's PCH_1 is at most 1 hop away from PCH_1, which is at most 2 hops away from X.

There are certain scenarios where the following exceptions should be considered:

Exception 1: the algorithm generates CM that has no link to its desired CH without passing through another cluster. In this case, CM joins this cluster if the D-hop limitation can be satisfied.

Exception 2: the algorithm generates CH which has already been selected as CM in another cluster. This means CH did not select itself as CH. In this case, CH and its connected CMs introduce themselves as CMs and select CH of the constructed cluster as their own CH. If the D-hop condition cannot be satisfied with switching CH to CM, CH remains as such and forms a cluster.

Exception 3: the algorithm select CH that has no CMs. In this case, the CH introduces itself as CM and selects CNN_1 which has the least relative mobility with it. If CNN_1 is CH, CM selects it as CH, unless it selects the $CNN_1 s$ CH as its own CH.

Merging:

In this phase, every CH tries to merge with another cluster, if it contains less than K members. In that case, CH joins the second best PCH, only if all nodes can satisfy the D-distance limitation. If the D-hop condition cannot be satisfied the cluster remains as is.

Mobility metrics

Mobility metrics can be used to characterize the mobility level of vehicles. We assume each vehicle knows its speed and position through GPS. Every vehicle broadcasts a beacon message with speed and location information to its neighboring nodes at every beacon interval. Whenever a neighboring vehicle gets the beacon frame, it determines the relative mobility and chooses the node that has the least relative mobility with it. In our algorithm, to calculate the relative mobility between vehicles, each node calculates the speed, location, and degree differences with all its neighbors.

Let $\boldsymbol{D_{XY}}$ denote the distance between node X and node Y (one of the neighboring nodes of X)

$$D_{xy} = \sqrt{(x_x - x_y)^2 + (y_x - y_y)^2}$$

where x_i and y_i represent the coordinates of node i.

Let $\boldsymbol{V_{XY}}$ denote the velocity difference between node X and node Y and $\boldsymbol{\theta}$ is the velocity vector angle between the two nodes.For example, $\boldsymbol{\theta}$ is 0 if they are moving in the same direction.

$$V_{xy} = |V_x \cdot \cos\Theta - V_y \cdot \cos\Theta|$$

After D_{XY} and V_{XY} computations, vehicle X calculates the relative mobility with its neighboring nodes as follows:

$$RM_{XY} = a \cdot D_{xy} + b \cdot V_{xy} + c \cdot (\text{Deg}(x) - \text{Deg}(y))$$

where $\boldsymbol{RM_{XY}}$ is the relative mobility between X and Y. a, b and c are parameters defined as distance, speed and degree factors, respectively. These values are weights to control the efficiency of the metrics.

3.2 Federated Learning

For the purposes of this thesis, we chose to work with a single model, specifically a GRU, which will be presented in detail below. Our approach does not impose any constraints on the

type of neural model used, as long as it is trained through weight exchange.

3.2.1 Training Process

Our federated learning approach can be summarized through the following steps:

1. Each node trains its own model locally.

2. After each training round, nodes send their model weights to their respective cluster head.

3. Upon receiving the trainable parameters from all cluster members, the cluster heads compute the local average.

4. The cluster heads then exchange their local averages every n rounds, to compute a global average of the trainable parameters, which is then distributed back to all nodes.

After these steps are completed, all nodes in the network share the same set of trainable parameters and the process can begin again.

3.2.2 Communication Protocols

Since the cluster head functions as a local server for each cluster, receiving all local parameters, it is important to minimize the communication overhead within the cluster. To address this and improve the scalability and practicality of our approach in real-world scenarios, we introduce two communication protocols that help reduce the amount of data exchanged within the cluster.

Random Nodes Participation

In the first protocol, the cluster head randomly selects a percentage of nodes to send their weights for the current round. This method is simple and easy to implement and effectively reduces the communication cost within the cluster.

Although this technique is promising, it introduces certain limitations in terms of model quality, as it does not account for the relative importance of each node in the training process.

Data Similarity-Based Participation

In our more advanced protocol, the cluster head determines which nodes will participate in each round based on the uniqueness or similarity of their weights. Identifying similarities among nodes is a challenging task due to the volume and nature of the data involved. To address this, we use the Discrete Fourier Transform (DFT) to convert time sequences into the frequency domain. By keeping only the first few frequency components, we capture the characteristics of each dataset while significantly reducing its complexity.[23]

More specifically, before each round, nodes send a list of integers corresponding to the selected DFT coefficients. The cluster head then compares them using the Mean Square Error (MSE). If the similarity between two nodes falls below a defined threshold ϵ, the cluster head selects only one of them, based on the smallest velocity difference, to contribute its weights in the upcoming training rounds.

