Universidade Estadual de Campinas
Instituto de Computação

Luiz Fernando Rodrigues da Fonseca

HVCFL: A Study of Centralized, Decentralized and Hybrid Federated Learning in Vehicle Networks with non-IID Data

HVCFL: Um Estudo de Aprendizado Federado Centralizado, Descentralizado e Híbrido em Redes Veiculares com Dados não-IID

CAMPINAS
2025

Luiz Fernando Rodrigues da Fonseca

HVCFL: A Study of Centralized, Decentralized and Hybrid Federated Learning in Vehicle Networks with non-IID Data

HVCFL: Um Estudo de Aprendizado Federado Centralizado, Descentralizado e Híbrido em Redes Veiculares com Dados não-IID

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Luiz Fernando Bittencourt

Este exemplar corresponde à versão final da Dissertação defendida por Luiz Fernando Rodrigues da Fonseca e orientada pelo Prof. Dr. Luiz Fernando Bittencourt.

CAMPINAS
2025

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

F733h

Fonseca, Luiz Fernando Rodrigues da, 1996-
HVCFL : a study of centralized, decentralized and hybrid federated learning in vehicle networks with non-IID data / Luiz Fernando Rodrigues da Fonseca. – Campinas, SP : [s.n.], 2025.

Orientador: Luiz Fernando Bittencourt.
Dissertação (mestrado) – Universidade Estadual de Campinas (UNICAMP), Instituto de Computação.

1. Aprendizado federado. 2. Redes ad hoc veiculares (Redes de computadores). I. Bittencourt, Luiz Fernando, 1981-. II. Universidade Estadual de Campinas (UNICAMP). Instituto de Computação. III. Título.

Informações complementares

Título em outro idioma: HVCFL : um estudo de aprendizado federado centralizado, descentralizado e híbrido em redes veiculares com dados não-IID
Palavras-chave em inglês:
Federated learning
Vehicular ad hoc networks (Computing networks)
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Luiz Fernando Bittencourt [Orientador]
Allan Mariano de Souza
Vinícius Fernandes Soares Mota
Data de defesa: 06-06-2025
Programa de Pós-Graduação: Ciência da Computação

Objetivos de Desenvolvimento Sustentável (ODS)
ODS: 9. Inovação e infraestrutura

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0009-0002-3249-2137
- Currículo Lattes do autor: https://lattes.cnpq.br/6220362529977978

Acknowledgments

I would like to express my deepest appreciation to Prof. Luiz Fernando Bittencourt, my advisor, for providing instructions, support, and guidance throughout my research journey in Federated Learning. The ideas suggested and the bridge established with other research teams, such as *Hub de Inteligência Artificial e Arquiteturas Cognitivas* (H.IAAC), were very helpful throughout the process.

I also express my deepest gratitude to Prof. Hélio Pedrini for introducing me to the research community during my undergraduate course. Prof. Pedrini also helped a lot during the qualification exam, with a meticulous examination of the research proposal and suggestions.

I could not have undertaken this journey without the full support of my family, especially my mother Cátia, my father Reinaldo, my brother João and my fiancee Mariane. They provided the support and love to overcome challenges during difficult times. I am also grateful to my uncle Carlos and my late grandfather Fernando.

Lastly, I would like to mention friends and colleagues I have made throughout my life. A special thanks to Gabriel and Matheus, for the entertainment moments between the studying sessions.

Resumo

Nos últimos anos, o Aprendizado Federado vem ganhando grande atenção por parte da comunidade científica, principalmente por permitir que modelos de aprendizado de máquina sejam treinados com grandes quantidades de dados de forma distribuída, e mantendo a privacidade dos dados dos usuários. Enquanto a arquitetura original desta abordagem foca em servidores centralizados que agregam os modelos treinados nos clientes, novas técnicas descentralizadas estão sendo propostas para melhorar a escalabilidade e robustez, já que não existe um agente centralizador como único ponto de falha. Entretanto, métodos descentralizados possuem uma maior deficiência quanto a quão rápida a informação se propaga na rede, pois cada cliente somente se comunica com seus vizinhos. Outro desafio que impacta ambas as abordagens são dados não-IID (Não Independentes e Identicamente Distribuídos), pois cada nó da rede pode possuir diferentes distribuições de dados, e um único modelo pode não ser suficiente para atender todos os clientes com uma boa performance.

Ao mesmo tempo, com o advento hoje das redes móveis 5G, e posteriormente 6G, muitas aplicações que antes eram difíceis de serem implementadas estão ganhando espaço, como as VANETs (Redes Veiculares Ad Hoc). Estas são redes com protocolos de comunicação curta, onde veículos se comunicam somente com seus vizinhos próximos e com a infraestrutura de borda, como as Roadside Units (RSUs), apresentando desafios tanto para abordagens de aprendizado federado centralizadas, já que veículos podem não conseguir se comunicar a todo momento com a infraestrutura da rede, quanto para descentralizadas, uma vez que a rede é altamente dinâmica devido a mobilidade, dificultando a disseminação da informação.

Desta forma, este trabalho propôs um método centralizado e outro híbrido de aprendizado federado para redes veiculares, chamados WSVC (Weight-Similarity Vehicle Clustering) e HVCFL (Hybrid Vehicle Clustered Federated Learning). Foram feitos extensos experimentos com diferentes conjuntos de dados e diversas distribuições não-IID, comparando seus resultados com métodos da literatura. Foram utilizadas ferramentas para simulação de redes veiculares, de forma que o ambiente experimental consiga reproduzir com maior fidelidade as características particulares destas redes, como potência de sinal de curto alcance, objetos que bloqueiam o sinal (Shadowing), e a mobilidade. Assim, com os resultados obtidos, foi possível listar pontos positivos e negativos de cada abordagem, como o WSVC convergindo mais rapidamente com menos mensagens trocadas quando a cobertura de RSUs era maior, e o HVCFL atingindo mais veículos, cobrindo toda a rede e conseguindo uma acurácia balanceada maior.

Abstract

In the last few years, Federated Learning has gained an increased attention in the scientific community, especially for allowing machine learning models to be trained with a large quantity of data in a distributed way and keeping the privacy of user data. Although the original architecture of this approach focuses on centralized servers that aggregate the models trained by the clients, new decentralized techniques are being proposed to improve the scalability and robustness because there is no centralized agent as a single point of failure. However, decentralized methods have a greater deficiency in the speed with which information propagates through the network, since each client only communicates with its neighbors. Another challenge which impacts both strategies is non-IID data (Non-Independent and Identically Distributed), because each node in the network may possess different data distributions, and a single model might not be enough to accommodate every client with a good performance.

At the same time, with the advent of the mobile 5G networks, and later 6G, many application that were hard to implement are gaining space, like VANETs (Vehicle Ad Hoc Networks). They are networks with short-range message protocols, where the vehicles only communicate with their nearest neighbors and with the edge infrastructure, like Roadside Units (RSUs), presenting challenges not only to centralized federated learning strategies, as vehicles might not be able to reach the network infrastructure all the time, but also to decentralized strategies, because the network is highly dynamic due to mobility, hindering the dissemination of information.

Therefore, this work proposed one centralized and one hybrid method for federated learning in vehicle networks, called WSVC (Weight-Similarity Vehicle Clustering) and HVCFL (Hybrid Vehicle Clustered Federated Learning). Extensive experiments were performed with different datasets and non-IID data distributions, comparing their results with methods from the literature. Vehicle network simulation tools were used in a way that the experimental environment was able to reproduce with higher fidelity the specific features of these networks, such as short-range signal power, objects that block the signal (Shadowing), and mobility. Thus, with the results obtained, it was possible to list the pros and cons of each approach, such as WSVC having a faster convergence with fewer messages exchanged when the RSU coverage was larger, and HVCFL reaching more vehicles, covering all the network and achieving a higher balanced accuracy.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

This chapter describes the problem addressed during this work, the research questions that guide the study, the objectives that are within the scope of the research, the contributions achieved at the end, and the organization of the rest of the dissertation.

1.1 Problem Definition

Intelligent vehicles have gained a lot of attention in recent years, especially after the deployment of 5G mobile networks and 6G in the future. Some kinds of applications that were difficult to implement due to technology and infrastructure limitations are becoming a new focus for the industry and the scientific community. For example, vehicles are now capable of making route predictions to help with traffic control tasks [15, 16], or they can use sensors and cameras to control the steering wheel or speed, autonomously driving without human intervention [53, 81].

Consequently, technological advances are redefining the way people live in cities, especially with the introduction of the concept of Smart Cities [10]. Inside them, a lot of actors are connected with one another and with a infrastructure to provide services for users, and VANETs (Vehicle Ad Hoc Networks) are an important part of this environment [52]. VANETs allow communication Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I), with elements called Roadside Units (RSUs). A general VANET architecture can be seen in Figure 1.1.

VANETs bring a series of challenges, such as short-range communication protocols, an unreliable communication channel, city elements that obstruct the wireless signal between transmitters and receivers (Shadowing) [70], units with less computing power inside vehicles, and mobility. Data privacy is another issue that arises when artificial intelligence algorithms are brought from the servers to the edge nodes. As a result, Google introduced the concept of Federated Learning [48] to handle the challenge of private data in the context of machine learning. In this paradigm, there is a centralized server with a global model and distributed nodes with private data. Only the nodes perform the task of training the machine learning models, and the server receives in each round the models from the clients to perform an aggregation. In this way, all data remain in each node, and only the models or the gradients are shared to create a global intelligent model.

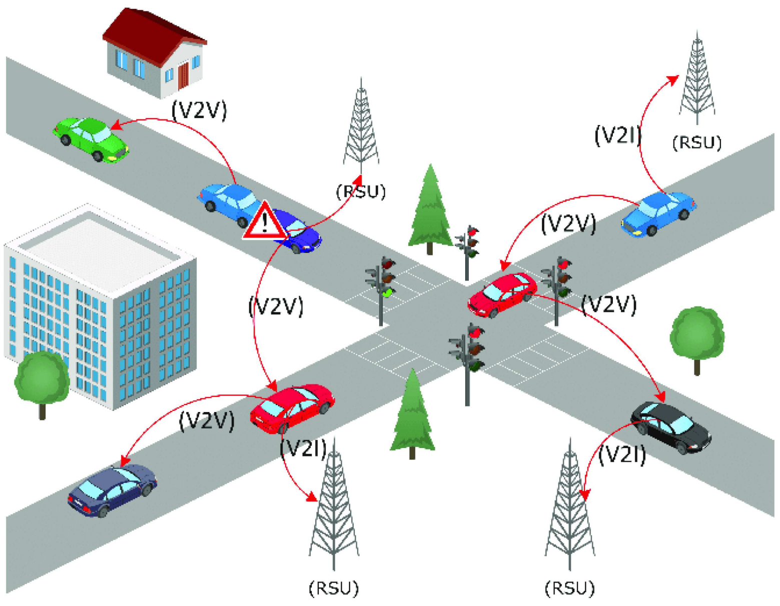


Figure 1.1: General VANET Architecture [17].

There are challenges associated with Federated Learning, such as the server as a single point of failure and scalability when more nodes join the training process. Another issue that arises is when the data distributions between the nodes are different, a concept called non-IID data (Non-Independent and Identically Distributed) [45]. Non-IID data are encountered in many real-world scenarios, like different users with different interests in recommendation systems, or vehicles that collect different quantities of images or under different lighting conditions, so they might not have a local dataset that is representative enough to contain the many different types of images needed to train a model. When these scenarios happen, it means that a single global model might not be enough to accommodate all the clients with a good performance, because their data is not the same nor follows the same statistical distribution, and additional strategies need to be deployed to the system to achieve better results.

To address these problems, some solutions have been proposed in the literature. Methods have been proposed to train models with better performance in non-IID environments. For example, some of them adapt the optimization problem to stabilize the training convergence of the different clients [58, 80], while others train multiple models at the same time based on the assumption that the clients could be separated into clusters [21], and there are many more strategies [45].

The strategy adopted in this work tries to separate clients into different dynamic clusters during the server aggregation process [68]. The method is called WSCC (Weight-Similarity Client Clustering), and it uses the Cosine Similarity of the weights of the model as a similarity metric, and the Affinity Propagation [19] clustering algorithm to separate the clients in a way that different models are generated during the aggregation process, only with similar ones. This strategy was originally developed for static networks, so in this work, we adapt the static nature of the WSCC strategy to the dynamic scenarios resulting from vehicle mobility in VANETs.

While traditional Federated Learning relies on a centralized aggregation entity, Decen-

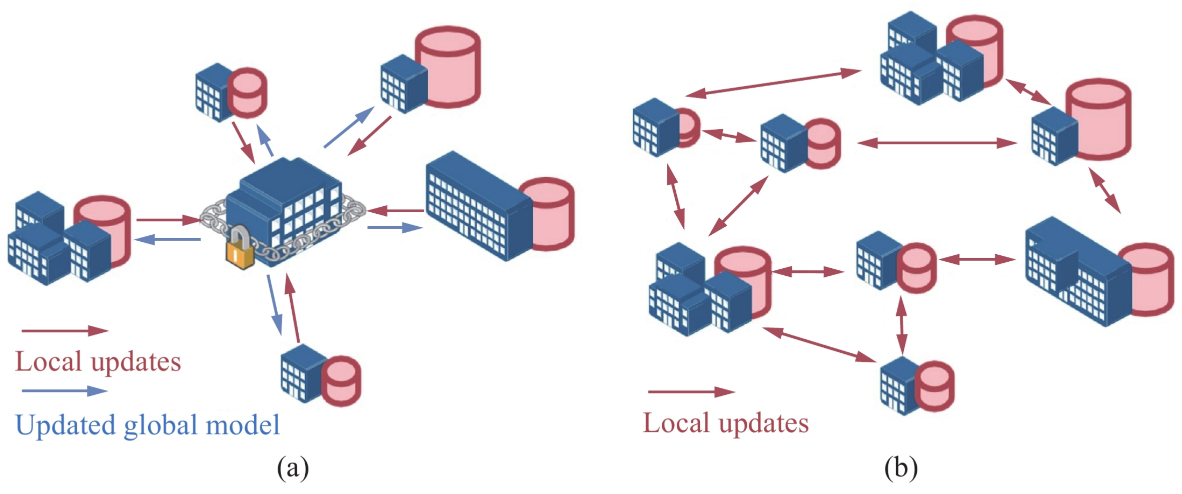


Figure 1.2: Architectures for: (a) Centralized Federated Learning with central coordinator; (b) Fully Decentralized Federated Learning [24].

tralized Federated Learning was proposed as a fully decentralized architecture without a central server. One strategy that implements this new architecture is called Gossip Learning [28, 29]. In this paradigm, nodes exchange and aggregate models directly between each other opportunistically, without a centralized coordinator but also maintaining data privacy. A comparison between centralized and decentralized Federated Learning can be seen in Figure 1.2. Furthermore, since vehicles can exchange messages directly with their neighbors and sometimes cannot reach the infrastructure, a decentralized approach could benefit the learning procedure in VANETs.

Some challenges also arise with a decentralized architecture, such as the slower speed with which the information propagates through the network, since each client only communicates with its neighbors. This communication is performed one hop at a time, so nodes far from a given node will need some communication rounds until the information reaches them. When non-IID data are taken into account, the Gossip Learning information dissemination issue and the network topology also hinder the training process, because each node is only able to exchange models with its close neighbors, and all might have different data distributions [22].

Considering the case of VANETs, both centralized and decentralized federated learning are affected by the way the network environment works. Centralized solutions are affected because some vehicles might not be able to communicate with the central server all the time, because an RSU might not be in range of the vehicle, or because the messages were lost in the unreliable communication channel (shadowing). This means that the positioning of the RSU can affect the number of vehicle messages that arrive at the server, so in scenarios with less coverage, some vehicles may not receive any model at all (more details in Chapter 5). Decentralized solutions are also hindered by the highly dynamic mobile network that changes constantly, increasing the effects of the information dissemination issue. Different types of vehicle trips might also affect model convergence, because short trips mean that the vehicle spends less time inside the network, whereas longer trips, such as buses, taxis, and app drivers, are the opposite.

With all these topics presented, developing a way to handle the issues of centralized and decentralized learning architectures for VANETs is a good approach to improve machine learning tasks using the vehicle networks architecture. Merging the strengths of both centralized and decentralized approaches in a hybrid way could help to train better models with higher performance in these highly dynamic environments.

1.2 Research Questions

Some questions have been formulated to guide the research. They serve as a way of better understanding the motivation and goals of the work. The questions are the following:

Q1. How does Federated Learning perform in a dynamic environment such as vehicle networks (VANETs)?

Q2. How does Centralized Federated Learning perform compared to Decentralized approaches in VANETs?

Q3. Is it possible to improve the performance of a trained model with a hybrid approach?

Q4. Do different non-IID data distributions perform differently in different approaches?

Q5. How does environment configuration (especially RSU positioning) affect the convergence of machine learning models for different methods?

Q6. How do long vehicle trips compare to short ones in model convergence?

During this dissertation, the methods and experiments developed to address these questions are presented, with a thorough discussion of the results in Chapter 5

1.3 Objectives

The main objectives of this work are: (1) to adapt methods developed to work in static networks, such as WSCC (Weight-Similarity Client Clustering) [68], to work with dynamic vehicle networks; (2) to develop a Hybrid Centralized/Decentralized Federated Learning method for Vehicle Networks for non-IID scenarios that achieves better performance when training machine learning models, and using metrics like balanced accuracy, compared to other strategies of the literature. Comparison methods include centralized strategies such as FedAvg [48] and FedProx [58], and decentralized strategies like Gossip Learning [28, 29] and FedPC [80].

To achieve the main objectives, other secondary objectives have been defined, such as: prepare simulated environments for the VANETs that approximate as best as possible real-world scenarios of highly dynamic and wireless short-range networks (OMNeT++, Veins, and SUMO [71, 63, 42]), prepare the datasets and simulate experiments using different non-IID data distributions, compare existing literature methods and solutions in these specific scenarios, and validate that the proposed methods achieve higher balanced accuracy in most of the scenarios tested.

1.4 Contributions

The main contributions of this work are the following:

- Proposal of the WSVC (Weight-Similarity Vehicle Clustering) method in dynamic vehicle networks, which adapts the WSCC (Weight-Similarity Client Clustering) [68] method from the literature, developed for static networks.
- Design of a new method for distributed learning in vehicle networks, called HVCFL (Hybrid Vehicle Clustered Federated Learning).
- Studies and extensive experiments comparing solutions from the area of VANETs and general Federated Learning in different mobility scenarios.
- Comparison of different non-IID data distributions and observations of positive and negative sides of each method.
- Open source code available in `https://github.com/Luizfrf3/veins-5.2`

1.5 Text Organization

The rest of this dissertation is organized from chapters 2 to 6. In Chapter 2 all the concepts needed to understand this work are introduced, such as VANETs, Mobility Scenarios, Neural Networks, Federated Learning and its variations, and non-IID data. Chapter 3 introduces all the methods from the literature that were studied in this dissertation, either being used for comparison or as inspiration for the proposed solution.

Chapter 4 provides a detailed description of the proposed solution of this work, while Chapter 5 describes the simulation scenarios, the experimental methods, and a detailed discussion of the results obtained. In the end, Chapter 6 provides the final comments and future work.

Chapter 2

Theoretical Concepts

This chapter introduces and explains the theoretical foundations that underlie the remainder of this dissertation. There is an introduction of Vehicle Networks with its challenges and a brief description of mobility models. Next, neural networks are explained, especially the Convolutional Neural Networks, with some performance metrics and the Data Augmentation concept. After that, there is an introduction to Centralized and Decentralized Federated Learning, also with its main challenges. In addition, there is a section related to Non-Independent and Identically Distributed data, together with some concepts of clustering and similarity metrics. To conclude, the Weight-Similarity Client Clustering method (WSCC) [68] is introduced because it is a basis for the development of the rest of this dissertation proposals, followed by a summary of the concepts presented and how they are utilized in the proposals.

2.1 Vehicle Networks (VANETs)

2.1.1 Fundamentals

Vehicle Ad Hoc Networks (VANETs) are composed of vehicles that exchange messages with each other, through Vehicle-to-Vehicle (V2V) communication, and with the infrastructure, through Vehicle-to-Infrastructure (V2I, I2V) communication [52]. Usually, the infrastructure of the network is composed of Roadside Units (RSUs), which are fixed signal sender/receiver equipment positioned on the side of roads and highways. Although vehicles have less computational power, less storage, and are inherently wireless, RSUs are higher-power edge servers that are connected to the wired network, being a gateway to the network core [62].

In the VANETs, all the communication is made through dedicated DSRC protocols (Dedicated Short Range Communication). For this reason, the Institute of Electrical Electronics and Engineering (IEEE) developed the IEEE 802.11p standard together with IEEE 1609.4 DSRC/WAVE [37, 33]. The communication bandwidth is usually around 3-27 Mbps and the range is 200-500m.

The standards together provide reliability for the operation of security and safety applications. For these reasons, there are different channels available for different types of applications: Control Channels and Service Channels (some of them are exclusive for safety messages). In this work, the service messages have been used because Federated Learning includes different kinds of applications, and most of them do not need to be used for safety reasons.

Furthermore, by default, since it is implemented in a wireless medium, when someone sends a message, all other clients on the network can hear the message, which is known as Broadcasting [56]. It is also possible to implement multiple hop messaging protocols, but in this work message broadcasting was used, as in other works [15, 16, 40].

2.1.2 Challenges

There are some challenges associated with VANETs and its protocols that may impact Federated Learning in this environment:

- **Short-Range Communication Protocols**: VANETs operate with short range messages and not super fast network speeds. This hinders the data transfer process between vehicles and decreases the number of neighbors available in the network.
- **Mobility**: Vehicles are moving agents that quickly go from one place to another, and they also change direction following the streets and highways. These facts can cause some effects:
  - **Variable Vehicle Density**: The number of vehicles in each region of the city varies over time depending on traffic. There are moments with peak congestion and moments with not as many vehicles, so this affects the number of neighbors each client has in a given time [37].
  - **Dynamic Topologies**: High mobility causes the network topology to change constantly, which can also shorten the contact time between the elements.
- **Data Dissemination**: Due to the challenges listed above, it is much more difficult to disseminate information in VANETs, i.e., to get information from a vehicle to every other interested in those data. The following challenges are associated with this topic [47]:
  - **Broadcast Storming**: It is possible to use broadcasting to disseminate the messages without knowing the final destination. However, if many vehicles try to transmit simultaneously, there will be network congestion due to more packet collisions, increasing the delay of the network.
  - **Shadowing** [70]: Shadowing is the effect that the signal power fluctuates between sender and receiver due to obstacles in the signal path, like buildings and even other vehicles.
  - **Temporal Fragmentation**: Shadowing, packet loss, and short-range communication can cause the partition of the network. In this way, during a period of time or while there are few vehicles connected, the network may be fragmented, causing some vehicles to be disconnected from the rest.

2.1.3 Mobility Models

Mobility Models are used to simulate how mobile agents traverse an environment, defining the speed and directions a user takes over time [8]. They are ways to approximate real-world scenarios when implementing VANET simulations. The models used in this work are the following:

- **Manhattan Mobility Model** [4]: This model is based on the city of Manhattan, where the streets form a quadrangular grid. When the vehicle reaches each intersection, there is a probability that 50% will go straight, 25% will turn left, and 25% will turn right. This model has been used by some works [66, 40], and it approximates vehicle behavior especially for big cities.

