



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Federated Transportation Mode Detection: A Client Selection Proposal

Pedro Arthur Rangel Linhares Lima

Orientador

Dr. Carlos Alberto Vieira Campos

RIO DE JANEIRO, RJ - BRASIL
AGOSTO DE 2023

Federated Transportation Mode Detection: A Client Selection Proposal

PEDRO ARTHUR RANGEL LINHARES LIMA

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:



Prof. Dr. Carlos Alberto Vieira Campos — UNIRIO



Prof. Dr. Pedro Nuno de Souza Moura — UNIRIO



Prof. Dr. Guilherme de Melo Baptista Domingues — UERJ

RIO DE JANEIRO, RJ - BRASIL

AGOSTO DE 2023.

L732 Lima, Pedro Arthur Rangel Linhares
Federated Transportation Mode Detection: A Client Selection Proposal / Pedro Arthur
Rangel Linhares Lima, 2023 – Rio de Janeiro, 2023
44p

Orientador: Carlos Alberto Vieira Campos
Dissertação (Mestrado) - Universidade Federal do Estado do Rio de Janeiro, Programa de
Pós-Graduação em Informática, 2023.

1. Sistemas de Transporte Inteligentes. 2. Aprendizado de máquina. 3. Aprendizado
federado. I. Campos, Carlos Alberto Vieira II. Título.

Gostaria de dedicar este estudo a todas as pessoas, encarnadas e desencarnadas, que me ajudaram de alguma forma a concluí-lo. O apoio e incentivo delas serviram como combustíveis, alimentando constantemente os motores da perseverança e determinação ao longo da jornada

Agradecimentos

Meu primeiro agradecimento formal pela realização deste trabalho não poderia deixar de ser direcionado aquele quem mais teve paciência para me ajudar a construí-lo, ao meu orientador Dr. Carlos Alberto Vieira. Durante todo processo de pesquisa, seu exemplo e sua paciência como educador não somente serviram de baluarte fundamental e seguro, como também inspirou a refletir que o caminho da docência na área de computação poderia ser um feliz caminho a ser trilhado, repleta de dinamismo e produção.

Seguindo com os agradecimentos, não poderia deixar de registrar meus mais sinceros e humildes agradecimentos ao meu colega de pesquisa, Elton Soares. Ao lado do meu orientador, ele ensinou valiosas lições mesmo não sabendo que estava fazendo isso, compartilhando seu conhecimento pacientemente, incentivando nos momentos de desânimos, e estimulando a fazer o melhor que estiver ao alcance, aprendendo com os erros para seguir em frente com resiliência, organização, disciplina e bom humor.

Extendo meus agradecimentos também À CAPES pelo suporte financeiro recebido ao longo desta pesquisa, contribuindo fundamentalmente para a continuidade e conclusão da mesma. Graças a fomentos desta natureza, pesquisadores acadêmicos de todas as áreas encontram meios de sustentação e crescimento para seus estudos.

Agradeço também à minha família pelo apoio incondicional e incentivo ao longo desta empreitada. Seu amor, orientação e crença foram fundamentais para meu sucesso. Sou verdadeiramente feliz por ter uma família tão amorosa e solidária ao meu lado.

Gostaria de estender um agradecimento especial ao meu leal companheiro Marola, meu amigo peludo, cujo rabo abanando e amor incondicional proporcionaram conforto durante as longas horas de estudo. Sua presença trouxe alegria aos meus dias e ajudou a criar um ambiente equilibrado e estimulante.

Por fim, expresso minha sincera gratidão ao meu querido companheiro Pedro, cujo

apoio, compreensão e encorajamento me deram força e motivação para superar desafios. Sua presença em minha vida tem sido uma fonte constante de inspiração, e sou grato por seu amor e companheirismo ao longo desta jornada acadêmica.

A todos aqueles que desempenharam um papel significativo no meu crescimento acadêmico e pessoal, expresso os meus sinceros agradecimentos. Esta dissertação é o culminar do apoio coletivo, amor e orientação que recebi. Sinto-me humilde e honrado em compartilhar essa conquista com cada um de vocês.

Lima, Pedro Arthur R. Linhares **Federated Transportation Mode Detection: A Client Selection Proposal**. UNIRIO, 2023. 49 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

O uso generalizado de smartphones, com seus diversos sensores, possibilitou que Sistemas Inteligentes de Transporte (ITS) coletassem uma variedade de dados sobre a mobilidade do usuário. Esses dados são utilizados para gerar informações valiosas, como o modo de transporte utilizado pelo usuário, sendo explorados por aplicativos e serviços baseados em localização (LBS). No entanto, a utilização desses dados sensíveis levanta preocupações sobre a privacidade dos usuários quando são empregados em algoritmos de aprendizado de máquina. Uma alternativa promissora para abordar esse desafio é a Aprendizagem Federada, que protege a privacidade dos usuários por meio de seu paradigma simplificado. No entanto, o uso do Aprendizado Federado enfrenta obstáculos, como a heterogeneidade dos dados, que pode exigir mais rodadas de treinamento federado para obter um bom desempenho, aumentando os custos de comunicação de forma geral. Isso não é desejável para aplicações federadas. Uma possível solução para esse problema é a seleção de clientes que participarão do treinamento federado. No entanto, poucos estudos na literatura apresentaram soluções federadas para o problema da inferência do modo de transporte, propondo uma seleção de clientes para reduzir os custos de comunicação no treinamento federado. Neste trabalho, propomos uma solução federada para o problema da inferência do modo de transporte do usuário, que aplica uma seleção de clientes para filtrar aqueles que podem contribuir melhor para o desempenho médio do treinamento federado. Nosso objetivo é demonstrar que a seleção de clientes pode reduzir os custos de comunicação em termos de rodadas de treinamento federado, sem comprometer o desempenho médio geral do modelo. Realizamos experimentos de avaliação em um framework desenvolvido para simular treinamentos em Aprendizado Federado. Criamos diferentes clientes com conjuntos de dados desbalanceados para simular a heterogeneidade dos dados no treinamento e avaliar o desempenho médio geral do modelo em condições mais próximas do mundo real. Ao final das simulações, comparamos o desempenho médio

geral do modelo utilizando e não utilizando a seleção de clientes para diferentes quantidades de rodadas de treinamento federado, demonstrando a eficiência da nossa proposta de seleção.

Palavras-chave: Sistemas Inteligentes de Transporte, Aprendizado de Máquina, Aprendizado Federado

ABSTRACT

The widespread use of smartphones, with their various sensors, has enabled Intelligent Transportation Systems (ITS) to collect a variety of data about user mobility. This data is used to generate valuable information, such as the mode of transportation used by the user, and is leveraged by location-based services (LBS) and applications. However, the use of this sensitive data raises concerns about user privacy when employed in machine learning algorithms. A promising alternative to address this challenge is Federated Learning, which protects user privacy through its simplified paradigm. However, the use of Federated Learning faces obstacles such as data heterogeneity, which may require more rounds of Federated Training to achieve good performance, increasing communication costs overall. This is not desirable for federated applications. One possible solution to this problem is client selection for Federated Training. However, few studies in the literature have presented federated solutions to the problem of transportation mode inference, proposing client selection to reduce communication costs in Federated Training. In this work, we propose a federated solution to the problem of user transportation mode inference that applies client selection to filter those who can contribute better to the average performance of Federated Training. Our goal is to demonstrate that client selection can reduce communication costs in terms of Federated Training rounds without compromising the overall average performance of the model. We conducted evaluation experiments in a framework developed to simulate Federated Learning training. We created different clients with unbalanced datasets to simulate data heterogeneity in training and thus evaluate the overall average performance of the model under conditions closer to the real world. At the end of the simulations, we compared the overall average performance of the model using and not using client selection for different amounts of Federated Training rounds, demonstrating the efficiency of our selection proposal.

Keywords: Intelligent Transportation Systems, Machine Learning, Federated Learning

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Justification	3
1.3	Objectives	4
1.3.1	Main Objective	4
1.3.2	Research Questions	4
1.4	Contributions	4
1.5	Structure	5
2	Theoretical Background	6
2.1	Features Engineering	7
2.1.1	Window Segmentation	7
2.1.2	Feature Extraction	8
2.2	Machine Learning	8
2.2.1	Feedforward Neural Networks	8
2.3	Federated Learning	9
2.3.1	Federated Averaging	11
3	Related works	13

3.1	Transportation Mode Detection	13
3.2	Federated Learning	15
3.2.1	Client Selection strategies	17
3.3	Discussion	18
4	Transportation Mode Detection based on Federated Learning	19
4.1	Introduction	19
4.2	Proposed Solution	20
4.2.1	Federated Transportation Mode Detection (FedTMD)	20
4.2.1.1	Data Collection and Preprocessing	23
4.2.1.2	Window Segmentation	24
4.2.1.3	Features Extraction	24
4.2.1.4	Federated Transportation Mode Classification	24
4.2.2	FedTMD Client Selection	25
4.2.2.1	Client Selection Strategy	26
4.2.3	Communication Costs	26
5	Experimental Evaluations	27
5.1	Implementation Details	27
5.1.1	Training Settings	27
5.1.2	Simulation Framework	28
5.2	Performance Metrics	29
5.2.1	Accuracy	30
5.2.2	Precision	30
5.2.3	Recall	31
5.3	FedTMD Client Selection Performance	31

5.3.1	Results Obtained	32
5.3.2	Analysis	32
5.4	FedTMD Communication Costs Comparison	39
5.4.1	Results Obtained	40
5.4.2	Analysis	40
6	Conclusions and Future Directions	41

List of Figures

4.1	FedTMD training with client selection	21
4.2	FedTMD architecture	22
5.1	Average performance results for one federated training round	33
5.2	Average performance results for five rounds of federated training	34
5.3	Average performance results for ten rounds of federated training	35
5.4	Average performance results for fifty rounds of federated training	36
5.5	Confusion Matrix	37

List of Tables

4.1	Sensors sets	23
5.1	Configuration values applied for each hyperparameter.	28
5.2	Performance results of centralized experimental reproduction	32
5.3	Comparison of average performance metrics with and without client selection in the best performing configuration	37
5.4	Communication costs results with and without the client selection.	40

1. Introduction

In the last decades, cities have become symbols of modernity and progress, and the emergence of Information and Communication Technologies (ICT) has given rise to a new proposal of cities, known as “Smart Cities”. These cities are characterized as urban areas that use telecommunication infrastructure, integrated management and control systems, and context-sensitive processes to help manager the city.

As in every big city, the transportation system is one of the most essential systems, allowing mobility to people to do their day-a-day activities. However, the disordered growth of many urban areas causes severe problems in the management of the transportation systems in general. It had stimulated the development of a dynamic research area focusing on the application of technology in solutions for human mobility called Intelligent Transportation Systems (ITS).

At the same time, technical advances in data collection have enabled the collection of a massive amount of data from varied sources, which has enable the development of data-driven new ITS applications, i.g. surveillance of traffic, detection of congestion and accidents, collection of toll tax, terrorist monitoring, travel speed estimation, traffic flow prediction [7, 13, 14, 26]

In that context, within the domain of ITS the transportation mode that the user is taking has become an essential and valuable information for several Location Based Services (LBS) to bring context-awareness for ITS data-driven applications (i.g. recommendations systems, personal assistants, human mobility behavior analysis, automatic carbon footprint estimator, real-time trajectory planners). This information can also help governments, companies and institutions to deal with transport-related problems, either in planning, operation, or control areas [6, 13, 14, 20, 21].