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 (Mean Square Error). *The Mean Square Error (MSE) is a metric to measure the average of the squares of the errors, that is, the average squared difference between the estimated values and the actual value. Let us assume that $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are pairs of actual and predicted values. The MSE is given by:*

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2$$

where MSE represents the average squared error between the predicted and actual values.

Either way, it is decided that cluster heads will participate in every training despite the algorithms' results since they do not have to send their parameters in the cluster members.

3.2.3 Federated Averaging Algorithm

Each node i trains the same machine learning model with the same configurations (e.g batch size, epochs, loss function, etc.). 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 and $Fi(x)$ the local objective function. Our Federated Average process is divided into two steps.

1. **Local federation and averaging**

   In every round, all cluster members train their local model and send the weights in the cluster head. The cluster head determines which nodes will participate based on the protocols described above, and then calculates the local average of the collected weights.

2. **Global federation and averaging**

   Every n rounds, once each cluster head has computed the local average, they exchange these averages to calculate a global average of all parameters. This global average is then sent back to all nodes, ensuring that every cluster member updates its model to the same parameter values and can start the next training round.

Chapter 4

Experiments

4.1 Experimental Setup

4.1.1 Vehicles' Network

Our network of vehicles is generated using SUMO[24]. It consists of 50 snapshots of the vehicles, taken every 5 seconds, including their position and velocity. We consider a vehicular network of 60 vehicles with varying speeds from 5 to 30 km/h. The simulation scenario uses a local urban area characterized by residential zones, buildings, and intersections.

For our experiments, two vehicles are considered connected if the euclidean distance between them is less than a value k. In addition, if a link exists, we consider that no obstacles such as buildings can prevent the signal from reaching its destination, and thus no packets can be lost. Finally, we assume that every vehicle can be trusted and that it will always send the right type of messages.

Table 4.1: Simulation Parameters

Parameters	**Value**
Simulation Time	250 s
Number of vehicle	60
α	0.1
β	0.9
c	1
k	200 m

4.1.2 Neural Network Architecture

Our selected model consists of a GRU layer followed by a Dropout layer followed by Dense output layer. For the first layer, we used a GRU with 16 cells, as it offers optimal performance while minimizing training time.

After the GRU, we used a Dropout layer. 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 - rate).

The final layer is a dense layer with a single unit, suitable for our regression task.

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 1, 15)	945
dropout (Dropout)	(None, 1, 15)	0
dense (Dense)	(None, 1, 1)	16
Total params		961
Trainable params		961
Non-trainable params		0

Table 4.2: Model Parameter Summary

4.1.3 Data set details and pre-process

Data selection is a key factor that affects the performance of our proposed federated learning model. Given that federated learning is mainly used in IoT environments such as smartphones, wearable devices, and autonomous vehicles, we decided to work with time series data, which reflects the behavior of real-world sensing and prediction tasks in such environments.

For this purpose, we used the real-world dataset from Kaggle [25], containing temperature records from five Chinese cities over a span of a five-year period (2010-2015). Each city's dataset contains 52.585 temperature measurements. This dataset provides a realistic foundation for evaluating our model's learning and predictive capabilities under federated conditions. As a part of cleaning and preprocessing the dataset, we dropped all the rows with Null/NaN values and we scaled the data using a MinMaxScaler. Finally, to perform the ML training, we split the data into two parts: 1) the training data set (80% of the original) and 2) the test data set (20% of

the original).

4.1.4 Vehicle's Dataset

Each vehicle is equipped with the neural network model as described above, that can be trained locally. In each round, the model is trained using a part of the entire dataset and more specifically it uses roughly 876 temperature records. Then the data is split into chunks of 4 consecutive values and the model predicts the 5th.

$$data\_size = \frac{len(dataset)}{\#vehicles} = \frac{52.585}{60} = 876 \text{ temperature records}$$

4.2 Experiments Structure

We performed different experiments using the above data for a network of maximum 60 vehicles.

1. In our first experiment, all nodes within the cluster participate in the federated learning and all cluster heads participate in the global averaging process.