- **Mobility Traces**: Traces are generated by collecting real data or using routing algorithms, such as some vehicle scenarios [13]. For example, it is possible to generate routes from point A to point B in the grid for multiple vehicles, representing traffic through regions of the city.

2.2 Neural Networks

This section presents a brief introduction about the concepts of Neural Networks that are used throughout this dissertation. The focus is especially on Convolutional Neural Networks (CNNs), performance metrics, challenges, and data augmentation.

2.2.1 Fundamentals

Neural Networks are defined as mathematical functions that map an input to an output and are inspired by brain neurons. They can be composed of simple computations or even have many layers of operations in a deep configuration. Convolutional Neural Networks (CNNs) [61] are a specialized class that focuses on computer vision problems, such as image classification (such as traffic sign classification in the vehicle domain), object detection (for example, for autonomous vehicles) or even steering wheel angle prediction [23]. All these problems have an image (or video) as an input, perform mathematical computations, and generate an output (or multiple outputs).

CNNs use spatial convolutions as the base operation with learned parameter filters (weights), along with other operations such as activation functions, for example the Rectified Linear Unit (ReLU), subsampling, and others. An example CNN can be seen in Figure 2.1. Given X an input, C the activation function, and w the weights, a neural network is represented by the following Equation 2.1.

$$\hat{y} = C(X, w) \tag{2.1}$$

After the function is defined, the neural network needs to be trained for some epochs with example data so that the weights w fit as best as possible the given examples. For this process, backpropagation is used to adjust w little by little passing pairs of X, y that are already known, and an epoch is reached when all the local training data passes

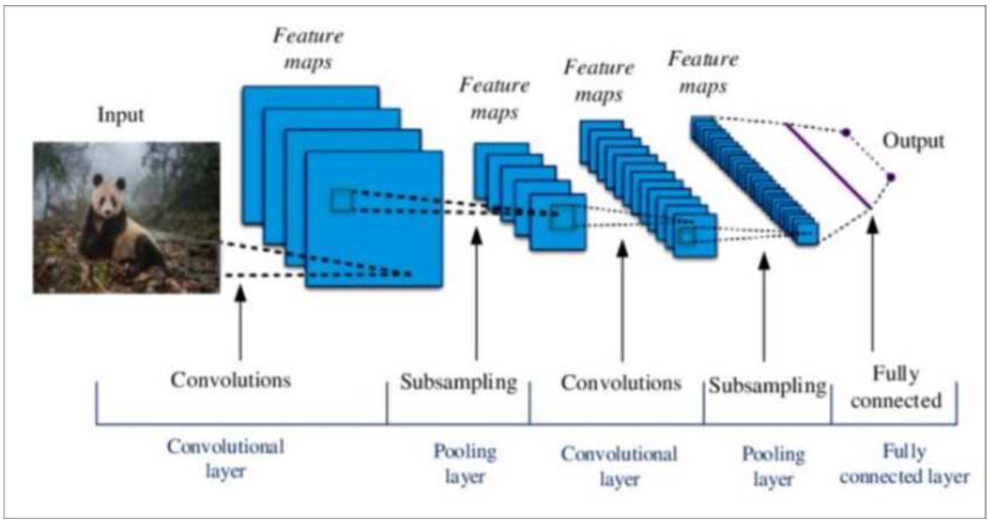


Figure 2.1: Simple Convolutional Neural Network (CNN) example [26].

through the network once. This concept uses cost function minimization, so an error is calculated between the $\hat{y}$ generated by C and a known example of y.

This work uses the image classification problem in its experiments, so given an input image X, the output y will be an array of size n. Each position of the array defines the probability that the image X belongs to class (or label) L, such as a turn right sign, a 20 Km/h sign, etc. In these problems, the Categorical Cross-Entropy error function 2.2 is commonly used during the training process to measure how good the model is and adjust its weights.

$$CCE = -\sum_{i=1}^{L} y_i \cdot \log(\hat{y}_i) \tag{2.2}$$

2.2.2 Performance Metrics

Performance metrics measure how the neural network model performs with example data. There are also protocols to separate a dataset into different training, validation, and testing sets, but this will be explained in detail in Chapter 5.

Given some data examples that were not used in the training procedure, it is possible to calculate these metrics. For classification problems, *Accuracy*, that is, the number of correct classified examples over the total is a simple metric to use. There are some disadvantages of using accuracy, especially when the dataset is non-IID. For example, if the distribution of the number of labels is very unequal and one class constitutes 90% of the validation examples, if your accuracy gives 90%, it is possible that the model always outputs the same class.

Therefore, some alternatives have been proposed to mitigate these issues, such as *BalancedAccuracy* 2.3. The balanced accuracy is the average of the *Recall* of each individual class 2.4. The recall of each class is the number of True Positives for class l, divided by the sum of True Positives and False Negatives for class l. True Positives are the examples that the model correctly classifies and False Negatives are the incorrectly classified examples. Taking into account the same example as before, the recall of the first class that constitutes 90% of the data would be 100% if the model always outputs the same class, but the second class would be 0%. The balanced accuracy would result in 50%, which better reflects the quality of the model compared to an accuracy of 90%.

$$BalancedAccuracy = \frac{1}{L}\sum_{l} Recall_l \tag{2.3}$$

$$Recall_l = \frac{TP_l}{TP_l + FN_l} \tag{2.4}$$

2.2.3 Challenges

Neural networks present some challenges in different aspects [20]. Here are some of them:

- **Underfitting**: This happens when the model cannot have a good performance with the training data, and even with more training, the error does not decrease. It might happen due to: few data examples or a very simple model for the data patterns.
- **Overfitting**: This is the most common issue, and it happens when the training performance is good but the validation metrics get worse. This means that the model fits the patterns of the training data, but it adjusts so well that it cannot generalize to new examples. It might happen due to: few data for complex models, and model is very complex.

2.2.4 Data Augmentation

Some strategies exist to better generalize a model such as Dropout [64] and Data Augmentation [60]. Dropout is an operation introduced inside the neural network during the training process that randomly drops weights (such as removing the connections between neurons). This helps in training all the weights of a layer and in not getting some of them biased, improving the model generalization.

Data Augmentation is the process of generating new training examples based on existing data. It not only generates more examples, but also different kinds of examples that did not exist in the training data but that could happen in the real world. This is especially useful for distributed scenarios where each client has few data examples as a way to decrease the effects of underfitting, but especially overfitting.

Taking images as the main input used in this work, it is possible to perform different operations to enhance the training data. The main strategies that have been used in this work are:

- **Zooming**: Applies zooming on the image and cropping the borders.
- **Rotation**: Rotates the images by some degrees.
- **Translation**: Dislocate the image by some pixels horizontally and vertically, cropping the rest.

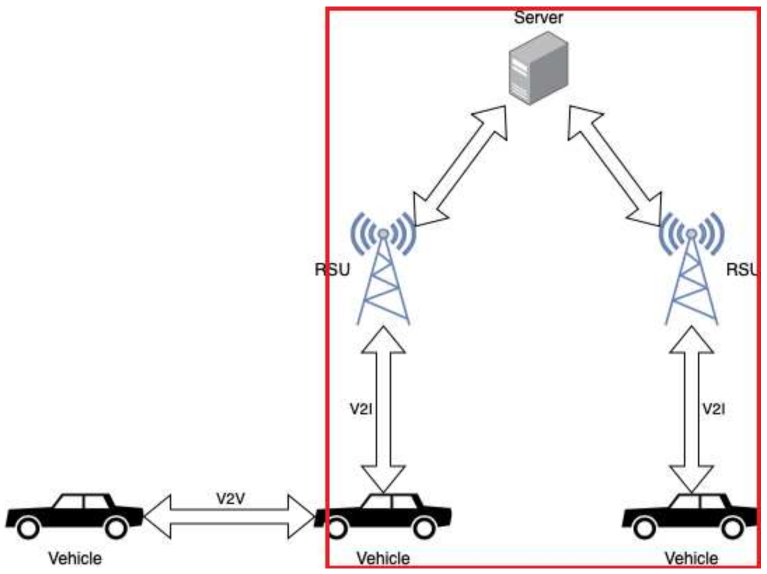


Figure 2.2: Solution architecture with a central server that performs centralized Federated Learning procedures, the RSUs that are the bridge between the vehicles and the server, and the vehicles.

Just an additional note on this topic; It is important to understand the data before applying data augmentation. For example, it might not make sense to apply a 180 degree rotation of the image on a traffic signs dataset, because the model will not be able to know which sign is the turn right and which is the turn left.

2.3 Centralized Federated Learning

2.3.1 Fundamentals

Federated Learning is a framework for training machine learning models in a distributed way, keeping the privacy of client data [48]. In this strategy, there is a centralized server that holds a global model, which is sent to the client devices (or vehicles). Clients receive the global model, train with their local data, and send only the weights or gradients back to the centralized server. The server finishes the round aggregating all the received models to generate a new global one, and sends them again back to the clients to start a new round. This procedure is repeated until the model convergence, or after a fixed number of rounds. This strategy improves data privacy because only the models (weights or gradients) are shared between the client and the server, so all data is kept private in the local edge devices. Figure 2.2 highlights only the centralized process in the system architecture.

The default Federated Learning method, which defines a model aggregation strategy, is called FedAvg [48], and the server side logic can be seen in Algorithm 1. In the beginning, the server initializes the global model weights w_g and sends them to the clients. Then, the round loop starts, waiting for a period of time Δ_a, usually called the round time. During this period, the server can receive a trained model from a client and store it, calling the $onReceiveModel$ procedure.

Algorithm 1 FedAvg Server Side

```
1: $w_g \leftarrow \mathbf{0}$
2: Send initial $w_g$ to clients
3: loop
4:     wait($\Delta_a$)
5:     Call the AGGREGATION procedure
6: end loop
7: procedure onReceiveModel($w_r, |D_r^t|$)
8:     Store $w_r$ and $|D_r^t|$ on the server
9: end procedure
10: procedure AGGREGATION
11:     With received models weights from clients $w_1, w_2, ..., w_N$
12:     Merge the weights received to generate a new $w_g$ using Equation 2.5
13:     Send the new $w_g$ to the clients
14: end procedure
```

At the end of every round, the *AGGREGATION* procedure is called. During aggregation, the server uses the received models from the clients and aggregates (merges) them to generate a new global model w_g, using Equation 2.5. This equation is the basic aggregation strategy for FedAvg, which basically calculates the new global weights using an average weighted by the size of each client's dataset, increasing the importance of clients with more data [48]. After that, it sends the new global model to the clients. If a defined stop criterion has been reached, the main-server loop can end, so here and in the following server-side algorithms, this logic has been removed for easier visualization.

$$w = \frac{\sum_{i=1}^{N} |D_i| w_i}{|D|} \tag{2.5}$$

The client side of FedAvg can be visualized in Algorithm 2. It starts by initializing the local model w_i for the client i and having a local dataset for training called D_i^t. When receiving a model from the server, it calls the procedure *onReceiveModel* and overwrites the local model with the one w_r from the server. After that, it trains the local model for E epochs calling the *UPDATE* procedure, and sends the new model trained with local data back to the server.

Algorithm 2 FedAvg Client Side

```
1: $w_i \leftarrow \mathbf{0}$
2: procedure onReceiveModel($w_r$)
3:     $w_i \leftarrow w_r$
4:     $w_i \leftarrow update(w_i, D_i^t, E)$
5:     Send $w_i$ and $|D_i^t|$ to the server
6: end procedure
```

The *UPDATE* procedure in Algorithm 3, trains the model w_i with the training data D_i^t for E epochs. In this step, it is possible to use Data Augmentation [60] to increase the size and improve the diversity of local data, to help the issues described in the last sections, such as overfitting. The training also uses the backpropagation strategy described in the previous sections, through Equation 2.6. Here, w_i is the model being trained, η is the learning rate that controls the speed at which the model is trained, and $\nabla F_i(w_i)$ is the gradient of a loss function, described in the section above during the function minimization procedure. The $ProxTerm$ mentioned in line 3 will be explained in the next sections.

Algorithm 3 Update

```
1: procedure UPDATE($w_i, D_i^t, E$)
2:     Train the model for $E$ epochs, using Data Augmentation if necessary
3:     Use backpropagation to update the weights, Equation 2.6 (if using the $ProxTerm$,
       add it to the minimization process)
4: end procedure
```

$$w_i = w_i - \eta \nabla F_i(w_i) \tag{2.6}$$

2.3.2 Challenges

Since Federated Learning is about training machine learning models in a distributed way, underfitting and overfitting described in the last section also have an impact here. However, some new challenges arise with the framework:

- **Single Point of Failure**: The server (or group of centralized servers) is a single point of failure for the system. If a node cannot reach the server, for example if it is a vehicle outside the coverage area of an RSU, it will not be able to continue the training process.

- **Scalability**: Executing the process for a few clients should be totally fine, but when escalating to millions of users, a single centralized strategy might not be able to use the models from every node, which hinders the training process.

- **Messages in Network Core**: Similarly to the scalability problem, if every node is included in the training process, a huge number of messages will flow through the network core, causing network congestion.

- **Non-IID Data** [45]: Non-IID data are probably the biggest challenge for Federated Learning, because it makes the training process of a single global model a much more difficult process. Non-IID data occur when the distribution of data among clients is different, and due to its complexity, it will be presented in section 2.5.

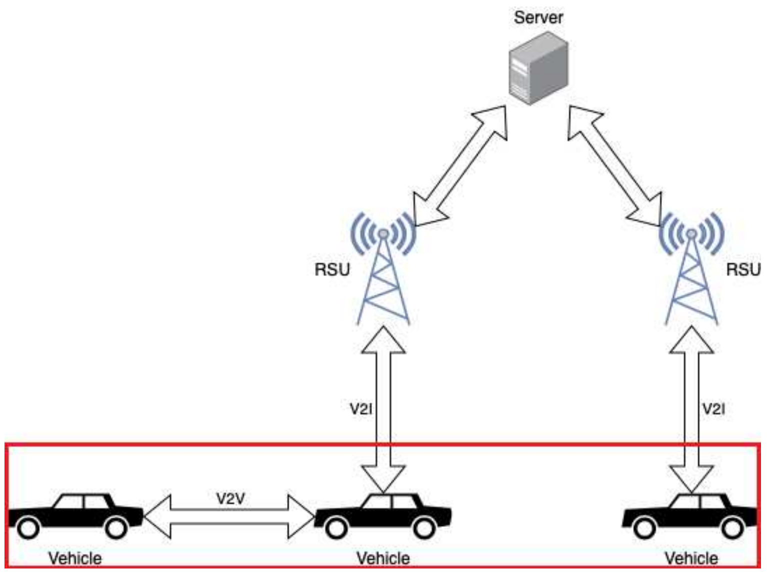


Figure 2.3: Solution architecture considering only the vehicles that perform decentralized Federated Learning procedures.

2.4 Decentralized Federated Learning

2.4.1 Fundamentals

The method that popularized decentralized approaches for Federated Learning is called Gossip Learning [28]. In this framework, there is no centralized server that aggregates the client models and synchronizes the training rounds, but the nodes opportunistically exchange and merge models with their neighbors. Figure 2.3 highlights only the decentralized process in the system architecture.

The basic logic of Gossip Learning can be seen in Algorithm 4. Each client also starts initializing the local model weights, but in this case there is a loop which handles the gossip round. During this loop, each node waits a period of time Δ_g, called gossip round time, and at the end they select a peer neighbor p and send the model w_i to the neighbor. In line 7, there is also a *onReceiveModel* procedure that receives a model w_r from a neighbor, merges the local model with the received one using Equation 2.5, and trains the local model with the same strategy as in the FedAvg Algorithm 3.

2.4.2 Challenges

Gossip Learning handles some of the problems of the Centralized Federated Learning approach, such as the single point of failure, scalability, and core network traffic, but it introduces new challenges [22]:

- **Synchronization**: Although the centralized approach has a server to sync the clients and training rounds, Gossip Learning by default does not impose any synchronization requirements. The nodes are free to join the training group at any time and start training, so this may impact slower devices that train less times compared to faster ones, generating data bias.

- **Information Dissemination**: Without a single global model being trained with data from the nodes, Gossip Learning nodes only communicate with their nearest neighbors, and the messages pass one hop at a time. As a consequence, nodes far away from a given one will need more communication rounds until the information reaches them, which increases the training time and time to convergence.
- **Overfitting**: Due to the problem above, overfitting also tends to happen more often in decentralized scenarios, especially because more data (information) is important to train neural networks.
- **Non-IID**: Also, another aspect that is more difficult because of the information dissemination problem is non-IID data. Since the data distribution among clients may vary, some nodes might only have very different neighbors at a given point in time, while others may have very similar neighbors. This hinders the training of some of them, while it may also make the model of a group of users biased.

Algorithm 4 Gossip Learning

```
1: $w_i \leftarrow \mathbf{0}$
2: loop
3:     wait($\Delta_g$)
4:     $p \leftarrow$ selectPeer()
5:     Send $w_i$ and $|D_i^t|$ to $p$
6: end loop
7: procedure onReceiveModel($w_r, |D_r^t|$)
8:     $w_i \leftarrow merge(w_i, |D_i^t|, w_r, |D_r^t|)$
9:     $w_i \leftarrow update(w_i, D_i^t, E)$
10: end procedure
11: procedure merge($w_i, |D_i^t|, w_r, |D_r^t|$)
12:     Use Equation 2.5 with $w_i$, $w_r$ and $N = 2$
13: end procedure
```

2.5 Non-Independent and Identically Distributed Data (non-IID)

Non-Independent and Identically Distributed data (non-IID) means that the distribution of data between different clients may vary, i.e., the assumption that each data point is drawn from the same distribution does not hold. For example, a dataset to detect spam contains the e-mails information and an indication if they are spam or not. Usually, there are many more examples of non-spam e-mails than spam ones, so this fact causes an unbalanced label distribution between the classes (spam or not). Moving to the distributed world, while some clients may contain both spam and not-spam e-mails, some of them might only have not-spam ones, which makes the training process biased in some clients.

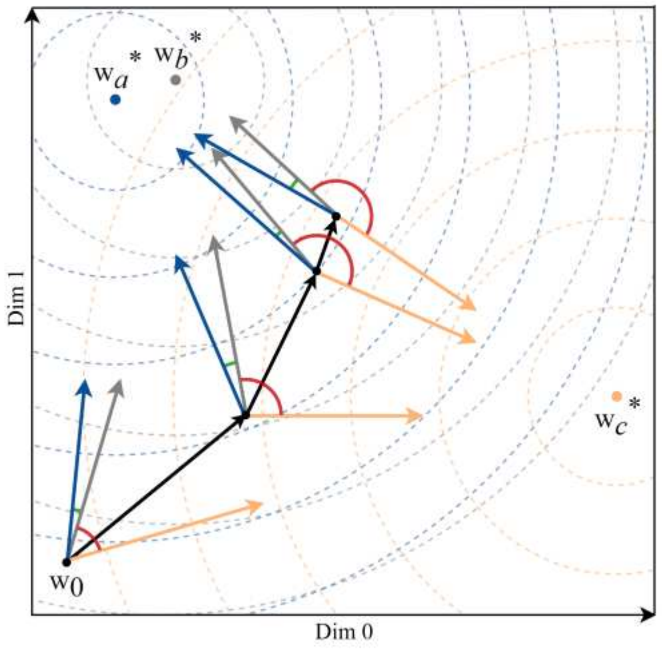


Figure 2.4: Image displaying the divergence of gradients during a Federated Learning global model training with 3 clients having different optimal models [38].

Another example of this occurs when there are groups of similar users. For example, if the original data follow 2 different distributions, there would be 2 different optimal points for the weights of the model, one for each group [38]. This is called the gradient divergence problem and can be visualized in Figure 2.4. In this figure, w_a^*, w_b^* and w_c^* represent the optimal model for clients a, b and c, and w_0 is the initial model being trained (the model contains only two features for visual simplicity). While clients a and b pull the model gradients to the top left, client c pull them to the lower right of the dimensions space. The result is that the trained global model stays in the middle of these 2 groups, indicating that a single model is not enough to train optimally for the 2 client groups.

Formally, some of the different non-IID data distributions can be of the following [45, 83], but usually they are a mixture of more than one of them:

- **Feature Distribution Skew**: This happens when the features of the data have different distributions, but the distributions of the labels are the same. For example, different writers can write the number 5 using different styles, changing the characteristics of the handwritten digit.

- **Label Distribution Skew**: This happens when the label distribution between the clients is different, but the features of the labels follow the same distribution. For example, when clients have a maximum number of labels assigned to them, or when this value varies, like some works that use the Dirichlet Distribution to create non-IID scenarios [77, 51].

- **Label Preference Skew**: This happens when the distribution of the features is the same, but the label is different. For example, some users might have different movie tastes, so while some of them like movie A, some of them dislike the same movie.

- **Quantity Skew**: This happens when the data quantities between different clients vary a lot. For example, client i has 50 data entries and client j has 5000.

Some strategies have been proposed to handle non-IID data for Federated Learning. One of the first proposals is a strategy to be applied during the function minimization procedure, called the FedProx method [58]. This strategy is similar to FedAvg, but it includes a regularization term inside the function minimization procedure, called $ProxTerm$, which can be seen in Equations 2.7 and 2.8. This regularization term minimizes not only the error but also the difference between the trained and the global model received, making the model updates closer to this last one, and this can be adjusted by setting the hyperparameter μ. FedProx was proposed for devices with different speeds and data quantities, so every update is guided to be closer to the global model to alleviate device heterogeneity.

$$ProxTerm = \frac{\mu}{2} \left\| w - w_i \right\|^2 \tag{2.7}$$

$$min\{F_i(w_i)\} \Rightarrow F_i(w_i) + ProxTerm = F_i(w_i) + \frac{\mu}{2} \left\| w - w_i \right\|^2 \tag{2.8}$$

2.5.1 Clustering

There are other types of strategies for other types of non-IID data. Some strategies that are studied in this work are in the area of client selection, especially by using clustering algorithms to calculate similar user groups. This is especially useful when the original user data can be separated into different clusters [21, 68], which happen in different real-world scenarios, such as users in different regions, users with different tastes, or vehicles with different mobility profiles, such as taxis/buses and family vehicles.

Clustering algorithms are also in the area of machine learning, and their main focus is to split the data examples (clients in this case) into different groups in a way that only similar data are assigned to the same group. Although there are some methods that require you to know the number of clusters in the data distribution beforehand, such as K-Means [46], some newer solutions do not need to know the number of clusters, such as Affinity Propagation [19]. Both strategies have advantages and disadvantages, but since in the real world the data used for Federated Learning usually do not have a well-known number of clusters, in this work Affinity Propagation will be used for clustering of clients, as in [68].