Thanks to that, there is a significant interesting from academic and commercial fields in developing brand new solution to better estimate the user transportation mode. As

the benefits, it can bring more accurate estimation of aggregated user flows for the public transportation companies [7]. Analyzing such transportation flows can allow a better schedule and plan of vehicle routes. Another valuable application of transportation mode detection is in estimating environmental impacts, such as fuel consumption and greenhouse gas [21]. Even in financial fields, transportation mode information can be helpful to companies providing a data-supported analysis for their transportation system expansion to investors [7].

1.1 Motivation

Data as transportation mode, location data, or even smartphone sensor measurements are example of spatial and temporal sensitive data that are closely related to user privacy. This kind of data related to people's activities represents personal habits, preferences, and behaviors. Therefore, it is crucial to be careful while handling them, especially in the development of machine learning solutions. Otherwise, it could represent a risk for data privacy and a disobedience of data privacy regulations such as the Brazilian General Data Protection Law (LGPD) [5], or the General Data Protection Regulation (GDPR) [10] of the European Union.

To achieve these and other challenges, a method called Federated Learning (FL) [19] emerged as an alternative to collaboratively train a machine learning model without sharing raw data. FL has been widely applied in different research areas (i.g., medical and financial areas).

The main idea of FL is to keep data locally and to transmit the model parameters and its updates between server and clients. Instead of clients transmitting their data to train a model in a remote server, is the server who send the model to clients to be trained by the client's local data and using the available device's resources. When each round of federated training is concluded, the server receives model parameters updates from each client. Then, the server calculates an updated average for the model parameters, and sends the updated average parameters to clients for the next federated training round. This process continues until a condition is satisfied, such as the model achieving convergence, or a pre-defined timeout has been achieved, or the number of federated training rounds finished.

Besides that, although several sensors are available on smartphone devices, most studies conducting transportation mode detection research still rely on GPS as the primary data source; maybe it is the most popular sensor built into almost all smartphones. How-

ever, considering just GPS as a data source implies a narrow of the spectrum of solutions for the domain. Instead, considering more than GPS as data source can amplify the possibilities of technical solutions.

Meanwhile, the development of Federated Learning systems faces some challenges, such as data heterogeneity. Data heterogeneity refers to the differences in data distributions, formats, or characteristics across different devices in a federated learning system. During federated learning, the server aggregates the model updates from multiple devices. With data heterogeneity, the models trained on different data distributions or characteristics may have divergent updates. As a result, more communication rounds may be needed to synchronize the models, leading to increased communication costs. One feasible solution is to sample a sub-group of clients, implementing a client selection strategy. With a client selection strategy, only a subset of devices or participants are chosen to participate in each round of federated learning. By selecting clients based on certain criteria (e.g., data diversity), the overall communication volume can be reduced. Client selection strategies can optimize the aggregation process during federated learning. Instead of blindly aggregating updates from all clients, a selective strategy allows for more efficient aggregation by focusing on the most informative or representative clients.

1.2 Justification

Transportation mode detection and Federated Learning are both challenging research fields that still have a lot to be explored together. Nowadays, data privacy concerns emerged as a mandatory issue, while data collection advances has enable LBS to gather more data from different resources, expanding the horizon of technical solutions but also imposing other challenges, such as the tenue line among data heterogeneity, average model performance and communication costs in Federated Learning training.

Findings from literature showed that although there are transportation mode solutions employing Federated Learning, few of them have explored a client selection strategy as alternative to increase the average training performance without demanding many communication rounds. Also, few of them use more than GPS as data source, and almost none are evaluating their solutions in scenarios closest to real-world conditions.

With a client selection strategy, it ensures that the selected clients represent a diverse range of data distributions and characteristics. This diversity can help to capture a comprehensive view of the overall data landscape while minimizing redundancy. By selecting clients with diverse data, the need for excessive communication rounds or transmission of

similar updates can be reduced, leading to lower communication costs.

1.3 Objectives

This section presents the main objective of this work and the research questions addressed in the following chapters.

1.3.1 Main Objective

The main goal of this research is to create a decentralized architecture for transportation mode detection applying Federated Learning and the user smartphone sensors. Furthermore, the architecture also proposes a client selection intended to reduce the communication costs during training.

1.3.2 Research Questions

The main research questions of this work are summarized as follows:

1. Can the federated algorithm achieve an average metric performance close to the average metric performance achieved by the centralized version of this algorithm without increasing the communication costs?
2. Can the federated client selection proposal reduce the communication costs in Federated Learning training?

1.4 Contributions

The main contributions of this work are summarized as follows:

1. A federated architecture for transportation mode detection using a Feedforward Neural Network algorithm.
2. A client selection strategy for the federated architecture proposal designed to decrease the communication costs, in terms of bytes.

1.5 Structure

This research is structured as follows. Initially on Chapter 2, we introduced the theoretical background related to transportation mode detection, Machine Learning and Federated Learning research domains. Both challenging research domains whose combination into a single solution is not so much explored in previous studies on literature. Following by Chapter 3, in which the findings in recent literature are discussed.

Within Chapter 4, details of our Federated Learning proposal are presented as well as the client selection strategy. Our proposed client selection strategy proves to be a promising approach to improve the overall average performance of the federated training and reduce communication costs. By considering the quantity of unique transportation modes on local dataset, we can select more diversity clients who has a high-quality contribution for the federated training, leading to enhanced model performance in terms of accuracy, precision and recall.

Finally, on Chapter 5, the simulation evaluation experiments are presented, and the results are discussed to answer the two research questions. To answer the first research question, experiments results presented in section 5.3.1 showed that the federated client selection proposal achieved an average accuracy of 93.89% with 94.11% of precision and 93.67% of recall. This result outperforms the result of 90,94% achieved by the centralized approach. The second research question is related to the comparison of communication costs with and without the client selection in a federated learning training. The experiment results presented on section 5.4 confirms that the client selection proposal has the lowest communication costs, achieving an improvement of 25% in terms of bytes.

2. Theoretical Background

In this chapter, we introduce the theoretical background concerning about Features Engineering, Machine Learning and Federated Learning. These topics are the fundamental concepts behind this research.

Transportation mode detection is a domain study of academia research that intent to infer the transportation mode used by the users during their displacement (i.g., bus, car, walk, bicycle).

The possibilities to infer the transportation mode have changing a lot during the time,improving technically according to available technologies. First researches related to this domain were surveys that collected the transportation mode information from user after they finish their activities. These surveys tried to kept all required information from them to infer their transportation mode. However, these solutions were expensive, time-consuming tasks, and rely on handcrafted features.

Over time, technological development brought alternatives to researchers than surveys, and therefore the problem of inferring the user's mode of transport has become an issue that can always be improved as new technologies and paradigms emerge.

With the diffusion of mobile devices, transportation mode detection solutions started to explore other solutions rather than surveys to explore the data generated by the sensors built in these devices (i.g., GPS, gyroscopic, barometers). Thus, devices such as wearable, cameras, and mainly smartphones, become possible automatic data collectors, speeding up and improving studies. Nowadays, studies apply Feature Engineering techniques to extract features from data collected by these devices, and in most cases, some Machine Learning algorithm is used to classify the transportation mode.

In the following section, we introduce the concepts behind Features Engineering and Machine Learning. Also, we explain about the Federated Learning paradigm, another

essential concept to better understand this work.

2.1 Features Engineering

Feature Engineering is the process of selecting and transforming the features of input data to improve the performance of a Machine Learning model. Features are the information that describes each instance of the dataset, and are used to build a model that can predict the correct output for new instances. Choosing the correct features is crucial for the model's performance, and the Feature Engineering process involves both selecting and creating new features from existing ones [3, 16]

In the context of Deep Learning, Feature Engineering can benefit deep architectures to learn more complex and abstract features from raw data, reducing the need for manual Feature Engineering [16]. Feature Engineering is an important part of the process of building deep learning models, especially when data is limited or when relevant features are not easily extracted from raw data [3]

2.1.1 Window Segmentation

Temporal data series are kind of data collected through a continuous amount of time, segmented in intervals with the same size [16]. Temporal data segmentation into fixed-size windows is a common technique in time series analysis and data mining. It consists of dividing a time series into segments of predetermined size before extracting data features. The advantages of this approach include dimensional reduction, preservation of temporal structure, and ease of feature extraction. Window segmentation allows adjusting the window size according to specific problem requirements and simplifies feature extraction.

The window segmentation is a technique used to partition the dataset in time windows to make it easier to extract features from data. Segmenting the raw sensor data into smaller slices or windows is the first step to produce features used to train the models. The time series for each sensor are divided in time windows of a fixed size. The size of time windows depends on the types of actions to be recognized, i.e., if time windows need to be sufficiently long to capture relevant activity information, while also providing enough overlap to avoid losing important information at the window boundaries.

2.1.2 Feature Extraction

The process of feature extraction from time series involves identifying patterns from data among the time. For each sequence and sensor, four features were extracted: the minimum value, the maximum value, the average value, and the standard deviation value of each sensor over each current sequence created.

2.2 Machine Learning

Machine Learning can be understood as a learning paradigm that allows a system to improve its performance on a specific task based on past examples [3]. It is a research field of artificial intelligence that focuses on the development of algorithms that can learn from data [16] to perform complex tasks by using statistical methods [11]. The goal of Machine Learning is to find a model that generalizes from training data to make accurate predictions on new data [3]. There are several types of Machine Learning algorithms, including supervised and unsupervised models.

The use of machine learning is constantly present in contemporaneous life, from searching the web to filtering content on social networks and recommendations on e-commerce sites. Machine learning systems are used to identify objects in images, transcribe speech into text, match news, posts or products to users' interests, and curate relevant search results [16].

Machine learning domain considers transportation mode detection as a classification task, and it begins to be largely adopted in transportation mode detection studies to replace handcrafted features by features learned by algorithms. Artificial Neural Networks structures such as Feedforward Neural Networks were introduced in solutions, speeding up developments and reducing time and financial costs demanded.

2.2.1 Feedforward Neural Networks

In this work, we are building a Feedforward Neural Network (FNN), also known as a Forward Propagation Neural Network. These networks have a layered structure that allows information to flow unidirectionally from input to output without feedback. The Dense layers connect each neuron to all neurons in the preceding layer, resulting in a highly interconnected network.

Feedforward Neural Networks (FNN) are utilized for various tasks, including classi-

fication. The FNN's primary objective is to create a mapping from inputs to outputs. In order to accomplish this, the FNN uses adaptable weights and biases that are fine-tuned during the model training process.

To understand how FNNs are used in classification, we can examine a basic example of binary classification. We have a dataset with instances represented by feature vectors and corresponding labels indicating membership in either the positive or negative class. Creating a classification model with an FNN involves defining the network architecture, compiling the model with a loss function and optimizer, and training it with the provided data.

The typical neural network architecture includes one or more dense layers, often using non-linear activation functions like ReLU. The final layer usually consists of a single unit with a sigmoid activation function, producing an output between 0 and 1, indicating the likelihood of the instance belonging to the positive class. During training, the model adjusts network weights and biases to minimize loss by comparing predicted outputs to actual labels. An optimizer updates these parameters based on the loss function's gradient.

For multiclass classification, the network architecture is similar to binary classification but includes distinctions. The output layer contains units corresponding to the number of classes, each one with the softmax activation function that normalizes outputs to interpret them as probabilities for each respective class.

During training, the model adjusts its weights and biases to minimize the loss function, usually a categorical cross-entropy function or another suitable option for multi-class problems. The optimizer continuously updates the parameters based on the loss function's gradient.

2.3 Federated Learning

Federated learning is a distributed machine learning paradigm that enables the training of machine learning models on decentralized data [19].

