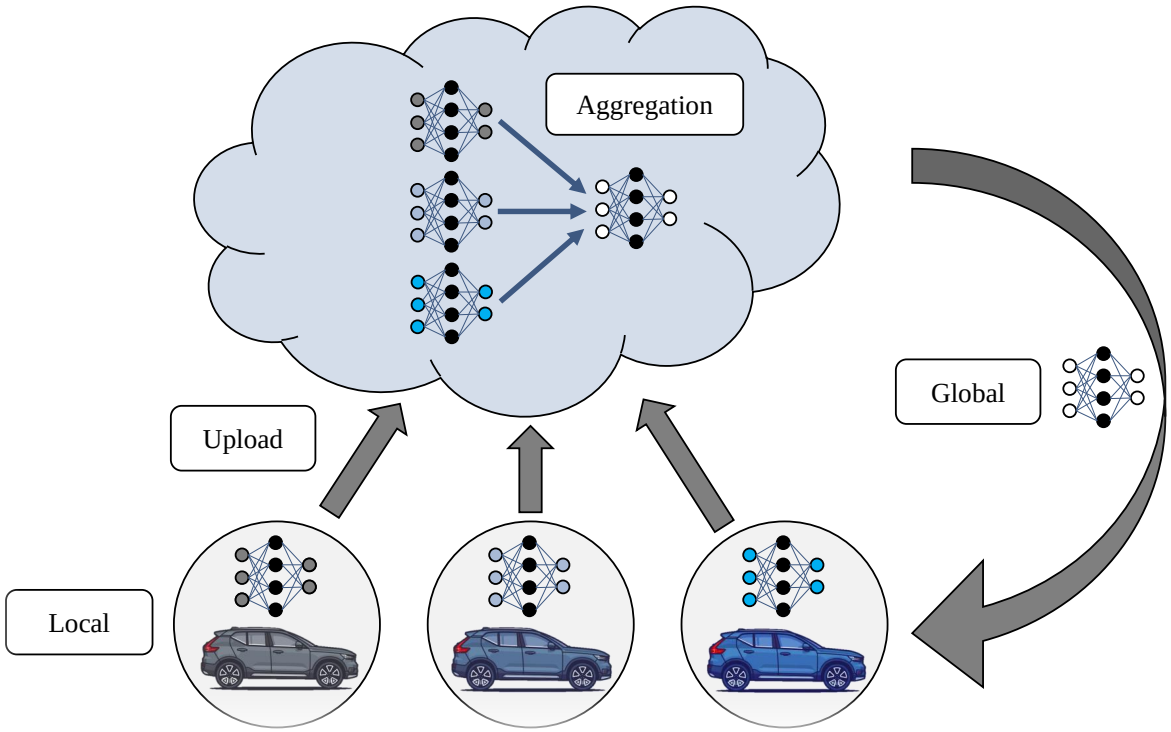
CHALMERS
UNIVERSITY OF TECHNOLOGY



Trajectory Prediction for Automotive Applications using Federated Learning

Master's thesis in Complex Adaptive Systems

HANNES JOHANSSON
DANIEL OLANDER

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Trajectory Prediction for Automotive Applications using Federated Learning

HANNES JOHANSSON
DANIEL OLANDER

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2023

Trajectory Prediction for Automotive Applications using Federated Learning

HANNES JOHANSSON
DANIEL OLANDER

Supervisor: Koen Vellenga, Analytics & AI at R&D, Volvo Cars
Examiner: Mats Granath, Department of Physics

Master's thesis 2023
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of the federated learning process applied to a fleet of cars.

Typeset in LaTeX
Printing /Department of Physics
Gothenburg, Sweden 2023

Abstract

Advanced Driver Assistance Systems (ADAS) and Autonomous Driving Systems (ADS) increasingly rely on Deep Learning (DL) models. While DL models achieve state-of-the-art performance for a variety of tasks, they are not robust across a wide range of traffic scenarios, require large quantities of continuously collected data, and must follow safety and privacy regulations. Federated learning (FL) enables the training of DL models to be done locally at each client (vehicle), sharing and aggregating the trained models while keeping the data local. This allows previously unreachable data to be harnessed for improved performance. Here, performance refers to the maximum reached accuracy. FL also enables models to be trained across regions, limiting sensitive data sharing while training on diverse datasets. This master thesis evaluates the performance of FL for trajectory prediction (a central part of ADAS and ADS) compared to a centralized learning (CL) approach. An FL framework was implemented and validated through classification experiments on the MNIST dataset using a convolutional neural network (CNN). The performance of a CL setup (one client) was used as a benchmark, achieving a validation accuracy of approximately 99 %. Results show that FL with multiple clients requires more training to converge but eventually saturates at a similar level of performance. Training on independent and identically distributed (IID) data yielded the best performance, while non-IID data introduced more noise and overall lower performance for the FL approach. Selecting a smaller fraction of clients each round (client fraction) corresponded to lower performance when evaluating non-IID data, while speeding up training due to processing fewer data samples each round. To test the effects of FL algorithms on trajectory prediction performance, the nuScenes dataset, a collection of data from vehicles driving in Boston and Singapore, was used. The data was transformed into 2D bird's eye-view (BEV) images and fed to CoverNet, a trajectory prediction model based on the residual network ResNet-50 which applies classification over a set of trajectories. The experiments included varying federated optimization algorithms [FedAvg, FedAvgM, FedProx], number of clients, client fraction, data distribution techniques to test IID/non-IID data, and a direct comparison of FL to CL. The federated optimization algorithms had no notable impact on the results. More clients resulted in a slower convergence rate but similar maximum performance to the benchmark CL setup, close to the results of the original CoverNet publication. Reducing the client fraction resulted in faster training and, contrary to the MNIST results, no notable performance or convergence rate difference. FL performed similarly to CL on both IID and non-IID data. Simulating the case where FL unlocks data from both Boston and Singapore to be used showed substantially improved performance, compared to CL using local data from only one city. Using FL and half the data from each city also showed improved performance over CL, displaying the importance of the data diversity FL can enable.

Keywords: Advanced Driver Assistance Systems, Autonomous Driving Systems, Federated Learning, Machine Learning, Trajectory Prediction

Acknowledgements

We would like to express our deepest gratitude to our supervisor, Koen Vellenga, who is currently pursuing his industrial Ph.D. at Volvo Cars. His help and guidance throughout this master's thesis have been instrumental in our progress. Koen's great knowledge, support, and insightful feedback have consistently motivated us and improved our understanding of the subject matter. Additionally, we would like to thank our examiner Mats Granath and Alicia Rey Alonso for being our opponent and teaching us Spanish.

We would also like to thank Volvo Cars, especially the R&D Analytics & AI group, for the great opportunity to carry out this master's thesis. Lastly, our families' support during our education at Chalmers has helped us get to where we are today. Thank you!

Hannes & Daniel, Gothenburg, June 2023

Contents

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ADAS	Advanced Driver Assistance Systems
ADS	Autonomous Driving Systems
AI	Artificial Intelligence
ANN	Artificial Neural Network
BEV	Bird's Eye View
CL	Centralized Learning
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
FL	Federated Learning
IID	Independent and Identically Distributed
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent

List of Figures

List of Tables

List of Algorithms

1
Introduction

The rapid advancements in artificial intelligence (AI), machine learning (ML), and deep learning (DL) have impacted various industries, showcasing promising results in, for example, natural language processing, visual data processing, and speech and audio processing [1]. Another domain where such advancements have significant effects is autonomous driving, where integrating DL techniques can enhance safety features [2]. Developing efficient safety features for autonomous driving applications relies on accurate predictions of vehicle trajectories. This thesis aims to investigate the performance of a trajectory prediction model when applying a strategy for decentralized learning, namely federated learning (FL).