Affinity Propagation works to create clusters for N data points. It starts by calculating the distance matrix using Equation 2.9. The diagonal $d(i, i)$ controls how likely each instance is to become an exemplar, that is, the main representative data point of a cluster, and it is usually initialized as the median distance.

$$d(i, k) = -\left\| x_i - x_k \right\|^2 \tag{2.9}$$

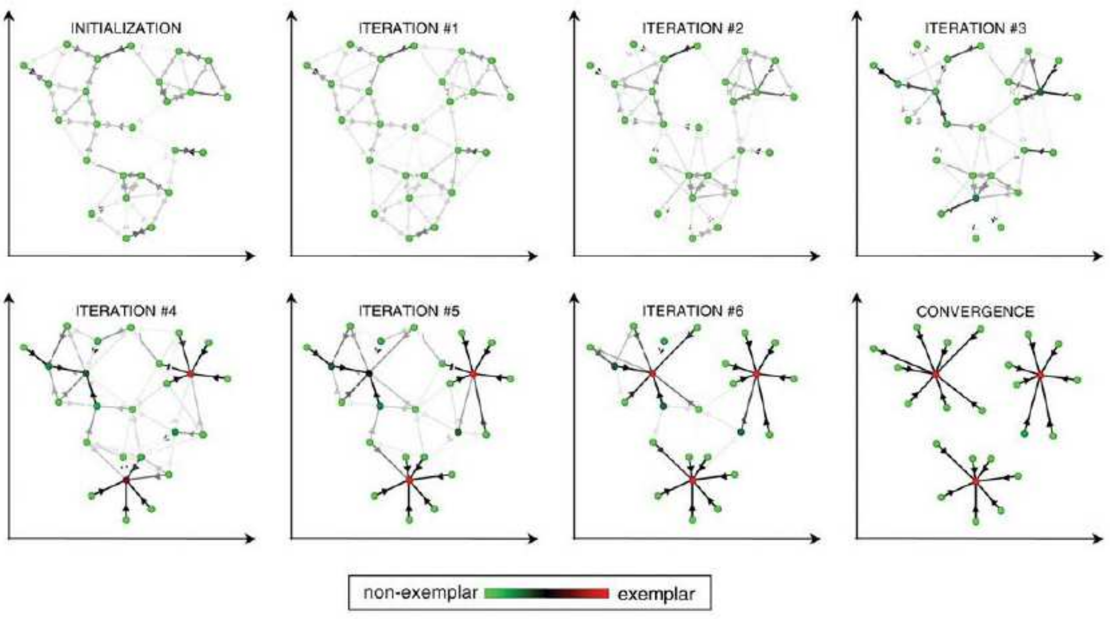


Figure 2.5: Image displaying the iterations of Affinity Propagation clustering until exemplars convergence [69].

After initialization, the algorithm starts an iterative process between 2 stages, represented by Equations 2.10 and 2.11. The first calculates another matrix called responsibility that calculates $r(i,k)$, i.e., how good k can be an exemplar of i, taking into account other possible exemplars. Second, the availability matrix, $a(i,k)$ represents how good it would be for i to pick k as its exemplar, considering the other points preferences to pick k as an exemplar.

$$r(i,k) = d(i,k) - \max_{k' \neq k}\{a(i,k') + d(i,k')\} \tag{2.10}$$

$$\begin{aligned} a(i,k) &= \min(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k)))\ for\ i \neq k \\ a(k,k) &= \sum_{i' \neq k} \max(0, r(i',k)) \end{aligned} \tag{2.11}$$

The exemplars are chosen at the end using Equation 2.12, that is, every point where the sum of responsibility and availability is greater than zero is chosen as an exemplar. The clusters are calculated with the matrices, for example, for point i the k with maximum $r(i,k) + a(i,k)$ gives the exemplar of i. The algorithm loop stops when the clusters have not changed for a number of iterations. Figure 2.5 displays the iterations of the algorithm, representing the exchange of both responsibilities and availabilities until the final convergence and the exemplars are found.

$$r(i,i) + a(i,i) > 0 \tag{2.12}$$

2.5.2 Similarity Metrics

To use clustering to select similar clients, it is common for works to use similarity metrics. A similarity metric (or measure) is a function such that $s(i, j) > s(i, k)$ if and only if the data point i is more similar to j than to k. Therefore, if a given neighbor is more similar than another, it probably has a more similar non-IID data distribution and is better for merging.

The cosine distance is commonly used as a similarity metric in the context of Federated Learning [68, 38], to measure how two models are similar to each other, and it is described in Equation 2.13. Given models w_i and w_j, the dot product of the flattened vectors are divided by the norm of both of them. This metric always gives a value in the range $[-1, +1]$, -1 meaning the data points are different and $+1$ similar.

$$CosSim(w_i, w_j) = \frac{w_i \cdot w_j}{\|w_i\| \, \|w_j\|} \tag{2.13}$$

Other possible functions that could be considered similarity measures are the inverse of the error 2.14 or the balanced accuracy 2.15. The inverse of the error is calculated in node i with local data D_i using the model from client j, and when the error is smaller, the similarity measure is bigger and they are more similar. Balanced accuracy has the same idea, being a metric inside the range $[0, 1]$.

$$s_{i,j} = 1/F_i(D_i, w_j) \tag{2.14}$$

$$s_{i,j} = BalancedAccuracy(D_i, w_j) \tag{2.15}$$

Since this is the context of neural networks, a metric used to calculate the similarity between two models is called Centered Kernel Alignment (CKA) [35]. The linear version of the CKA that is used in this dissertation can be seen in Equation 2.16.

$$X \in \mathbb{R}^{n \times p1}, Y \in \mathbb{R}^{n \times p2}, CKA = \frac{\|Y^T X\|}{\|X^T X\| \, \|Y^T Y\|} \tag{2.16}$$

X and Y are activation matrices for two different neural networks, considering only one layer of the first model and one layer of the second model (these matrices are preprocessed to center the columns). The activation of a layer is defined after passing some data examples through the network, calculating the intermediate steps, and not only the final values. In this way, the generated matrices have n data points and $p1$ or $p2$ neurons. If we compare the same layer between models, $p1 = p2$. This measure outputs a value in the range $[0, 1]$ for every pair of layers, so getting the average of these values outputs a similarity metric for the whole network. It is also possible to use just some layers, and some experiments have been performed with only the last layers of the models in this calculation.

2.6 Weight-Similarity Client Clustering (WSCC)

The Weight-Similarity Client Clustering method (WSCC) [68] uses some of the concepts described in the last sections to propose a solution for Centralized Federated Learning under non-IID data. This method leverages the cosine distance as a similarity metric, using a pair of models weights from different clients. It also uses the Affinity Propagation clustering to group similar nodes together and performs the FedAvg [48] aggregation only between the nodes of each cluster. The detailed description of the algorithm is given below.

Algorithm 5 WSCC Server Side

```
1: $w_g \leftarrow \mathbf{0}$
2: Send initial $w_g$ to clients
3: loop
4:     wait($\Delta_a$)
5:     Call the AGGREGATION procedure
6: end loop
7: procedure onReceiveModel($w_r, |D_r^t|$)
8:     Store $w_r$ and $|D_r^t|$ on the server
9: end procedure
10: procedure AGGREGATION
11:     With received models weights from clients $w_1, w_2, ..., w_N$
12:     Select a random benchmark $j \in [1, N]$
13:     Empty distance list $\theta \leftarrow []$
14:     for $i \in 1, 2, ..., N$ do
15:         $\theta_i \leftarrow CosSim(w_i, w_j)$
16:         $\theta.append(\theta_i)$
17:     end for
18:     $Clusters \leftarrow AffinityPropagation(\theta)$
19:     for $c \in Clusters$ do
20:         Merge the weights of models from cluster $c$ to generate $w_c$, using Equation 2.5
21:         Send $w_c$ to the clients belonging to cluster $c$
22:     end for
23:     Merge all the weights received to generate a new $w_g$ using Equation 2.5
24: end procedure
```

Algorithm 5 presents the logic implemented on the server side. It starts the same as the original FedAvg Algorithm 1, initializing the global model w_g, sending it to clients, and starting the main loop that waits for the aggregation round time Δ_a and calls the *AGGREGATION* procedure. The *onReceiveModel* procedure is also present to store the information from the received model.

During the aggregation process, WSCC implements additional steps compared to the original FedAvg server solution, to better handle non-IID datasets. It starts selecting a random benchmark j from the received models, and between lines 13-17 it calculates the cosine distance between the benchmark and all the other received models. Next, Affinity Propagation clustering is used, passing the cosine distance list as data points, to generate the client clusters. After that, on lines 19-22, for each cluster c the server creates an aggregated model w_c using the models from the clients of that cluster and sends the model to the cluster's clients. In the end, the regular FedAvg aggregation algorithm is used to calculate a new global model w_g, in case new nodes enter the training process.

On the client side, described by Algorithm 6, it begins initializing the local model w_i, and a threshold hyperparameter δ that will be used. The *onReceiveModel* procedure starts with an *acceptanceChecking* procedure, described in Algorithm 7. In this context, the terms acceptance checking are used to define a process to check whether or not a model that has been received will be accepted by the local vehicle. This step calculates the performance metric, in this case the balanced accuracy, both using the local model and the received model, with a local acceptance dataset D_a, which will be explained in Chapter 5. After that, the client accepts the model only if the performance metrics of the received model are better than the local model, or if the difference between them is within the threshold δ. This strategy guarantees better stability during the first steps of the training process and prevents potentially malicious nodes from attacking the whole system [68]. After this step, the rest of the algorithm follows the regular FedAvg Algorithm 2, training the model locally and sending it back to the server.

Algorithm 6 WSCC Client Side

```
1: $w_i \leftarrow \mathbf{0}$
2: Initialize threshold $\delta$ for global weight evaluation
3: procedure ONRECEIVEMODEL($w_r$)
4:     // The acceptance checking below is a step that is removed during some tests
5:     $w_i \leftarrow acceptanceChecking(w_i, w_r, D_i^a, \delta)$
6:     $w_i \leftarrow update(w_i, D_i^t, E)$
7:     Send $w_i$ and $|D_i^t|$ to the server
8: end procedure
```

Algorithm 7 Acceptance Checking

```
1: procedure ACCEPTANCECHECKING($w_i, w_j, D^a, \delta$)
2:     $y_i \leftarrow BalancedAccuracy(w_i, D^a)$
3:     $y_j \leftarrow BalancedAccuracy(w_j, D^a)$
4:     if $y_j \geq y_i$ or $diff(y_j, y_i) \leq \delta$ then
5:         $w_i \leftarrow w_j$
6:     end if
7:     Return $w_i$
8: end procedure
```

2.6.1 Summary

The concepts introduced in this chapter were used as a base to develop a hybrid solution for Federated Learning in VANETs, under different non-IID data distributions. Neural networks, especially convolutional neural networks, are the state of the art solution to general problems involving images, such as autonomous driving. The different mobility models allow the simulations to replicate real scenarios of VANETs, which improves the reliability of the experiments.

Decentralized Federated Learning, such as Gossip Learning, allows the vehicles to communicate even if they cannot communicate with the centralized server in a given moment in time. Clustering and similarity metrics are used as a solution to handle client selection under non-IID data distributions, that is, which clients are similar or should belong in the same group. With all these concepts and the WSCC method presented, a hybrid solution was proposed to dynamically select similar received models in each aggregation round both in the server and in the vehicle, which is capable of keeping the training process running even when vehicles cannot communicate with the network infrastructure.

Chapter 3

Related Work

With the theoretical concepts introduced in the last chapter, this next part summarizes the related literature works that have been used or influenced the development of this work. To better present the information, the works have been separated in Centralized, Decentralized and Hybrid Federated Learning. Each section presents a table with some main characteristics of each method, if they include vehicles or strategies to handle non-IID data, and a final note about how this dissertation is different and expands other works.

3.1 Centralized Federated Learning

This section describes some relevant works for Centralized Federated Learning that have inspired and influenced the development of this dissertation, and they are listed in Table 3.1. It started in 2016 with the original FedAvg paper [48] from Google, which was described in the last chapter. Although FedAvg was mainly targeted at IID data, FedProx [58] followed to tackle non-IID data, creating the $ProxTerm$ that was added as regularization to the function minimization during the training process and that was also described in the last chapter.

Some more strategies have been proposed to handle non-IID data, especially using clustering. The CFL (Clustered Federated Learning) [59] method first uses Federated Learning to find a global model, and at the end it iteratively splits the clients into 2 clusters using a bipartitioning algorithm with the cosine distance, until a convergence criterion is matched. Another proposed solution was the IFCA (Iterative Federated Clustering Algorithm) [21] method, in which the server keeps in parallel K models, one for each cluster, and the clients are responsible for picking one of them, using the local error to choose the best one. The downside of this method is that the number of clusters needs to be known in advance.

FeSEM (Federated Stochastic Expectation Maximization) [41] also has the disadvantage of needing the number of clusters, but it uses FedProx regularization and Expectation Maximization to dynamically calculate the clusters in each step. FedCC [79] uses the performance of the local model to accept the global model or not, similar to WSCC [68] described in the last chapter, and it also performs the clustering of clients after using the

SVD (Singular Value Decomposition) algorithm in part of the model weights and merging pairs of clusters using the Euclidean distance until the N clusters are found.

Another very similar method to the last one is FedSim [55], which uses PCA (Principal Component Analysis) on the gradients to reduce the dimensionality of the weights before applying K-Means++ for clustering. FLIS (Federated Learning by Inference Similarity) [51] is another similar approach, which uses the cosine distance on neural network activations from the last layer and applies hierarchical clustering. To conclude non-IID works, FedSNGP (Federated Spectral-normalized Neural Gaussian Process) [43] applies SNGP strategies within the neural network architecture, which calculates similarities for the clustering process.

Now, in the area of FL with vehicles, Elbir et al. [18] performed some initial comparison studies between a centralized learning solution and a distributed vehicle federated solution. FedCPF (Customized, Partial, and Flexible) [39] selects and aggregates vehicle models with a probability defined by the number of data samples, accelerating the training process.

Flexe [40] was an implementation of Federated Learning inside the OMNeT++/Veins simulation tools, which also performed practical studies of the environment under an FL system. FedSNN-NRFE (Spike Neural Networks, Neuronal Receptive Field Encoding) [77] used Spike Neural Networks for fast model training in traffic sign recognition. To conclude vehicle studies, PDP-PFL (Personalized Federated Learning with Personalized Differential Privacy) [57], which uses differential privacy for security, the model is split into shared layers and local layers only for personalization, and dynamic convolutions are used for the classification of traffic signs.

The last centralized works to be mentioned are the Meta-FL [2], which uses a 2 level hierarchical aggregation and applies security mechanisms in the intermediary layer to handle attackers, like Krum algorithm and Trimmed Mean. To conclude, Lu et al. [44] merges vehicles and non-IID data, where each vehicle uses a Bayesian Gaussian Mixture model on local data and sends the result to the server, who is responsible for running weighted K-Means to create clusters. After that, FL is applied to every cluster separately.

This dissertation contributes to the area of Centralized Federated Learning expanding the ideas implemented in WSCC [68] and FLIS [51]. These methods handle Non-IID data on the server using clustering and similarities, so this work proposes WSVC (Weight-Similarity Vehicle Clustering) that adapts these ideas to be used in VANETs.

3.2 Decentralized Federated Learning

This next section focuses on Decentralized Federated Learning works, and they are listed in Table 3.2. The original Gossip Learning (GL) [28] paper was introduced in 2019 and explained in the last chapter, and immediately afterward, Giaretta et al. [22] prepared a study to pinpoint the main disadvantages of this strategy, presenting some solutions. They showed that network connectivity, non-IID data, and devices with different speeds can affect the GL training process, leading to slow convergence. They propose to store the received models and use one of them only at the end of the gossip round, to get the most

Table 3.1: Related work methods for Centralized Federated Learning, indicating if they include vehicles or non-IID strategies, and their main characteristics.

Method	Year	Vehicles	Non-IID	Characteristics
FedAvg [48]	2016			Original work
FedProx [58]	2018		✓	Proximal term regularization
CFL [59]	2019		✓	Clustering after FL, iteratively splits the clients in 2 clusters until convergence
IFCA [21]	2020		✓	Server trains K models, 1 for each cluster, clients pick the best one
FeSEM [41]	2020		✓	FedProx and clustering made with Expectation Maximization
Elbir et al. [18]	2020	✓		Comparison with server centralized learning
FedCPF [39]	2021	✓		Select clients and aggregate with a probability defined by the number of data samples
Meta-FL [2]	2021			2 level aggregation using security mechanisms in the intermediary layer
FedCC [79]	2021		✓	Client selection based on global model performance, clustering after using SVD in part of the model weights
WSCC [68]	2022		✓	Clustering with Affinity Propagation using cosine similarity of a benchmark node and the rest
Flexe [40]	2022	✓		Framework and tests for FL in OMNeT++/Veins
FedSNN-NRFE [77]	2022	✓		Spike Neural Networks for fast model training
FedSim [55]	2022		✓	Uses PCA on the gradients and K-Means++ for clustering
FLIS [51]	2022		✓	Clustering after calculating the similarities using cosine distance on neural network last layer
Lu et al. [44]	2023	✓	✓	Bayesian Gaussian Mixture on local data, server performs weighted K-Means from BGM
PDP-PFL [57]	2023	✓		Personalized differential privacy, dynamic convolution, last layers are local only
FedSNGP [43]	2023		✓	Spectral-normalized Neural Gaussian Process to calculate the similarities for clustering

recent one. This is also similar to what Bagoly et al. [3] proposed, where the neighbors' models are stored locally, and all of them are used when the round ends, meaning that each vehicle behaves as a FedAvg client and server at the same time.

The next step after the initial proposals was to diversify the studies and start to think about other important aspects in a decentralized environment, like IoT (Internet of Things). A method called BACombo (Bandwidth-Aware Combo) [31], was proposed to use network resources efficiently in a decentralized environment. They introduce model split in segments, and also worker selection based on bandwidth prediction, to better use the network resources. After that, the original authors of Gossip Learning extended their study [29], also using the idea of model partitioning and adding token accounts to each device, to control the number of messages through the network. They also applied the concepts of proactive messages, that is, the ones sent at the end of the gossip round, and reactive messages that are sent after a model is received, to speed up the convergence process.

The last method that does not tackle vehicles or non-IID data is the following. CGL (Centrality-Aware Gossiping for Distributed Learning) [49] uses the network topology of nodes to calculate some metrics, such as centrality. The metric is used to give importance to the nodes, and the more important nodes have the gossip round time decreased, so they send their models more times to neighbors.

To start non-IID decentralized papers, PENS (Performance-Based Neighbor Selection) [54] is a GL method that only trains the local model after receiving N models from neighbors. During the merging process, they select the M top neighbors with lowest error to create the new local model. PANM (Personalized Adaptive Neighbor Matching) [38] follows similar strategies as PENS, but with additional steps. Every node here keeps a bag of potential clients, and they leverage first a Monte Carlo random expansion approach to include more neighbors inside this list, and a second phase using Expectation Maximization with similarity metrics as input (cosine distance), to also keep nodes with similar distributions for merging.

The PEPPER [6] method is focused on recommendation algorithms, and each time a client receives a gossiped model, it uses a local weighting data set to calculate the accuracy of both local and received models, to perform a performance-weighted merge. They also implemented a personalized neighbor selection on each node, storing the performance of the peers to choose the best ones, but also with an expansion phase to include potential new peers. To conclude non-IID without vehicles, MGM-4-FL (Model Gossiping Method for FL) [50] applies many strategies already listed, but each node also executes the local training choosing more times neighbors with lower error.

Going to papers with vehicles, Barbieri et al. [5] tested a strategy that uses all neighbors' models for aggregation, and only the last layers of the models are exchanged, the rest is local only. Dinani et al. [15] went for a similar approach, that each vehicle acts as a client and a server at the same time, but they also implemented OMNeT++/Veins realistic scenarios for the simulations, using a dataset of vehicle trajectory predictions. They also continued their work [16] testing a decentralized powerloss function for model aggregation.

The last 2 decentralized works use both vehicles and non-IID data. DFL-DDS (Decentralized Federated Learning with Diversified Data Sources) [66] keeps and exchanges a state vector that quantifies how much each neighbor contributes to the local model of the current vehicle, updated using the learning rate. The idea is then to minimize the Kullback-Leibler divergence to get the aggregation weights of the received models and recalculate the state vector. The last method is FedPC [80], which uses a dataset of driving action recognition. They use FedProx [58] $ProxTerm$, leverage transfer learning with the first layers of the model frozen, and there is no aggregation when receiving a model (the local model is replaced).

The main contribution of this dissertation to pure Decentralized Federated Learning is the adaptation of the centralized method WSCC [68] to execute such as an implementation of Gossip Learning. The experiments performed in Chapter 5 show the advantages and disadvantages of this approach, comparing it with centralized and hybrid approaches.

3.3 Hybrid Federated Learning

This last section introduces the hybrid methods studied, and they are listed in Table 3.3. For regular use cases, Gossip-PGA (Periodic Global Averaging) [11] implements a regular Gossip Learning algorithm, but in every couple of gossip rounds, it performs a centralized global average round, to speed up information dissemination. HL-SGD (Hybrid Local Stochastic Gradient Descent) [25] expands the last idea treating each node as a server. First, there is a step to average the neighbors' models, followed by a global averaging round.

To conclude the works without vehicles and non-IID data, there is FedPareto [32]. This work focuses on efficient device communication, so there is a hierarchy with local leaders and the global server. Each leader aggregates the models of their neighbors before sending only one model to the server for global aggregation. The leaders are selected using location and distance as criteria for K-Means clustering, and in each round, the leader uses the loss to pick the worst models to train.

In the vehicle area, IoV-SFDL (Swarm-Federated Deep Learning in IoV) [73] uses a new proprietary framework implemented by Hewlett Packard called Swarm Learning [74]. In their case, each swarm group sends the models to the RSU, which predicts an aggregated weight using a credibility-aware algorithm, which takes into account the number of times the model group was better or worse than the new global model. The last step is for the RSUs to send their models for global aggregation.

The TFL-CNN (Two-Layer FL) [82] method also considers a 2 level hierarchy with the RSUs and the global server, with aggregations in both levels. However, in the RSUs the aggregation is weighted by a parameter that is calculated based on the velocity, location, and computational power of the vehicle. These values try to map the quality of the data captured by each vehicle.

Another hybrid method is EAGLE (Edge-Assisted Gossiping Learning) [14], which uses GL between vehicles and the RSU is the FL server. When a vehicle receives a model from the server (RSU), they can use gossip for N rounds, or until no neighbor is available, before returning the model to the server for global aggregation.

Table 3.2: Related work methods for Decentralized Federated Learning, indicating if they include vehicles or non-IID strategies, and their main characteristics.