In Federated Learning, instead of to concentrate all data in one location, the data is kept on local devices and the models are collaboratively trained through parameter updates. The typical Federated Learning process can be divided into three main phases:

1. **Model Initialization:** In this phase, the central server initializes the model and distributes it to the participating devices. This model can be a pre-trained model or a

randomly initialized model. The central server sends the initial model parameters to each device, and each device uses these parameters as the starting point for training using its local data.

2. **Local training:** In this phase, each device trains the model on its local data. The local training process can be done using various machine learning algorithms, such as Gradient Descent or Stochastic Gradient Descent. During local training, each device updates the model parameters based on its local data and sends the updated parameters back to the server.
3. **Model Aggregation:** In this phase, the server aggregates the updated model parameters from the participating devices and computes a new global model. The aggregation process can be done using methods as Federated Averaging. Once the server finishes computing the new global model, it sends the new updated model parameters to each device again for further local training.

These three phases are typically repeated iteratively until the global model converges, or another condition is satisfied (i.g., timeout or number of total federated rounds).

One of the key benefits of Federated Learning is that it allows multiple devices to collaboratively train a model without sharing their data with a central server, which can address privacy concerns and reduce communication costs. Additionally, FL can handle data heterogeneity and data distribution differences across clients, which makes it useful for scenarios in which the training data is spread across various locations.

However, there are also some limitations in applying FL. Firstly, FL is computationally more expensive since it requires multiple iterations of model training and model aggregation. In comparison with centralized approaches, the federated training paradigm implies in a computational resource consuming in client's devices. For that reason, to look for lighter approaches is recommended in FL training, intending to consume the minimum quantity of computational resources from client's devices.

The process of aggregating models from multiple clients can introduce noise, which can affect the model's accuracy. Thus, the quality of the global model can be affected by the quality and size of the local datasets, leading to a degradation for the model to achieve the convergence.

In the context of intelligent transportation systems and in transportation mode detection research area, FL could be used to train a model to classify the transportation mode of the user based on its smartphone sensor data. Applying FL, the model can be trained

on the user’s device without transmitting the raw sensor data to a central server, which enhances data privacy preservation. Additionally, FL can handle the heterogeneity and distribution differences of user sensor data, which can improve the model’s performance.

2.3.1 Federated Averaging

The Federated Averaging algorithm (FedAvg) [19] is widely applied in federated optimization [9]. FedAvg relies on Stochastic Gradient Descent (SGD) methods, which allow the adaptation of the model’s structure (including the loss function) by adjusting the parameter value of the gradient. In a federated setting, FedAvg selects a fraction of N clients in each round and computes the gradient of the loss over all the data held by these clients [9].

The FedAvg formula is defined in Equation 2.1:

$$\bar{w}_k = \frac{\sum_{i=1}^N n_i w_{i,k}}{\sum_{i=1}^N n_i} \quad (2.1)$$

In Equation 2.1, $\bar{w}_k$ is the vector containing the mean of the model’s weights during the k th iteration. N denotes the total number of clients, while n_i represents the number of training samples available for client i . Lastly, $w_{i,k}$ means the model’s weight vector for client i in the k th iteration.

FedAvg enables clients to perform multiple update batches with their local data, resulting in communication efficiency. This is achieved as the client only sends back to the server the latest update [9]. Federated Learning (FL) offers several advantages, including smarter model development, reduced latency, lower power consumption, and privacy preservation. Additionally, FL allows the immediate utilization of the enhanced model on users’ devices, leveraging personalized experiences based on individual phone usage patterns.

However, federated optimization still faces various challenges. Communication costs arise from the periodic exchange of model parameters between clients and the server. The size of models becomes problematic in terms of communication costs, but researchers have proposed advanced methods such as efficient model updates, stochastic gradient descent (SGD), and gradient compression to mitigate these challenges [17, 18].

Another challenge is the statistical heterogeneity of non-IID datasets in federated scenarios. Each user’s unique characteristics result in significant statistical heterogeneity,

leading to challenges related to model convergence [17, 28]. Large-scale federated learning for training also introduces difficulties in achieving model convergence due to distributed optimization [17, 28].

System heterogeneity is another factor that poses problems in federated learning. Different operation systems, heterogeneous devices, varied network conditions, and the absence of a guarantee that every client will stay continuously connected to the server during each learning round contribute to the challenges faced in federated learning [17, 28].

Privacy concerns are one of the most importance in federated learning, as user data may be exposed during communication with the server. Federated systems are reported to be more vulnerable to poison attacks compared to traditional approaches [17, 18, 28]. Ensuring data privacy remains a critical aspect of federated learning.

Moreover, labeled data availability is another challenge in federated learning. Users are encouraged to generate data rather than label it, which affects the availability of labeled data for training [18, 26].

3. Related works

In this chapter, we organize the related works about transportation mode detection using Machine Learning and Federated Learning algorithms. We searched in IEEE OpenAccess, ACMLibrary, and Google Scholar for state-of-the-art papers published between 2018-2022.

3.1 Transportation Mode Detection

In this section, we organized the finding on the literature about transportation mode detection.

[6] explores the potential of detecting transportation modes through smartphone sensors. The article highlights the benefits of transportation mode detection, including the possibility of enabling novel services such as smart parking and vehicular traffic monitoring. The use of smartphone sensors can improve the accuracy of transportation mode detection by exploiting sensor diversity. The article proposes a custom dual transportation mode detection algorithm that uses a combination of accelerometer, gyroscope, and magnetometer sensors to detect walking and driving modes. The algorithm is evaluated using a dataset collected from 10 participants, and the results show that the proposed algorithm achieves high accuracy rates. However, there are several open issues related to transportation mode detection using smartphone sensors, such as the variability of sensor data due to different factors and the need for a large and diverse labeled dataset to train and evaluate transportation mode detection algorithms. The article suggests that the development of standardized datasets and evaluation metrics can help address these challenges. In summary, the article proposes a promising approach for detecting transportation modes using smartphone sensors, but further research is needed to improve the accuracy and reliability of the algorithm in real-world scenarios.

In [23], it was proposed the utilization of recurrent neural networks (RNNs) in detecting travel modes in real-time using smartphone sensor data. The algorithm combines accelerometer and GPS data to identify six modes of transportation: walking, running, cycling, car, bus, and train. The model was trained on a labeled dataset and evaluated on a separate dataset from ten participants, showing high accuracy rates for all six transportation modes. However, several challenges were identified, including the need for a large and diverse labeled dataset, computational complexity, and potential privacy concerns. Addressing these challenges through the development of standardized datasets, optimization techniques, and privacy-preserving algorithms can improve the accuracy and reliability of the algorithm in real-world scenarios. Overall, the article presents a promising approach for online travel mode detection using RNNs, but further research is needed to address the open issues.

[22] proposes a method for improving the accuracy and computational cost trade-off of online travel mode detection. The proposed method uses state-of-the-art feature engineering and automated machine learning techniques to detect the mode of transportation in real-time using smartphone sensors. The paper highlights the importance of travel mode detection for various applications in the field of Smart Mobility and Internet of Things, especially with respect to Intelligent Transportation Systems. The proposed method is evaluated on a real mobility dataset using different features and parameters, and the results show that it achieves high accuracy rates with low computational cost. In general, the use of automated machine learning and feature engineering techniques for online travel mode detection is a promising approach.

[20] presents a method for detecting the mode of transportation used by an individual using an efficient convolutional neural network (CNN). The proposed method uses accelerometer and gyroscope data collected from a smartphone to identify whether the user is walking, running, cycling, or driving a car. The article describes in detail the architecture of the proposed neural network, which consists of several convolutional and pooling layers, followed by dense and output layers. The network was trained on a dataset of over 10,000 samples collected from 50 participants in different environments and conditions. The article explores the effectiveness of the proposed neural network for detecting transport modes in various scenarios, achieving an average accuracy of 93.1% and surpassing existing methods. However, the article has some open points. For instance, the dataset used to train and test the network was collected in a limited environment of four transportation modes. Consequently, it's unclear how the network would perform in more complex environments or with other transport modes. Additionally, the article doesn't address the ethical or privacy implications of using smartphones to gather people's move-

ment data.

The paper [21] proposes a methodology for identifying the mode of transportation used by individuals using GPS data. The process includes data preprocessing, feature extraction, and classification. In the data preprocessing stage, the raw GPS data is cleaned and filtered to eliminate any noise or errors. After that, features like speed, acceleration, and direction are extracted from the processed data. These features serve as inputs for the classification model, which is based on machine learning algorithms like decision trees, support vector machines, and random forests. The model is trained on a labeled dataset containing GPS data for various modes of transportation, such as walking, cycling, driving, and public transportation. Once trained, the model can predict the mode of transportation for new GPS data. The article outlines several issues with this methodology, including the lack of a standardized dataset for evaluating the performance of different transport mode detection algorithms. Additionally, the variability of GPS data due to factors such as signal quality, device accuracy, and environmental conditions can impact the accuracy of the transport mode detection algorithm and make it difficult to achieve high accuracy rates. Despite this, the article presents a promising methodology for detecting the mode of transportation based on GPS tracking data, but further research is required to address the open issues and increase the algorithm's accuracy and reliability.

3.2 Federated Learning

In this section, we summarized the studies related to transportation mode detection that applies Federated Learning paradigms on their solutions, mainly client selection solutions.

[24] introduces a deep learning method for identifying a user's mode of transportation and speed using limited GPS data. The proposed method employs a multitasking deep learning approach that simultaneously identifies the user's mode of transportation and speed. The paper provides an in-depth description of the proposed neural network's architecture, which includes several convolutional, pooling, dense, and output layers. The network was trained on a dataset collected from users who used various modes of transportation in urban environments. The results indicate that the proposed neural network achieves an average accuracy of 92.1% in detecting transportation modes and an average accuracy of 89.4% in estimating user speed, outperforming other existing methods. The proposed method also performs well with limited GPS data, which is beneficial in scenarios where GPS accuracy is low or restricted. However, the article has some open points.

For example, the dataset used to train and test the network was collected in a specific urban environment and may not be representative of other environments. Furthermore, the article does not discuss the ethical or privacy implications of using location data to track people's movements.

In their paper [28], the authors propose a Federated Learning approach to predicting transportation mode detection using an augmented attention CNN network. They implement a labeled data sharing strategy to make client local dataset distribution more independent and identically distributed (IID). The server trains the global model initially among clients from a selected group using labeled data from a public dataset. After the global model converges, the server selects another group of different clients and broadcasts the global model along with a set of IID-labeled data from the public dataset. The clients then select the necessary data to make their distribution IID and merge it with their local dataset to train the global model. The authors evaluate the impact of different percentages of labeled data shared with clients (10, 30, 50, and 100), showing that accuracy improves until a certain level, after which the improvement is insignificant. They conclude that 50 percent of labeled data is sufficient to avoid communication overhead.

In [13], the authors proposed a semi-supervised deep ensemble learning approach to identify travel modes using smartphone sensor data. The study uses deep learning models and ensemble learning techniques to improve the accuracy of travel mode identification. The authors also introduce a semi-supervised approach that leverages both labeled and unlabeled data to improve the model's performance. One of the main contributions of this study is that it achieves state-of-the-art performance on a publicly available dataset for travel mode identification. However, the proposed approach requires a large amount of labeled data to achieve high accuracy, which can be a challenge in real-world scenarios where labeled data may be scarce or expensive to obtain. Additionally, the study only considers a single dataset for evaluation, and it would be interesting to see how the proposed approach performs on other datasets or in different contexts.

In their paper [27], the authors introduced FedOVA - a new federated learning technique that addresses the challenge of non-IID data by training multiple local models for each class and combining them into a global model. The authors demonstrate that FedOVA outperforms existing methods on various benchmark datasets with non-IID data. However, the proposed method may be computationally expensive since it requires a large number of local models to achieve high accuracy, and the authors only evaluated it on a limited number of benchmark datasets.