Safety systems in vehicles include both passive and active systems [3]. Passive safety includes seatbelts, airbags, and crumple zones. Active safety, more prominent in recent years, includes a wide spectrum of capabilities. This ranges from advanced driver-assistance systems (ADAS) functionalities, such as adaptive cruise control, lane-keeping assist, and automatic emergency braking to fully autonomous driving systems (ADS) [4]. Components of an ADAS or ADS depend on accurate trajectory prediction and environmental awareness to safely navigate through complex traffic scenarios, including diverse driving conditions. However, ensuring the reliable performance of these active systems is an ongoing challenge, due to dynamic and unpredictable real-world driving environments [5].

The performance and efficiency of ADAS and ADS are indicated by their ability to generalize and perform consistently in different scenarios (robustness). In addition, they need to meet increasingly stricter safety and privacy requirements from governing bodies in the coming years [6] [7]. To enable continued safe deployment of AI implementations in the safety systems, there is a need for sustained improvements through continuous updates (continuous learning) [8]. A challenge in this process is to collect data from diverse traffic scenarios, a vital aspect to ensure robustness and reliability.

The active safety capabilities of ADAS and ADS rely on data collected from a variety of sensors, including cameras, LiDAR, RADAR, and GPS. The sensors capture information about the vehicle's surroundings and interior, including other road users, pedestrians, and objects, enabling the vehicle to build an understanding of its environment. Newly produced cars on the road often have various sensors installed. If utilized, the recorded sensor data could prove to be a valuable asset over time by, for example, improving ADAS and ADS functionalities. Currently, car manufacturers

use specific fleets of test vehicles to capture the data used for training ADAS and ADS. Hence, the opportunity for improvement could be substantial if more data were to be used. It is not currently possible to collect all data from a fleet of customer cars. Accessing and processing this vast amount of data to represent diverse driving scenarios require costly data transfers and lead to additional data privacy overhead [9].

Ensuring data privacy, addressing the limitations of data collection, and keeping compliance with increasingly strict regulations are notable challenges in the development of active safety systems. Federated learning (FL), presented in 2015 [10][11], is a promising solution to address these challenges. FL provides a decentralized framework for training ML models. It allows data from distributed clients, such as vehicles or servers, to be used during training without the need for moving the local data from the clients. In the standard centralized learning (CL) approach, the client data is transferred to a central server for processing. FL offers several potential benefits, such as improved data privacy, reduced communication costs of sharing large quantities of data, and the ability to enable previously unreachable data from diverse and geographically distributed clients to improve model robustness and generalization.

Many studies have been done evaluating FL, including automotive applications. Examples include steering wheel angle prediction [12] and object detection [13]. However, no other studies evaluated the influence of FL on the performance of road user trajectory prediction methods learning from different geographical locations. Therefore, to contribute to the ongoing research to improve the safety and robustness of ADAS and ADS, this thesis evaluates the performance and generalization effects of applying FL for trajectory prediction tasks on data from different geographical regions.

1.1 Aim, scope and limitations

The main aim of this study is to conduct an empirical performance evaluation of FL compared to CL in a trajectory prediction task. This is done by answering the following questions:

1. Influence of training ML models in a federated way in the context of ADAS/ADS functionalities:
   - What are the effects of FL compared to CL training methods on trajectory prediction models?
   - What are the effects of sampling data from different geographical regions?

2. Generalization of FL: How is the performance and convergence rate on validation data from datasets A and B affected when trained by:
   - FL using datasets A and B?
   - CL using dataset A or B?
   - CL using datasets A and B?

The scope and limitations of the study include the following assumptions:

(1) We assume that each vehicle can detect and obtain annotations of all surrounding objects by using a pre-trained object detection model. We will therefore not consider self-labeling.

(2) The goal is not to find which model architecture yields the best performance. Instead, we will stick to a single underlying model architecture for a specific task to evaluate the effect of FL on validation performance.

(3) The available computing power further limits the scope of the work. Each experiment was done on a single machine, simulating a server and client structure.

1.2 Thesis outline

In Chapter 2 the underlying background and theory of deep neural networks, trajectory prediction, and FL is outlined. Chapter 3 covers the methodology of the study. The FL implementation validation experiments are explained, using the MNIST dataset. This is followed by a trajectory prediction section, introducing the nuScenes dataset and the specific experiments conducted following the aim of this study. The results are presented in Chapter 4. The first section includes the results from the implementation validation, based on the MNIST dataset. The second section presents the results from the trajectory prediction experiments on the nuScenes dataset. The results are discussed in Chapter 5. The key findings and implications are discussed, including limitations and challenges. A section focused on ethical considerations is included at the end. Chapter 6 is a summary of the study and presents possibilities for future studies following our work.

2
Theory

The theory chapter provides the necessary background and theoretical foundation for understanding the relevance and implications of this study. By presenting a review of the relevant concepts related to deep neural networks (DNNs) and FL, this chapter prepares the reader with the knowledge required for the following chapters. In this chapter, we first discuss DNNs, starting with an overview of the basics and how DNNs are trained and optimized. This is followed by presenting two types of DNNs, convolutional neural networks (CNNs) and residual neural networks, and DNN applications within trajectory prediction. Thereafter, FL is thoroughly introduced to the reader, covering the high-level concept of FL, edge computing, privacy and security, data distribution techniques, and FL optimization algorithms.

2.1 Deep neural networks

Artificial Neural Networks (ANNs) are mathematical models inspired by biological neural networks. They perform complex computations based on behavior learned from patterns of data and enable progressive learning by taking biological neurons as inspiration. Here, the idea of connections is replicated by connections in the form of weights and biases. The term deep neural networks (DNNs) is given to an artificial network with at least two hidden layers [14]. There are various types of DNNs, but they all follow the structure of containing neurons, weights (weighted connections between neurons), biases (thresholds), and activation functions. The flexibility and vast capability of DNNs allow them to be used in a wide range of applications, in various domains. This section provides an introduction to DNNs, explains how the parameters are trained and optimized, and present relevant types of DNNs (convolutional neural networks (CNNs) and residual neural networks).

2.1.1 The basics

DNNs are referred to as ”deep” since they have multiple hidden layers between their input layer and output layer, as opposed to ”shallow” networks with at most one hidden layer. The presence of these extra layers enables composition of features from lower layers, thereby modeling complex data with a lower amount of parameters/units than shallow networks requires [15]. What is learned in these layers depends on the specific network type and task. DNNs leverage layers to learn different levels of information about the data. Each layer learns something new and more abstract than the previous layer, helping the network understand the data better. An ex-

ample is image recognition models, where the first layer can learn simple patterns such as edges, and the second layer learns combinations of edges forming patterns. Progressing deeper means more complex, high-level features. DNNs are capable to learn complex non-linear relationships from different input data formats (e.g., images, sequences, audio) through the use of non-linear activation functions, such as the sigmoid function, the hyperbolic tangent (tanh) function, and the Rectified Linear Unit (ReLU) function.

A common type of DNN is feed-forward networks. These consist of layers of interconnected neurons, each receiving input from neurons in the previous layer, thus only feeding the information forward in the network. Figure 2.1 illustrates the structure of a classic feed-forward network, displaying the multiple hidden layers between the input and output layers.