Method	Year	Vehicles	Non-IID	Characteristics
Gossip Learning [28]	2019			Original work
Giaretta et al. [22]	2019			Experiments and extension for devices with different speeds
Bagoly et al. [3]	2020			Merge all the received models at the end of the gossip round
BACombo [31]	2020			Considerations about bandwidth and model split in segments
Hegedus et al. [29]	2021			Model partitioning and token accounts to control the number of models sent
Barbieri et al. [5]	2021	✓		Only the last layers of the models are exchanged, aggregation with all the neighbors
PENS [54]	2021		✓	Train after receiving N models, select M neighbors with top performance on local dataset
Dinani et al. [15]	2021	✓		Vehicle acts as server and client at the same time
DFL-DDS [66]	2022	✓	✓	Vehicles keep a state vector, Kullback-Leibler divergence to measure diversity
PANM [38]	2022		✓	Cosine distance to measure similarity, Monte Carlo method and Expectation Maximization to create the clusters
PEPPER [6]	2022		✓	Aggregation weighted by performance on local data, top neighbors selected with expansion phase
Dinani et al. [16]	2022	✓		Powerloss function for aggregation
MGM-4-FL [50]	2022		✓	Nodes share their error to neighbors, training executed more times in nodes with lower error
CGL [49]	2022			Gossip round is smaller for nodes with more importance
FedPC [80]	2023	✓	✓	FedProx regularization, first layers of the model are frozen, no aggregation when receiving a model

Table 3.3: Related work methods for Hybrid Federated Learning, indicating if they include vehicles or non-IID strategies, and their main characteristics.

Method	Year	Vehicles	Non-IID	Characteristics
Gossip-PGA [11]	2021			Periodic global average after some gossip rounds
IoV-SFDL [73]	2021	✓		Swarm learning, groups of vehicles generate a model for the RSU, global aggregation
TFL-CNN [82]	2021	✓		Hierarchy, RSU and global aggregations, parameters like velocity and power for weighted aggregation
HL-SGD [25]	2022			Gossip averaging of neighbors and global aggregation
EAGLE [14]	2022	✓		RSU is the FL server, gossip for N rounds before returning the model to the RSU
FedPareto [32]	2022			Hierarchy with local leaders and global server, leaders selection using the distance
Taïk et al. [67]	2022	✓	✓	Cluster heads and server, leader selected using vehicles information

The last method merges vehicles with non-IID data and was proposed by Taïk et al. [67]. The environment is composed of servers and vehicle cluster heads and both perform aggregation following the hierarchy. The differential aspect here is the process to schedule the cluster heads, the vehicles of each cluster, and the global aggregation. The cluster formation is performed by the server considering different parameters such as velocity, data diversity, or model age, and the server creates a different aggregated model for each cluster.

The main objective and contribution of this dissertation is in the area of Hybrid Federated Learning, proposing the new method HVCFL (Hybrid Vehicle Clustered Federated Learning). This strategy works with vehicles and Non-IID data, without relying on building a hierarchy of vehicles, as in Taïk et al. [67].

Chapter 4

Proposed Methods

This chapter presents the proposed methods of this dissertation. It is organized in a way that builds the solution step by step, leveraging existing solutions, and integrating new ideas. It starts talking about the message broadcasting in vehicular networks and the adaptations to Gossip Learning to benefit from this environmental behavior of VANETs.

Next, two ideas are introduced which are based on the Weight-Similarity Client Clustering (WSCC) [68] method, and these ideas are the basis and will be combined to build the final hybrid solution. One proposal is Decentralized WSCC, applying the ideas of similarity metrics and clustering without a central server. The other proposal is a practical implementation of WSCC in vehicle networks. Some structural changes needed to be made to the original framework so that it was able to execute in VANETs, and this method was named Weight-Similarity Vehicle Clustering (WSVC).

To conclude, the complete solution which includes both decentralized and centralized WSCC/WSVC implementations is fully explained, which was called Hybrid Vehicle Clustered Federated Learning (HVCFL). The results of experiments, advantages, and disadvantages of the methods presented are explained and discussed in Chapter 5.

4.1 Message Broadcasting

As explained in Chapter 2, VANETs are implemented in a wireless medium, so when a vehicle sends a message, every other client close to the sender vehicle will be able to hear it. It is also possible to implement routing and multiple-hop protocols, but in this work the message broadcasting [56] is used, as in [15, 16, 40].

The disadvantage of using message broadcasting is when many nodes try to transmit a message at the same time, originating the already explained phenomenon of Broadcast Storming. In this work, to prevent many messages from colliding at the same time, a similar strategy to [15, 16] was implemented, that is, to use a larger federated learning round time. In this work, the gossip round time and the centralized approaches both use 30 seconds for each round. It makes sense to not use very small round times, since the vehicles do not have a powerful CPU or GPU to train the local models so quickly, and the network would be overwhelmed with messages of lower priority compared to other kinds of services, like safety.

As an example of broadcasting, the original Gossip Learning algorithm was slightly modified, which can be seen in Algorithm 8. The only change made is inside the main gossip loop, and instead of selecting a neighbor p and sending the model to it, the modified algorithm broadcasts the message to every other node that can hear it, being vehicles and RSUs. For communication between vehicle and central server, the RSU is the responsible agent to receive and send messages, also using broadcast to send data to vehicles, as in [82, 39, 73].

Algorithm 8 Gossip Learning with Message Broadcasting

```
1: $w_i \leftarrow \mathbf{0}$
2: loop
3:     wait($\Delta_g$)
4:     Broadcast $w_i$ and $|D_i^t|$ to neighbors
5: end loop
6: procedure onReceiveModel($w_r, |D_r^t|$)
7:     $w_i \leftarrow merge(w_i, |D_i^t|, w_r, |D_r^t|)$
8:     $w_i \leftarrow update(w_i, D_i^t, E)$
9: end procedure
```

4.2 Decentralized WSCC

This and the next section leverage the Weight-Similarity Client Clustering (WSCC) [68] method and expand it for the context of decentralized federated learning and VANETs. The first strategy that was implemented and assessed was to use the WSCC logic and implement it in a decentralized way, as shown in Algorithm 9. This method is fully decentralized, so there are no central servers or RSUs acting, but communication among vehicles only. An analysis and discussion about this method, with experiments and results, will be given in Chapter 5.

Each vehicle starts by initializing the local model w_i, an empty list of received models w_{ir}, and the threshold hyperparameter δ, used for receiving model evaluation like in the original WSCC. The *onReceiveModel* procedure is called when the vehicle receives a model from a neighbor, and only the information received is added to the w_{ir} list.

The main gossip loop of the algorithm has been modified to perform steps similar to WSCC. It starts waiting for the gossip round time Δ_g, then it checks if the vehicle has received models in the last round, R being the size of the received models list. In lines 8-12, the cosine distance similarities θ are calculated between the local model w_i and the ones received from w_{ir}. Between lines 13-15, Affinity Propagation clustering [19] is used to group the received models and find the cluster c for the current vehicle. Note that the clustering procedure excluded the current vehicle i.

Having found the cluster c with the current vehicle, representing the models with similar distributions, the merging process inside the cluster calculates the model w_c. After that, *acceptanceChecking* with Algorithm 7 using local acceptance dataset D_i^a is executed, and then the model is trained locally for E epochs. In the end, the model information is broadcast to neighbors, and the w_{ir} list is emptied.

Algorithm 9 Decentralized WSCC

```
1: $w_i \leftarrow \mathbf{0}$
2: $w_{ir} \leftarrow []$
3: Initialize threshold $\delta$ for merged weight evaluation
4: **loop**
5:     wait($\Delta_g$)
6:     $R \leftarrow w_{ir}.size$
7:     **if** $R > 0$ **then**
8:         Empty distance list $\theta \leftarrow []$
9:         **for** $j \in \{1, 2, ..., R\}$ **do**
10:            $\theta_j \leftarrow CosSim(w_i, w_{ir}[j])$
11:            $\theta.append(\theta_j)$
12:        **end for**
13:        $Clusters \leftarrow AffinityPropagation(\theta)$
14:        $c \leftarrow Clusters.predict(CosSim(w_i, w_i))$
15:        $c.add(i)$
16:        Merge the weights of models from cluster $c$ to generate $w_c$, using Equation 2.5
17:        // The acceptance checking below is a step that is removed during some tests
18:        $w_i \leftarrow acceptanceChecking(w_i, w_c, D_i^a, \delta)$
19:        $w_i \leftarrow update(w_i, D_i^t, E)$
20:    **end if**
21:    Broadcast $w_i$ and $|D_i^t|$ to neighbors
22:    $w_{ir} \leftarrow []$
23: **end loop**
24: **procedure** onReceiveModel($w_r, |D_r^t|$)
25:    $w_{ir}.append(w_r, |D_r^t|)$
26: **end procedure**
```

This algorithm executes the same steps as WSCC but in a decentralized way and with some adaptations, to check if models have been received or not. It is also important to broadcast the local model w_i at the end of the gossip loop even if no model was received, to prevent cases where every node cannot communicate with another and the training process would stop. Note that it uses the modified version of Gossip Learning that trains the local model at the end [22, 3].

4.3 Weight-Similarity Vehicle Clustering

The original WSCC work [68] was designed without considering that some nodes of the network might not be reachable at all times, as often occurs in VANETs. For example, in every round the server sends the models to all the nodes, separating them by cluster, but if the node did not participate in the last round, the node does not have a current cluster. Also, this is a broadcasting medium, so a strategy is needed to differentiate which vehicles belong to each cluster, since each should be able to get the correct model assigned to their current cluster.

With these issues listed, some modifications have been made to WSCC to adapt it to vehicle networks, and this method was called Weight-Similarity Vehicle Clustering (WSVC). This new method is a centralized federated learning solution, not including communication between the vehicles.

The Algorithm 10 represents the server (it is also used for the hybrid method presented in the next section). Here, the basic algorithm steps follow the WSCC server-side logic, with some additional steps to identify the vehicles of each cluster and to disseminate the global model to vehicles that did not participate in the last round.

It starts the same as the WSCC server, initializing the global model w_g, sending it to clients, and waiting for the round time of aggregation Δ_a, before calling the function $AGGREGATION$. In line 7, the *onReceiveModel* procedure is called, and in this case it stores the $VehicleId$ in addition to the vehicle model information, for further use.

The $AGGREGATION$ function was modified from the WSCC method to create a list of $ParticipatingVehicles$ with the $VehicleIds$ received by the server in the last round. If the server received models, between lines 14-20, it uses the same logic as WSCC to select a benchmark vehicle j, calculate the cosine similarities and use Affinity Propagation [19] to create the clusters.

At lines 21-25, for each cluster, a list $ClusterVehicles$ is created with the $VehicleIds$ of each cluster c. Next, the models of the cluster c are merged to generate the cluster model w_c. At the end of this loop, the server sends w_c, the $ParticipatingVehicles$ list, and the $ClusterVehicles$ list to the clients belonging to the cluster c. With this information, each vehicle receiving a server model will be able to identify if they have participated in the last round and which cluster they belong to.

To conclude the server-side algorithm, all participating models are merged to generate a new global model w_g. Then in line 28, there is an important additional step to send the w_g and $ParticipatingVehicles$ list to vehicles, indicating that this message is the global model. This is important because some vehicles might not have participated in the last round of aggregation, so they can use the global model or check the models in the clusters to choose the best.

Algorithm 11 presents the vehicle logic. It starts by initializing the local model w_i, the local $VehicleId$, and the threshold δ for acceptance check. Then the *onReceiveModel* procedure receives a model w_r, the $ParticipatingVehicles$ list of the last round of aggregation, and the $ClusterVehicles$ for model w_r.

Next in line 5, the vehicle performs some checks to accept or not the model. It first checks if the data are from the server and if the local $VehicleId$ belongs to the $ClusterVehicles$ list, indicating that the vehicle is from the cluster represented by the received model. It also checks if $VehicleId$ is not in $ParticipatingVehicles$, which means that the current vehicle did not participate in the last round of aggregation. If either of the conditions above is true, it means that the cluster is correct or the vehicle did not participate in the last aggregation round, so it should pick the best model for its training. If both of them are false, either the vehicle belongs to another cluster, or the message is from another vehicle and not from an RSU.

If the vehicle did not participate in the last round, it will wait for the server to send all the models generated in the last centralized round $w_{c1}, w_{c2}, \ldots, w_g$, for all clusters and the global model. It waits a time Δ_c, because some models might not arrive to the vehicle in the event of network failures. Then, it defines w_r as the best received model considering *BalancedAccuracy* in the local acceptance dataset D_i^a. Doing this strategy, the vehicle is able to choose which cluster model best fits its current data, or pick the global model if none of them are good.

Algorithm 10 WSVC/HVCFL Server Side

```
 1: $w_g \leftarrow \mathbf{0}$
 2: Send initial $w_g$ to clients
 3: loop
 4:     wait($\Delta_a$)
 5:     Call the AGGREGATION procedure
 6: end loop
 7: procedure onReceiveModel($w_r, |D_r^t|, VehicleId$)
 8:     Store $w_r$, $|D_r^t|$ and $VehicleId$ on the server
 9: end procedure
10: procedure AGGREGATION
11:     With received models weights $w_r$ and $VehicleIds$ from clients $w_1, w_2, ..., w_N$
12:     $ParticipatingVehicles \leftarrow VehicleIds$
13:     if $w_r.size > 0$ then
14:         Select a random benchmark $j \in [1, N]$
15:         Empty distance list $\theta \leftarrow []$
16:         for $i \in 1, 2, ..., N$ do
17:             $\theta_i \leftarrow CosSim(w_i, w_j)$
18:             $\theta.append(\theta_i)$
19:         end for
20:         $Clusters \leftarrow AffinityPropagation(\theta)$
21:         for $c \in Clusters$ do
22:             $ClusterVehicles \leftarrow VehicleIds$ in $c$
23:             Merge the weights of models from cluster $c$ to generate $w_c$, using Equation
    2.5
24:             Send $w_c$ to the clients belonging to cluster $c$, including
    $ParticipatingVehicles$ and $ClusterVehicles$
25:         end for
26:         Merge all the weights received to generate a new $w_g$ using Equation 2.5
27:     end if
28:     Send $w_g$ and $ParticipatingVehicles$ to the vehicles, indicating this is the global
    model
29: end procedure
```

If the vehicle accepts the received models, the steps are the same as the WSCC logic. First, it performs the local acceptance check using the local acceptance dataset D_i^a and the threshold δ. After that, it trains the model for E epochs on the local training dataset D_i^t, before sending the new local model information and the $VehicleId$ back to the server.

Algorithm 11 WSVC Client Side

```
1: $w_i \leftarrow \mathbf{0}$
2: Initialize $VehicleId$
3: Initialize threshold $\delta$ for global weight evaluation
4: procedure onReceiveModel($w_r, ParticipatingVehicles, ClusterVehicles$)
5:     if $Is\ from\ server\ and\ (VehicleId \in ClusterVehicles\ or\ VehicleId \notin ParticipatingVehicles)$ then
6:         if $VehicleId \notin ParticipatingVehicles$ then
7:             Wait $\Delta_c$ time for server to send all the centralized models $w_{c1}, w_{c2}, \dots, w_g$
8:             Overwrites $w_r$ using the model with the best $BalancedAccuracy$ on the local acceptance dataset $D_i^a$
9:         end if
10:        // The acceptance checking below is a step that is removed during some tests
11:        $w_i \leftarrow acceptanceChecking(w_i, w_r, D_i^a, \delta)$
12:        $w_i \leftarrow update(w_i, D_i^t, E)$
13:        Send $w_i$, $|D_i^t|$ and $VehicleId$ to the server
14:    end if
15: end procedure
```

4.4 Hybrid Vehicle Clustered Federated Learning

The centralized methods are good for aggregating all the information from multiple parts of the network, but the centralized server entity might be a problem if nodes in the network cannot reach it. Decentralized methods are capable of training models even with partially disconnected networks, allowing communication between vehicles, but have the information dissemination issue [22] that also hinders the training process.

Therefore, to obtain the strengths of both centralized and decentralized methods, a hybrid strategy is proposed in this work, called Hybrid Vehicle Clustered Federated Learning (HVCFL). It merges the implementations from the last two sections, Decentralized WSCC and WSVC (Weight-Similarity Vehicle Clustering).

Taking a look at Figure 4.1, there is a central server that executes the WSVC logic, performing synchronized rounds of centralized federated learning aggregation for vehicles based on the WSCC [68] method. The server communicates with the RSUs to receive the models from the vehicles and send the server models to them. Vehicles can leverage Vehicle-to-Infrastructure (V2I) communication with the RSUs to receive the clustered or global models from the server, or send the trained models for centralized aggregation.

On the vehicle side, they also leverage Vehicle-to-Vehicle (V2V) communication to gossip the models between them. In doing this, they are able to merge models from neighbors in case they are not in range of an RSU, so that they can continue the training process even with the network partially disconnected. Note that it is also important to

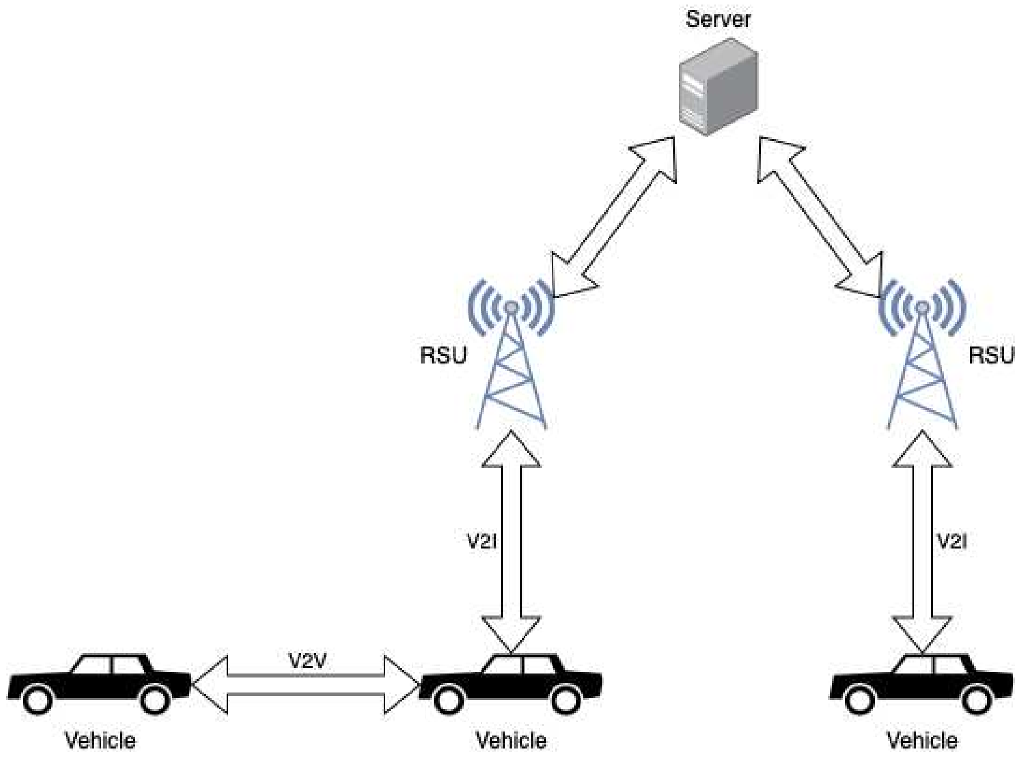


Figure 4.1: Solution architecture with a central server that performs centralized Federated Learning procedures, the RSUs that are the bridge between the vehicles and the server, and the vehicles that perform both centralized and decentralized Federated Learning procedures.

correctly synchronize the gossip or centralized rounds, so that the vehicles can decide if they are going for the decentralized approach or the centralized one at a given point in time. The details of this logic will be described in the client-side algorithm.

For the server side, it also uses the algorithm for WSVC presented in the last section 10. An important note here is that the messages sent by the server need to have some kind of identification, so that the vehicles can distinguish which ones are from vehicle neighbors and which ones come from RSUs.

On the client side, the Algorithm 12 describes the steps for HVCFL, including the logic to handle both centralized and decentralized approaches at the same time. The initialization starts with the local model, the received models list w_{ir}, the $VehicleId$, and the hyperparameter δ for the acceptance check of the new merged model.

The main loop of the application handles the decentralized gossip logic, similar to Algorithm 9 of the Decentralized WSCC. It waits for the gossip round time Δ_g, then checks if models have been received. From lines 9-16, it creates the cosine similarity list θ between the local model and the received ones, and uses Affinity Propagation [19] to find the clusters. Finally, it gets the cluster c of the current vehicle. After merging the models of c to generate the new model w_c, there is the *acceptanceChecking* procedure and local model training. In the end, the model information and $VehicleId$ are broadcast to neighbors, including the server through the RSUs, and w_{ir} is reset.

This first part covers the logic of the vehicle executed when there is no contact with the server, using a purely decentralized gossip approach. It is also possible that during the training procedure the vehicle entered the coverage area of an RSU, so it also sends its model to the server for aggregation, even if it had not participated in the last centralized round. However, to include the centralized logic, it is necessary to implement some synchronization mechanisms.

The *onReceiveModel* procedure is responsible for deciding whether the vehicle will use the decentralized or centralized approach. If the message received is not from the server (through an RSU), it only stores the model locally in w_{ir}, at line 38. However, if it is a model from the central server, there are some possibilities. If $VehicleId$ is on the list $ClusterVehicles$, it means that the vehicle participated in the last aggregation round, and the model received w_r is of the correct cluster. If $VehicleId$ is not on the $ParticipatingVehicles$ list, it means that the vehicle did not participate in the last centralized aggregation round. If that is the case, the vehicle will use the same strategy as the WSVC client Algorithm 11 to pick the best model from the server, between lines $28 - 31$ of the HVCFL client Algorithm 12.

If the vehicle accepts the model from the server, it stops the gossip loop and the ongoing training if it is executing it, to synchronize with the server. It restarts the loop from the beginning of Δ_g for synchronization, because the vehicle can exit the RSU coverage area, so it is important to continue the training process with the decentralized approach in the next round if the server is not reachable. After that, the process is the same as the other algorithms presented, to perform the *acceptanceChecking* procedure, local training, broadcasting of model information to server and neighbors, and reset w_{ir}.

With all the server and client logics explained, there are some execution possibilities for message communication between the agents. Figure 4.2 displays a sequence diagram with an example of the message exchange procedure between the server (S), an RSU, and two vehicles (V1 and V2).

This sequence diagram starts with server S performing the centralized aggregation procedure, and vehicle V2 is participating in this aggregation round, i.e., it has sent its model to the server. In this example, for the sake of simplicity, the server sends only one cluster model, named C1, and one global model, and the vehicles will accept them, V1 with the global model and V2 with C1. In a real scenario, the vehicles could drop the received models after the acceptance checking step; however, it will be considered that the models received in this diagram are accepted.

After the server aggregation, it sends both cluster C1 and global models to the RSU, to be sent over IEEE 802.11p to the vehicles. The vehicle V2, in this case, was used to train the model for the cluster C1, and since V2 is in the range of the RSU, it will start V2's local training procedure after picking C1.

At the same time, vehicle V1 is not in the coverage area of an RSU, so it will not receive the global model from the server. Considering that V1 received some models from other vehicles, it will start the decentralized V1 local training, triggered at the end of the gossip round time, which in this case is not synchronized with the server aggregation rounds because V1 has not communicated with the server yet.