3.2.1 Client Selection strategies

[1] proposes a federated edge learning approach that combines edge computing and federated learning to train a global model on unlabeled data from multiple sources. The approach addresses the challenge of limited labeled data in smart cities by using unlabeled data without sharing raw data, and overcomes privacy and security concerns. However, the proposed approach requires a large number of edge devices to participate in the federated learning process, which can be challenging in real-world settings. Additionally, the authors only evaluate the approach on a single dataset, making it unclear how it performs in different smart city environments or on other datasets.

The article [18] proposes RC-SSFL, a communication-efficient semi-supervised federated learning system that uses robust optimization and semi-supervised learning to improve performance. RC-SSFL addresses the challenge of robustness in federated learning by improving the stability of the training process through robust optimization techniques. The authors demonstrate that RC-SSFL outperforms existing methods on several benchmark datasets. However, there are limitations to the proposed system, such as its requirement for a large amount of labeled and unlabeled data to achieve high accuracy and the evaluation on a limited number of benchmark datasets.

[12] proposed an algorithm called FedFa to achieve more fairness and accuracy in horizontal federated learning. The proposed algorithm uses a double momentum gradient optimization scheme to accelerate the convergence rate of the model and improve fairness and accuracy in the training process. The authors show that FedFa outperforms existing federated learning methods on several benchmark datasets. One of the main contributions of this study is that it addresses the challenge of fairness in federated learning. Federated learning systems are vulnerable to bias and discrimination due to the distributed nature of the training process. FedFa tackles this problem by promoting fairness in the training process. However, there is limited information available on the open issues addressed in this study. Further research is needed to evaluate the performance of FedFa on other datasets or in different contexts, and to identify any potential limitations or open issues with the proposed algorithm.

The paper [25] proposed an architecture that clusters vehicles based on their mobility patterns to improve learning accuracy under mobility constraints. The architecture combines clustering and federated learning techniques to train a global model on data collected from multiple vehicles. The authors demonstrate that the proposed process can improve learning accuracy in several non-independent and identically distributed datasets distributions, under mobility constraints, compared to standard federated learning. The

architecture overcomes the challenge of mobility in vehicular federated learning by clustering vehicles based on their mobility patterns and using this information to optimize the training process. However, further research is necessary to fully understand its capabilities and limitations.

The paper [8] discusses a federated learning method called FedCPF for the vehicular edge computing (VEC) environment in 6G communication networks. FedCPF addresses the problem of high communication costs in traditional FL, which can take up regular communication resources. The FedCPF approach proposes a customized local training strategy with a constraint term on the local objective function at clients to improve the convergence speed and reduce the number of communication rounds. FedCPF also introduces a flexible aggregation policy to reduce communication overhead and speed up the training process. Experimental results show that the FedCPF approach is effective in reducing communication costs in vehicular networks compared to the traditional FL algorithm.

3.3 Discussion

From the findings presented early section, it can be observed that studies related to both transportation mode detection and Federated Learning still have several opportunities to be explored for new researches.

The articles presented in Section 3.1, offered different approaches for detecting transport modes, using smartphone sensors, convolutional neural networks, and GPS data. The proposals presented promising results, but some challenges still need to be overcome, such as the variability of sensor data, the need for large, standardized, labeled datasets to train and evaluate algorithms, and the ethical and privacy implications of using smartphones to collect people's movement data.

In Section 3.2, the Federated Learning proposals summarized addressed challenges such as lack of labeled data, training process robustness, fairness, and mobility in federated systems. Proposals include optimization techniques, semi-supervised approaches, and client selection algorithms to improve accuracy and efficiency. Further research is needed to evaluate effectiveness in real-world scenarios. Additionally, studies should explore other data sources besides GPS.

4. Transportation Mode Detection based on Federated Learning

4.1 Introduction

In recent times, remarkable progress in data collection and mobile technologies has enabled a extensive collection of data daily, primarily sourced from sensors embedded within smartphones. Within the domain of Intelligent Transportation Systems (ITS), the acquisition of user transportation mode data assumes great importance, facilitating the development of Location-Based Services (LBS) that engender context-awareness for ITS data-driven applications (i.g. recommendations systems, personal assistants, human mobility behavior analysis, automatic carbon footprint estimator, real-time trajectory planners).

The information of transport, location data, and sensor measurements have inherent sensitivity on its collected raw data, profoundly connected to the imperative of safeguarding user privacy, i.g. GPDR, LGPD. Thus, it is imperative to have prudence and diligence in handling with this kind of data data. In response of this concern, Federated Learning emerged as a decentralized paradigm, presenting an alternative to train machine learning models while preserving data privacy without imposing a substantial computational resource consuming. FL operates by training the model on client's device using their local data, thereby circumventing data privacy concerns. Despite the fact that FL has been widely adopted in diverse research domains, few studies have explored its utilization in the context of transportation mode detection to reduce communication costs.

In the deployment of FL systems, data heterogeneity brings challenges for the federated training. In federated training, dealing with multiple independent clients means to handle many unbalanced datasets. It is reflected in more time to achieve convergence and in higher communication costs in federated training.

In light of these considerations, we introduce FedTMD to address both challenges of privacy concerns and communication costs faced by federated systems. This work has the following contributions, detailed in the next sections:

- Creation of a data privacy preserve transportation mode detection system through the strategic application of Federated Learning paradigms.
- Implementation of a client selection, designed to decrease communication costs
- Validation through the utilization of an authentic unbalanced dataset collected from smartphone sensors
- Evaluation through extensive experiments within a federated simulation framework capable to handle large-scale simulations

4.2 Proposed Solution

In the following sections, we described the proposed solution and its methods in more details.

4.2.1 Federated Transportation Mode Detection (FedTMD)

The term FedTMD was defined to refer to our federated learning proposal, as an acronym for Federated Transportation Mode Detection. FedTMD consists of a supervised algorithm able to classify the user transportation mode based on features extracted from smartphone sensors data trained collaboratively applying Federated Learning paradigm, in which clients do not have to share their local data with a centralized server. Meanwhile, it also implements a client selection strategy that filters clients with more valuable contribution at each federated training round to reduce communication costs. At each federated training round, the strategy selects clients that have a large variety of transportation modes in their local dataset. The strategy selects the quantity of half of the total of clients to participate on federated training round. By applying the size and variability restrictions at each round, only the selected clients will participate on training, thus reducing communication costs for the whole federated learning process.

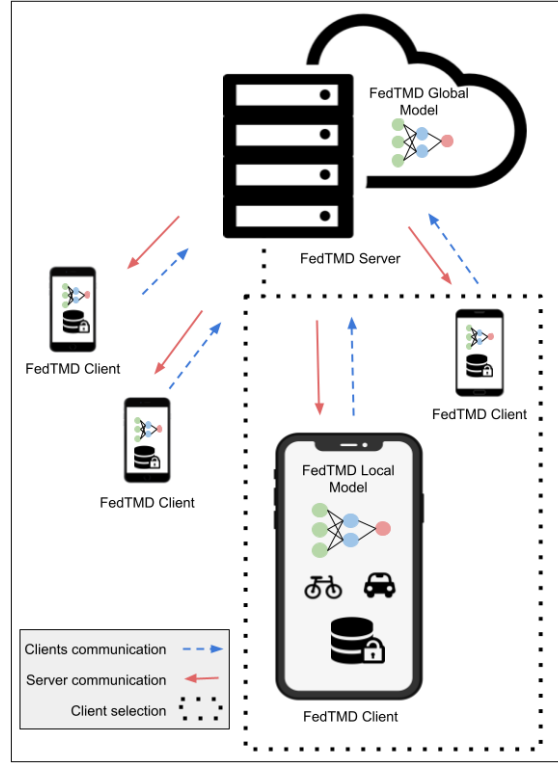


Figure 4.1: FedTMD training with client selection

Figure 4.1 illustrates the components behind the federated learning process and the communication exchanged among them. The federated server (i.e., FedTMD Server) holds the global model (i.e., FedTMD Global Model) and a labeled dataset. It initiates the communications sending to clients (i.e., FedTMD Clients) initial parameters and configurations that will be used on client-side during training and evaluation steps. During federated training, each client will fit its local model (i.e., FedTMD Local Model) on its own data. This allows local customer models to be trained decentrally while the global model is kept in sync through gradient aggregation during federated training rounds, which means that the global model and local models are adjusted and updated in a coordinated manner during federated training. While local models are trained independently on each client, they are periodically combined and aggregated to update the global model.

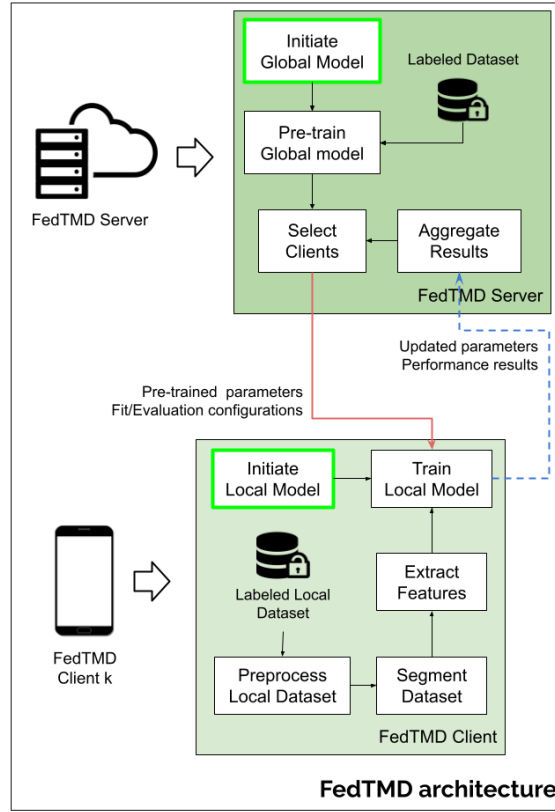


Figure 4.2: FedTMD architecture

Figure 4.2 describes in more details the activities behind the federated training rounds.

The figure 4.2 describes the FedTMD architecture proposal. It begins on server side with the initiation of the global model. Next, the global model is pre-trained on server-side using a labeled dataset adopted to pre-train and validate the global model. Pre-training a model before sharing its parameters is a positive practice for federated learning training if a labeled dataset is available, it allows the model to learn the general characteristics of whole dataset, benefiting local training on client side.

Then, the FedTMD Server selects clients for the federated training round. Based on the quantity of transportation modes present on their local dataset, the server selects the most valuable clients to participate on training and sends to them the pre-trained model parameters.

In parallel, the local model is also initiated on client side and it is trained using the local dataset and the pre-trained parameters received from server. On each client, there is labeled dataset composed by the data collected from smartphone sensors. The data is pre-processed and segmented in fixed windows size before the features are extracted to be used on local training. After concluding the local training, each client sends back to server its updated model parameter and the local performance results.

The server receives the updated parameters from all clients and applies a weighted average aggregation to update the global model. Finally, the server selects another subset of clients for another round of federated training, sending to selected clients the aggregated parameters. This iteration follows until the number of total federated training rounds is achieved.

4.2.1.1 Data Collection and Preprocessing

The data collection stage constitutes a foundational step in the utilization of machine learning algorithms, often preceding the preprocessing stage. In this initial phase, data is acquired from nine sensors embedded within the user’s smartphone. Specifically, sensor data was collected from 16 volunteer participants. The delineated set of transportation modes encompassed pedestrian locomotion, automobile usage, stationary rest, train travel, and bus commuting. Spanning a duration of three months, data collection occurred within a diverse array of settings, encompassing urban, suburban, and rural environments. The centralized dataset distribution entails 26% of data instances attributed to walking, 25% to driving, 24% to stationary activities, 20% to train journeys, and 5% to bus voyages.