2. Next, we test our model using the RNP protocol as described above where a random number of nodes participate in each round. Particularly, we consider two different participation thresholds: 0.5 and 0.75 that implies the 50% and the 75% of the cluster participate respectively.

3. We also study the case where the nodes participating to federation are selected based on their data similarity. We experiment keeping the first 5, 10 or 20 integers of the DFT coefficients to implement the comparison and select the nodes participating.

Finally, we repeat all the above scenarios but this time, we randomly select some of the cluster head to participate in the global averaging process. Cluster heads that don't participate in global averaging at the particular round continue training and implement the averaging locally (into the cluster).

4.3 Performance Measures

The goal of our model is to predict the upcoming value of a time-series dataset, thus we can consider it a regression problem.

Since the evaluation of the model is one of the most important stages in the machine learning field, there is a variety of performance measures that can be used such as error measures, accuracy measures, distance and similarity, dissimilarity etc. Although the main focus of this thesis is not as related to the performance as it is to the innovative ideas of clustering the network and implementing the federating learning, we have used two measures to evaluate our model in each iteration.

4.3.1 Mean Squared Error (MSE)

We decided to use the Mean Squared Error as our loss function.It is widely used and it eliminates any outlier predictions with significant mistakes as it gives more weight to these errors thanks to the squaring component of the function. Our goal is to achieve a low loss number which indicates that our model performed really well.

4.3.2 Accuracy

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

$$Accuracy = \frac{\#correctGuesses}{\#totalGuesses}$$

A prediction is considered correct if the following percentage:

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

where predY is the prediction and testY is the actual value, is lower than a threshold of 0.3. After having the number of correct predictions we divide it by our test size to find the model's final accuracy.

4.4 Results

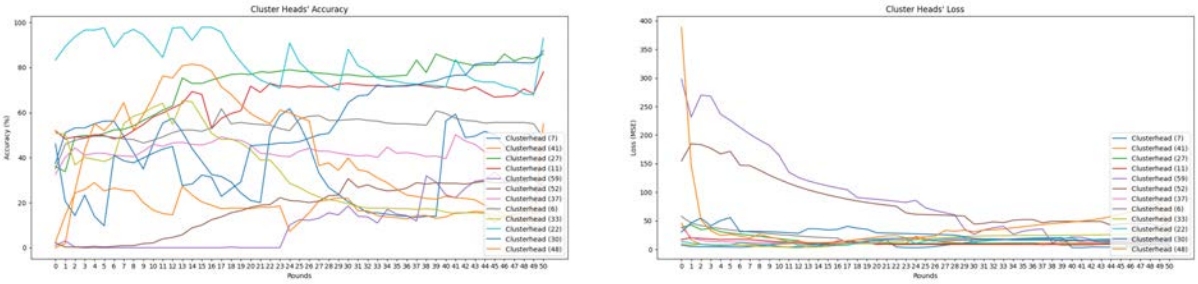
4.4.1 An overview

In this section, we show the results obtained from the experiments discussed in 4.2 and evaluate our proposed model. Our goal is not to achieve the highest accuracy possible but to evaluate how well our proposed approach can behave in vehicular ad-hoc networks.

Since we cannot present the results of all nodes of the network, we will present only the results of the cluster heads, considering that they represent each cluster.

4.4.2 All-Node Participation Experiment (ANP)

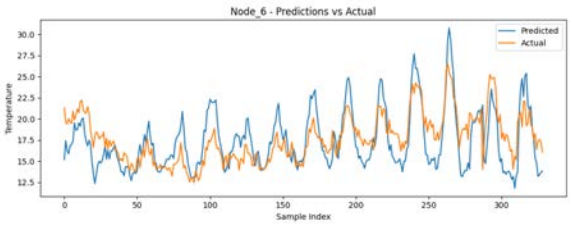
For the ANP protocol, where all nodes inside the cluster participate in the local averaging we got the following results:



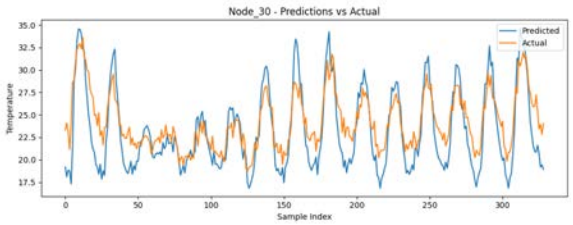
(α′) Accuracy Progress (β′) Loss Progress

Figure 4.1: Loss and MSE for the cluster heads.