Algorithm 12 HVCFL Client Side

```
1: $w_i \leftarrow \mathbf{0}$
2: $w_{ir} \leftarrow []$
3: Initialize $VehicleId$
4: Initialize threshold $\delta$ for merged weight evaluation
5: loop
6:     wait($\Delta_g$)
7:     $R \leftarrow w_{ir}.size$
8:     if $R > 0$ then
9:         Empty distance list $\theta \leftarrow []$
10:        for $j \in \{1, 2, ..., R\}$ do
11:            $\theta_j \leftarrow CosSim(w_i, w_{ir}[j])$
12:            $\theta.append(\theta_j)$
13:        end for
14:        $Clusters \leftarrow AffinityPropagation(\theta)$
15:        $c \leftarrow Clusters.predict(CosSim(w_i, w_i))$
16:        $c.add(i)$
17:        Merge the weights of models from cluster $c$ to generate $w_c$, using Equation 2.5
18:        // The acceptance checking below is a step that is removed during some tests
19:        $w_i \leftarrow acceptanceChecking(w_i, w_c, D_i^a, \delta)$
20:        $w_i \leftarrow update(w_i, D_i^t, E)$
21:    end if
22:    Broadcast $w_i$, $|D_i^t|$ and $VehicleId$ to the server and neighbors
23:    $w_{ir} \leftarrow []$
24: end loop
25: procedure onReceiveModel($w_r, |D_r^t|, ParticipatingVehicles, ClusterVehicles$)
26:    if $Is\ from\ server\ and\ (VehicleId \in ClusterVehicles\ or\ VehicleId \notin ParticipatingVehicles)$ then
27:        Stop gossip Loop (and ongoing training) and restart from the beginning of $\Delta_g$
28:        if $VehicleId \notin ParticipatingVehicles$ then
29:            Wait $\Delta_c$ time for server to send all the centralized models $w_{c1}, w_{c2}, \ldots, w_g$
30:            Overwrites $w_r$ using the model with the best $BalancedAccuracy$ on the local acceptance dataset $D_i^a$
31:        end if
32:        // The acceptance checking below is a step that is removed during some tests
33:        $w_i \leftarrow acceptanceChecking(w_i, w_r, D_i^a, \delta)$
34:        $w_i \leftarrow update(w_i, D_i^t, E)$
35:        Broadcast $w_i$, $|D_i^t|$ and $VehicleId$ to the server and neighbors
36:        $w_{ir} \leftarrow []$
37:    else if $Not\ from\ server$ then
38:        $w_{ir}.append(w_r)$
39:    end if
40: end procedure
```

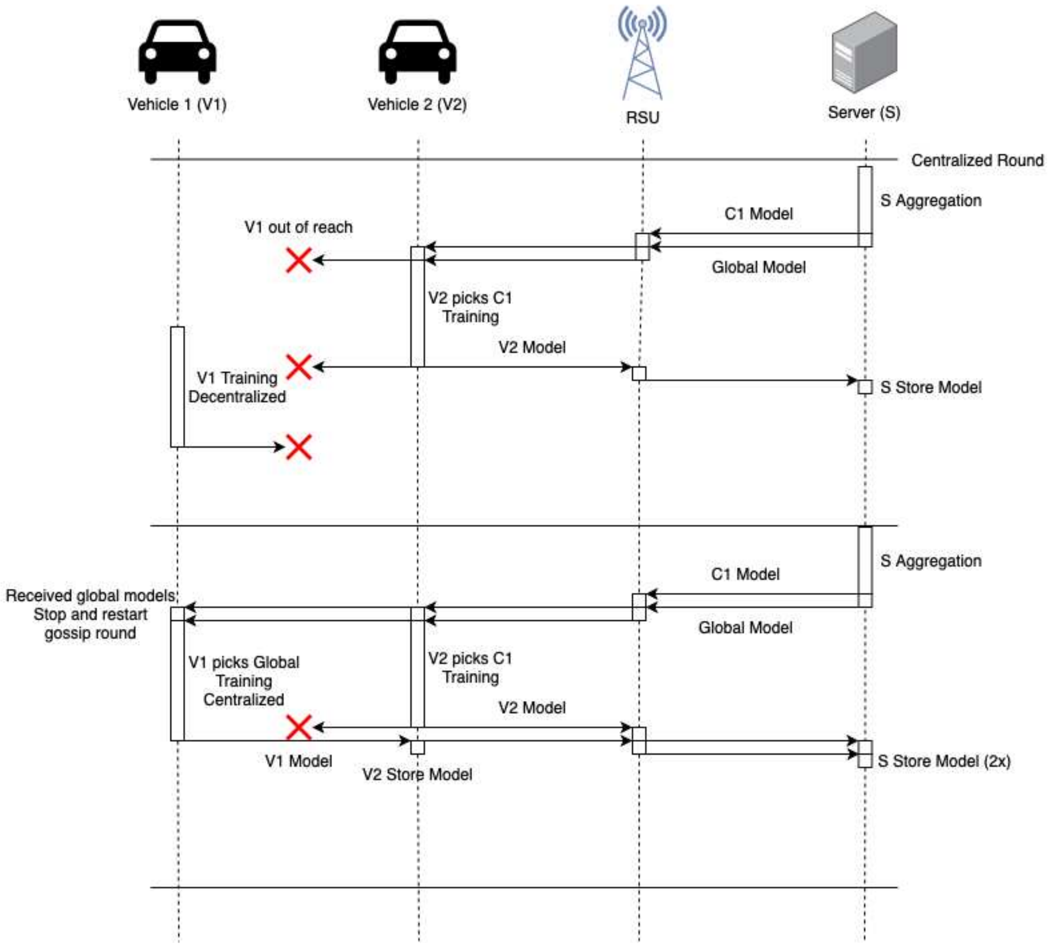


Figure 4.2: Sequence diagram of Hybrid Vehicle Clustered Federated Learning (HVCFL), displaying some examples of interactions between vehicles, RSUs and the central server, and the synchronization solutions adopted.

While V1 is training, V2 finishes the local training and broadcasts the V2 model to its neighbors. The RSU is in coverage of V2, so it receives the model and sends it to the server to store it, but it does not reach V1, because V1 and V2 in this case are not in the coverage space of one another. After that, V1 finishes the local decentralized training and broadcasts its model, but neither V2 nor the RSU receive the message.

After the above steps, the server triggers another round of aggregation. The server generates another cluster model C1 and the global model, and sends them to the RSU to broadcast to the vehicles. For simplicity, the C1 cluster model is considered to also contain the vehicle V2, but in practice it could be a different cluster assigned to vehicle V2.

The process for V2 starts in a similar way, receiving model C1 from the server and starting the local training. However, now V1 has entered the RSU coverage area, so it is capable of correctly receiving the centralized models. After that, V1 stops and restarts the local gossip round, to synchronize with the server. It also triggers the procedure to pick which model is the best one using the local acceptance dataset, and chooses the global model over C1 in this case.

In this diagram example, V2 finishes the local model training before V1, so it broadcasts the model to its neighbors. While the RSU receives the V2 model and the server stores it, the message does not reach the V1 vehicle. This is an example that could occur even if both vehicles are in the RSU coverage area. Since the wireless protocol has a short transmission range, the vehicles could be in range of the RSU but not in range of each other, or there is an obstacle between them.

Another interesting aspect that could occur is depicted after V1 finishes the local training. In this case, it is able to send the model to the RSU, so the server stores it, and also to vehicle V2 that stores the V1 model locally. Between the time V2 finished training and V1 finished training, the vehicles could have been in contact, so V2 was not seeing V1 at first, but V1 was able to reach V2.

Another possibility that could happen with any vehicle is the following. At first, they were able to receive a server model from the RSU, but during the local training procedure, they moved outside of the coverage area. If this happens, they will not be able to send the new trained model to the server again, but neighboring vehicles could receive the model. This is important because the training procedure of the nodes does not stop, even if the network is partially disconnected, and when the vehicles cannot participate in the centralized aggregation round.

Chapter 5

Experiments

This chapter presents the experiments that were carried out with existing methods from the literature and the proposals of this dissertation. It starts with the methodology, presenting the tools used, mobility scenarios, the simulation parameters like round times or message range, the datasets preparation, neural networks and metrics. Then a variety of experiment results are displayed, with a thorough analysis and discussion of them, including the advantages and disadvantages of each method tested.

5.1 Methodology

5.1.1 Tools

To try as best as possible to simulate a real-world scenario and environment of highly dynamic and wireless short-range networks, the tools OMNeT++, Veins and SUMO [71, 63, 42] have been used in the experiments. OMNeT++ (Objective Modular Network Testbed in C++) [71] is a discrete event-based network simulation tool, while Veins (Vehicles in Network Simulation) [63] is an OMNeT++ module that implements a framework for VANET simulation, and SUMO (Simulation of Urban Mobility) [42] simulates road vehicle traffic and mobility in a map scenario.

In Figure 5.1, it is possible to see how the integration of OMNeT++, Veins, and SUMO works. Veins is a module of OMNeT++, so it runs inside it, and SUMO is a separated application. Thus, Veins bridges OMNeT++ and SUMO implement a protocol called TraCI (Traffic Control Interface), so that the pair OMNeT++/Veins handles wireless channel simulation and Veins/SUMO the position, velocity, and route of the vehicles.

Another important feature embedded is the simulation of wireless message collisions and object shadowing, such as buildings, which helps the simulation to be realistic. For the experiments, a part of the Luxembourg SUMO Traffic Scenario (LuST) [13] was used because it represents a real city with different roads with different max speeds, and buildings. Figure 5.2 displays the complete map.

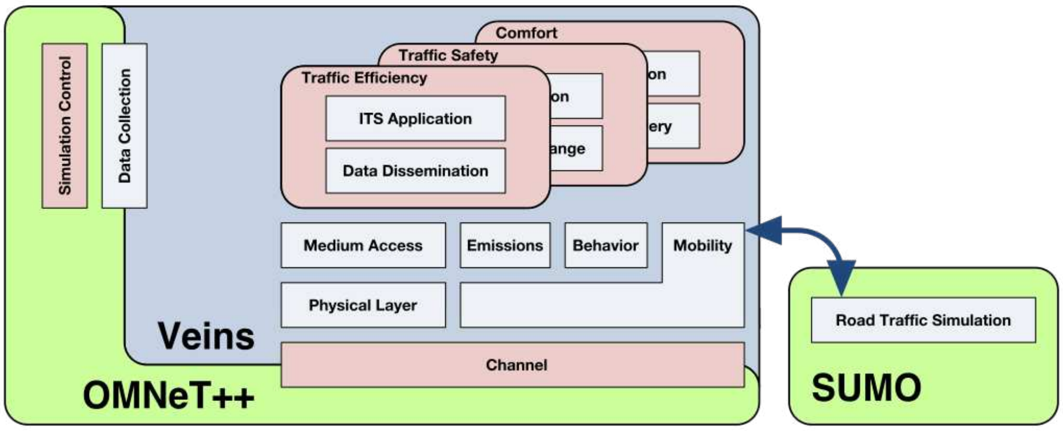


Figure 5.1: Architecture of the integration between OMNeT++, SUMO and Veins [63].

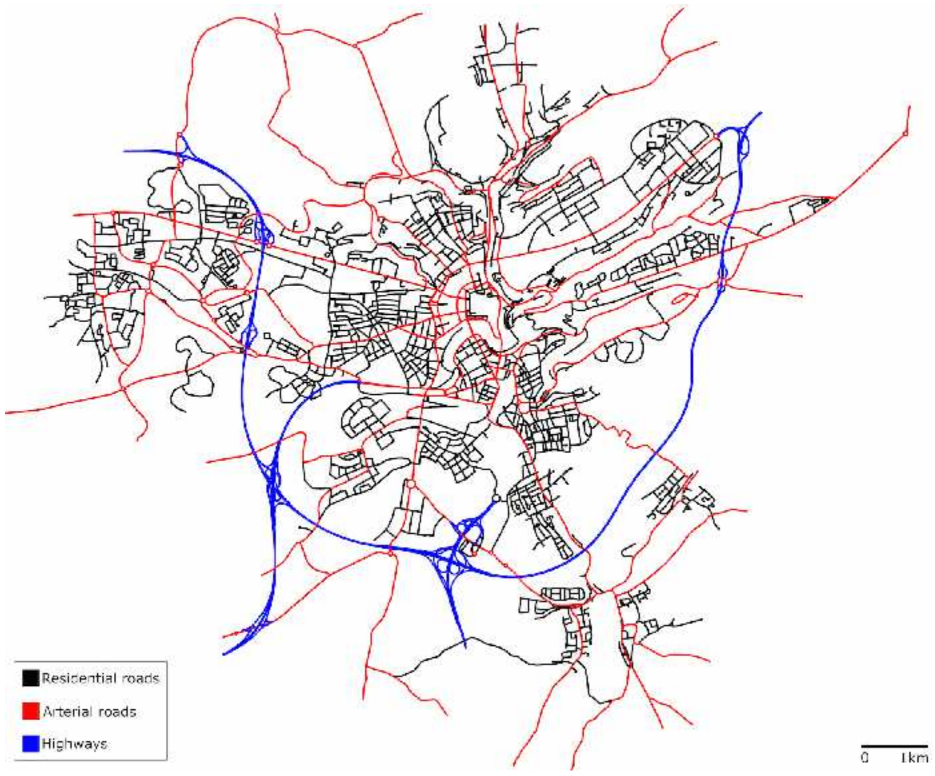


Figure 5.2: Luxembourg SUMO Traffic Scenario (LuST) [13].

5.1.2 Scenarios

In terms of scenarios used for testing, Figures 5.3, 5.4 and 5.5 show the same squared region of the original LuST scenario that was used in the experiments. The region's size is approximately $3000m \times 2200m$, and the 3 different images represent the same map with 12, 8 and 6 RSUs. During sections of the experimental results, it will be possible to see how the RSU coverage improves or hinders the training process, especially of centralized federated learning methods. There is also a centralized server in all scenarios that is connected to all the RSUs.

To generate different experiments, two mobility models have been used, the Manhattan Mobility Model and the vehicle traces that come with LuST. The Manhattan model is good for representing vehicles that have long trips around the city, such as buses, taxis, and app drivers, while the traces usually represent short trips from one point to another. The main focus of the experiments is on the Manhattan model, with initial tests using traces.

The vehicles generated in both scenarios always try to use the maximum speed of their current road if possible, and 100 vehicles were used in all experiments. In the Manhattan model, vehicles are added one by one to the map, starting at time 0.0 and with a gap of 0.2 seconds to add the next vehicle. In this way, all vehicles are added until 20 seconds have passed, and since 30 seconds is the base round time for all the methods, the vehicles are spread through the round period. This is important to simulate real examples for which the vehicles can join the training process at any time, and 0.2 seconds of separation is small enough for some vehicles to transmit models at the same time and test message conflicts. The total execution time of the experiments varied from 900 to 3600 seconds (15 minutes to 1 hour).

For the vehicles traces, they also were added one by one, but passing 6 seconds between each vehicle. In doing this, it was possible to generate a scenario with a variable number of vehicles on the road at each time, while the Manhattan scenario always had 100 vehicles. The total time for the short trips experiments was 900 seconds (15 minutes).

5.1.3 Simulation Parameters

To approximate the simulations as best as possible to a real-world scenario, some parameters for the simulation and timing were established. As already stated, a rectangle of $3000m \times 2200m$ of the LuST [13] scenario was used, and a maximum number of 100 vehicles was chosen for all experiments. Their speeds are always adjusted according to the traffic on the road, but by default they try to drive using the maximum allowed speed, which is 13.89 m/s in most cases.

In terms of communication parameters, the idea was to establish a maximum transmission range of 500 meters for both vehicles and RSUs, as in [78]. An experiment was carried out to test the transmission range using two static elements within the scenario, without objects blocking the message path. The parameters set were 20 mW of transmission power, -92 dBm of minimum power level, and 6 Mbit/s to achieve the desired range. Note that during the experiments, this range usually decreases a lot due to objects that hinder signal transmission on the wireless medium.

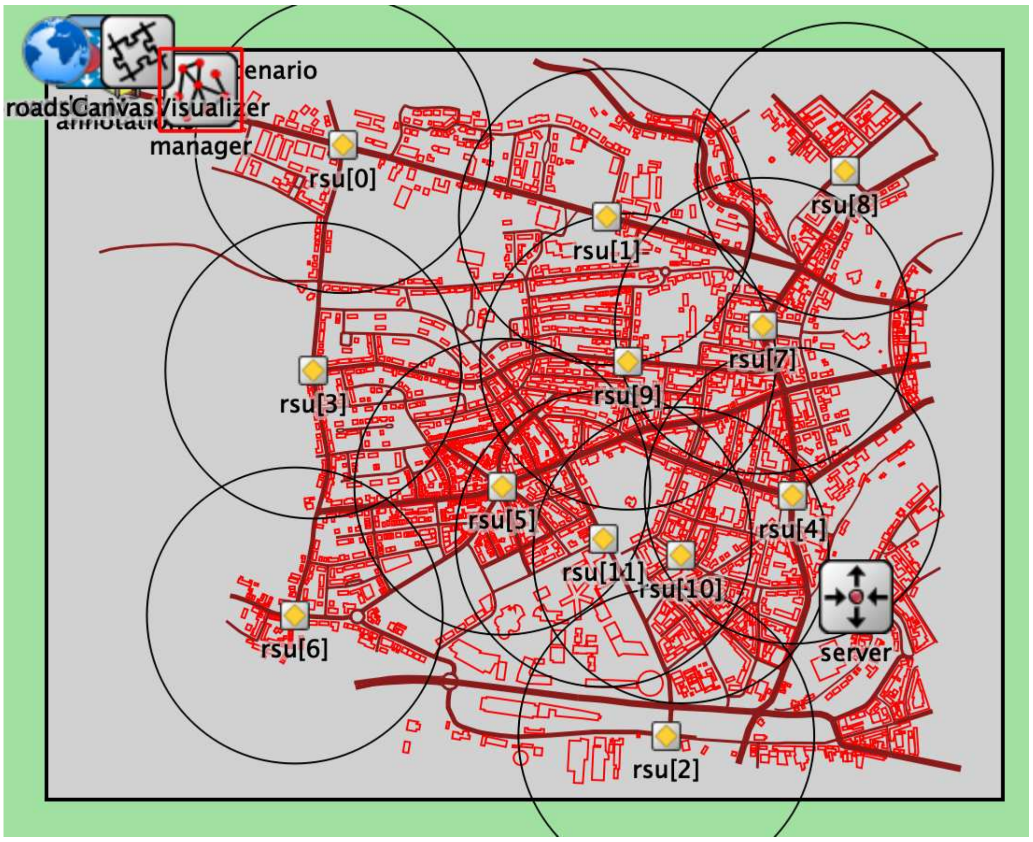


Figure 5.3: LuST Scenario with 12 RSUs.

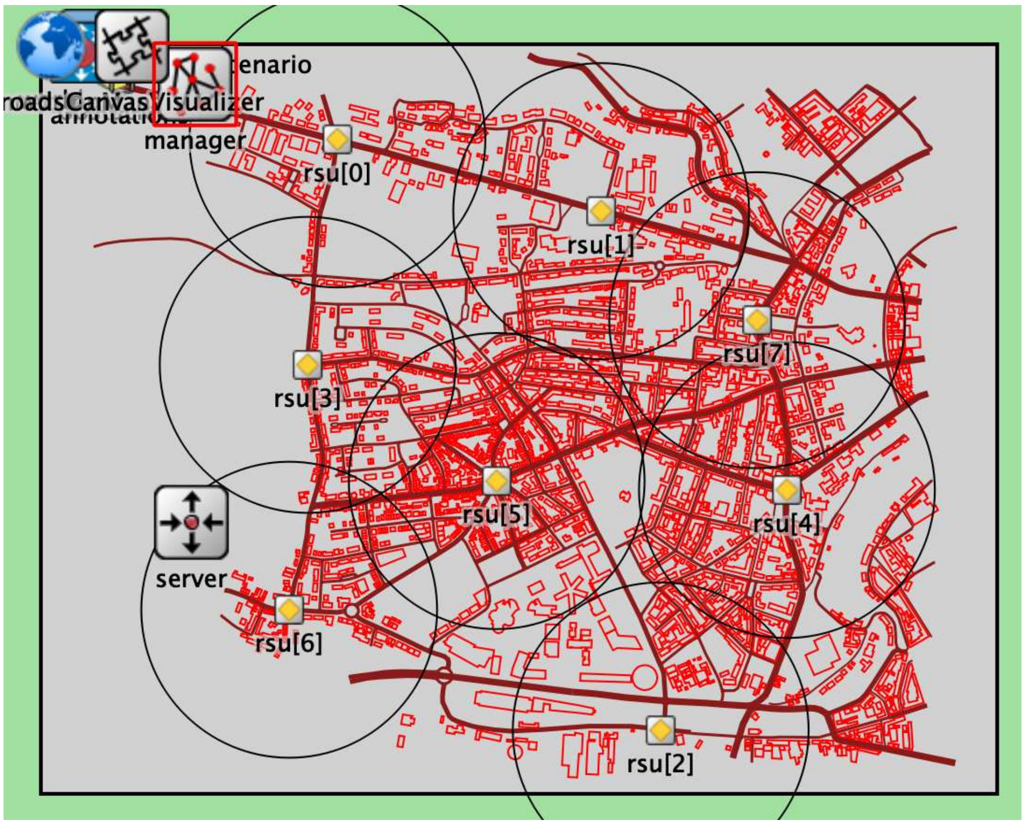


Figure 5.4: LuST Scenario with 8 RSUs.

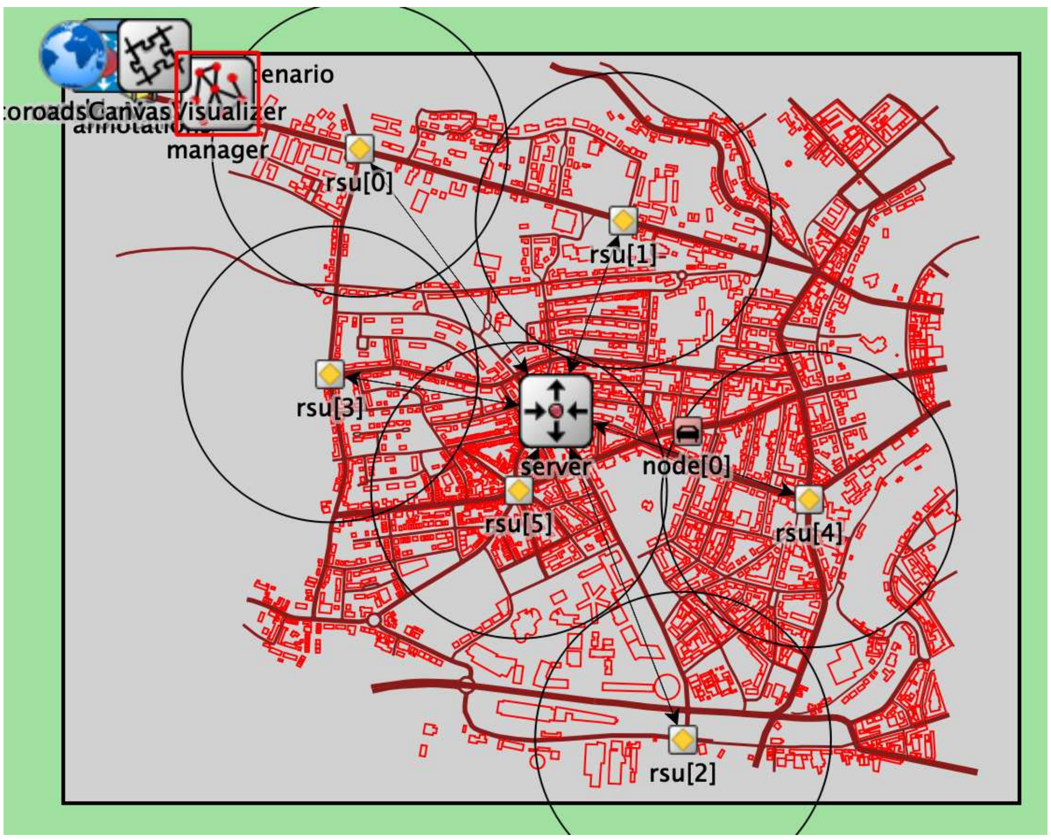


Figure 5.5: LuST Scenario with 6 RSUs.