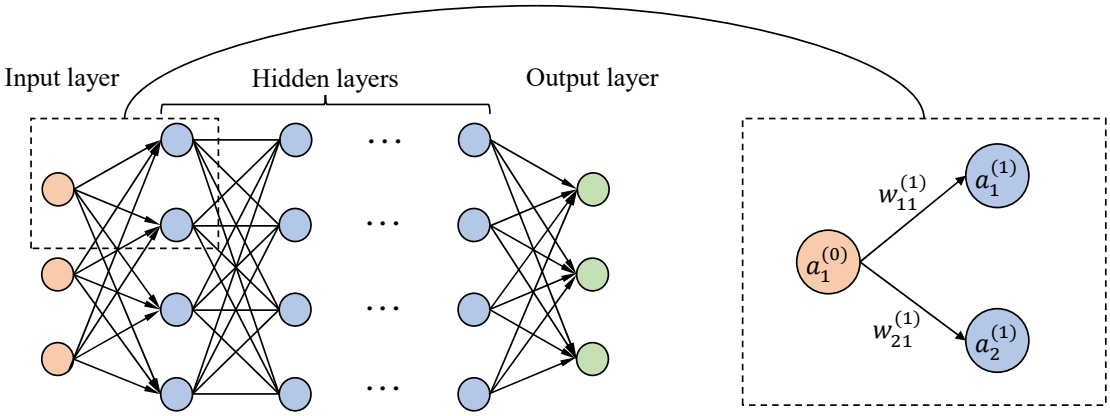


Figure 2.1: Illustration of the structure of a feed-forward DNN.

Figure 2.1 highlights some neurons and their interconnecting weights. Mathematically, the output of a neuron i in layer l can be expressed as:

$$a_i^{(l)} = f\left(\sum_{j=1}^{N} w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)}\right), \tag{2.1}$$

where:
$a_j^{(l-1)}$ is the output of neuron j in the previous layer $(l-1)$,
f is the activation function,
$w_{ij}^{(l)}$ is the weight connecting neuron j in layer $(l-1)$ to neuron i in layer l, and
$b_i^{(l)}$ is the bias term for neuron i in layer l.

2.1.2 Training and optimization

In order to learn, a DNN needs to be trained. During training, the parameters of the network (the weights and biases) are updated. The training process, where the

model is tuned by using training data, is done is multiple steps. The training data consist of both model input and the corresponding correct values/labels (ground truth). Training a DNN involves updating the weight and biases w to minimize a chosen loss function F, which is a function that quantifies the difference between the network's prediction (output) and the ground truth. The choice of loss function depends on the type of task the model is trying to learn and the format of this. Some common loss functions are mean squared error (MSE), likelihood loss, and log loss (cross-entropy loss). The latter two are often used in classification problems, where the model should classify the input among a given set of classes.

Training starts with a forward pass, where the input is propagated through the network, layer by layer. The error (or loss) is calculated by applying the loss function to the produced output and the ground truth. This loss is then used for a backward pass, so-called backpropagation, where the error is propagated backward through the network, done layer-wise but now starting from the output layer. During the backpropagation, the derivative (gradient) of the loss with respect to each weight in the network, $\frac{\partial F}{\partial w}$, is calculated. This measures how each small parameter change would affect the overall error. The calculated gradients are then used by an optimization algorithm to adjust the weights, as

$$w \leftarrow w - L_r \frac{\partial F}{\partial w},$$

where L_r is the decided learning rate, affecting the magnitude of the parameter updates. Most optimizations algorithms are based on stochastic gradient descent (SGD) and variants of this, for example, Adam [16] and RMSprop [17]. By iteratively repeating this process, often in batches of data in-between each adjustment of the parameters, the loss function is minimized. Performing backpropagation over the entire available dataset is referred to as training one epoch. Normally, the dataset is recycled and used for training a model many epochs.

When training a network, the available data is typically split between a training set and a validation set. Only the training set is used in the previously described training process. The validation set is reserved to use for measuring the model's performance on unseen data, that is, data that the model has not been trained on. This process is called evaluation. One challenge when training an ANN is overfitting, the phenomenon that the model learns too specific features of the particular training data rather than learning more useful, generalizable patterns. Overfitting can lead to a model that is performing well on the training data used for backpropagation, but showing low performance on the validation data reserved for testing only. However, there are regularization techniques to combat overfitting. Dropout [18] and weight decay are often employed to prevent overfitting and thus improve the generalization of DNNs. The concept of dropout is to randomly temporarily deactivate a certain percentage of neurons during training, reducing excessive interdependencies. Weight decay penalizes large weights and thereby encourages the network to learn simpler functions, making it less likely to overfit. Apart from iteratively training the network parameters (weights and biases) over epochs, optimization can also include changing hyperparameters that govern the training process and the structure

of an ML model. Examples of hyperparameters are (structural) numbers of layers, neurons in each layer, (training-affecting) batch size, learning rate of the parameter tuning and learning momentum.

2.1.3 Convolutional neural networks

A CNN is a type of ANN designed for processing structured grid structured data such as images. CNNs excel in vision tasks such as object detection, classification, and image segmentation, the concept of partitioning an image into image segments or regions. They employ convolution, a sliding operation that uses convolutional filters (so-called kernels) and element-wise multiplications to detect local features and patterns in the input [19]. Each convolutional filter slides over the input grid, applying the same weights for each sub-region (input patch) of the data. By repeating for all filters, the CNNs produce feature maps of the input. This makes CNNs particularly effective at detecting local patterns and spatial hierarchies in the input data. An illustration of how a kernel is applied over input data is presented in Figure 2.2.

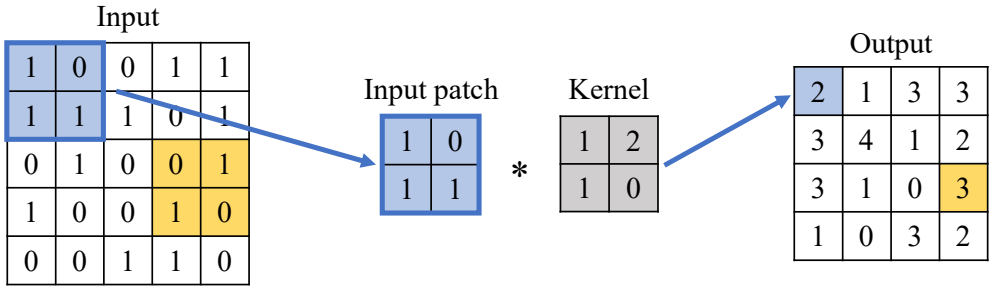


Figure 2.2: A kernel of size 2x2 is applied to an input of size 5x5. Sliding the kernel over the input and repeatedly applying element-wise multiplications results in the output matrix of size 4x4. Two areas in the input matrix, and the resulting values in the output matrix, are highlighted. Each input patch (2x2 sub-region of the input matrix) is element-wise multiplied with the kernel to produce the sums in the output matrix.

CNNs are usually constructed of multiple convolutional layers, each consisting of several filters, alternated with pooling layers that have a down-sampling effect. The first convolutional layers extract smaller features from the input, that are aggregated into larger patterns in later convolutional layers. CNNs can use padding, the concept of adding extra border pixels to the input images, to preserve the input dimension throughout convolution operations, if desired. After the sequence of convolutional and pooling layers, the resulting output is normally flattened and connected to fully-connected layers, where the information gained through the convolutions is used to solve the problem-specific task.

2.1.4 Residual neural networks

A residual neural network (often referred to as residual network or ResNet) is a DNN model that incorporates residual connections. These are identity mappings, allowing the input of a residual block a shortcut to the end of the block, where it is merged with the regular block output via addition [20]. The structure is illustrated in Figure 2.3. The residual connections are also known as skip connections since they allow the input to "skip" the block that the residual connection passes.

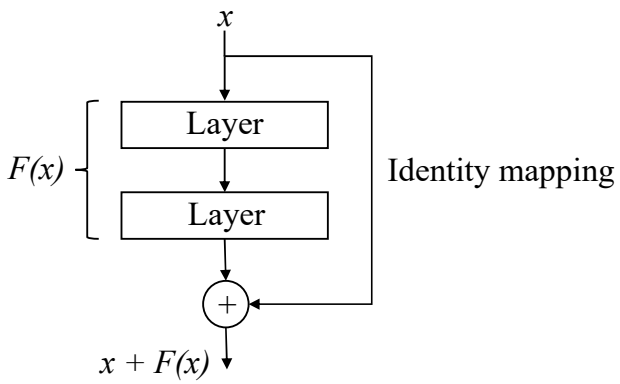


Figure 2.3: Illustration of a residual block. An input x is passed both trough the layers and directly to the addition module, via the identity mapping. The identity mapping can be seen as a shortcut, letting the input pass the layers without being affected by their operations F. At the addition module, the resulting output from the layers, $F(x)$, is summed with the original input x to form the final output $F(x) + x$.