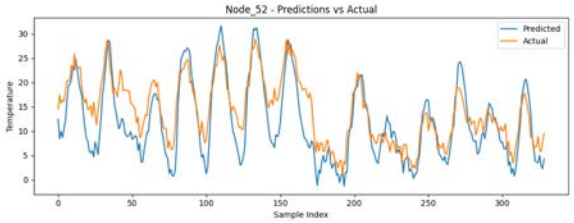
As we notice, there are some cluster heads, like 22, that achieve high accuracies close to 98%, while others, like 52, achieve a lower accuracy and cannot converge. The main reason for this is that the distribution of the data set differs from node to node, and considering that the network is not fully connected and that the topology changes, a node has significantly more chances to converge,when its connected with clusters that have a data distribution that is closer to its own.



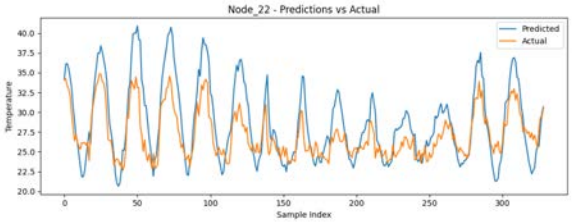
(α′) Train Accuracy for CH 6



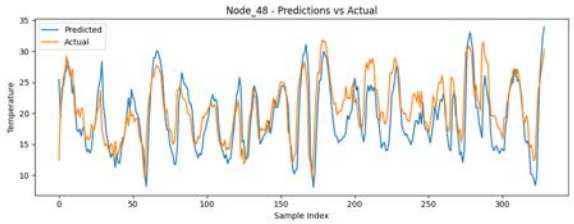
(β′) Train Accuracy for CH 30



(γ′) Train Accuracy for CH 52



(δ′) Train Accuracy for CH 22

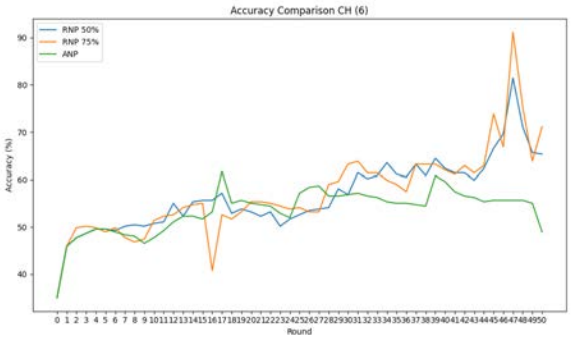


(ε′) Train Accuracy for CH 48

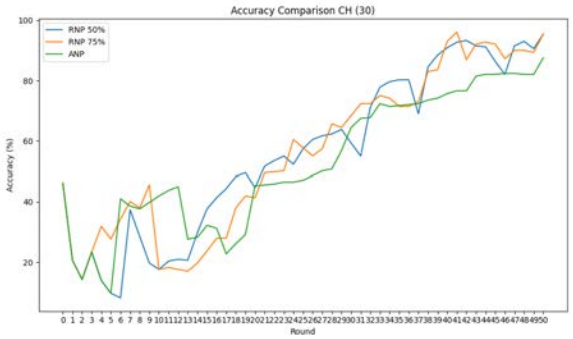
Figure 4.2: Train accuracy for round 50.

4.4.3 Random-Node Participation Experiment (RNP)

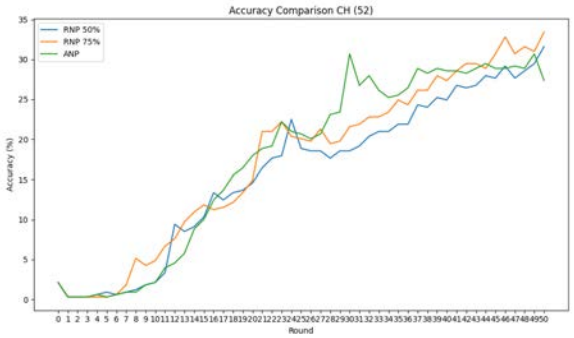
For the RNP protocol, where a random percentage of nodes inside the cluster participate in the local averaging we got the following results:



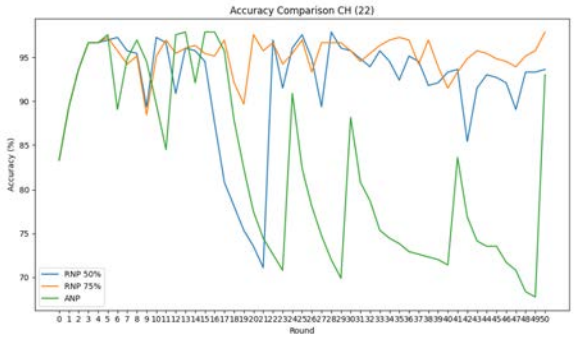
(α′) RNP Accuracy Comparison for CH 6



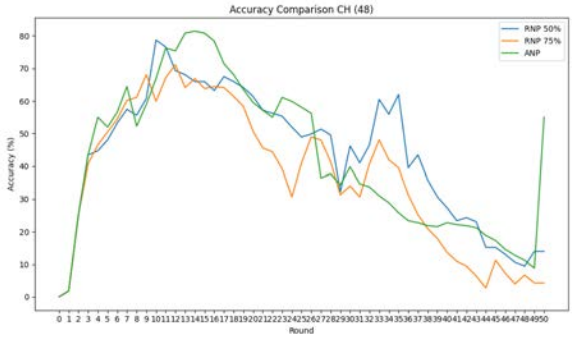
(β′) RNP Accuracy Comparison for CH 30



(γ′) RNP Accuracy Comparison for CH 52



(δ′) RNP Accuracy Comparison for CH 22

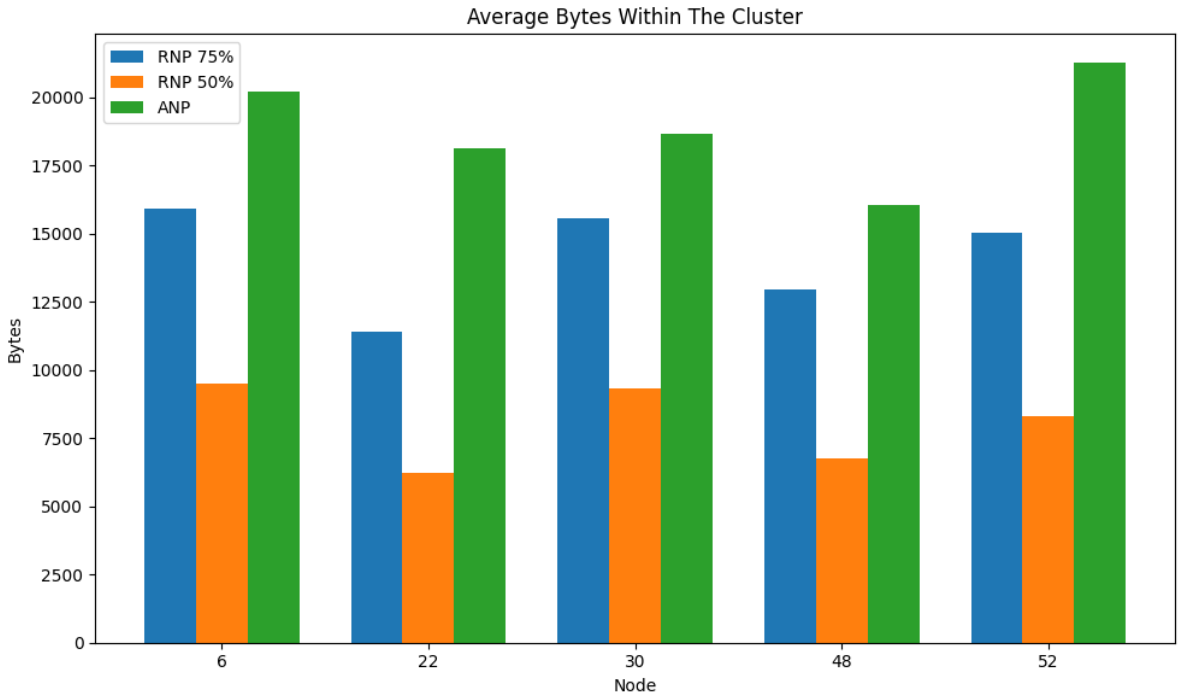


(ε′) RNP Accuracy Comparison for CH 48

Figure 4.3: Accuracy Comparison between RNP percentages.