Table 5.1: Simulation parameters common to all experiments.

Parameter	**Value**
Simulation area	3000m x 2200m
Scenario	LuST
Number of vehicles	100
Vehicle speeds	Maximum of road (mostly 13.89 m/s)
Transmission power	20 mW
Min power level	-92 dBm
Bitrate	6 Mbit/s
Transmission range	500m

Table 5.2: Parameters related to the simulated timing of the operations executed by the vehicles and the server inside OMNeT++/Veins experiments.

Parameter	**Value**
Centralized round time	30 seconds
Server aggregation time	5 seconds
Gossip round time	30 seconds
Local training time	12 seconds + uniform(0.0, 5.0)
Local time with clustering/acceptance	15 seconds + uniform(0.0, 5.0)

However, it is also important to simulate the timing of activities within the simulation. For example, the time it takes for a message to be sent by vehicle V1 and completely received by vehicle V2 is already simulated by OMNeT++/Veins.

To avoid broadcast storming of messages, a small round time is usually not used for centralized/decentralized federated learning. In this work, 30 seconds are used for each round, as in [7, 27].

The processing time is also important, so the 30 seconds were divided, starting with 5 seconds for server aggregation. In addition, the time each vehicle takes to perform local processing also affects when a message will be sent through the network. For local training, 12 seconds with an additional value from 0 to 5 derived from a random uniform distribution were used, to make each vehicle behave differently each time. For experiments with methods that perform local clustering or acceptance checking, 15 seconds were used with the same random value added as well.

5.1.4 Datasets

For the experiments, 4 datasets have been used to capture different types of images and try to reproduce different data distributions, as can be seen in Figure 5.6:

- **Fashion MNIST (FMNIST)** [76]: Collection of clothes images used for image classification tasks, which contains 10 different types of clothes.
- **CIFAR10** [36]: Collection of images of 10 different types, including animals like frogs and horses, and vehicles like trucks and ships.
- **GTSRB (German Traffic Sign Recognition Benchmark)** [65]: Dataset containing 43 different traffic signs from Germany.
- **Federated MNIST (FEMNIST)** [9]: Images of handwritten characters from different writers, containing 62 types of images.

Some preparations needed to be done on each dataset before submitting them to the neural networks. For all datasets, except Federated MNIST, the images pixels were divided by 255 to normalize the values between 0 and 1, but no color change was applied to any dataset (no RGB to grayscale). For Fashion MNIST and CIFAR10, the original image sizes were used, while in Federated MNIST the images were resized to 28 x 28 pixels. In GTSRB, the same preparation as in [75] was used, cropping the rectangular images in the center to create a square and resizing them to 48 x 48 pixels. For Federated MNIST, the data were generated using the scripts provided by [9], using the following parameters:

- **-s niid**: to retrieve the data in a non-IID manner.
- **–sf 0.25**: fraction of the total dataset used.
- **-k 200**: minimum number of samples per user.
- **-t sample**: partition each user's samples into train-test groups.
- **–tf 0.8**: fraction of data in the training set.
- **–smplseed 123**: seed to be used before random sampling of data.
- **–spltseed 456**: seed to be used before random split of data.

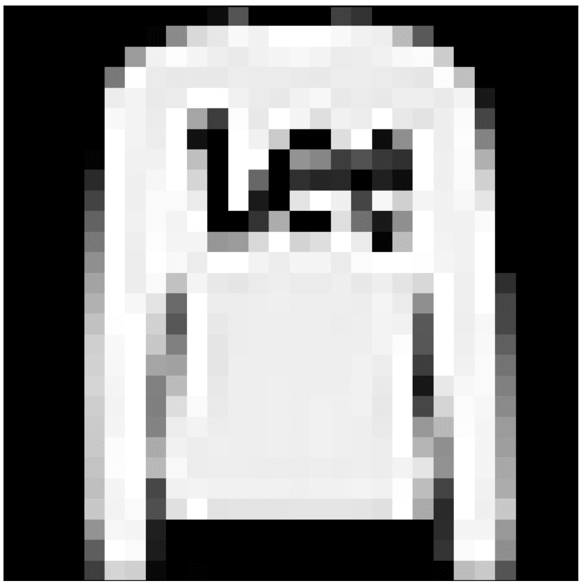
For the experiments, all datasets were separated following the strategy in Figure 5.7. All datasets have training and testing examples, so each of these sets was divided among the 100 vehicles of the experiments. The testing dataset was used only at the end to validate the experiments with new unseen data. An important note here is that the data inside the vehicles is static, so each vehicle maintains the same data points throughout the experiment.

The training dataset was also divided into 6 parts to create special subsets. 4 parts (66.6%) were used effectively as training samples, while 1 part (16.7%) was used as acceptance data and 1 part (16.7%) as validation data. The validation data are important to tune the hyperparameters of the model and check the solutions and methods implemented, while the acceptance dataset is used locally in each vehicle to test whether a received or merged model should be accepted. The acceptance dataset needs to be separated from the other data so that it uses new data not seen during training and to prevent a biased model that always improves based on the validation dataset.

Also, the experiments were run 3 times using K-fold cross-validation (except in some experiments that were pointed out), shifting the acceptance and validation sets to provide a better validation of the performance of each solution, with the same seed to reproduce the exact events that would be random. To calculate the metrics displayed in the next sections, the average value was used for each vehicle and each fold of the cross-validation.

To generate data in a non-IID manner some techniques were used, except in Federated MNIST that already provided a standard way to generate such distribution:

- **Image Rotations**: Create different clusters of users by rotating the images in either 0 and 180 degrees or 0, 90, 180 and 270 [54, 21, 38].
- **Dirichlet Distribution**: Separate the data according to the Dirichlet distribution, assigning a very different number of samples per vehicle [77, 51]. There is a parameter α that was set to 0.9 in GTSRB and 0.5 in other datasets. Also, note that here the users are not separated into clusters.



(a) Fashion MNIST image example.

(b) CIFAR10 image example.

(c) GTSRB image example.

(d) Federated MNIST image example.

Figure 5.6: Example images of the datasets used in the experiments.

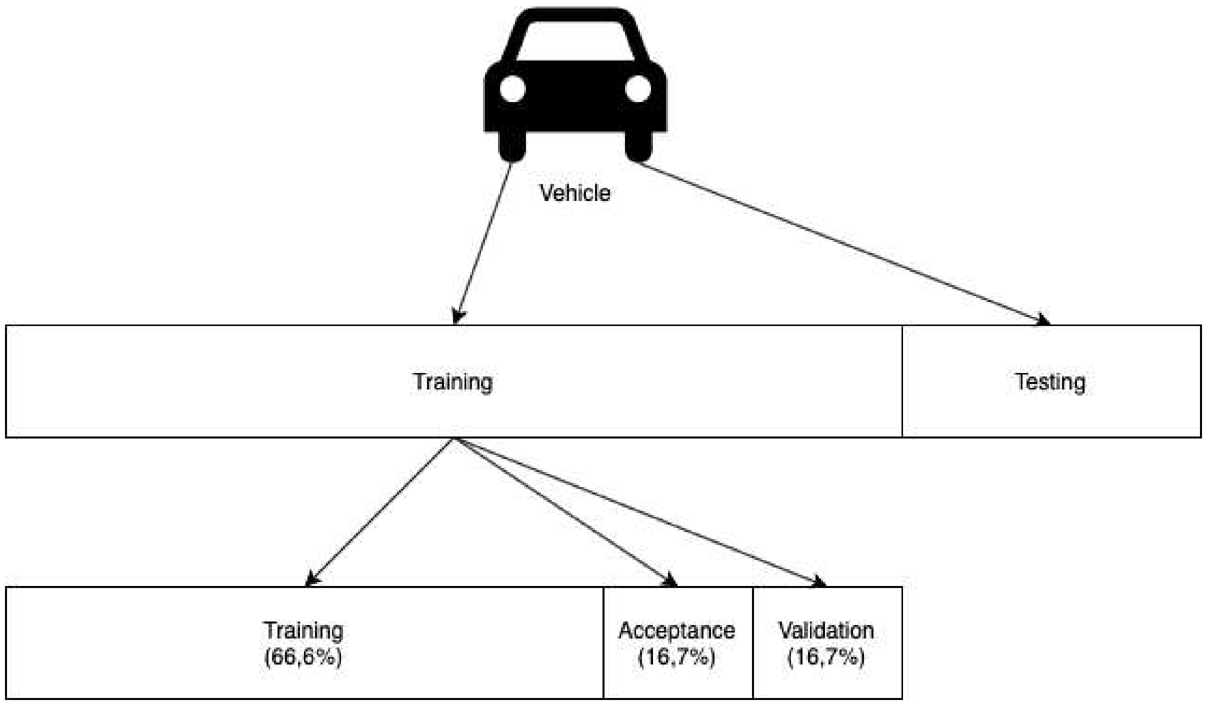


Figure 5.7: Datasets separation between testing, training, validation and acceptance sets.

Table 5.3: Data distribution for Fashion MNIST dataset similar to the one used in WSCC paper [68].

Group	Label Set	Training Samples	Vehicles
1	{0,1,2,3,4}	800	30
2	{5,6,7,8,9}	800	30
3	{2,3,4,5,6,7}	960	30
4	{3,4}	320	10

- **WSCC** [68]: Similar distributions for the Fashion MNIST and CIFAR10 datasets used in the WSCC paper. For simplification of the explanation, we will number the labels of the datasets from 0 to 9, and the distributions can be seen in Tables 5.3 and 5.4. Note that the training samples number refers to the quantity of data only in the training set, excluding the acceptance, validation, and testing sets.

5.1.5 Neural Networks

For the experiments, 2 neural network architectures have been used depending on the dataset. For Fashion MNIST and CIFAR10, the Lenet-5 architecture was used, as in [51], and for GTSRB and Federated MNIST, MicronNet [75]. Figures 5.8 and 5.9 display a diagram of Lenet-5 and MicronNet, both based on convolutional layers. A detailed description of each network, with the number of parameters in each layer and the total size, can be seen in Tables 5.5, 5.6, 5.7, and 5.8.

In the training process, all experiments used a batch size of 50. For Fashion MNIST,

Table 5.4: Data distribution for CIFAR10 dataset similar to the one used in WSCC paper [68].

Group	Label Set	Training Samples	Vehicles
1	{1,2,3,4,5}	300	40
2	{1,2,3,4,5}	370	15
3	{0,6,7,8,9}	300	30
4	{0,6,7,8,9}	370	15

the number of local training epochs was set to $E = 3$, while in the other datasets $E = 5$. Some experiments with different optimizers were used, so for the Adam optimizer, the learning rate was $\eta = 0.001$, and for SGD and FedProx $\eta = 0.0625$. In addition, for FedProx, a value of $\mu = 0.001$ was chosen after some values were tested. In some experiments, different values of local training epochs were used, which will be outlined in the following sections.

For the metrics, the Categorical Cross-Entropy error function 2.2 was used to train and validate the trained model. Balanced accuracy 2.3 was used as the default performance metric of the experiments, but also some other parameters related to the environment, such as the number of message collisions and the number of vehicles training in each round.

Data Augmentation was used in all experiments with the same parameters for all datasets and data distributions. For zooming, horizontal translation, and vertical translation, a maximum value of 10% was allowed, that is, the maximum number of pixels in a translation is 10% of the total size of the image. For rotation, a maximum value of 15 degrees was used.

For comparison methods, FedAvg [48], FedProx [58], Gossip Learning [28, 29], and FedPC [80] were used. For FedAvg and FedProx, the global server model was not used at the end of the experiments for the metrics calculation, but the local vehicle models that were trained and personalized with the local datasets. This allows for a more fair comparison with the methods that consider clusters of users, because it allows the global model to be tuned locally, and it is the model that the vehicle would have locally after exiting the training process.

5.2 Validation Set Results

First, the validation dataset was used to calculate performance metrics and analyze the results of the experiments. It is possible to compare the different methods under different scenarios, validate the hyperparameters chosen, and guarantee that neither overfitting nor underfitting were impacting the methods' performances.

Table 5.5: Fashion MNIST Lenet-5 neural network architecture, with ReLU activation in every convolutional and dense layer, and a total of 107786 parameters (~ 0.4 MB).

Type / Stride / Pad	**Shape**
Input	$28 \times 28 \times 1$
Conv / 1 / same	$5 \times 5 \times 6$
Max-Pool / 1 / no	2×2
Conv / 1 / same	$5 \times 5 \times 16$
Max-Pool / 1 / no	2×2
Flatten	-
Dropout	25%
Dense	120
Dropout	50%
Dense	84
Dropout	50%
Softmax Dense	10

Table 5.6: CIFAR10 Lenet-5 neural network architecture, with ReLU activation in every convolutional and dense layer, and a total of 136886 parameters (~ 0.5 MB).

Type / Stride / Pad	**Shape**
Input	$32 \times 32 \times 1$
Conv / 1 / same	$5 \times 5 \times 6$
Max-Pool / 1 / no	2×2
Conv / 1 / same	$5 \times 5 \times 16$
Max-Pool / 1 / no	2×2
Flatten	-
Dropout	25%
Dense	120
Dropout	50%
Dense	84
Dropout	50%
Softmax Dense	10

Table 5.7: GTSRB MicronNet neural network architecture, with ReLU activation in every convolutional and dense layer, and a total of 625273 parameters ($\sim$ 2.4 MB).

Type / Stride / Pad	**Shape**
Input	$48 \times 48 \times 3$
Conv / 1 / same	$5 \times 5 \times 29$
Max-Pool / 2 / no	3×3
Conv / 1 / same	$3 \times 3 \times 59$
Max-Pool / 2 / no	3×3
Conv / 1 / same	$3 \times 3 \times 74$
Max-Pool / 2 / no	3×3
Flatten	-
Dropout	50%
Dense	300
Dropout	50%
Softmax Dense	43

Table 5.8: Federated MNIST MicronNet neural network architecture, with ReLU activation in every convolutional and dense layer, and a total of 163342 parameters ($\sim$ 0.6 MB).

Type / Stride / Pad	**Shape**
Input	$28 \times 28 \times 1$
Conv / 1 / same	$5 \times 5 \times 29$
Max-Pool / 2 / no	3×3
Conv / 1 / same	$3 \times 3 \times 59$
Max-Pool / 2 / no	3×3
Conv / 1 / same	$3 \times 3 \times 74$
Max-Pool / 2 / no	3×3
Flatten	-
Dropout	50%
Dense	300
Dropout	50%
Softmax Dense	62

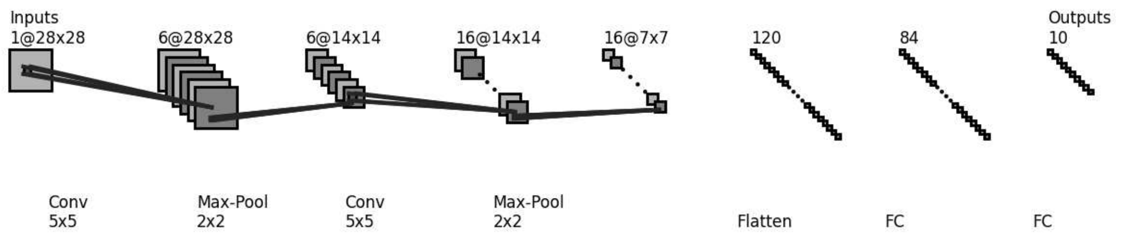


Figure 5.8: Lenet-5 architecture diagram for Fashion MNIST, but also used for CIFAR10 with a different input size.

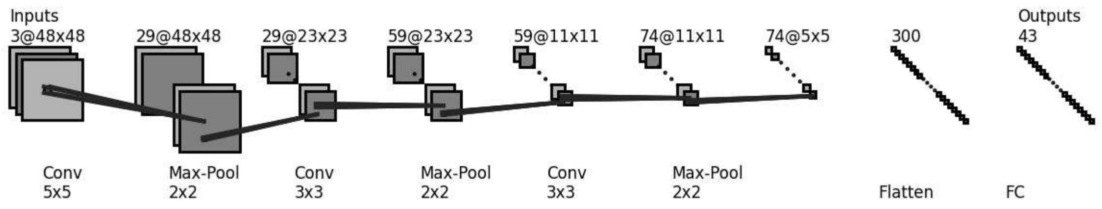


Figure 5.9: MicronNet architecture diagram for GTSRB, but also used for Federated MNIST with a different input and output sizes.

Table 5.9: Results of the experiments of FMNIST dataset using validation data, 8 RSUs scenario, Manhattan Mobility, and 0.1 acceptance threshold.

Methods	Balanced Accuracy / Loss		
	FMNIST 4 Rotations	FMNIST Dirichlet	FMNIST WSCC
Personalized FedAvg	75.21% / 0.5153	72.63% / **0.3124**	84.18% / 0.2996
WSVC	77.49% / 0.4543	72.03% / 0.3282	84.29% / **0.2718**
Gossip Learning	78.62% / 0.5347	70.79% / 0.3813	85.90% / 0.3595
Decentralized WSCC	84.09% / 0.4528	71.34% / 0.3672	87.78% / 0.3244
HVCFL	**84.31%** / **0.4201**	**74.83%** / 0.3238	**87.83%** / 0.3159

Table 5.10: Results of the experiments of CIFAR10 dataset using validation data, 8 RSUs scenario, Manhattan Mobility, and 0.1 acceptance threshold.

Methods	Balanced Accuracy / Loss		
	CIFAR10 2 Rotations	CIFAR10 Dirichlet	CIFAR10 WSCC
Personalized FedAvg	51.98% / 1.2554	44.67% / 0.8712	56.41% / 0.9658
WSVC	55.23% / 1.1715	45.38% / 0.8913	65.74% / **0.7834**
Gossip Learning	56.34% / 1.2342	**50.11%** / 0.8698	59.20% / 0.9650
Decentralized WSCC	55.12% / 1.2767	39.92% / 1.0166	60.68% / 0.9608
HVCFL	**59.63%** / **1.1480**	48.34% / **0.8694**	**67.69%** / 0.8104

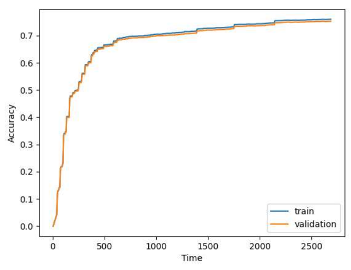
5.2.1 Initial Validation

The first experiments were performed with Personalized FedAvg [48], WSVC, Gossip Learning [28, 29], Decentralized WSCC, and HVCFL. All datasets were used with different data distributions, in the 8 RSUs scenario, Manhattan Mobility, Adam optimizer, and the acceptance threshold $\delta = 0.1$.

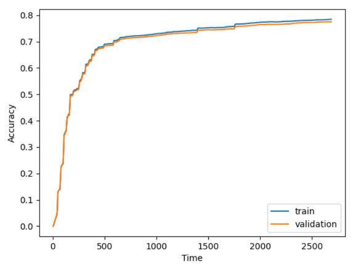
Table 5.9 presents the results of the balanced accuracy and Cross-Entropy loss for the Fashion MNIST dataset under the following data distributions: 4 rotations, Dirichlet and the WSCC paper [68]. Fashion MNIST showed faster convergence compared to other datasets, so the total simulated time was 2700 seconds.

From the results, it can be seen that HVCFL outperformed the other methods for all distributions. Looking at the 4 rotations and WSCC distributions that indeed create groups of users, we can see that both Decentralized WSCC and HVCFL have a good balanced accuracy, with HVCFL achieving lower loss values. We also see that Gossip Learning achieved better results compared to centralized solutions in this case. However, by checking the Dirichlet distribution, it is possible to see that centralized methods outperformed purely decentralized ones.

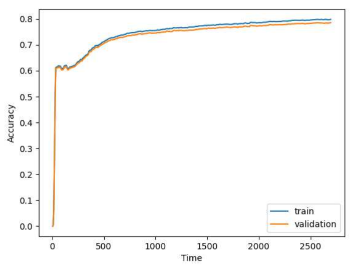
If the analysis is expanded to the accuracy through the time charts in Figure 5.10, it is possible to see that the methods that have decentralized agents converge faster, but display a little more overfitting compared to the pure centralized ones. Decentralized WSCC clearly has a larger gap between the training and validation curves, and this could impact the performance in other datasets as will be shown next.



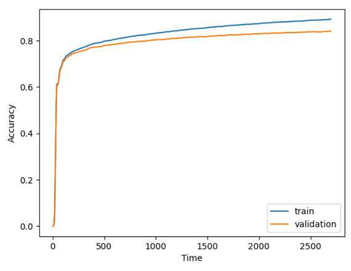
(a) FedAvg



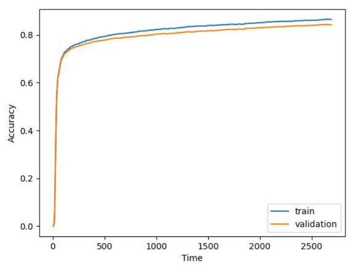
(b) WSVC



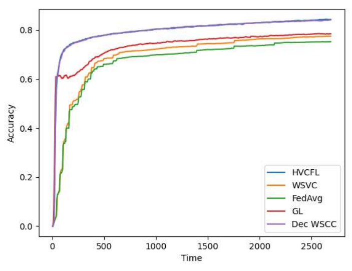
(c) Gossip Learning



(d) Decentralized WSCC



(e) HVCFL



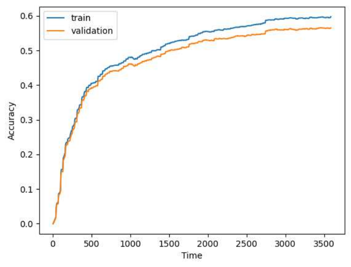
(f) Validation curves

Figure 5.10: Experiment results of balanced accuracy for different methods using the Fashion MNIST dataset with the 4 Rotations data distribution, 8 RSUs scenario and Manhattan Mobility.

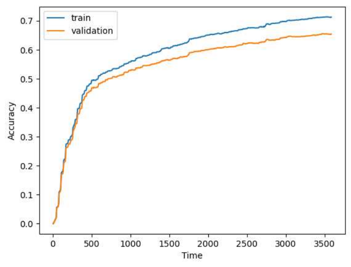
Table 5.10 displays the results for similar experiments carried out on CIFAR10, with 2 rotations instead of 4, and 3600 seconds simulation time. Again, it is possible to see that HVCFL had the best overall performance, but this time on the Dirichlet distribution, Gossip Learning is able to outperform HVCFL. Gossip Learning is capable of carrying out more training rounds in each vehicle, because the vehicle starts training as soon as it receives a model from a neighbor, instead of waiting until the end of the gossip/aggregation round, like all the other methods. However, it is also possible to see that this does not happen with Fashion MNIST, which presents a much faster convergence compared to CIFAR10, so each method has advantages and disadvantages depending either on the dataset and data distribution.