In the initial phase of data pre-processing, a series of operations are executed to ensure the quality of the data. These operations involve tasks such as removing erroneous measurements from the sensors, excluding certain measures, and rectifying values from sound and speed sensors to ensure they are positive. Additionally, specific sensors, including environmental sensors such as sound, light, pressure, and proximity sensors, produce singular data values that directly contribute to the dataset. Conversely, other sensors yield multiple values that are inherently tied to the chosen coordinate system, signifying a strong orientation dependence. To address this issue, the magnitude metric was defined as an uniform and orientation-independent metric, which was extracted from all three sets of sensors.

Sensors of first set	Sensors of second set	Sensors of third set
Accelerometer Sound Gyroscope	Sensors of first set + Orientation Linear acceleration Rotation vector Game rotation vector Gyroscope uncalibrated	Sensors of first set + Sensors of second set + Speed

Table 4.1: Sensors sets

The nine sensors were divided into three sets as presented in Table 4.1. First set con-

tain three sensors which are considered to be the basic sensors (accelerometer, gyroscope, and sound). The second set contains sensors from first set plus orientation, linear acceleration, rotation vector, game rotation vector, and gyroscope uncalibrated. Lastly, the third set contains all sensors, including speed that was removed from second set. Each sensor event is collected with a maximum frequency of 20 Hz to ensure the small size file to be stored in the device.

In Figure 4.2, the collection and preprocessing stages are done in both client and server-side following the same procedures, but the obtained datasets are used for distinct ways in each side. The preprocessed dataset enables training and evaluation of the global model on server-side along the federated training rounds. On client-side, the process of collecting and preprocessing the smartphone sensors data are done entirely, and creating in the end an unbalanced dataset composed by the preprocessed smartphone sensors data collected for the client itself by its own device.

4.2.1.2 Window Segmentation

As discussed in Chapter 2, segmenting the raw sensor data into smaller slices or windows is the first step to produce features used to train the models. Window segmentation is the process which the time series is divided in time windows of a fixed size. In FedTMD, the window segmentation is defined as a fixed size of 5 seconds. This value is a common value adopted in similar studies in literature [6,22], and also it is following the best results found in [2].

4.2.1.3 Features Extraction

The next stage involves extracting features from the segmented raw sensors readings. Each window segment is processed using feature extraction techniques, resulting in a set of calculated features vectors for each window. In Figure 4.2, for each window segment, four different statistical features are computed: the maximum value, the minimum value, the mean and standard deviation. For each segment, a total of $\Lambda \cdot 4$ features were calculated, where Λ is the number of sensors. In FedTMD, a total of 36 features are extracted from each fixed size window segment. Then, the features extracted are passed to the federated model, which finally classifies the transportation mode.

4.2.1.4 Federated Transportation Mode Classification

The following stage involves training the federated model to classify the current transportation mode based on features described previously. In Figure 4.2, we created three different federated models for each set of sensors presented in Table 4.1. Our intention

is to understand which of the models achieves a better performance predicting the user transportation mode. For each set of sensors, we built a federated model which is trained and evaluated using the dataset described in Chapter 5.

The federated models consist of supervised machine learning algorithms dealing with a multiple class classification task. For each instance of features received as input, the models learn the corresponded label (i.e., the transportation mode). Thus, for a new instance of feature received, the model classifies it with an unique label from five total options (i.e., walking, car, bus, train, still), which is validated with labeled datasets.

On server-side, this federated model is the FedTMD global model. The server conducts an initial training of the global model with the labeled dataset before sending its initial parameters to clients. This pre-trained global model continues to be collaboratively trained among clients through federated training rounds. On client-sides, this federated model is called FedTMD local model and each clients has its own local model. The local models are trained using the preprocessed local dataset, which contains only the preprocessed sensors data of that specific client. After this, the evaluation metrics obtained are returned to server. Here, we can observe how FL is important to guarantee local data privacy but also allowing a collaboratively model training.

4.2.2 FedTMD Client Selection

When dealing with clients holding each one an unbalanced dataset (i.e., each one has its own data distribution), it may imply in increase communication costs of the federated learning. Data heterogeneity brings the challenge to increase of rounds necessary for the model to achieve the convergence. In vanilla FedAvg, the server receives updated weights and parameters results from clients in aggregation step and calculates an weighted average of them. As higher the data heterogeneity in the federated training is, more rounds will be necessary for the model to achieve the convergence. Usually on vanilla FedAvg, all clients are randomly selected to participate in the round of training, and any kind of sampling criterion is implemented.

Sampling all available clients to participate in every round of training may introduce noise or biased information into the training process from clients which may have a less contribute dataset. As a consequence, more rounds are necessary for the model to achieve the convergence, increasing the communication costs of the whole federated learning.

4.2.2.1 Client Selection Strategy

In this section, we explain in details the client selection strategy employed within the FedTMD architecture, as visualized in Figure 4.2.

Our FedTMD architecture incorporates a client selection process for sampling participants based on the number of unique transportation modes in their local dataset. The criterion for sampling clients is based on the quantity of unique transportation modes to reduce communication costs in the federated learning process.

Our approach seek to use the simplest criterion available instead of implementing an equation to be executed on the client-side, to avoid creating extra computational overhead on the client’s devices. The quantity of unique transportation modes is a feasible solution due to its simplicity, and clients with two or more transportation modes in their local dataset provide more variability of labels for training and evaluating the model, which improves its classification performance when faced with new unclassified instances.

On each federated round, the server demands from N available clients the quantity of unique transportation modes on their local dataset after the collection and pre-processing stages are done, in which N is the number of total clients. On server-side, with N denoting the total number of clients, the server ranks the clients in descending order by the value of the quantity of unique transportation modes on their local dataset.

Subsequently, the server selects $N/2$ of clients to participate in the next federated training round. Our sampling selection process minimizes communication costs and maximizes resource utilization, leading to improved model convergence and generalization.

In summary, the client selection strategy we present capitalizes on the simplicity of counting unique transportation modes in local datasets. This pragmatic approach aligns with our goal of minimizing communication complexity and maximizing the utility of available resources in the federated learning framework. As a result, it contributes to the advancement of model convergence and overall performance within the FedTMD architecture.

4.2.3 Communication Costs

To measure the communication costs in our federated learning process, we implemented two variables proposed at [8], the upper bound that the communication costs can achieve and the total communication costs for the entire federated learning process. Each one of them are explained in following.

5. Experimental Evaluations

In this chapter, we present the simulation experiments conducted to assess the efficacy of our federated approach. We discuss the attained average performance outcomes in terms of accuracy, precision, and recall. Initially, we present implementation details, outlining the construction of our model. Following section explains the performance metrics considered during evaluation experiments (i.e., accuracy, precision, recall). Then, the results obtained from simulations are presented and discussed in the last section.

5.1 Implementation Details

The execution of the simulation experiments was conducted using Python, with the integration of various libraries to facilitate stages such as pre-processing, selection, and evaluation. These libraries encompassed tools such as NumPy, scikit-learn, pandas, TensorFlow, Keras, matplotlib, and seaborn, which collectively contributed to the robustness of this work.

The experimental evaluations was conducted on an Ubuntu 18.04 environment with an Intel(R) Core(TM) i5-3317U CPU operating at 1.70GHz, with a 4GB RAM.

The federated models were implemented using the Fully Connected Layer provided by Keras library. The configuration of its hyperparameters follows the approach outlined in [6].

5.1.1 Training Settings

In our simulations, we diverge from the traditional approach of employing a single labeled dataset for evaluation and validation in supervised classification tasks. Instead, we partitioned the dataset among various clients, simulating a scenario that aligns with

real-world conditions. The main intention behind this division is to investigate learning efficacy in the face of data heterogeneity variability. As elaborated in Chapter 4, our simulation experiments created 16 distinct clients, each holding a distinct dataset that are not fully balanced (i. e., composed by different amounts of samples per transportation mode).

For every simulation instance, we orchestrated the configuration of distinct hyperparameters and then assessed the corresponding performance metric outcomes achieved. Our simulation encompassed four simulation parameters (i.e, total number of federated training rounds, batch size, learning rate, local epochs). For each simulation experiment, these parameters were assigned distinct values, and the cumulative results were aggregated and discussed.

Hyperparameters	Configuration values
Federated training rounds	[1, 5, 10, 50]
Batch size	[4, 8, 16, 32, 64, 128, 256]
Learning rate	[0.1, 0.01, 0.001]
Local epochs	[1, 2, 5, 10, 50]

Table 5.1: Configuration values applied for each hyperparameter.

Table 5.1 summarizes a comprehensive overview of the hyperparameters configurations implemented throughout the simulation experiments, commonly found in literature. The scope of experimentation included varying the number of federated rounds within a range of 1 to 50. Additionally, three different learning rate values were examined: 0.1, 0.01, and 0.001. Extensive exploration was dedicated to batch sizes, aiming to evaluate the minimal value required for optimal performance. The concept of local epochs, while not so much explored in other Federated Learning studies, also has its relevance. Thus, this gap incentives us to explore it in our simulations, analysing its potential impact on the Federated Learning process.

5.1.2 Simulation Framework

In Federated Learning systems, researches face various challenges, as underscored in previous chapters. In addition to these challenges, there is another practical concern in conducting realistic federated simulations: the training and assessment steps must be in conditions more close to real-world as possible.

To address this concern, we adopted Flower [4] as framework to support our deterministic simulations. Flower is an open-source framework especially crafted to support the

development and evaluation of single-machine simulations of Federated Learning systems. It was designed to be scalable and highly customizable, that is, it can handle large-scale simulations with many clients, and it allows to tailor the framework to suit our specific use case. The framework has been widely applied to support evaluation of Federated Learning solutions in several different domains

Within Flower framework, there are two principal approaches available to evaluate the aggregated models: evaluating the aggregated parameters on server-side (i.e., centralized evaluation), and/or evaluating the aggregated parameters on client-side (i.e., federated evaluation). The evaluation approaches are similar, however they have particularities, explained as following:

- **Server-side evaluation** (or, centralized evaluation): this approach mirrors the evaluation mechanisms of conventional centralized machine learning methodologies. A centralized dataset is available all times, employed for evaluation purposes on server-side. On each round, the server evaluates current aggregated parameters using global model and the centralized dataset. Here, we evaluated the global model performance on each federated round.
- **Client-side evaluation** (or, federated evaluation): unlike server-side evaluation, this approach does not require a centralized dataset. Instead, it empower clients to evaluate the present aggregated parameters through their local models and dataset. On each federated training round, the aggregated parameters are trained through local models and a set of performance metrics are sent back to server, which receives and averages them. The major advantage of this approach is to attain a more representative evaluations by using datasets with different distributions.

Our simulation experiments encompassed both evaluations approaches, and the average performance results are presented and discussed in following subsections.

5.2 Performance Metrics

In this section, we explain about the performance metrics used to evaluate our federated classification proposal (i. e., accuracy, precision and recall). These metrics are commonly used to evaluate the performance of classification models, and they provide different insights into the model's performance [15] .

5.2.1 Accuracy

The accuracy metric is a common performance metric used in supervised classification machine learning. It measures the proportion of correctly classified samples over the total number of samples in a dataset [15], and the formula is defined as following, in Equation 5.2.1:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total of Samples}} \quad (5.1)$$

Accuracy can be a helpful metric for monitoring the progress of a model at learning the underlying patterns in the data. However, accuracy do not always provide an accurate representation of a model's performance, it is important to be cautious about overfitting. It occurs when a model becomes too complex and starts to memorize the training data instead of learning the general patterns. As a result, the model may perform extremely well on the training data but fail to generalize to unseen data. For that reason, other metrics such as precision and recall may provide a more comprehensive understanding of the model's performance.