From the comparisons above, we notice that the accuracy in some cluster heads, like 52, does not change significantly, even when half of the nodes participate in the local averaging and some other nodes, like 6, even have a significant increase in their accuracy. However, there are nodes like 48, which have a decrease in the accuracy. This can be explained, if many nodes of the same distribution are selected to not participate, then nodes that have the same distribution

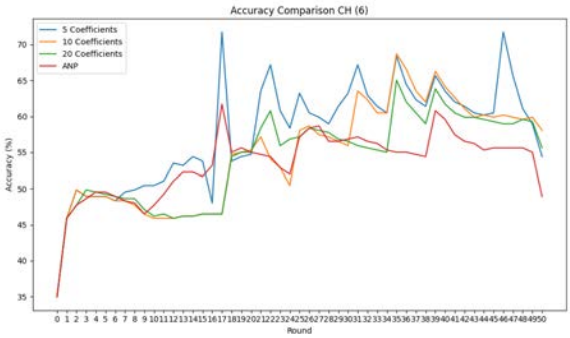
get affected and achieve lower accuracies. Finally, the data that is exchanged within the cluster reduces significantly.



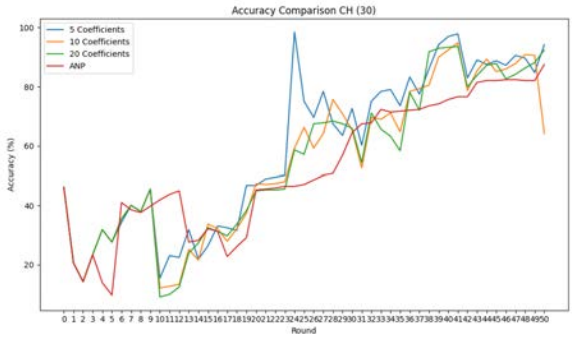
(α′) Average Bytes Exchanged Within The Clusters

4.4.4 Similarity-based-Node Participation Experiment (SNP)

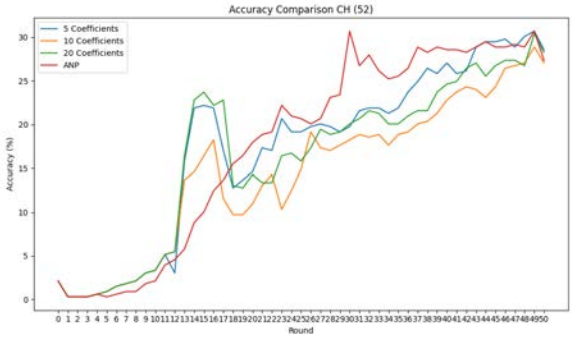
For our network the progress of MSE and accuracy during the whole federation process using 5,10 and 20 coefficients respectively is shown bellow:



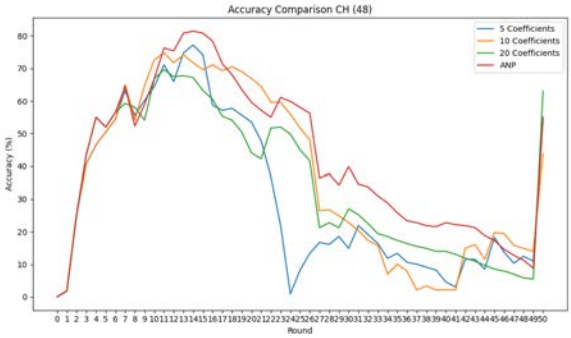
(α′) SNP Accuracy Comparison for CH 6



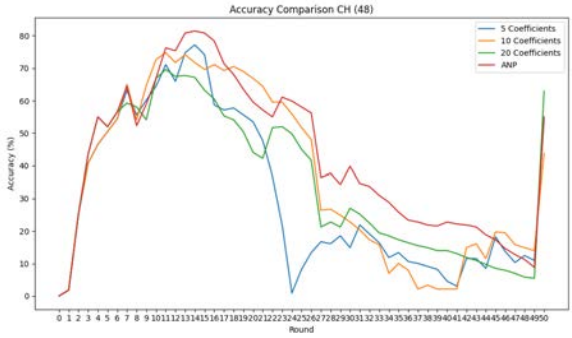
(β′) SNP Accuracy Comparison for CH 30



(γ′) SNP Accuracy Comparison for CH 52



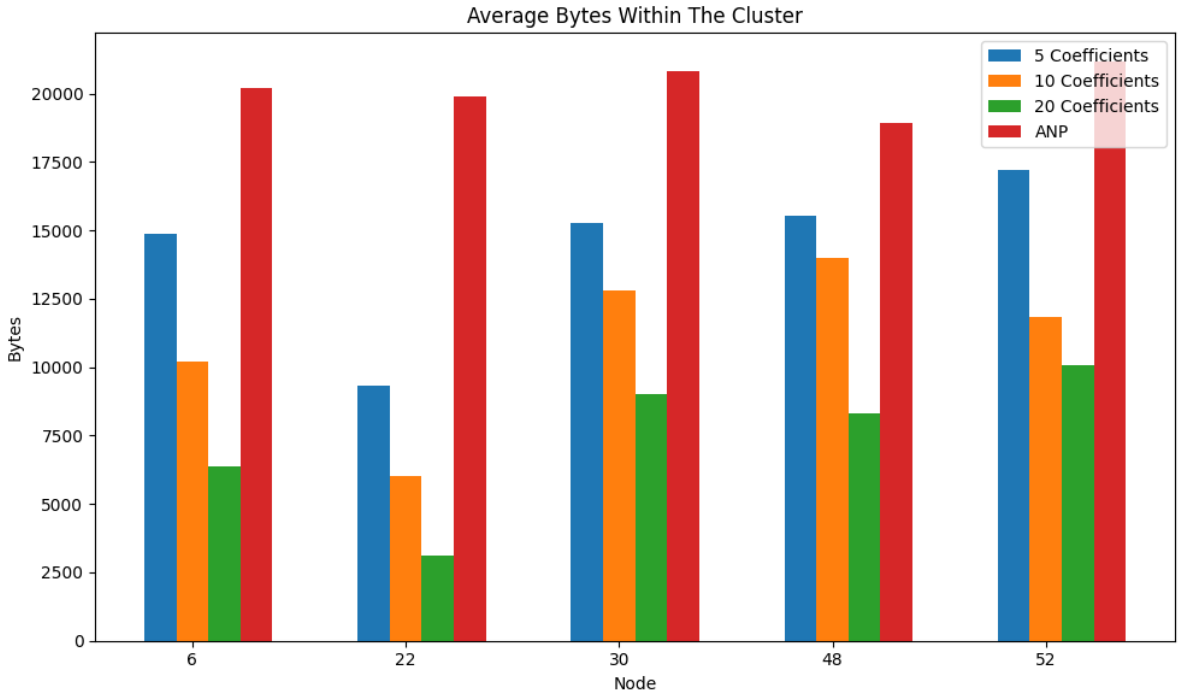
(δ′) SNP Accuracy Comparison for CH 48



(ε′) SNP Accuracy Comparison for CH 22

Figure 4.5: Accuracy Comparison between SNP percentages.

The accuracy remains relatively unchanged and continues to depend on the specific cluster head. No significant improvements in node accuracy were observed. However, the amount of data exchanged within the cluster is notably lower compared to the ANP protocol. For instance, 20 coefficients can lead to an average reduction in communication cost of about 70%.



(α′) Average Bytes Exchanged Within The Clusters

4.4.5 Final Conclusion

During the whole experimental process we notice that some nodes converge a lot faster, while others need more rounds to achieve convergence. The impact of global averaging in most situations can be seem uncooperative at first sight, since we are changing the calculated weights by force, but in the long run its effect is very optimistic.

In all the 3 protocols we used for our experiments, the accuracy of the nodes, do not seem to be significantly affected. This is due to the nature of the dataset and the network not being fully connected, so not all cluster heads participate in the global averaging process.

The impact of the random participation of the cluster heads in the global averaging, in which we run the same experiments but cluster heads participate randomly in the global averaging, is not that different than the original one.

As an overall conclusion for our approach, we could say that for the given dataset and topology, the experiment with the best results is the RNP and more specifically the 50% participation variation since it can achieve the same accuracy as ANP but with a significantly lower amount of bytes sent within the network, combining quality and efficiency. Similar observations apply to the SNP protocol, suggesting that the choice of protocol should be guided by the characteristics of the dataset and the importance of each node's contribution to the averaging process. However, not all nodes managed to converge and achieve a high accuracy, so further exploration needs to be done to better understand why this is happening.

Chapter 5

Conclusions

5.1 Summary and Conclusion

In this thesis, we proposed a distributed cluster-based Federated Learning framework for VANETs. 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 cluster head is in charge of aggregating the model's updates into each cluster. Then, cluster heads (as representatives of the whole cluster) communicate with each other and exchange parameters to implement global aggregation. The data that we used was real and we experimented in a rural vehicualar network consisting of 60 vehicles. We proposed a set of protocols 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.