Upon checking the charts for the CIFAR10 WSCC distribution in Figure 5.11, it is possible to see in this case that even the centralized solutions present little overfitting, while the decentralized/hybrid methods are more affected by this behavior. It is also important to note that HVCFL is able to combine the strengths of both centralized and decentralized approaches, having a better and consistent end result in general for this dataset. Here, it is possible to see that HVCFL and WSVC have a faster convergence rate compared to the other methods.

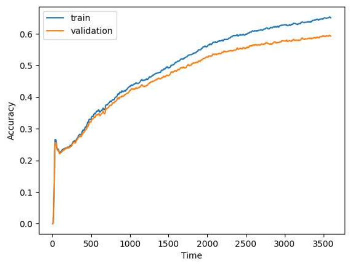
For the last 2 datasets, the results are shown in Table 5.11, using 3600 seconds for Federated MNIST and 2700 seconds for GTSRB. It is possible to see that the methods that have a decentralized component performed better for Federated MNIST, while the methods that have a centralized component were able to achieve lower loss values. For GTSRB, the result is almost the same. However, the Decentralized WSCC method presented a much higher loss value.



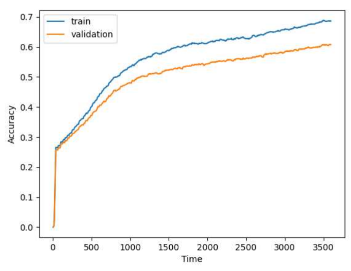
(a) FedAvg



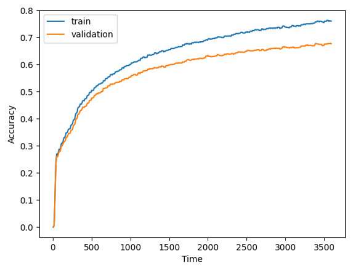
(b) WSVC



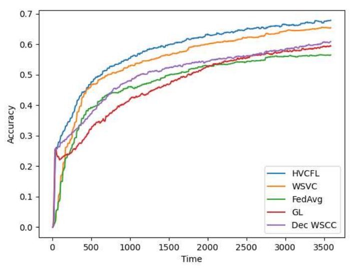
(c) Gossip Learning



(d) Decentralized WSCC



(e) HVCFL

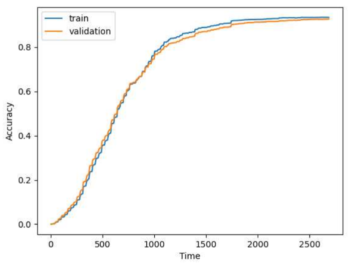


(f) Validation curves

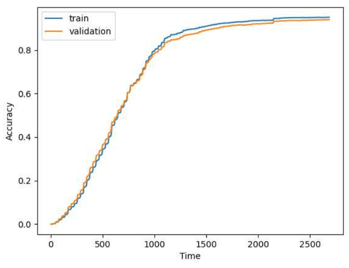
Figure 5.11: Experiment results of balanced accuracy for different methods using the CIFAR10 dataset with the WSCC data distribution, 8 RSUs scenario and Manhattan Mobility.

Table 5.11: Results of the experiments of Federated MNIST and GTSRB datasets using validation data, 8 RSUs scenario, Manhattan Mobility, and 0.1 acceptance threshold.

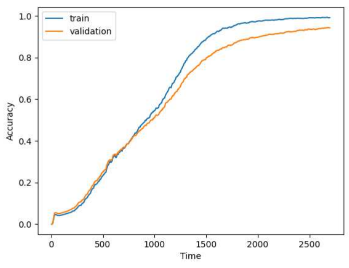
Methods	Balanced Accuracy / Loss	
	Federated MNIST	GTSRB Dirichlet
Personalized FedAvg	75.66% / 0.3835	92.70% / **0.0379**
WSVC	76.10% / 0.3916	93.85% / 0.0403
Gossip Learning	**80.73%** / 0.4021	94.16% / 0.1369
Decentralized WSCC	80.14% / 0.4252	90.33% / 0.2312
HVCFL	80.44% / **0.3829**	**96.97%** / 0.0667



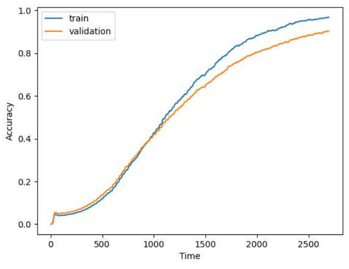
(a) FedAvg



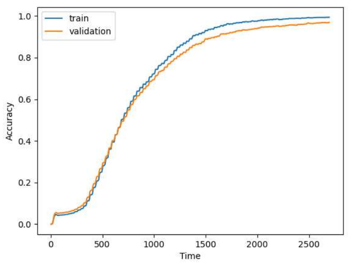
(b) WSVC



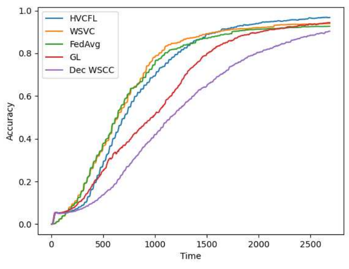
(c) Gossip Learning



(d) Decentralized WSCC



(e) HVCFL



(f) Validation curves

Figure 5.12: Experiment results of balanced accuracy for different methods using the GTSRB dataset with the Dirichlet data distribution, 8 RSUs scenario and Manhattan Mobility.

Looking at the charts in Figure 5.12, it is possible to check that Decentralized WSCC and Gossip Learning had much higher overfitting, while FedAvg and WSVC were able to keep it very controlled. Upon checking the HVCFL curves, the separation between training and validation is larger than that of the centralized methods, but it is controlled not to have the same issue as the decentralized solutions. During the analysis of the test results, it will be possible to see how the overfitting affects the model generalization. Here, it is possible to note that HVCFL has a slightly slower convergence rate compared to FedAvg and WSVC.

Lastly, for this initial validation, there are some metrics related to the environment that are important to analyze. Table 5.12 displays the total messages sent by each experiment, and the percentage of collisions that occur for the different methods and datasets. The other Table 5.13 displays the number of messages and the time to achieve a given accuracy for some data distributions. These tables are organized with increasing model size, as can be seen in the last section, and each message/model sent is considered one transmission.

The first consequence that can be analyzed from the tables is the fact that the larger the model, the larger the percentage of collisions. It is also possible to see that the decentralized methods have a much lower collision rate than the centralized ones. For the first models that are not large, this behavior does not affect much the messages; however, for the GTSRB model the collisions are very constant. Personalized FedAvg also had an issue with this dataset, because since the global model is sent to the clients at the same instant, even with different training times, most of the clients tried to send the trained models back to the server almost at the same time. There is an area for improvement here, especially for centralized methods, but that could be applied to all of them, like

Table 5.12: Results of average messages per round, total messages and collisions percentage in the experiments using 8 RSUs scenario and Manhattan Mobility.

Methods	Messages Per Round / Total Messages / Percentage of Collisions			
	FMNIST	CIFAR10	Federated MNIST	GTSRB
Per. FedAvg	35/3174/17%	35/4244/18%	35/4233/21%	34/3094/76%
WSVC	55/5132/13%	54/6818/16%	66/7976/19%	67/5986/41%
Gossip Learning	99/8900/2%	99/11900/3%	99/11900/4%	99/8900/21%
Dec. WSCC	99/8948/2%	100/11949/4%	100/11951/4%	99/8946/19%
HVCFL	138/12404/9%	145/15831/12%	148/17744/14%	141/12646/38%

Table 5.13: Results of number of messages and the time to achieve a given accuracy in the experiments using 8 RSUs scenario and Manhattan Mobility (one execution only).

Methods	Number Of Messages / Time			
	FMNIST WSCC (84%)	CIFAR10 2 Rot. (51%)	Fed. MNIST (75%)	GTSRB Dir. (90%)
Per. FedAvg	2989 / 2520s	4099 / 3465s	**3909** / 3315s	**2080** / 1790s
WSVC	4322 / 2255s	**3617** / 1970s	6480 / 2935s	3763 / 1675s
Gossip Learning	5791 / 1765s	7074 / 2150s	5991 / 1825s	7400 / 2250s
Dec. WSCC	**2342** / 720s	8669 / 2615s	6609 / 2000s	- / -
HVCFL	2990 / **625s**	6738 / **1435s**	7115 / **1470s**	5314 / **1135s**

using a message-exchange protocol or breaking the model into smaller parts [29].

Additionally, from Table 5.13 it is possible to see that in general centralized methods use fewer messages to converge to some accuracies, but they take much more time than HVCFL and even decentralized methods in some cases. HVCFL has the fastest convergences, but for Fashion MNIST WSCC, Decentralized WSCC uses fewer messages, while in CIFAR10 2 Rotations, WSVC uses fewer messages. For Federated MNIST and GTSRB Dirichlet, Personalized FedAvg uses less messages, but it usually takes more time to achieve the given accuracy. Therefore, it is possible to see that some methods have advantages and disadvantages considering the number of messages used to achieve a goal, but not always more messages mean faster convergence, as can be seen in the decentralized methods for CIFAR10 and GTSRB.

The final metrics are related to the number of rounds each vehicle trained on each method and the number of models received on Decentralized WSCC and HVCFL. For the Fashion MNIST experiments, vehicles trained 10∼60 rounds in FedAvg and WSVC, 90∼140 in Gossip Learning, and 60∼90 in Decentralized WSCC and HVCFL. This explains why in some experiments Gossip Learning is able to have a good performance even without specific non-IID special treatment. Some vehicles trained more than twice the number of rounds, but it also led to increased overfitting, so there are advantages and disadvantages.

In Decentralized WSCC and HVCFL, it could be seen that the vehicles received between 1∼20 models in each round. For some crowded areas, the information could be

Table 5.14: Results of the experiments using validation data, SGD optimizer, 8 RSUs scenario, Manhattan Mobility, and 0.1 acceptance threshold.

Methods	Balanced Accuracy / Loss		
	FMNIST 4 Rotations	CIFAR10 WSCC	GTSRB Dirichlet
Personalized FedAvg	74.86% / 0.5164	54.66% / 1.0419	90.73% / **0.0983**
Personalized FedProx	75.24% / 0.5144	55.13% / 1.0370	89.64% / 0.1299
WSVC	76.84% / 0.4575	59.98% / **0.9067**	**92.08%** / 0.1089
Gossip Learning	80.54% / 0.4916	61.00% / 1.0025	85.35% / 0.3763
FedPC	77.09% / 0.5624	- / -	88.21% / 0.3546
Decentralized WSCC	83.33% / 0.4548	58.77% / 1.0597	62.34% / 0.9247
HVCFL	**83.96%** / **0.4211**	**64.40%** / 0.9206	85.45% / 0.3542

distributed with high quality, while other vehicles received just a few models, turning the aggregation almost into a Gossip Learning scheme.

5.2.2 Optimizers

Some experiments have also been performed using the SGD optimizer, so it was possible to validate the FedProx [58] and FedPC [80] methods. In these experiments, some of the datasets and data distributions were chosen, 5 epochs were used for Fashion MNIST, and 8 epochs for the other datasets. Table 5.14 shows the results for these experiments.

From the table, it can be seen that HVCFL has a good overall performance for Fashion MNIST and CIFAR10, but it performed worse in GTSRB. Looking at Decentralized WSCC, it presented a poor model performance, so in this case HVCFL might have been impacted by the vehicle local client selection strategy. However, this behavior is much less present using the Adam optimizer, leading to better results for the experiments in general, so Adam was used in the rest of the study.

Also, analyzing the other methods, WSVC had a good performance for GTSRB, while FedPC is inconstant. In CIFAR10 the model is overfitting with the same scenario and parameters as the other methods. A possible cause for this issue is that FedPC does not perform local aggregation, it just uses the received model, and also does not have a mechanism of acceptance checking. In this dynamic environment, this implementation led to a more difficult convergence for the specific CIFAR10 dataset.

5.2.3 RSU Coverage

The next set of experiments focused on the different RSU scenarios and how they impact the performance of the models. Table 5.15 displays the results for the validation set of the centralized/hybrid methods, because they are the only ones that use RSUs. Figure 5.13 displays the learning curves for the 6 and 12 RSUs scenarios.

HVCFL presented the best results despite the scenario. However, it is possible to see that it is much less impacted by RSU changes than the other centralized methods. For the 12 RSUs scenario, it is possible to see that the centralized solutions performed

Table 5.15: Results of balanced accuracy of the experiments using validation data, different RSU scenarios, Manhattan Mobility, and 0.1 acceptance threshold.

Methods	6/8/12 RSUs Balanced Accuracy FMNIST WSCC	6/8/12 RSUs Balanced Accuracy GTSRB Dirichlet
Personalized FedAvg	71.43% / 84.18% / 86.54%	76.57% / 92.70% / 96.77%
WSVC	70.97% / 84.29% / 88.33%	78.75% / 93.85% / 96.84%
HVCFL	**88.73%** / **87.83%** / **89.28%**	**97.94%** / **96.97%** / **97.98%**

the same as HVCFL, and since they exchange much less messages and have the message collisions problem described in the last sections, they have room here to surpass HVCFL when there is better coverage.

Other metrics that were collected for these experiments on Fashion MNIST are the number of vehicles that did not participate in the centralized training for FedAvg and WSCC, and also the number of vehicles that participated in the centralized training. 2 vehicles did not participate in the 12 RSUs scenario, 5 in the 8 RSUs, and 19 in the 6 RSUs. This explains the performance drop for the 6 RSUs scenario, since less vehicles shared their data/knowledge with the system.

For the participating vehicles by round, the number was between 16-38 for the 12 RSUs scenario, 11-27 in the 8 RSUs, and 9-23 in the 6 RSUs. This also corroborates with the results analyzed, but shows that for the 12 RSUs scenario, sometimes a significant part of the system was able to reach the centralized server, almost 40% of the vehicles.

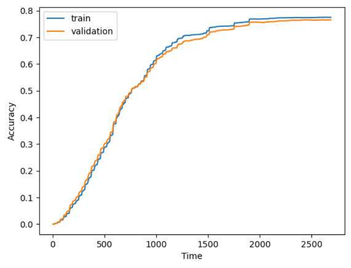
5.2.4 Acceptance Threshold

Next, different experiments with the acceptance thresholds for HVCFL were performed, and the results are summarized in Table 5.16. The tested values were 10%, 5%, 0%, and without a threshold, that is, always accepting the received model.

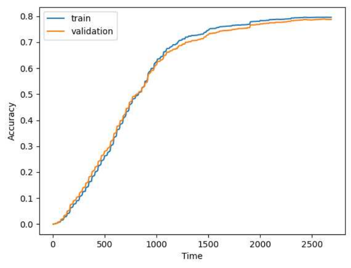
For Fashion MNIST, since it is a dataset that is less prone to overfitting and converges faster, smaller thresholds led to higher balanced accuracy and lower loss values, but all the experiments led to similar end results. For GTSRB, the smaller the threshold or without it, the worse the results. The threshold 10% had the best metrics for this specific dataset, so it was used in the rest of the studies. However, it is important to note that this hyperparameter depends on the dataset being used, so more experiments could be performed to analyze its full impact.

5.2.5 Similarity Metrics

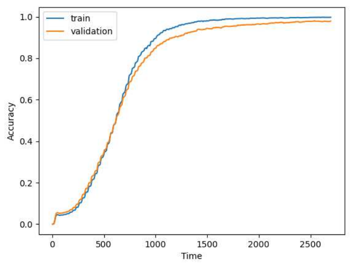
The last set of experiments explored the different similarity metrics and some client selection techniques. The similarity metrics tested are the Cosine Similarity (the default implementation of HVCFL that was used in all the latest experiments), balanced accuracy, and Centered Kernel Alignment, which was applied only in the last layer of the network, all the dense layers, or the full model. A client selection strategy that does



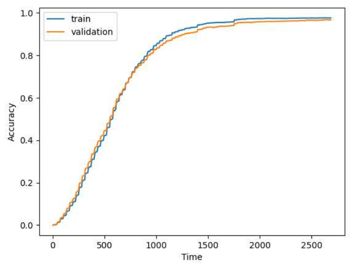
(a) FedAvg 6 RSUs



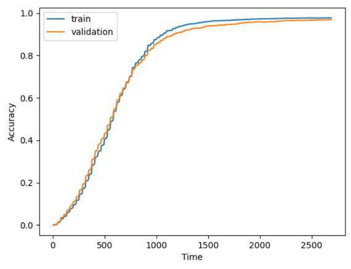
(b) WSVC 6 RSUs



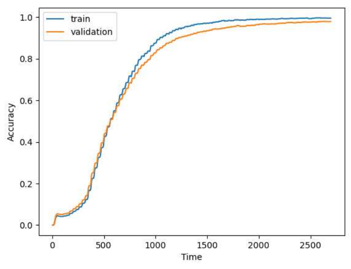
(c) HVCFL 6 RSUs



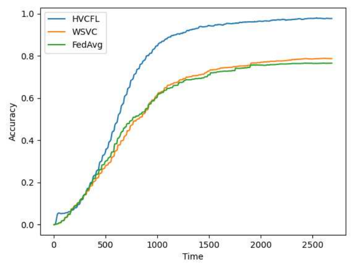
(d) FedAvg 12 RSUs



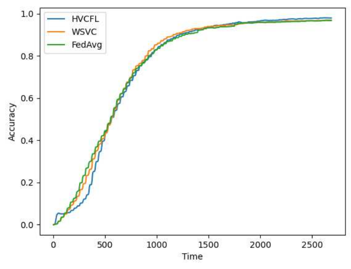
(e) WSVC 12 RSUs



(f) HVCFL 12 RSUs



(g) 6 RSUs validation curves



(h) 12 RSUs validation curves

Figure 5.13: Experiment results of balanced accuracy for different methods using the GTSRB dataset with the Dirichlet data distribution, 6/12 RSUs scenarios and Manhattan Mobility.

Table 5.16: Results of the experiments using validation data, 8 RSUs scenario, Manhattan Mobility, and different acceptance thresholds.

Methods	Balanced Accuracy / Loss	
	FMNIST WSCC	GTSRB Dirichlet
HVCFL 0.1 Threshold	87.83% / 0.3159	**96.97%** / **0.0667**
HVCFL 0.05 Threshold	88.39% / **0.2769**	96.33% / 0.0714
HVCFL 0.0 Threshold	**88.82%** / 0.2850	96.04% / 0.0944
HVCFL No Threshold	88.19% / 0.3031	93.09% / 0.1463

Table 5.17: Results of the experiments using validation data, different similarity metrics, 8 RSUs scenario, Manhattan Mobility, and 0.1 acceptance threshold.

Methods	Balanced Accuracy / Loss	
	FMNIST WSCC	GTSRB Dirichlet
HVCFL Cossim	87.83% / 0.3159	**96.97%** / **0.0667**
HVCFL Balanced Accuracy	88.28% / 0.3019	91.75% / 0.1824
HVCFL CKA Last	88.07% / 0.3098	88.44% / 0.2435
HVCFL CKA Dense	**88.83%** / **0.2931**	93.13% / 0.1527
HVCFL CKA Full	88.63% / 0.2946	94.48% / 0.1263
HVCFL 25 Percent	88.22% / 0.3046	95.96% / 0.0882

not use Affinity Propagation to create the local clusters was also used, it uses Cosine Similarity and gets the 25% more similar models in each round, just for testing purposes.

From the results in Table 5.17, it can be seen that HVCFL Cossim presents the best results in the GTSRB dataset while it has the lowest balanced accuracy in Fashion MNIST. CKA has the best results in Fashion MNIST using either all the dense layers or the full model, but is not as good as Cossim in GTSRB. Cossim was also able to achieve good results in Fashion MNIST, so it was chosen for the rest of the experiments, but it is important to note that the use of 25% also led to great results and should be further explored. In this case, it can also be concluded that the dataset/model may impact the model training, and with the variability of the results obtained, more tests need to be performed with more experiments and more similarity metrics to see the advantages and disadvantages of each one.

An important analysis that can also be done here and with WSVC is how well the vehicles and the server are creating the groups of users. By checking the clusters in the Fashion MNIST distribution WSCC in the final period of the experiments, WSVC is able to create better clusters than HVCFL in general. By comparing only server with server, the decentralized factor makes HVCFL create incorrect clusters more frequently, but even WSVC in this dynamic environment has incorrectly assigned vehicles frequently and there is an area for improvement here.

By analyzing only the participating nodes in each vehicle, HVCFL has a good performance with client selection in general, better than inside the server: it is able to correctly filter out models that belong to different clusters. However, sometimes there are some models that belong to the same clusters that are also filtered out, which represents an area for improvement in the methods.

By comparing HVCFL Cossim and CKA, it can be seen that CKA usually includes the correct models that Cossim usually filters out. However, it also includes some undesired models much more frequently than the Cossim implementation, which may explain why in GTSRB it performed worse than Cossim.

Table 5.18: Results of the experiments of FMNIST dataset using test data, 8 RSUs scenario, Manhattan Mobility, and 0.1 acceptance threshold.

Methods	Balanced Accuracy		
	FMNIST 4 Rotations	FMNIST Dirichlet	FMNIST WSCC
Personalized FedAvg	75.39%	73.09%	84.24%
WSVC	77.75%	72.40%	84.78%
Gossip Learning	77.81%	70.31%	85.05%
Decentralized WSCC	82.77%	70.53%	86.91%
HVCFL	**83.47%**	**74.83%**	**87.06%**

Table 5.19: Results of the experiments of CIFAR10 dataset using test data, 8 RSUs scenario, Manhattan Mobility, and 0.1 acceptance threshold.

Methods	Balanced Accuracy		
	CIFAR10 2 Rotations	CIFAR10 Dirichlet	CIFAR10 WSCC
Personalized FedAvg	50.63%	42.86%	56.48%
WSVC	53.49%	43.04%	64.21%
Gossip Learning	51.94%	43.63%	57.86%
Decentralized WSCC	50.47%	36.47%	59.10%
HVCFL	**56.13%**	**43.88%**	**65.94%**

5.3 Test Set Results of Centralized, Decentralized and Hybrid Methods

To conclude this first set of experiments and fully validate the methods, the balanced accuracy was also calculated in the test set. The experiments were carried out 3 times to ensure the quality of the results.

Table 5.18 displays the results for Fashion MNIST. Since this dataset had a fast convergence and did not present significant overfitting, the results for the test set followed the behavior of the validation set. HVCFL achieved the best results in all data distributions.

Table 5.19 displays the results for CIFAR10. Here, the overfitting is much more present and the result impacts could be seen. HVCFL had the best performance in all distributions, but in Dirichlet, it is Gossip Learning which has the best results in the validation set. This distribution presented a really slow convergence in general, but the bigger overfitting in Gossip Learning due to the lack of non-IID special treatment led to poorer results on the test set. It is also important to note that WSVC is the second-best method here for the other distributions, performing great on this dataset.