The residual connections enable easier training of DNNs and combat the vanishing gradient problem that stacking many layers often leads to [20].

2.1.5 Trajectory prediction

Predicting the trajectories of other road users can support both human and autonomously driven vehicles to safely anticipate future driving scenarios. Accurate trajectory prediction is essential for ensuring safe and efficient vehicle operation, as it enables the generation of feasible and safe driving paths. There are two main approaches to trajectory prediction: physics-based models and data-driven methods [21]. Physics-based models rely on physical models of vehicle dynamics to predict future movements [22], while data-driven methods use machine learning algorithms to learn patterns from large datasets of vehicle trajectories [23].

The use of data-driven methods, based on DNNs, has gained significant attention in this field in recent years due to their ability to learn patterns in large datasets of vehicle trajectories [24]. DNNs can be trained on various sources of information, such as vehicle speed, road geometry, and sensor readings, to improve the performance of the trajectory prediction. Some existing DNN models for trajectory prediction are CoverNet [25], MTP [26], and TNT [27]. Models for trajectory prediction can

either treat the task as a regression problem or a classification problem, where the former tries to directly predict the future positions of the vehicle in question, while the latter classifies over a given set of trajectories.

2.2 Federated learning

Federated learning (FL) is an approach for training ML models on decentralized data, offering potential advantages in data privacy and reduced data communication costs. In this chapter, we outline the key concepts and principles of FL, including an overview of FL, the training process, privacy and security concerns, the influence of independent and identically distributed (IID) data, and various federated optimization algorithms.

2.2.1 FL overview

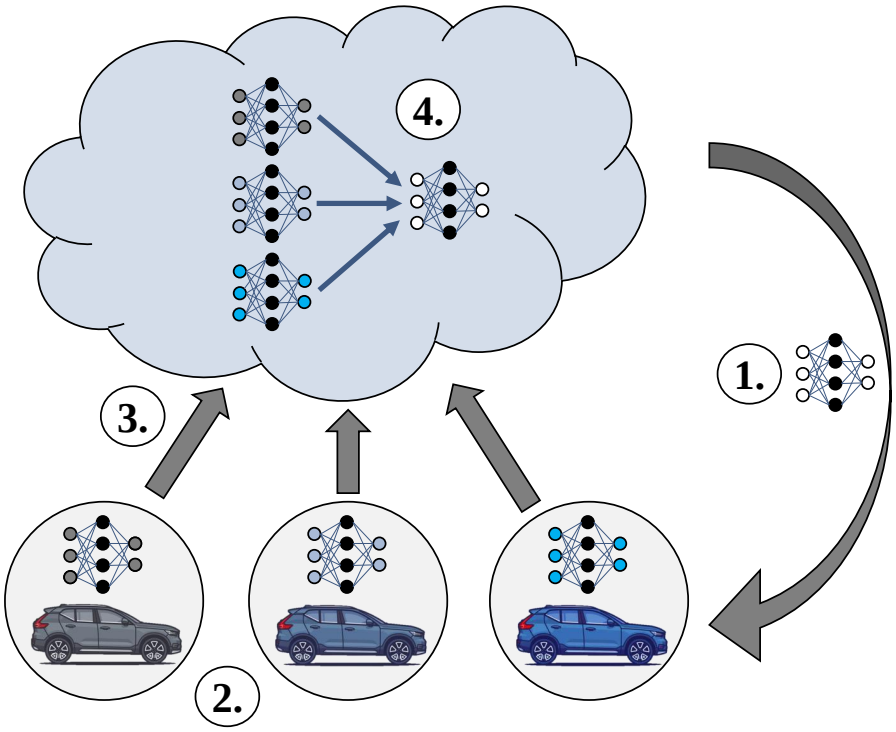


Figure 2.4: The FL training process. (1) copies of the global model are shared with each selected client for a training round. (2) the clients train the models on their locally collected data. (3) the trained models are uploaded to a central server and (4) aggregated into a new global model. The process is repeated for all training rounds R.

The process of how training in a federated way can be done is illustrated in Figure 2.4, where one loop corresponds to one training round. Instead of sharing the

local data to a central server and training centrally, as done in CL, copies of the global ML model are distributed to all selected clients. The now local models are separately trained on the individually observed data stored by each client. After training, all selected clients share their local model parameter updates to the central server, where they are aggregated into a new global model. This is done to improve and generalize the global model, which is thereafter shared with the clients again. Throughout training this cycle repeats, enabling global learning while keeping client data decentralized.

FL differs from (1) centralized learning (CL), where observations are collected into a large dataset that is used for training an ML model, (2) distributed ML, where the training process is parallelized by distributing the gathered data between multiple processors, called worker nodes, to speed up training and (3) decentralized ML, where the data usually is assumed to be independent and identically distributed (IID). Data being identically distributed means that samples are taken from the same probability distribution and therefore not subject to distribution fluctuations. Independent refers to items not being connected in any way and knowledge of one item does not give information of another. For example, the outcomes of a sequence of roulette wheel spins are IID, while the values of a stock price time series are usually non-IID. FL does not assume IID data or collection of all data to a central server, nor is it designed to only be used as a parallelization tool to speed up training.

Edge computing is a decentralized computing model that brings computation and data storage closer to the data source [28]. In this model, computing processes take place at edge devices in the network. For FL in an automotive setting, the computers in each vehicle i.e., at each client, can be seen as edge devices. FL using edge computing can reduce processing and storing sensitive data to avoid privacy or security issues, while also reducing latency and improving efficiency compared to CL [29].

2.2.2 Privacy and security

Although client data is not centrally collected in FL, privacy and security issues still remain [9] [30]. Parameter updates shared during training are direct consequences of the data, and reverse deduction methods could therefore be applied to the updates to get access to the data. There are several privacy techniques that could be used to prevent malicious activities of this kind. *Differential privacy* is a technique for sharing information about a dataset while protecting the data of individual elements. It works by collecting information into groups with visible patterns, while keeping individual information confidential. Differential privacy is based on the principle that small variations in the database caused by the substitution of a single element do not reveal substantial information about the individual, thus providing privacy. [31]. *Homomorphic encryption* is a method that when applied in an ML setting uses encryption of model parameters, while allowing computations to be made directly on the encrypted parameters, without the need for decryption [32]. *Secure Multi-party Computation* is a protocol for distributing computations across multiple parties,

without revealing their individual data [33]. By using a combination of some, or all, of these and other similar methods the privacy and security of FL can be improved.

2.2.3 IID and non-IID data

In ML, there is usually an assumption of IID data [9]. The IID assumption is used in the concepts of empirical risk minimization, law of large numbers and the central limit theorem, all useful to utilize in ML. Due to training on local data from distributed clients, the assumption of IID is usually not valid in FL [9]. Firstly, the observations from different clients or groups of clients might have different distributions, therefore preventing model convergence. Secondly, another shift in the probability distribution might occur over time if there is a change of participating clients or a change of behavior of the clients [34], called dataset shift due to a non-stationary environment. The implications of non-IID data on convergence rate and accuracy in FL have previously been studied [35] [36]. The results indicate that deterioration of accuracy is close to inevitable and the convergence rate slows down, compared to using IID data.

2.2.4 Federated optimization algorithms

FL training algorithms span multiple iterations, each involving local training at client level and communication of parameters to and from the central server. These rounds can be structured in different ways and there are several options for how the parameter aggregations are performed each round.