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. In addition, we would like to implement a new protocol where the vehicles will choose to join the federation process only if that cluster has similar data and it is beneficial for it.

Bibliography

[1] Mustafa Çevik and Gizem Tabaru Örnek. Comparison of matlab and spss software in the prediction of academic achievement with artificial neural networks: Modeling for elementary school students. pages 1689–1707, 10 2020.

[2] A deep architecture: Multi-layer perceptron. https://medium.com/@nlunge786/a-deep-architecture-multi-layer-perceptron-164bc5ff3842, May 24, 2024.

[3] Odi match prediction with elo scores and sklearn. https://googlyanalytics.sport.blog/2020/03/26/odi-match-prediction-with-elo-scores-and-sklearn/, Mar. 26, 2020.

[4] Introduction to recurrent neural networks. https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/, May 23, 2025.

[5] Gated recurrent unit networks. https://www.geeksforgeeks.org/machine-learning/gated-recurrent-unit-networks/, Jun 4, 2025.

[6] Understanding gated recurrent unit (gru) in deep learning. https://medium.com/@anishnama20/understanding-gated-recurrent-unit-gru-in-deep-learning-2e54923f3e2, May 4, 2023.

[7] When to use grus over lstms? https://www.analyticsvidhya.com/blog/2025/03/lstms-and-grus/, Mar. 7, 2025.

[8] Federated learning: 5 use cases real life examples. https://research.aimultiple.com/federated-learning/, May 28, 2025.

[9] What are the main challenges of federated learning? https://milvus.io/ai-quick-reference/what-are-the-main-challenges-of-federated-learning.

[10] Securing the future of ai: Innovations in decentralized federated learning. https://techxplore.com/news/2025-01-future-ai-decentralized-federated.html, Jan. 13, 2025.

[11] Ad hoc network: What is an ad hoc network? https://www.7signal.com/ad-hoc-network.

[12] What is ad-hoc network? https://ecomputernotes.com/computernetworkingnotes/network-technologies/ad-hoc-network.

[13] Mobile adhoc network (manet). https://www.sciencedirect.com/topics/computer-science/mobile-adhoc-network-manet.

[14] Atinder Singh, Avneet Chawla, and Surjeet Surjeet. Performance issues and behavioral analysis of routing protocols in manets. *International Journal of Communications, Network and System Sciences*, 09:431–439, 01 2016.

[15] Medium access control protocols for safety applications in vehicular ad-hoc network: A classification and comprehensive survey. https://www.sciencedirect.com/topics/computer-science/vehicular-ad-hoc-network#chapters-articles.

[16] Mahendra KumarJhariya, Piyush Shukla, and Raju Barskhar. Assessment of different attacks and security schemes in vehicular ad-hoc network. *International Journal of Computer Applications*, 98:24–30, 07 2014.

[17] Zoubeir Mlika Afaf Taïk and Soumaya Cherkaoui. Clustered vehicular federated learning: Process and optimization. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, 23(12):25371–25383, Dec. 2022.

[18] Soumaya Cherkaoui Meysam Azizian and Abdelhakim Senhaji Hafid. A distributed d-hop cluster formation for vanet. In *IEEE Wireless Communications and Networking Confer-*

ence (WCNC 2016) - Track 4 - Services, Applications, and Business, Doha, Qatar, Apr. 2016.

[19] Maria Papadopoulou Dimitrios Papakostas Dimitrios Katsaros Schahram Dustdar Evangelia Fragkou, Eleftheria Chini. Distributed federated deep learning in clustered internet of things wireless networks with data similarity-based client participation. *IEEE Internet Computing magazine*, 28(6):53–61, 2024.

[20] Yannis Manolopoulos Nikos Dimokas, Dimitrios Katsaros. Energy-efficient distributed clustering in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 70(4):371–383, 2010.

[21] Dimitrios Katsaros Leandros Maglaras. Social clustering of vehicles based on semi-markov processes. *IEEE Transactions on Vehicular Technology*, 65(1):318–332, 2016.

[22] Theodoros Kasidakis Dimitrios Katsaros Evangelia Fragkou, Dimitrios Papakostas. Multilayer backbones for internet of battlefield things. *Future Internet journal*, 14(6), 2022.

[23] 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.

[24] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[25] Pm2.5 data of five chinese cities kaggle. `https://www.kaggle.com/datasets/uciml/pm25-data-for-five-chinese-cities`.