5.2.2 Precision

In machine learning, precision refers to the fraction of true positive predictions among all positive predictions [15]. True positive refers to the instances where the model correctly predicts a positive class, while false positive refers to the instances where the model incorrectly predicts a positive class. Thus, precision measures the model's ability to correctly identify positive instances and avoid false positives. A higher precision value indicates that the model has fewer false positives. The precision formula is presented in Equation 5.2.2:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (5.2)$$

Precision is often used in conjunction with other evaluation metrics such as recall and accuracy to provide a more comprehensive understanding of the model's performance. While precision is an essential metric for evaluating the quality of a classification model, it may not be the only metric to consider. For instance, in some cases, a model with low

precision may still be valuable if its recall is high, meaning that it can correctly identify most positive instances, even if it produces many false positives. Thus, precision should be considered along with other metrics to provide a more complete picture of a model's performance.

5.2.3 Recall

In machine learning evaluation, recall is a performance metric that measures the percentage of true positive cases that are correctly identified out of the total number of positive cases [15]. It is one of the key metrics used in classification tasks where identifying all positive cases is critical. Recall is also known as sensitivity or hit rate and is calculated as the ratio of true positive cases to the sum of true positive and false negative cases. A higher recall value indicates a better performance of the model in correctly identifying positive cases.

The recall formula is presented in Equation 5.2.3:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (5.3)$$

Overall, recall is a crucial metric in evaluating classification models and helps in identifying cases where the model may be underperforming in identifying positive cases

5.3 FedTMD Client Selection Performance

In this section, the outcomes from the simulation experiments are presented, initiating by the replication of original experiments to validate the applicability of the smartphone sensors dataset [6] for classifying users' transportation modes.

The federated architecture is closely mirrored the original study, with two notable distinctions: the expansion of performance metrics and the calculus of the communication costs. The proposal incorporates two performance metrics that were not considered in centralized original solution (i.e., precision and recall) and calculates the communication costs for the centralized original solution. Our intention is to seek more comprehensive understanding of model performance and to compare communication costs between centralized and federated solutions. The results are summarized on Table 5.2

As it can be seen in Table 5.2, the best centralized performance was achieved by the

Algorithm	Sensor Set	Accuracy(%)	Precision(%)	Recall(%)	Costs
Decision Tree	1	72.38	72.81	72.38	228.93 Kbits
Decision Tree	2	78.08	78.26	78.08	171.33 Kbits
Decision Tree	3	83.60	83.64	83.60	142.91 Kbits
Random Forest	1	79.64	79.64	79.64	20.17 Mbits
Random Forest	2	87.68	87.72	87.68	16.21 Mbits
Random Forest	3	90.94	91.06	90.94	13.70 Mbits
Support Vector Machine	1	76.07	75.98	76.07	711.87 Kbits
Support Vector Machine	2	86.07	86.19	86.07	1.14 Mbits
Support Vector Machine	3	89.91	90.12	89.91	1.20 Mbits
Neural Network	1	78.51	74.87	74.51	805.48 Kbits
Neural Network	2	86.71	86.73	86.71	1.63 Mbits
Neural Network	3	90.15	90.18	90.15	1.23 Mbits

Table 5.2: Performance results of centralized experimental reproduction

Random Forest algorithm with 90.94% of accuracy and a communication costs of 13.70 Mbits. Following the rank by the Neural Network algorithm with 90.15% of accuracy and a communication costs of 1.23 Mbits. The results was achieved with the third set of sensors configured in both cases. The cheaper algorithm, in terms of communication costs, was the Decision Tree algorithm achieving 142.91 Kbits with the third set of sensors configured, 171.33 Kbits with the second set of sensors, and 228.93 Kbits with the first set of sensors.

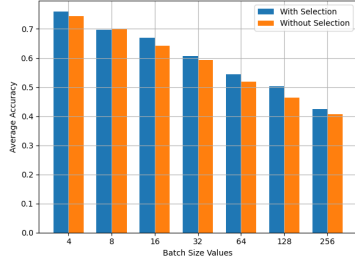
5.3.1 Results Obtained

Within this section, the average performance outcomes of the Federated Learning models applying the client selection strategy are presented. The core objective of these simulation experiments is to compare the impact of our selection approach on the communication costs and the model performance across several federated training rounds, varying values for batch size, learning rate and local epochs.

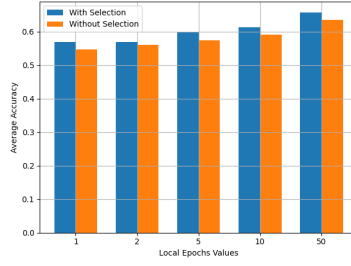
Given the extensive outcomes arising from the simulation experiments, the average performance results were organized in tables and figures to facilitate comprehension and visualization.

5.3.2 Analysis

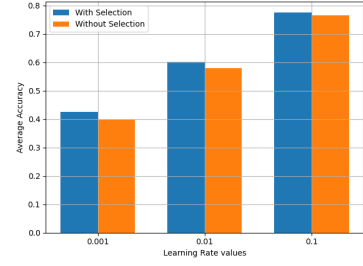
In this section, an analysis about the simulation experiments is presented, elucidating the advantages and opportunities of the federated proposal as well as the efficiency of the client selection over the performance of the federated training and the communication costs.



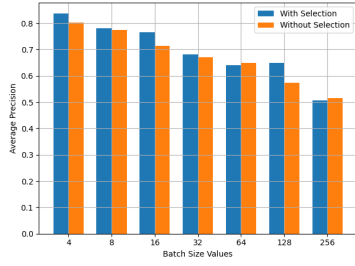
(a) Average accuracy performance comparison across various batch sizes



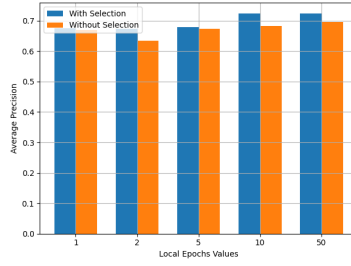
(b) Average accuracy performance comparison across various local epochs



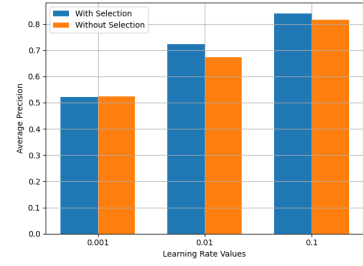
(c) Average accuracy performance comparison across various learning rates



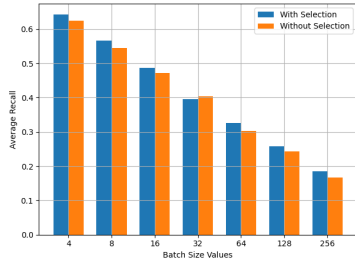
(d) Average precision performance comparison across various batch sizes



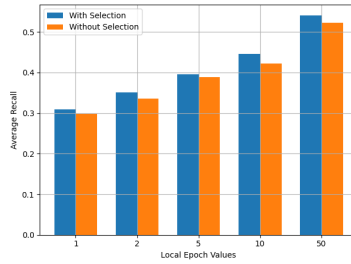
(e) Average precision performance comparison across various local epochs



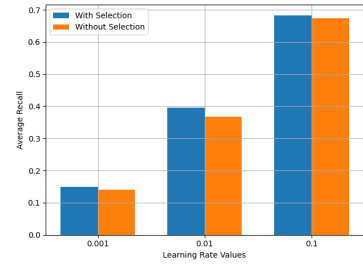
(f) Average precision performance comparison across various learning rates



(g) Average recall performance comparison across various batch sizes



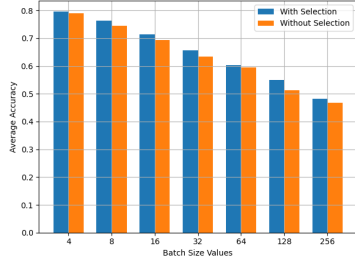
(h) Average recall performance comparison across various local epochs



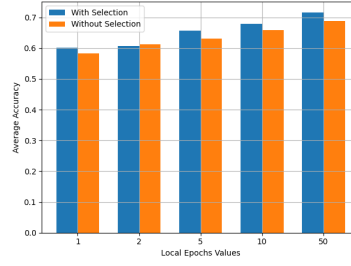
(i) Average recall performance comparison across various learning rates

Figure 5.1: Average performance results for one federated training round

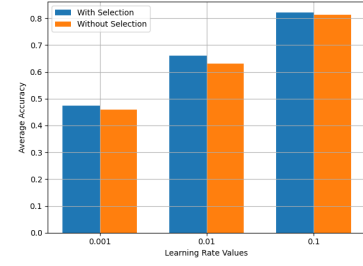
The first result analyzed is the confusion matrix for the classification model, presented in Figure 5.5. A confusion matrix is a table that allows evaluating the performance of a classification model. It is composed of cells that represent the different combinations of predicted and actual classifications. In 5.5, the rows represent the actual or expected classes, while the columns represent the classes predicted by the model based on local smartphone sensors. It compares the predicted classifications by the model with the actual classifications to determine the accuracy and errors rate in each class. Darker or lighter areas can indicate different patterns of model performance. Darker areas can represent cells with higher values, indicating a high frequency of correct predictions (true positives) or errors (false positives or false negatives). This can be interpreted as a higher confidence of the model in relation to these classes. Lighter areas can represent cells with values



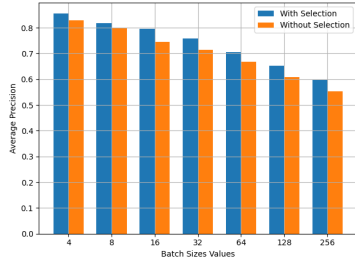
(a) Average accuracy performance comparison across various batch sizes



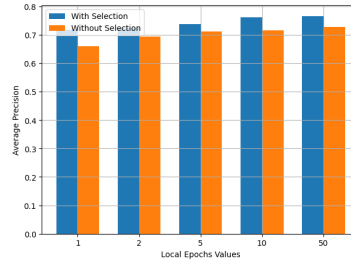
(b) Average accuracy performance comparison across various local epochs



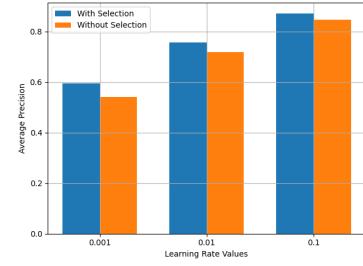
(c) Average accuracy performance comparison across various learning rates



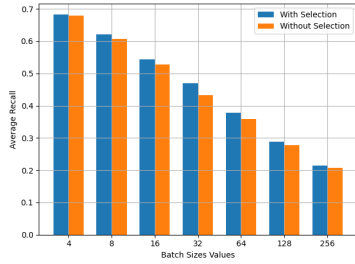
(d) Average precision performance comparison across various batch sizes



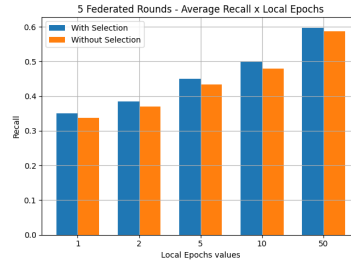
(e) Average precision performance comparison across various local epochs



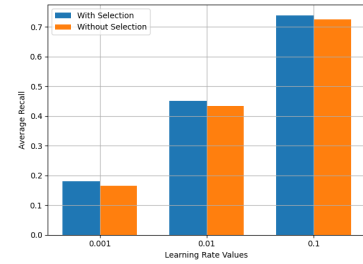
(f) Average precision performance comparison across various learning rates



(g) Average recall performance comparison across various batch sizes



(h) Average recall performance comparison across various local epochs

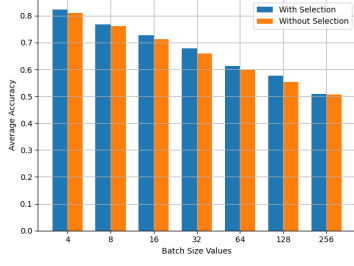


(i) Average recall performance comparison across various learning rates

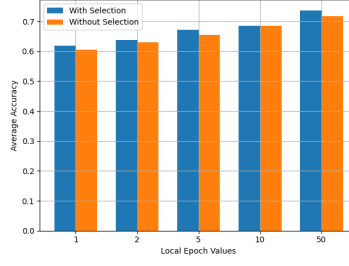
Figure 5.2: Average performance results for five rounds of federated training

equal to or smaller than zero, indicating that the model did not make incorrect predictions for these classes. This can be interpreted as a perfect performance of the model in these classes.