The strategies to perform FL are called federated optimization algorithms (later referred to as federated optimizers). Some of the relevant options to be considered are presented in Table 2.1. It provides an overview of these algorithms, their primary objectives, and the key techniques employed to address challenges in FL.

Table 2.1: Summary of Federated learning optimization algorithms.

Algorithm	Objective
FedAvg [37]	Weighted averaging of client models *Key Techniques:* Standard averaging
FedAvgM [38]	Weighted averaging of client models with momentum *Key Techniques:* FedAvg with momentum on the server side
FedProx [39]	Robust convergence for heterogeneity *Key Techniques:* proximal term
FedOpt [40]	Server-side adaptive optimizers *Key Techniques:* ADAGRAD, ADAM, YOGI optimizers

The algorithms aim to address various challenges in FL, such as convergence speed and heterogeneity in devices and data. The techniques employed by these algorithms

vary, ranging from weighted averaging, momentum, and proximal terms to adaptive optimizers. The choice of a suitable optimization algorithm depends on the specific requirements and constraints of the problem at hand, such as the nature of the data and the desired trade-off between convergence speed and model performance.

FedAvg (federated averaging) [37] is the most common optimization algorithm for FL. For each round, the algorithm selects a fraction of the clients (client fraction). These clients receive the current model and perform training on their local data, before a weighted average of the resulting client models are gathered and constructed into a new model [41]. Using FedAvg, the FL training follows Algorithm 1. Line 9 is characteristic for FedAvg, updating the server model by averaging over the updated weights from all selected clients in the round.

Algorithm 1 *FedAvg.* The C clients are indexed by c, E is the number of local epochs, B is the local mini-batch size, F_C is the client fraction (fraction of clients randomly selected for each round of FL), n_c is the number of local examples, ℓ is the loss function, and L_r is the learning rate. D_c is the dataset at client c and $D_{c,B}$ is the corresponding dataset split into mini-batches of size B. Re-presented algorithm from paper [37].

1: *// Server executes*
2: initialize w_1
3: **for** each round $r = 1, 2, \ldots$ **do**
4: $\quad N_r \leftarrow \max(F_C \cdot C, 1)$
5: $\quad S_r \leftarrow$ (random set of N_r clients)
6: $\quad$ **for** each client $c \in S_r$ **in parallel do**
7: $\quad\quad w_{r+1}^c \leftarrow \text{ClientUpdate}(c, w_r)$
8: $\quad m_r \leftarrow \sum_{c \in S_r} n_c$
9: $\quad w_{r+1} \leftarrow \sum_{c \in S_r} \frac{n_c}{m_r} w_{r+1}^c$
10:
11: **procedure** ClientUpdate(c, w) *// Run on client c*
12: $\quad D_{c,B} \leftarrow$ (split D_c into batches of size B)
13: $\quad$ **for** each local epoch i from 1 to E **do**
14: $\quad\quad$ **for** batch $b \in D_{c,B}$ **do**
15: $\quad\quad\quad w \leftarrow w - L_r \nabla \ell(w, b)$
16: $\quad$ **return** w

FedAvgM (Federated Averaging with Server Momentum) [38] is similar to FedAvg, but adds momentum on the server side. Implementation of FedAvgM follow Algorithm 2. Previous work shows that FedAvgM's performance can reach similar levels of performance faster than FedAvg.

Algorithm 2 *FedAvgM.* The C clients are indexed by c, E is the number of local epochs, B is the local mini-batch size, F_C is the client fraction, n_c is the number of local examples, ℓ is the loss function, and L_r is the learning rate. D_c is the dataset at client c and $D_{c,B}$ is the corresponding dataset split into mini-batches of size B. A new constant, β, is the momentum coefficient. In this algorithm, we introduce server momentum, where weight updates are computed as $v_{r+1} = \beta v_r + \Delta w$, and the model is updated with $w_{r+1} = w_r - v_{r+1}$.

// Server executes
initialize w_1, v_1
for each round $r = 1, 2, \ldots$ **do**
    $N_r \leftarrow \max(F_C \cdot C, 1)$
    $S_r \leftarrow$ (random set of N_r clients)
    for each client $c \in S_r$ **in parallel do**
        $w_{r+1}^c \leftarrow \text{ClientUpdate}(c, w_r)$
    $m_r \leftarrow \sum_{c \in S_r} n_c$
    $\Delta w \leftarrow \sum_{c \in S_r} \frac{n_c}{m_r}(w_{r+1}^c - w_r)$
    $v_{r+1} \leftarrow \beta v_r + \Delta w$
    $w_{r+1} \leftarrow w_r - v_{r+1}$

procedure ClientUpdate(c, w) *// Run on client c*
    $D_{c,B} \leftarrow$ (split D_c into batches of size B)
    for each local epoch i from 1 to E **do**
        for batch $b \in D_{c,B}$ **do**
            $w \leftarrow w - L_r \nabla \ell(\text{w, b}) -$
            return w

In order to address heterogeneity in FL settings, both in the devices' networks (systems heterogeneity) and in non-IID data (statistical heterogeneity), the framework *FedProx* was introduced, presented in Algorithm 3. FedProx is a generalization and re-parametrization of the previous mentioned FedAvg algorithm, showing more robust convergence [39].

Algorithm 3 *FedProx.* The C clients are indexed by c, E is the number of local epochs, B is the local mini-batch size, F_C is the client fraction, n_c is the number of local examples, and ℓ is the loss function. D_c is the dataset at client c and $D_{c,B}$ is the corresponding dataset split into mini-batches of size B. μ is the proximal term constant.

// Server executes
Initialize w_1
for each round $r = 1, 2, \dots$ **do**
    $N_r \leftarrow \max(F_C \cdot C, 1)$
    $S_r \leftarrow$ (random set of N_r clients)
    for each client $c \in S_r$ **in parallel do**
        $w_{r+1}^c \leftarrow \text{ClientUpdate}(c, w_r)$
    $m_r \leftarrow \sum_{c \in S_r} n_c$
    $w_{r+1} \leftarrow \sum_{c \in S_r} \frac{n_c}{m_r} w_{r+1}^c$

procedure ClientUpdate(c, w) *// Run on client c*
    $D_{c,B} \leftarrow$ (split D_c into batches of size B)
    for each local epoch i from 1 to E **do**
        for batch $b \in D_{c,B}$ **do**
            $w \leftarrow \arg\min_{w'} \left(\ell(w', b) + \frac{\mu}{2} \|w' - w\|^2 \right)$
    return w

FedOpt is a family of algorithms containing implementations of several adaptive optimizers (ADAGRAD, ADAM, YOGI) [40]. The optimizers are used on the server side while SGD is used on the client side, to ensure that communication costs are similar to FedAvg. Similarly to FedProx, FedOpt aims to improve performance on more diverse data compared to FedAvg. The reader is referred to the original publication for details about the algorithms.

3
Methodology

This chapter outlines the overall methodology, datasets, underlying model architectures, design of experiments, and experimental setups used in this thesis. To gain experience and confirm the functionality, an FL implementation validation was done on the MNIST dataset, using a simple model. This is followed by the main task, evaluating FL for trajectory prediction using the nuScenes dataset.

3.1 Implementation validation

The implementation validation methodology section starts with a description of the MNIST dataset and how it was used to evaluate FL for IID and non-IID data. Secondly, the specific experimental setup is presented. Lastly, the experiments done on MNIST are described in detail.

3.1.1 MNIST dataset

The MNIST (Modified National Institute of Standards and Technology) dataset [42] is an extensive collection of handwritten images that has been widely used for image classification tasks. The images are grayscale handwritten digits with 28x28 pixels resolution. In total, the set consists of 60 000 training images and 10 000 testing images. Some example images from the MNIST set are presented in Figure 3.1.