Finally, Table 5.20 shows the results for Federated MNIST and GTSRB. For Federated MNIST it is possible to see the same behavior as in CIFAR10 Dirichlet for Gossip Learning. It had the best validation results, but HVCFL surpassed it in the test set.

For the GTSRB, HVCFL showed the best performance, but is closely followed by WSVC. In the validation set, the gap between the results of these 2 methods was larger,

Table 5.20: Results of the experiments of Federated MNIST and GTSRB datasets using test data, 8 RSUs scenario, Manhattan Mobility, and 0.1 acceptance threshold.

Methods	Balanced Accuracy	
	Federated MNIST	GTSRB Dirichlet
Personalized FedAvg	69.56%	90.17%
WSVC	69.52%	91.24%
Gossip Learning	70.37%	84.38%
Decentralized WSCC	68.32%	77.52%
HVCFL	**72.12%**	**91.51%**

Table 5.21: Results of the experiments using validation data, 8 RSUs scenario, vehicle traces, and 0.1 acceptance threshold.

Methods	Balanced Accuracy / Loss	
	FMNIST WSCC	GTSRB Dirichlet
Personalized FedAvg	81.99% / 0.4571	**57.92%** / 0.9296
WSVC	82.12% / **0.4216**	57.29% / **0.9255**
HVCFL	**82.40%** / 0.4357	49.39% / 1.1603

so it can be seen that despite being small, the little overfitting here also had a little impact on HVCFL.

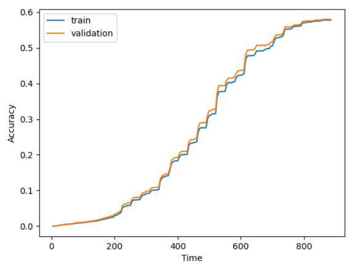
In conclusion, after the analysis of all results from the test set, the HVCFL performed very well in most scenarios. However, when the RSU coverage is better, centralized methods tend to generate better models in terms of overfitting and using fewer messages exchanged. The message collision issue of the last sections could also boost the performance of the centralized methods a little, if messaging exchanging is handled in a better way, but HVCFL-like strategies could still be applied to compensate for the lack of RSU coverage.

5.4 Long and Short Vehicle Trips

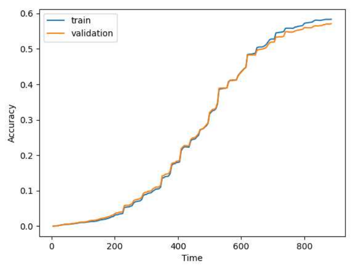
This section displays some initial experiments using vehicle traces instead of the Manhattan Mobility model. Table 5.21 has the results of the validation set and Figure 5.14 displays the learning curves for GTSRB. The experiments carried out here were performed only once, and for 900 seconds.

For Fashion MNIST, HVCFL is able to surpass the other methods but all the balanced accuracies are similar, while in GTSRB it did not perform well. Since these experiments are carried out in a shorter period of time, it can be seen that the convergence speeds of centralized methods are faster than those of HVCFL, as can also be seen on the charts.

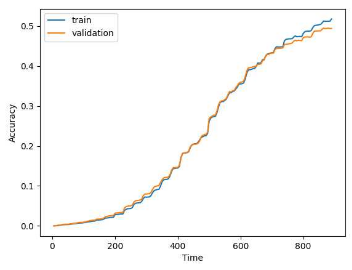
Another important aspect of the scenario used is that the RSUs were positioned in a way that high-traffic streets have good coverage, so the vehicle traces passed through these roads. The consequence was that only 1 vehicle did not participate in the cen-



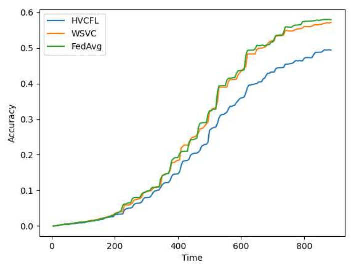
(a) FedAvg



(b) WSVC



(c) HVCFL



(d) HVCFL

Figure 5.14: Experiment results of balanced accuracy for different methods using the GTSRB dataset with the Dirichlet data distribution, 8 RSUs scenario and vehicle traces.

tralized aggregation round of the centralized methods, while the other experiments with the Manhattan model had more missing vehicles. This opens another path for studies with long and short trips, mixing them, and trying to include the best of centralized and decentralized solutions for these scenarios, such as using transfer learning [84].

5.5 Experiments Remarks

From the experiments performed, it was possible to see how WSVC and HVCFL performed compared to other methods, including their advantages and disadvantages. Using datasets and data distributions that did not present overfitting, such as Fashion MNIST, decentralized methods presented faster convergence. When data led to more overfitting, such as in CIFAR10 and GTSRB, centralized methods were able to achieve faster convergence with fewer messages exchanged in the environment. Another disadvantage of centralized approaches occurs when the environment does not have good RSU coverage, leading to slower convergence.

The hybrid HVCFL approach was able to combine the strengths of both strategies, while being less impacted by the disadvantages of them, and outperformed other methods in almost all of the scenarios. In addition, the tests performed with the acceptance hyperparameter and different similarity metrics for aggregation revealed that different data distributions behave differently, with Cosine Similarity a good overall metric. The main disadvantages of HVCFL occur when the RSU coverage is good, because the centralized methods achieve similar results with fewer messages exchanged, and when the vehicles have short trips, leading to slower convergence in the scenarios tested.

Chapter 6

Conclusions

The final chapter concludes the dissertation and is divided into the final remarks and future work.

6.1 Final Remarks

This dissertation aimed to study and develop federated learning methods for Vehicle Ad Hoc Networks (VANETs). Two methods were proposed: Weight-Similarity Vehicle Clustering (WSVC), based on the Weight-Similarity Client Clustering (WSCC) centralized method [68], and Hybrid Vehicle Federated Learning (HVCFL), which applies WSVC but also includes a decentralized component in the system architecture.

Through extensive experiments that simulated as best as possible real-world scenarios, it was possible to validate the new proposals against other solutions such as FedAvg [48], FedProx [58], Gossip Learning [28, 29], and FedPC [80]. HVCFL was able to surpass the other solutions in most of the scenarios tested, achieving a better balanced accuracy in the test set. However, in some specific data distributions and with short vehicle trips, there is still room for improvement.

In the beginning of the dissertation, some research questions were raised to guide the study. In conclusion, it is possible to answer each of them after analyzing the results.

Q1. How does Federated Learning perform in a dynamic environment such as vehicle networks (VANETs)?

Through the different experiments performed, it was possible to see that Federated Learning can be implemented in VANETs, but different non-IID data distributions impact the convergence of the models and cause more overfitting in local training for some methods, especially decentralized ones.

Q2. How does Centralized Federated Learning perform compared to Decentralized approaches in VANETs?

Centralized Federated Learning usually has less overfitting than decentralized approaches, but in some scenarios with less RSU coverage, it is not able to reach all the vehicles of the network, which is a disadvantage.

Q3. Is it possible to improve the performance of a trained model with a hybrid approach?

Hybrid solutions were able to combine the strengths of both approaches in most of the datasets and data distributions tested. However, they still have room for improvement in some scenarios, such as with short vehicle trips.

Q4. Do different non-IID data distributions perform differently in different approaches?

It was possible to check that while centralized solutions performed better in some distributions, such as GTSRB Dirichlet, decentralized solutions achieved better results in others, such as Fashion MNIST. Datasets that converge faster tend to have less overfitting and help decentralized methods, while slow convergence tends to help centralized solution.

Q5. How does environment configuration (especially RSU positioning) affect the convergence of machine learning models for different methods?

HVCFL was able to have a similar performance with 6, 8 or 12 RSUs in the environment, which surpassed centralized solutions when coverage is not good. Centralized methods were affected by the lack of coverage and in the worst case, almost 20% of the vehicles did not participate in the training process. However, when coverage is good, centralized methods have performance similar to that of HVCFL.

Q6. How do long vehicle trips compare to short ones in model convergence?

HVCFL presented a slower convergence compared to WSVC and FedAvg, especially due to the positioning of the RSUs in high-traffic streets. Therefore, for vehicles with short trips, centralized methods are achieving better results.

6.2 Future Work

In this final section, some future research avenues are introduced to further explore the different scenarios and methods that were tested. Throughout the dissertation, some alternatives have been proposed, such as:

- Implement a message-exchange protocol to decrease the negative effect of message collisions in centralized methods. In the experiments, it was possible to check that more collisions occur in these solutions, because all the models are sent back to the server after training almost at the same time.

- Another option for message collisions is to break the model into smaller parts and exchange only parts of it [29]. Since the models are becoming larger, especially with the new Large Language Models [72], smaller messages exchanged are important for the feasibility of a VANET implementation.

- During the experiments, it was possible to see that Gossip Learning sometimes achieved good results in some datasets, even without a non-IID data strategy. This happened because more training rounds were performed despite having the overfitting issue, but a way to decrease the round time in HVCFL could be investigated, such as [49].

- Centralized solutions tend to have better convergence when there is good RSU coverage, but hybrid methods reach vehicles that would not have received the server's global model. Some strategies could be studied to apply a less aggressive decentralized approach in the hybrid solution, such as forwarding global models or exploring local client selection for a smoother training convergence.

- Explore further the effects of acceptance local checking and similarity metrics in hybrid environments. These parameters and metrics have different impacts depending on the dataset and data distributions being used, so there is room for a more thorough investigation here.

- Expand experiments and strategies with short and long vehicle trips. Since HVCFL had a slow convergence, it did not perform well with short vehicle trips, so strategies like transfer learning [84] could be tested.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Omid Aramoon, Pin-Yu Chen, Gang Qu, and Yuan Tian. Meta federated learning. *ArXiv*, abs/2102.05561, 2021.

[3] Szabolcs-Márton Bagoly and Radu Gabriel Dănescu. Round based extension algorithm for gossip learning. In *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 251–257, 2020.

[4] F. Bai, Narayanan Sadagopan, and A. Helmy. Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, volume 2, pages 825–835 vol.2, 2003.

[5] Luca Barbieri, Stefano Savazzi, and Monica Nicoli. Decentralized federated learning for road user classification in enhanced v2x networks. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2021.

[6] Yacine Belal, Aurélien Bellet, Sonia Ben Mokhtar, and Vlad Nitu. Pepper: Empowering user-centric recommender systems over gossip learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–27, September 2022.

[7] Abdelwahab Boualouache and Thomas Engel. Federated learning-based scheme for detecting passive mobile attackers in 5g vehicular edge computing. *Ann. des Télécommunications*, 77(3-4):201–220, 2022.

[8] Shafinaz Buruhanudeen, Mohamed Othman, Mazliza Othman, and Borhanuddin Mohd Ali. Mobility models, broadcasting methods and factors contributing towards the efficiency of the manet routing protocols: Overview. In *2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications*, pages 226–230, 2007.

[9] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings, 2019.

[10] Andrea Caragliu, Chiara Del Bo, and Peter Nijkamp. Smart cities in europe. *VU University Amsterdam, Faculty of Economics, Business Administration and Econometrics, Serie Research Memoranda*, 18, 01 2009.

[11] Yiming Chen, Kun Yuan, Yingya Zhang, Pan Pan, Yinghui Xu, and Wotao Yin. Accelerating gossip SGD with periodic global averaging. *CoRR*, abs/2105.09080, 2021.

[12] François Chollet et al. Keras. `https://keras.io`, 2015.

[13] Lara Codecá, Raphaël Frank, Sébastien Faye, and Thomas Engel. Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63, 2017.

[14] Giuseppe Di Giacomo, Jérôme Härri, and Carla Fabiana Chiasserini. Edge-assisted gossiping learning: Leveraging v2v communications between connected vehicles. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3920–3927, 2022.

[15] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. Gossip learning of personalized models for vehicle trajectory prediction. In *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–7, 2021.

[16] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. Vehicle position nowcasting with gossip learning. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 728–733, 2022.

[17] Muhammad Ehtisham, Mahmood ul Hassan, Amin A. Al-Awady, Abid Ali, Muhammad Junaid, Jahangir Khan, Yahya Ali Abdelrahman Ali, and Muhammad Akram. Internet of vehicles (iov)-based task scheduling approach using fuzzy logic technique in fog computing enables vehicular ad hoc network (vanet). *Sensors*, 24(3), 2024.

[18] Ahmet M. Elbir, Burak Soner, Sinem Çöleri, Deniz Gündüz, and Mehdi Bennis. Federated learning in vehicular networks. In *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pages 72–77, 2022.

[19] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[20] Benyamin Ghojogh and Mark Crowley. The theory behind overfitting, cross validation, regularization, bagging, and boosting: Tutorial. *ArXiv*, abs/1905.12787, 2019.

[21] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *IEEE Transactions on Information Theory*, 68(12):8076–8091, 2022.

[22] Lodovico Giaretta and Šarūnas Girdzijauskas. Gossip learning: Off the beaten path. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1117–1124, 2019.

[23] Usman Manzo Gidado, Haruna Chiroma, Nahla Aljojo, Saidu Abubakar, Segun I. Popoola, and Mohammed Ali Al-Garadi. A survey on deep learning for steering angle prediction in autonomous vehicles. *IEEE Access*, 8:163797–163817, 2020.

[24] Alessandro Giuseppi, Sabato Manfredi, and Antonio Pietrabissa. A weighted average consensus approach for decentralized federated learning. *Machine Intelligence Research*, 19(4):319–330, 2022.

[25] Yuanxiong Guo, Ying Sun, Rui Hu, and Yanmin Gong. Hybrid local SGD for federated learning with heterogeneous communications. In *International Conference on Learning Representations*, 2022.

[26] Mastura Hanafiah, Mohd Adnan, Shuzlina Rahman, Sofianita Mutalib, Ariff Malik, and Mohd Razif Shamsuddin. Flower recognition using deep convolutional neural networks. *IOP Conference Series: Earth and Environmental Science*, 1019:012021, 04 2022.

[27] Bastian Havers, Marina Papatriantafilou, Ashok Koppisetty, and Vincenzo Gulisano. Proposing a framework for evaluating learning strategies in vehicular cpss. In *Proceedings of the 23rd International Middleware Conference Industrial Track*, Middleware Industrial Track '22, page 22–28, New York, NY, USA, 2022. Association for Computing Machinery.

[28] István Hegedűs, Gábor Danner, and Márk Jelasity. Gossip learning as a decentralized alternative to federated learning. In José Pereira and Laura Ricci, editors, *Distributed Applications and Interoperable Systems*, pages 74–90, Cham, 2019. Springer International Publishing.

[29] István Hegedűs, Gábor Danner, and Márk Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124, 2021.

[30] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability between c++11 and python, 2017. https://github.com/pybind/pybind11.

[31] Jingyan Jiang, Liang Hu, Chenghao Hu, Jiate Liu, and Zhi Wang. Bacombo—bandwidth-aware decentralized federated learning. *Electronics*, 9(3), 2020.

[32] June-Pyo Jung, Young-Bae Ko, and Sung-Hwa Lim. Resource efficient cluster-based federated learning for d2d communications. In *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, pages 1–5, 2022.

[33] Georgios Karagiannis, Onur Altintas, Eylem Ekici, Geert Heijenk, Boangoat Jarupan, Kenneth Lin, and Timothy Weil. Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *IEEE Communications Surveys and Tutorials*, 13(4):584–616, 2011.

[34] J.D. Kelleher, B.M. Namee, and A. D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press. MIT Press, 2015.

[35] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. *ArXiv*, abs/1905.00414, 2019.

[36] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5(4):1, 2010.

[37] Fan Li and Yu Wang. Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular Technology Magazine*, 2(2):12–22, 2007.

[38] Zexi Li, Jiaxun Lu, Shuang Luo, Didi Zhu, Yunfeng Shao, Yinchuan Li, Zhimeng Zhang, Yongheng Wang, and Chao Wu. Towards effective clustered federated learning: A peer-to-peer framework with adaptive neighbor matching. *IEEE Transactions on Big Data*, pages 1–16, 2022.

[39] Su Liu, Jiong Yu, Xiaoheng Deng, and Shaohua Wan. Fedcpf: An efficient-communication federated learning approach for vehicular edge computing in 6g communication networks. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1616–1629, 2022.

[40] Wellington Lobato, Joahannes B. D. Da Costa, Allan M. de Souza, Denis Rosário, Christoph Sommer, and Leandro A. Villas. Flexe: Investigating federated learning in connected autonomous vehicle simulations. In *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, pages 1–5, 2022.

[41] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. Multi-center federated learning: clients clustering for better personalization. *World Wide Web*, 26:1–20, 06 2022.

[42] 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.

[43] Hao Lu, Adam Thelen, Olga Fink, Chao Hu, and Simon Laflamme. Federated learning with uncertainty-based client clustering for fleet-wide fault diagnosis. *Mechanical Systems and Signal Processing*, 210:111068, 2024.

[44] Lin Lu, Yao Lin, Yuan Wen, Jinxiong Zhu, and Shengwu Xiong. Federated clustering for recognizing driving styles from private trajectories. *Engineering Applications of Artificial Intelligence*, 118:105714, 2023.

[45] Xiaodong Ma, Jia Zhu, Zhihao Lin, Shanxuan Chen, and Yangjie Qin. A state-of-the-art survey on solving non-iid data in federated learning. *Future Generation Computer Systems*, 135:244–258, 2022.

[46] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[47] Guilherme Maia, Cristiano Rezende, Leandro A. Villas, Azzedine Boukerche, Aline C. Viana, Andre L. Aquino, and Antonio A. Loureiro. Traffic aware video dissemination over vehicular ad hoc networks. In *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*, MSWiM '13, page 419–426, New York, NY, USA, 2013. Association for Computing Machinery.

[48] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[49] J.S. Mertens, L. Galluccio, and G. Morabito. Centrality-aware gossiping for distributed learning in wireless sensor networks. In *2022 IFIP Networking Conference (IFIP Networking)*, pages 1–6, 2022.

[50] J.S. Mertens, L. Galluccio, and G. Morabito. Mgm-4-fl: Combining federated learning and model gossiping in wsns. *Computer Networks*, 214:109144, 2022.

[51] M. Morafah, S. Vahidian, W. Wang, and B. Lin. Flis: Clustered federated learning via inference similarity for non-iid data distribution. *IEEE Open Journal of the Computer Society*, 4(01):109–120, jan 2023.

[52] Valery Naumov, Rainer Baumann, and Thomas Gross. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '06, page 108–119, New York, NY, USA, 2006. Association for Computing Machinery.

[53] Anh Nguyen, Tuong Do, Minh Tran, Binh X. Nguyen, Chien Duong, Tu Phan, Erman Tjiputra, and Quang D. Tran. Deep federated learning for autonomous driving. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1824–1830, 2022.

[54] Noa Onoszko, Gustav Karlsson, Olof Mogren, and Edvin Listo Zec. Decentralized federated learning of deep neural networks on non-iid data. *ArXiv*, abs/2107.08517, 2021.

[55] Chamath Palihawadana, Nirmalie Wiratunga, Anjana Wijekoon, and Harsha Kalutarage. Fedsim: Similarity guided model aggregation for federated learning. *Neurocomputing*, 483:432–445, 2022.

[56] Sooksan Panichpapiboon and Wasan Pattara-atikom. A review of information dissemination protocols for vehicular ad hoc networks. *IEEE Communications Surveys and Tutorials*, 14(3):784–798, 2012.

[57] Zhiguo Qu, Yang Tang, Ghulam Muhammad, and Prayag Tiwari. Privacy protection in intelligent vehicle networking: A novel federated learning algorithm based on information fusion. *Information Fusion*, 98:101824, 2023.

[58] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018.

[59] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3710–3722, 2021.

[60] Connor Shorten and Taghi Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 07 2019.

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[62] Christoph Sommer and Falko Dressler. *Vehicular Networking*. Cambridge University Press, 2014.

[63] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing (TMC)*, 10(1):3–15, January 2011.

[64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[65] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012. Selected Papers from IJCNN 2011.

[66] Dongyuan Su, Yipeng Zhou, and Laizhong Cui. Boost decentralized federated learning in vehicular networks by diversifying data sources. In *2022 IEEE 30th International Conference on Network Protocols (ICNP)*, pages 1–11, 2022.

[67] Afaf Taïk, Zoubeir Mlika, and Soumaya Cherkaoui. Clustered vehicular federated learning: Process and optimization. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):25371–25383, 2022.

[68] Pu Tian, Weixian Liao, Wei Yu, and Erik Blasch. Wscc: A weight-similarity-based client clustering approach for non-iid federated learning. *IEEE Internet of Things Journal*, 9(20):20243–20256, 2022.

[69] Ferran Torrent Fontbona. *DECISION SUPPORT METHODS FOR GLOBAL OPTIMIZATION*. PhD thesis, University of Girona, 09 2012.

[70] David Tse and Pramod Viswanath. *Fundamentals of wireless communication*. Cambridge University Press, USA, 2005.

[71] Andras Varga. *OMNeT++*, pages 35–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[72] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[73] Zhe Wang, Xinhang Li, Tianhao Wu, Chen Xu, and Lin Zhang. A credibility-aware swarm-federated deep learning framework in internet of vehicles. *Digital Communications and Networks*, 10(1):150–157, 2024.

[74] Stefanie Warnat-Herresthal, Hartmut Schultze, Krishnaprasad Lingadahalli Shastry, Sathyanarayanan Manamohan, Saikat Mukherjee, Vishesh Garg, Ravi Sarveswara, Kristian Händler, Peter Pickkers, N Ahmad Aziz, et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature*, 594(7862):265–270, 2021.

[75] Alexander Wong, Mohammad Javad Shafiee, and Michael St. Jules. Micronnet: A highly compact deep convolutional neural network architecture for real-time embedded traffic sign classification. *IEEE Access*, 6:59803–59810, 2018.

[76] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. cite arxiv:1708.07747 Comment: Dataset is freely available at https://github.com/zalandoresearch/fashion-mnist Benchmark is available at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/.

[77] Kan Xie, Zhe Zhang, Bo Li, Jiawen Kang, Dusit Niyato, Shengli Xie, and Yi Wu. Efficient federated learning with spike neural networks for traffic sign recognition. *IEEE Transactions on Vehicular Technology*, 71(9):9980–9992, 2022.

[78] Lixia Xue, Yuchen Yang, and Decun Dong. Roadside infrastructure planning scheme for the urban vehicular networks. *Transportation Research Procedia*, 25:1380–1396, 2017. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016.

[79] Zeng Yan, Yan Zhong Yi, Zhang JiLin, Zhao NaiLiang, Ren YongJian, Wan Jian, and Yu Jun. Federated learning model training method based on data features perception aggregation. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pages 1–7, 2021.

[80] Liangqi Yuan, Yunsheng Ma, Lu Su, and Ziran Wang. Peer-to-peer federated continual learning for naturalistic driving action recognition. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 5250–5259, 2023.

[81] Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson. End-to-end federated learning for autonomous driving vehicles. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[82] Xiaokang Zhou, Wei Liang, Jinhua She, Zheng Yan, and Kevin I-Kai Wang. Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles. *IEEE Transactions on Vehicular Technology*, 70(6):5308–5317, 2021.

[83] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

[84] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.