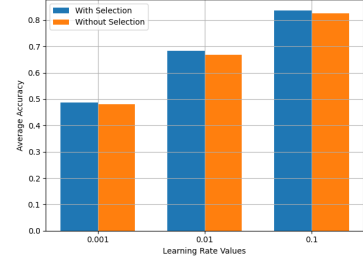
In Figure 5.5, it can be seen that in general the algorithm had a good performance classifying the transportation modes based on smartphone sensors. Almost all transportation modes were correctly predicted by the algorithm excepted for the Train mode that achieved the worst performance. According to the Figure, it appears that the algorithm had difficulty to distinguish Train, Bus and Car modes maybe because these transportation modes may have similar measurements, requiring more volume of data to better train the algorithm to distinguish each other.



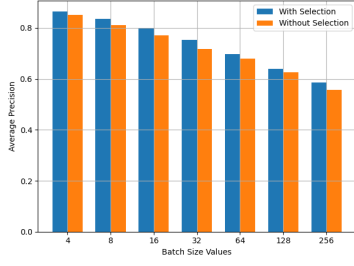
(a) Average accuracy performance comparison across various batch sizes



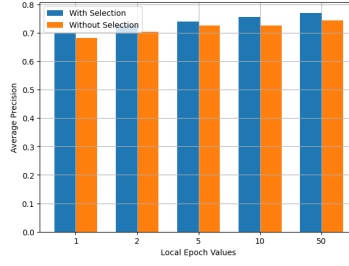
(b) Average accuracy performance comparison across various local epochs



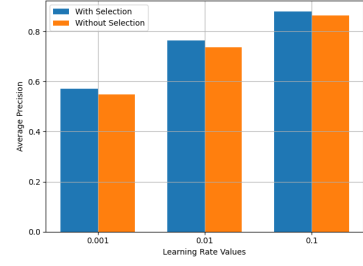
(c) Average accuracy performance comparison across various learning rates



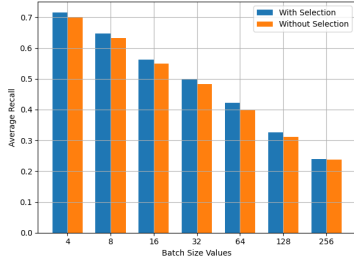
(d) Average precision performance comparison across various batch sizes



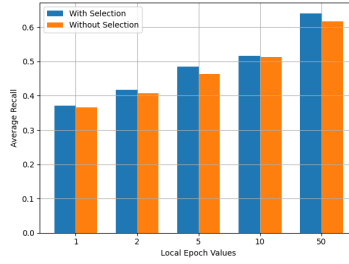
(e) Average precision performance comparison across various local epochs



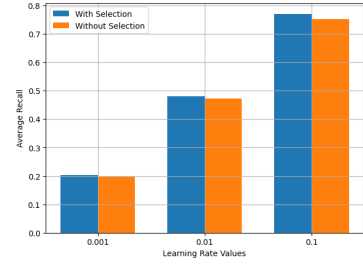
(f) Average precision performance comparison across various learning rates



(g) Average recall performance comparison across various batch sizes



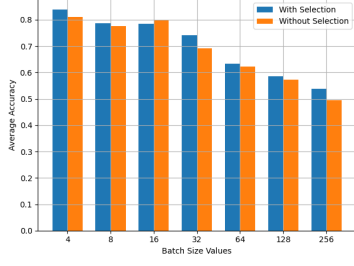
(h) Average recall performance comparison across various local epochs



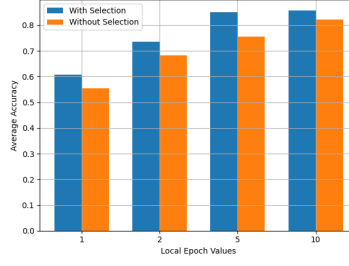
(i) Average recall performance comparison across various learning rates

Figure 5.3: Average performance results for ten rounds of federated training

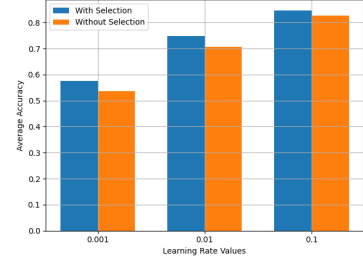
The Table 5.3 presents the average metric performance achieved by the the federated algorithm collaboratively trained across selected clients with their local datasets. The best average performance was 93.89% of accuracy when the client selection is applied and 93.69% when it is not. The average performance for the precision and recall metrics also confirms the result obtained, with the best average achieving 94.11% and 93.67%, respectively. These performances were achieved using a light set of configurations (i.e., batch size of 4, learning rate of 0.1, and local epochs of 10). To achieve this high score it required 50 federated training rounds, which implies an estimated communication costs of 537.60 Kbits. However, despite the best result required 50 federated rounds, the communication costs were smaller than the 13.70 Mbits estimated as communication costs for the centralized experiments, presented in Table 5.2.



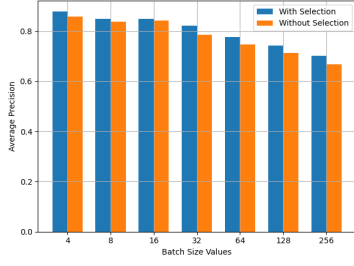
(a) Average accuracy performance comparison across various batch sizes



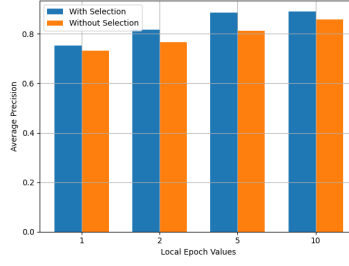
(b) Average accuracy performance comparison across various local epochs



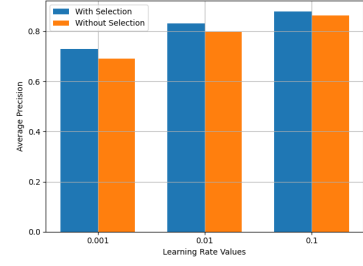
(c) Average accuracy performance comparison across various learning rates



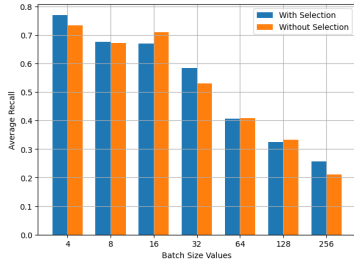
(d) Average precision performance comparison across various batch sizes



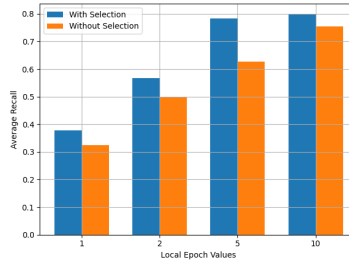
(e) Average precision performance comparison across various local epochs



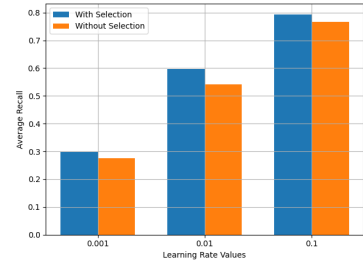
(f) Average precision performance comparison across various learning rates



(g) Average recall performance comparison across various batch sizes



(h) Average recall performance comparison across various local epochs



(i) Average recall performance comparison across various learning rates

Figure 5.4: Average performance results for fifty rounds of federated training

In Figure 5.1, the comparison of average performance results for simulations is shown, with and without the client selection, across different batch sizes, learning rates, and local epoch values for accuracy, precision, and recall during one federated round.

Figures 5.1(a), 5.1(d), and 5.1(g) show the average performance comparison across different batch sizes in terms of accuracy, precision, and recall. Our client selection had better average performance for almost all batch sizes, achieving the best average with size 4 and the worst with 256. We believe that with bigger batch sizes, the model has more difficulty learning patterns from the training dataset, leading to overfitting. It was expected that the biggest value of 256 achieved the worst average performance, given that we are not dealing with large datasets in our simulations.

Federated Rounds	Accuracy(%)		Precision(%)		Recall(%)	
	w/	w/o	w/	w/o	w/	w/o
1	92.38	91.94	93.33	92.72	91.74	91.12
5	90.32	91.96	91.01	91.75	89.67	90.38
10	91.04	91.42	91.54	91.94	90.57	90.99
50	93.89	93.69	94.11	93.94	93.67	93.95

Table 5.3: Comparison of average performance metrics with and without client selection in the best performing configuration

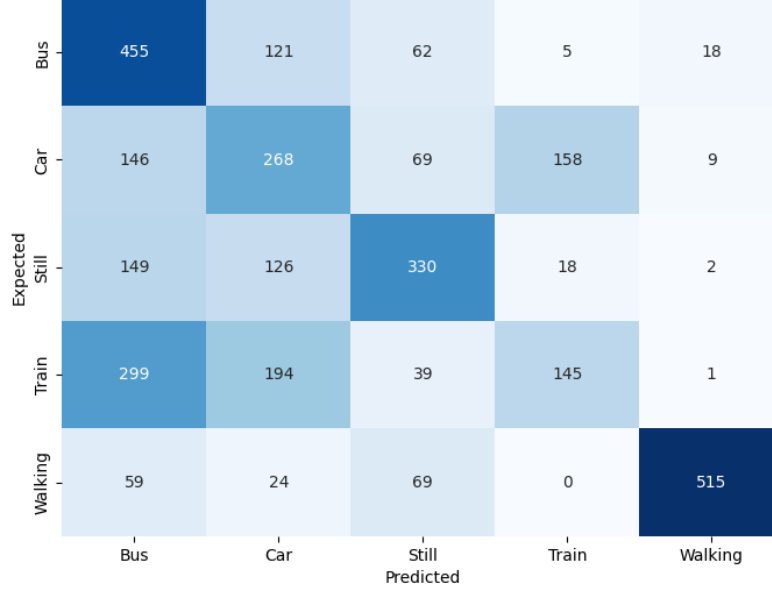


Figure 5.5: Confusion Matrix

In Figures 5.1(b), (e), and (h), the average performances for each local epoch value are compared. Our federated proposal achieved better performance when the local epoch was 50. This indicates the model is able to learn more when it has more iterations with local data.