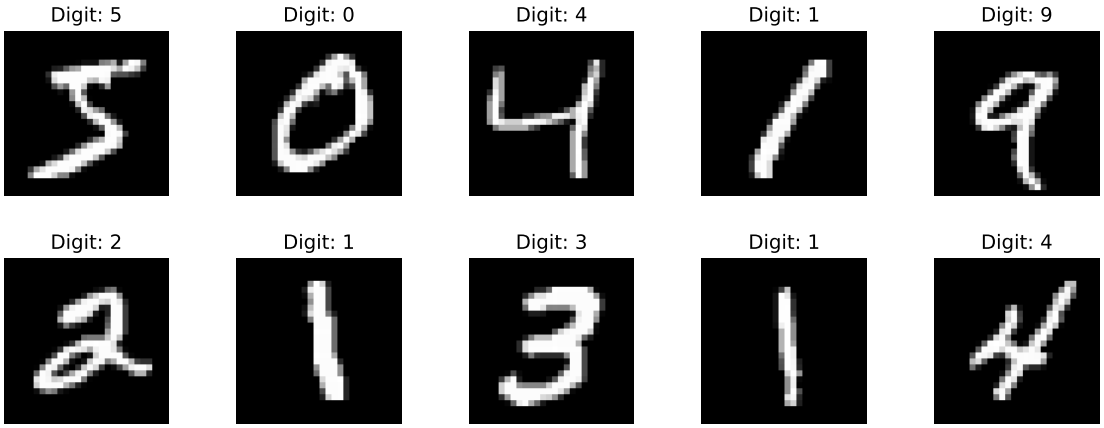


Figure 3.1: Images from the MNIST dataset

Although the MNIST dataset is not directly linked to trajectory prediction or federated learning, the fact that it has been extensively used in previous classification

tasks with well-established results makes it a great set to benchmark and validate our FL setup. Additionally, by doing this we gain insight into how FL performs on another dataset to compare to nuScenes, later used for trajectory prediction.

In our MNIST experiments, we employ different strategies for distributing the dataset among the clients. We categorize these strategies as either IID or non-IID. IID distribution involves creating sets with independently and identically distributed data. To achieve this, we randomly shuffle the training set before dividing it among the clients. This ensures that each client receives a subset of the dataset that represents the entire range of digits in a balanced manner. In contrast, non-IID distribution entails sorting the data based on the labels before allocating it to the clients. This means that each client receives a portion of the dataset that primarily consists of samples belonging to a specific digit class. To visualize this process, refer to Figure 3.2. It demonstrates that even when the data is sorted before distribution, there is still an overlap of digit classes between clients. This overlap persists even when the total number of classes is evenly divisible by the number of clients. It is important to note that this overlap occurs due to the inherent variation in the distribution of the digit classes. Each digit (0-9) has a slightly different frequency of occurrences, ranging from 5400 to 6700, summing up to a total of 60000 samples in the dataset. One exploring experiment was conducted where the 10 classes were, in fact, balanced and evenly dividable by the number of clients (10), leading to the case that each client only received instances of one class. This resulted in the model not learning to classify the numbers. Hence, the overlap is needed.

	Client 1	**Client 2**	**Client 3**	**Client 4**
IID	3 1 0 3 2 3 0 1 0 0	3 3 0 0 1 1 2 3 1 0	1 2 0 2 3 0 1 3 0 2	3 1 0 2 3 2 0 2 3 1
Non-IID	0 0 0 0 0 0 0 0 0 0	0 0 1 1 1 1 1 1 1 1	1 2 2 2 2 2 2 2 2 3	3 3 3 3 3 3 3 3 3 3

Figure 3.2: Illustration showing the effect of employing IID and non-IID distribution methods for dividing the training data. For simplicity, only four numbers and four clients are depicted. It is important to note that the numbers display slight variations in their total occurrences, resulting in the non-IID sets containing overlaps.

3.1.2 Experimental setup

The FL implementation used in this study is built on Flower, a framework for federated learning [43] [44]. Flower focuses on implementing FL realistically in terms of scalability and systems heterogeneity. The framework follows a structure that allows both local (client side) and global (server side) computations, and communications between the actors. Flower also aims to provide an easier transition from simulated FL to FL in practice on real hardware devices. Our setup utilizes the existing setup in Flower as a base for development. This includes, for example, methods for

data processing, server-client communications, client operations, and implementations of federated optimization algorithms. We used Tesla T4 GPUs for the training of our ML models. Everything was run through Python 3.8.16 using PyTorch 1.10.2.

To validate our setup and test the effects of settings and parameters, we performed classification experiments on the MNIST dataset, using a simple CNN model. The focus and motivation were not only on the results of the MNIST classification task but also to validate that all installations and implemented functions were working as intended, by reconstructing a task that has been thoroughly investigated with recognized results. However, extensive runs were still done to be able to compare the results to the following trajectory prediction experiments.

3.1.3 Experiments on MNIST

The experiments on the MNIST dataset are set to explore how the model performance and convergence are impacted by changes in various parameters. Performance, in this case, refers to the highest achieved level of accuracy. This is done by repeatedly training models with a change in one single parameter, while keeping the other parameters constant, measuring the loss and accuracy throughout the training interval. In these experiments, the clients and server are simulated, meaning that all operations are carried out locally, without the involvement of multiple hardware devices. The simulated clients represent individual devices with unique data and communicate accordingly, only sharing and receiving model updates.

Parameter	**Tested values**
Data distribution	**IID**, **non-IID**
Batch size B	10, 16, **32**, 64, 128
Learning rate L_r	0.0001, 0.001, 0.01, **0.1**
Number of clients C	1, 5, **10**, 20, 40, 80
Client fraction F_C	0.2, 0.4, 0.6, 0.8, **1.0**
Federated optimizers	**FedAvg**, FedAvgM, FedProx, FedAdagrad, FedAdam, FedYogi

Table 3.1: Table listing the parameters and values that will be tested. The values marked in bold represent the chosen standard baseline. Both IID and non-IID are marked bold since each setup will be repeated with both IID and non-IID.

First of all, we establish a ”standard baseline” in which we decide default values for all parameters that are investigated. These values have been chosen based on a combination of previous studies and results/tendencies from our initial exploring tests. The standard baseline includes batch size $B = 32$, learning rate $L_r = 0.1$, number of clients $C = 10$, client fraction $F_C = 1.0$, and FedAvg as federated optimizer. When conducting the experiments to explore the possible values listed in Table 3.1, all other parameters, apart from the specific one being tested, are set to the corresponding default values marked as bold. Note that both IID and non-IID are listed as standard baseline since both are tested, for each setup. Apart from the

chosen parameter values, each experiment is performed with 100 rounds of training R, and all clients are performing 1 epoch of training per round E.

3.1.3.1 Learning rate

Experiment 1 focuses on evaluating the impact of the learning rate L_r on the performance and convergence of the model. The learning rates tested are 0.0001, 0.001, 0.01, and 0.1. The remaining parameters are kept consistent with the standard baseline configuration, which includes a batch size B of 32, 100 rounds of training R, 10 clients C, each clients performing 1 epoch of training per round E, 1.0 as client fraction F_C, FedAvg as federated optimizer, and using IID data for training split. Theses values for E and R are kept constant for all experiments of the implementation validation. In short, Experiment 1 is therefore summarized as

1:	B	L_r	C	F_C	Optimizer	Data
	32	$[0.0001, 0.001, 0.01, 0.1]$	10	1.0	FedAvg	IID

using the same order as previously listed with the addition of the inserted learning rate vector. In *Experiment 2*, the same tests are repeated, but now using non-IID data:

2:	B	L_r	C	F_C	Optimizer	Data
	32	$[0.0001, 0.001, 0.01, 0.1]$	10	1.0	FedAvg	non-IID

3.1.3.2 Batch size

In *Experiment 3* and *Experiment 4*, different values of batch size are tested. Batch sizes 8, 16, 32, 64, and 128 are investigated, otherwise using the same parameters as the first experiments and a constant learning rate of 0.1:

3-4:	B	L_r	C	F_C	Optimizer	Data
	$[10, 16, 32, 64, 128]$	0.1	10	1.0	FedAvg	**3:** IID **4:** non-IID

3.1.3.3 Number of clients

The effects on performance and convergence rate from different numbers of clients C are investigated in *Experiment 5* and *Experiment 6*. The client values tested were 1 (equivalent to CL), 5, 10, 20, 40, and 80. In short:

5-6:	B	L_r	C	F_C	Optimizer	Data
	32	0.1	$[1, 5, 10, 20, 40, 80]$	1.0	FedAvg	**5:** IID **6:** non-IID

3.1.3.4 Client fraction

Experiment 7 and *Experiment 8* investigates the performance implication of varying the client fraction used in each training round. We test the client fractions 0.2, 0.4, 0.6, 0.8 and 1.0, once again performed for both IID and non-IID data distributions:

7-8:	B	L_r	C	F_C	Optimizer	Data
	32	0.1	10	$[0.2, 0.4, 0.6, 0.8, 1.0]$	FedAvg	**7:** IID **8:** non-IID

3.1.3.5 Federated optimization algorithms

The impact of the different federated optimization algorithms [FedAvg, FedAvgM, FedProx, FedAdagrad, FedAdam and FedYogi] (referred to as *optimizers* below) are tested in *Experiment 9* (IID) and *Experiment 10* (non-IID). Previous altered values are reverted to $C = 10$, $B = 32$ and $L_r = 0.1$, resulting in:

9-10:	B	L_r	C	F_C	Optimizer	Data
	32	0.1	10	1.0	*optimizers*	**9:** IID **10:** non-IID

3.1.3.6 Number of clients and client fraction

An experiment was designed to investigate the impact of the relationship between the number of clients and client fraction. The parameters are varied according to $C = [5, 10, 20, 40, 80]$ and $F_C = [0.2, 0.4, 0.6, 0.8, 1.0]$, resulting in a total of 25 combinations. This is done with the IID data distributions with the other parameters again kept constant. *Experiment 11* can be summarized as

11:	B	L_r	C	F_C	Optimizer	Data
	32	0.1	$[5, 10, 20, 40, 80]$	$[0.2, 0.4, 0.6, 0.8, 1.0]$	FedAvg	IID

3.2 Trajectory prediction

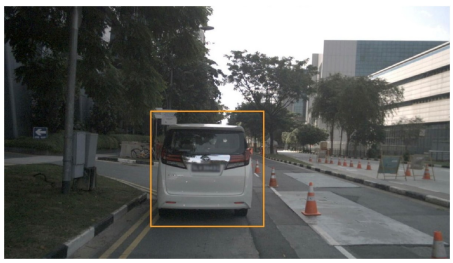
This section describes the dataset, models, and methodology used for the trajectory prediction experiments in this study. First, we present nuScenes [45], a dataset created for autonomous driving research. Secondly, we describe CoverNet [25], the model used for predicting trajectories. Lastly, we describe our experiments and choice of dataset parameters.

3.2.1 nuScenes dataset

For our trajectory prediction experiments, we chose to use nuScenes [45], a dataset created for autonomous driving research. It is a collection of data from vehicles driving in different cities/regions, i.e., Boston and Singapore. Therefore, it is suitable for FL experiments by providing a diverse set of traffic scenarios and environments. We can leverage this to examine performance and generalizability for geographic diversity, i.e., non-IID data.

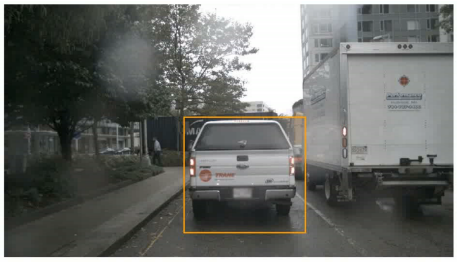
The dataset contains 1000 driving scenes collected from multiple cameras, a LiDAR, RADAR, and GPS at a frequency of 2Hz. Each scene is approximately 20 seconds long ($\sim$ 40 time steps), with 2D and 3D bounding boxes annotations tracking all surrounding vehicles, pedestrians, and objects simultaneously over time. Figure 3.3 shows a vehicle tracked over time. Figure 3.4 shows two different vehicles being tracked simultaneously. The positions, velocities, and accelerations of all tracked objects are also available.

(a) First sample.



(b) Second sample at a later time step.

Figure 3.3: Tracked vehicle with 2D bounding boxes for two different time steps.



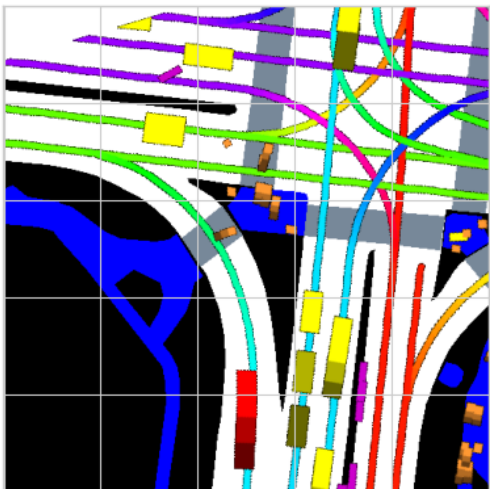
(a) Car with bounding box.

(b) Truck with bounding box.

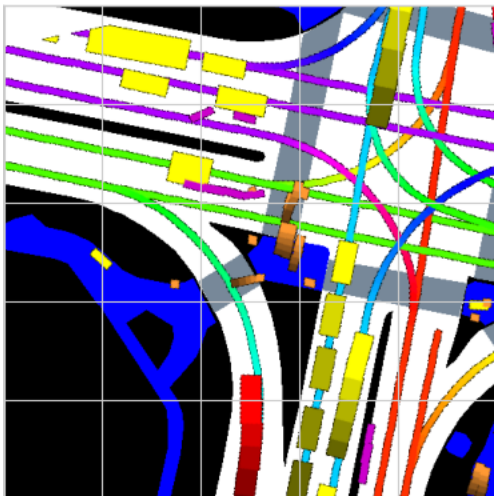
Figure 3.4: Tracked car and truck with 2D bounding boxes.

The data can be transformed into bird's eye view (BEV) images of the surrounding area, illustrating the vehicle to predict (in red), all surrounding vehicles (in yellow), and pedestrians (in orange), seen in Figure 3.5 and 3.6. This simplifies the dataset but keeps important spatial information necessary for 2D trajectory prediction. The vehicle collecting the data is not present in the images. Hence, it is the trajectories of all surrounding vehicles we predict. All moving objects have their own history illustrated by the faded trail of boxes. The history represents the chosen number of previous time steps at 2 Hz. The image resolution is 0.1 meter per pixel and the total image size is 500x500 pixels, which is equivalent to a 50mx50m square. In Figure 3.5, two samples are presented with 1.0 and 2.0 seconds of history, respectively. Figure 3.6 shows one sample from Boston and one from Singapore, apparent by the right-hand traffic in Boston and left-hand traffic in Singapore.

For each scene, several images can be created. All visible vehicles can be focused (red) and used for prediction. This is under the assumption that they are visible and tracked for at least the time period of the chosen history, plus the current frame, plus the prediction horizon (always set to 6.0 seconds). With 2 Hz, 2.0 seconds of history demands $2 \cdot (2.0 + 0.5 + 6.0) = 17$ time steps to be available. The data can be sampled in several ways. Either by including one image for each eligible vehicle in a scene, or several by creating images with non-overlapping histories for each vehicle in a scene, resulting in a larger dataset. This was later tested with results presented in Figure 3.12.