In Figure 5.1(c), (f), and (i), the average performances across learning rate values are compared. Differences between the averages are noted. The precision average performances indicate better performance when selection is applied, despite almost none difference is noted when learning rate is 0.001. The recall average performance comparison demonstrates better performance when the selection strategy is used, with the higher difference when learning rate is 0.01. These results indicate that one federated round is not enough to distinguish the better learning rate value.

Figure 5.2 shows the average performance results for five federated training rounds. The presentation order follows Figure 5.1. The average performance comparisons across various batch sizes, local epochs, and learning rates are presented. In all cases, applying

client selection resulted in better average performance compared when no client selection is applied.

In Figures 5.2(a), (d), and (g) we present the average performance comparison across batch size. Our client selection demonstrated the best average performance in all cases, achieving almost 80% of accuracy, and more than 80% of precision when batch size was set to 4.

Figure 5.2(b), 5.2(e), and 5.2(h) show the average performance comparison for local epoch values. Our selection strategy outperforms all metrics for all local epoch values. Compared to Figure 5.1, the results in Figure 5.2 indicate better performance of our federated proposal. An interesting behavior is observed in accuracy average performance for local epochs 2, which no client selection performs better than when client selection is used.

In Figures 5.2(c), 5.2(f), and 5.2(i), we present a comparison of the average performances across various learning rate values for five rounds of federated training. Our client selection proposal achieves the best average results for all learning rate values analyzed, with a culmination average measurement at a learning rate of 0.1, the highest value compared to the others, achieving over 80% average accuracy and precision performance, with a recall score of over 70%.

Following the discussion about the simulation outcomes, Figure 5.3 summarizes the average performance results for simulations with ten federated rounds. In Figures 5.3(a), 5.3(d) and 5.3(g), the average results for batch size values are summarized. From the figures, it can be seen that the batch size of 4 achieved the best average performance, achieving more than 85% of average accuracy, precision, and recall performance metrics. We believe that is because the value of 4 perfectly matched with average local dataset size and speed up the finding for the local minimum.

Following with the ten federated rounds analysis, Figures 5.3(b), 5.3(e) and 5.3(h) show the average performance metrics for local epochs. Our client selection again outperformed the comparison with no selection, with a greater margin this time, particularly with local epochs 50.

Figures 5.3(c), (f), and (i) show that our client selection has a better average performance comparing across the learning rate values, especially with the value of 0.1 which achieved an score of more than 80% of average accuracy, precision and recall.

Figure 5.4 summarizes the average performance results for simulations configured

with fifty federated rounds. Figures 5.4(a), 5.4(d), and 5.4(g) show the average performance results, and although the best average accuracy performance was achieved by our proposal when the batch size is 4, we can note the good average performance was achieved when the batch size is 16. This is an interesting fact to notice because when the batch size is 16, the federated models can learn better patterns from all clients than when only selected clients participate in federated learning. Our client selection cannot select a representative set of clients when the batch size is 16, not enabling the model to learn patterns to achieve better average performance. This can be confirmed by taking a look at Figure 5.4(g), in which the average recall performance for batch size 16 also outperformed our federated proposal.

In Figures 5.4(b), (e), and (h), the average performance metrics are compared across local epoch values. Our client selection proposal had the best average performance for all values, with a larger difference between the results than in Figure 5.3. Notably, in Figure 5.4(h) at local epoch 5, our selection proposal had significantly higher average recall performance compared to no selection in federated training rounds. We can also note a very close similarity between the average performance with the values 5 and 10, leading to realize that the value 5 is the more appropriate giving to consume less local computational resources to achieve almost the same average performance.

In Figures 5.4(c), (f), and (i), the average performance comparison for different learning rate values during fifty federated training rounds is shown. It can be observed that our client selection had the best average performances, especially when the learning rate was 0.1, achieving more than 80% of average accuracy and precision performances. For fifty federated training rounds, better average performance for learning rate values was expected, which indicates that higher values can achieve better average performances.

5.4 FedTMD Communication Costs Comparison

In this section, the estimations for the communication costs are presented and the comparison between centralized and federated results are analyzed.

5.4.1 Results Obtained

Federated Rounds	Communication Costs	
	with selection	without selection
1	10.75 Kbits	14.33 Kbits
5	53.76 Kbits	71.68 Kbits
10	107.52 Kbits	143.36 Kbits
50	537.60 Kbits	716.80 Kbits

Table 5.4: Communication costs results with and without the client selection.

5.4.2 Analysis

Table 5.4 represents the comparison between communication costs with and without the client selection being applied in federated training. The values in the first column represent the total of federated training rounds used in each experiment, i.e. 1, 5, 10, 50. The second column represents the costs of communication with the client selection, which represent the best results. It indicates that communication with client selection is less costly in all comparisons, achieving the lowest value of 10.75 Kbits for one federated training round and 537.60 Kbits for fifty federated training rounds. The third column represents the cost of communication without the client selection, which represents the highest values obtained. Therefore, the results show that applying the client selection FedTMD reduces the communication costs in the federated learning training, while the absence of client selection causes federated training to result in higher costs.

In summary, the tables and figures presented in this chapter demonstrated the importance of incorporating the client selection strategy. Increasing the number of local epochs improves the average performance of the federated model, achieving the best average performance with 50 local epochs in most of the cases. The best average performance achieved was 93.89% of accuracy and 94.11% of precision with 93.67% of recall. The high average performance in recall metric indicates that the model can correctly identify a large number of positive examples concerning the total number of positive examples present in the data distributed among the devices of the participating users. This means that the model is managing to learn well from the decentralized data and that the communication between the devices is taking place efficiently. Communication costs with the client selection proposal applied achieved 10.75 Kbits for 1 federated training round and 537.60 Kbits for 50 federated training rounds. The comparison between the communication costs with and without the client selection showed that the proposal can save more than 30% of the communication costs.

6. Conclusions and Future Directions

In this chapter, the main conclusions and the future directions of this research are summarized.

This study addressed two research problems: the user data privacy concerns in transportation mode detection domain, and the communication costs in federated learning systems as a consequence of data heterogeneity. To answer the problems, we propose FedTMD, a federated learning architecture to classify the user transportation mode which also implements a client selection strategy to sample clients based on the quantity of transportation modes present in their local dataset. Clients with multiple transportation modes in their dataset provide more variability of labels for the training step, leading to improve classification performance when facing new unclassified instances. The proposal trains a supervised machine learning algorithm with a real-world smartphone sensors dataset applying Federated Learning paradigms to preserve user data privacy and select a sub-set of clients to mitigate the communication costs during federated training. Our approach of using the number of unique transportation modes as the criterion to select client offers a simple yet effective solution. Experimental results demonstrated that FedTMD architecture is able to classify the transportation mode up to 90% of accuracy, precision and recall, and it also minimizes the communication costs from 13.08 Mbits of centralized solution to 564.0 Kbits of the federated proposal.

Future researches can improve the client selection criteria to evaluate the quality of local datasets (i.e., data reliability, distribution, diversity). The use of Self-Supervised Learning to deal with unlabeled data in Federated Learning systems can also be evaluated in future directions. Future investigations can also implement data augmentation techniques to leverage datasets used in simulations maintaining the same distributions. Additionally, comparing the total execution time consumed can also be an improvement to analyze possible performance trade-offs.

Bibliography

- [1] Abdullatif Albaseer, Bekir Sait Ciftler, Mohamed Abdallah, and Ala Al-Fuqaha. Exploiting unlabeled data in smart cities using federated edge learning. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1666–1671. IEEE, 2020.
- [2] Luca Bedogni, Marco Di Felice, and Luciano Bononi. Context-aware android applications through transportation mode detection techniques. *Wireless communications and mobile computing*, 16(16):2523–2541, 2016.
- [3] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.
- [4] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchu Qiu, Titouan Parcollet, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.
- [5] Brazil. Lei nº 13.709, de 14 de agosto de 2018. lei geral de proteção de dados pessoais (lgpd). *Diário Oficial [da] República Federativa do Brasil*, 2018.
- [6] Claudia Carpineti, Vincenzo Lomonaco, Luca Bedogni, Marco Di Felice, and Luciano Bononi. Custom dual transportation mode detection by smartphone devices exploiting sensor diversity. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 367–372. IEEE, 2018.
- [7] Francesco Delli Priscoli, Alessandro Giuseppe, and Federico Lisi. Automatic transportation mode recognition on smartphone data based on deep neural networks. *Sensors*, 20(24):7228, 2020.
- [8] Xiaoheng Deng, Leilei Wang, Jinsong Gui, Ping Jiang, Xuechen Chen, Feng Zeng, and Shaohua Wan. A review of 6g autonomous intelligent transportation systems:

- Mechanisms, applications and challenges. *Journal of Systems Architecture*, page 102929, 2023.
- [9] Zhaoyang Du, Celimuge Wu, Tsutomu Yoshinaga, Kok-Lim Alvin Yau, Yusheng Ji, and Jie Li. Federated learning for vehicular internet of things: Recent advances and open issues. *IEEE Open Journal of the Computer Society*, 1:45–61, 2020.
 - [10] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council.
 - [11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
 - [12] Wei Huang, Tianrui Li, Dexian Wang, Shengdong Du, Junbo Zhang, and Tianqiang Huang. Fairness and accuracy in horizontal federated learning. *Information Sciences*, 589:170–185, 2022.
 - [13] JQ James. Semi-supervised deep ensemble learning for travel mode identification. *Transportation Research Part C: Emerging Technologies*, 112:120–135, 2020.
 - [14] JQ James. Travel mode identification with gps trajectories using wavelet transform and deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):1093–1103, 2020.
 - [15] Jean-Yves Le Boudec. *Performance Evaluation of Computer and Communication Systems*. EPFL Press, Lausanne, Switzerland, 2010.
 - [16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
 - [17] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.
 - [18] Yi Liu, Xingliang Yuan, Ruihui Zhao, Yifeng Zheng, and Yefeng Zheng. Rc-ssfl: Towards robust and communication-efficient semi-supervised federated learning system. *arXiv preprint arXiv:2012.04432*, 2020.
 - [19] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

- [20] Hugues Moreau, Andrea Vassilev, and Liming Chen. The devil is in the details: An efficient convolutional neural network for transport mode detection. *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [21] Paria Sadeghian, Xiaoyun Zhao, Arman Golshan, and Johan Håkansson. A stepwise methodology for transport mode detection in gps tracking data. *Travel Behaviour and Society*, 26:159–167, 2022.
- [22] Elton F de S Soares, Carlos Alberto V Campos, and Sidney C de Lucena. Online travel mode detection method using automated machine learning and feature engineering. *Future Generation Computer Systems*, 101:1201–1212, 2019.
- [23] Elton F de S Soares, Hojjat Salehinejad, Carlos Alberto V Campos, and Shahrokh Valaee. Recurrent neural networks for online travel mode detection. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.
- [24] Xiaozhuang Song, Christos Markos, and JQ James. Multimix: A multi-task deep learning approach for travel mode identification with few gps data. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.
- [25] Afaf Taik, Zoubeir Mlika, and Soumaya Cherkaoui. Clustered vehicular federated learning: Process and optimization. *arXiv preprint arXiv:2201.11271*, 2022.
- [26] Yuanshao Zhu, Yi Liu, JQ James, and Xingliang Yuan. Semi-supervised federated learning for travel mode identification from gps trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [27] Yuanshao Zhu, Christos Markos, Ruihui Zhao, Yefeng Zheng, and JQ James. Fedova: One-vs-all training method for federated learning with non-iid data. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2021.
- [28] Yuanshao Zhu, Shuyu Zhang, Yi Liu, Dusit Niyato, and JQ James. Robust federated learning approach for travel mode identification from non-iid gps trajectories. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 585–592. IEEE, 2020.