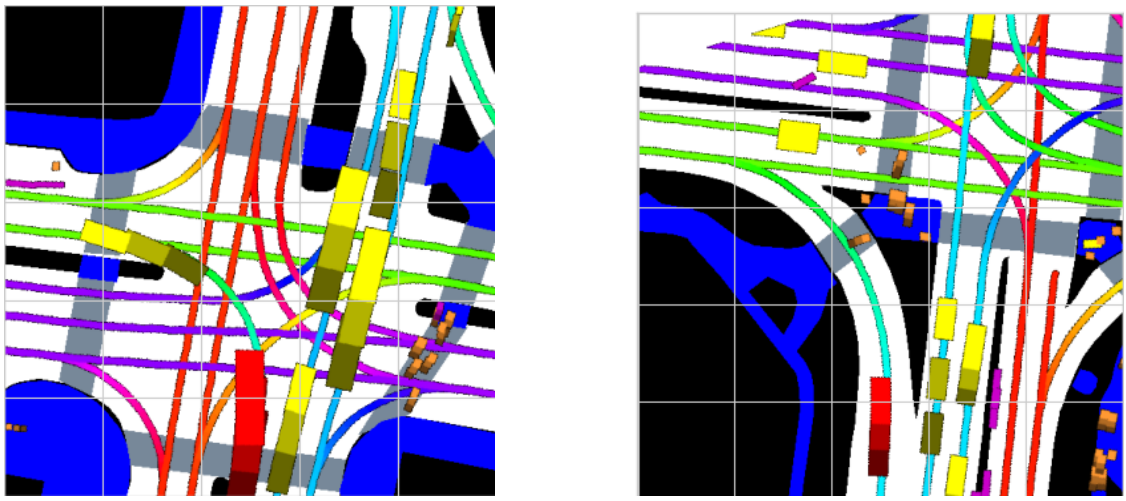


(a) BEV image with 1.0 second of history.



(b) BEV image with 2.0 seconds of history.

Figure 3.5: BEV images with varying lengths of history used. For each agent, the bounding boxes displaying the previous positions are ranging from darker to brighter (current position).



(a) BEV image from Boston with right-hand traffic.

(b) BEV image from Singapore with left-hand traffic

Figure 3.6: BEV images from Boston and Singapore, respectively.

The dataset can be changed and distributed to clients in an FL setting to simulate all combinations of IID, non-IID, balanced, and non-balanced data distributions. Here, data from Boston and Singapore are considered to be non-IID if sorted by city and distributed to separate clients, as shown in Figure 3.7. This is motivated by the two locations driving on different sides of the road, being different in terms of infrastructure, and having different driving culture standards. If distributed randomly to clients, the data is considered IID. The dataset is balanced if an equal number of samples from Boston and Singapore are present, and non-balanced if not. Our balanced dataset was constructed of 864 samples from Boston and 864 samples from Singapore, while the non-balanced dataset was constructed of 1152 samples from Boston and 864 samples from Singapore. The validation set consists of 512 samples from each city in all cases.

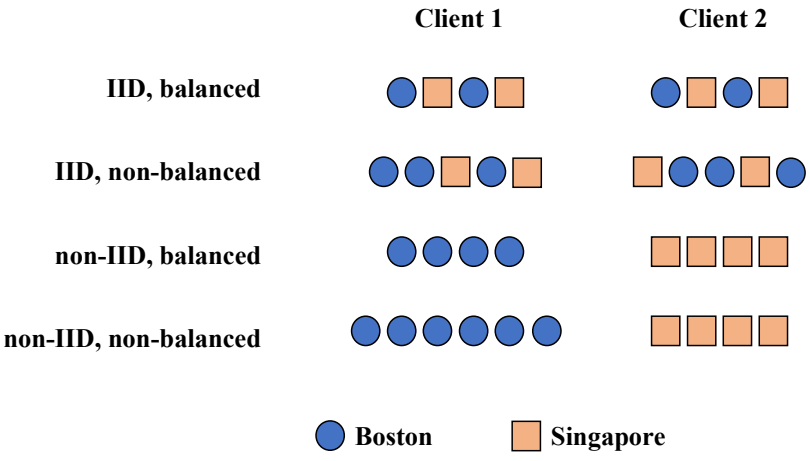


Figure 3.7: The different variations of IID, non-IID, balance, and non-balance dataset splits.

3.2.2 Trajectory prediction model architecture

We used CoverNet [25] as our underlying trajectory prediction model. It is a multi-modal, probabilistic trajectory prediction model based on classification over a set of trajectories. Using the previously discussed BEV image together with an agent state vector (speed, acceleration, and yaw rate) as input, CoverNet handles the trajectory prediction task as a classification problem. It was created to improve trajectory prediction for autonomous driving in urban environments, which is characterized by dynamic and interactive scenarios including various types of agents, for example, motor vehicles, pedestrians, and bicycles among others.

CoverNet is based on a CNN backbone, followed by fully connected layers with the last layers representing the probabilities for each mode in a chosen trajectory set. This architecture is presented in Figure 3.8. The backbone CNN consists of a ResNet model, in particular ResNet50. It contains 48 sequential convolutional layers, one maxpool layer and one average pool layer. The ResNet backbone outputs 2058 image features, which are flattened and concatenated with the agent state scalar vector, resulting in a combined vector of length 2061. The head of CoverNet consists of two fully connected layers: the first with dimension 4096 and the second with a dimension equal to the number of modes in the chosen trajectory set.

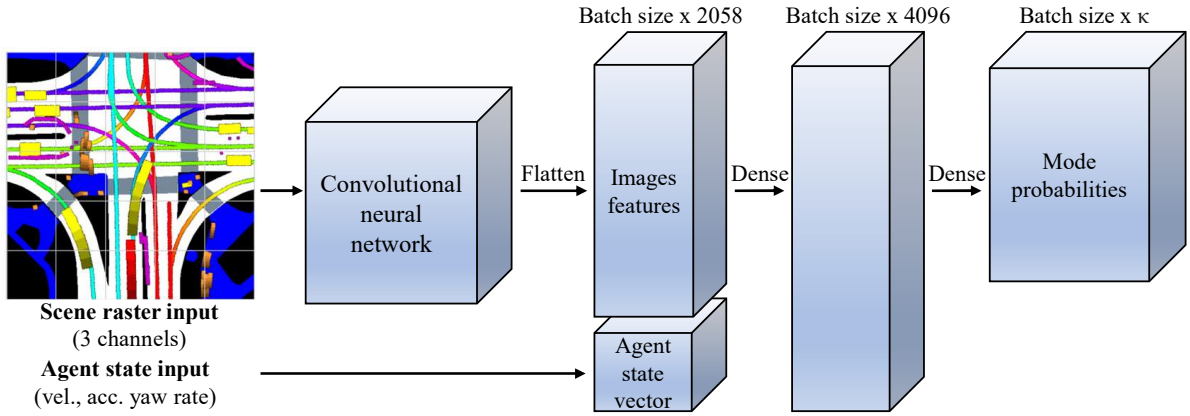


Figure 3.8: Illustration of the CoverNet architecture.

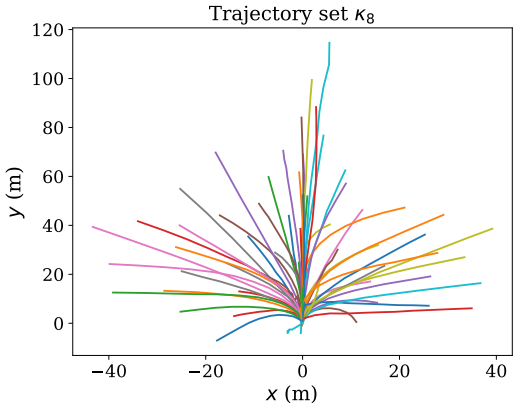
In the original paper, CoverNet was used to predict future agent state over a 6.0-second horizon. Two versions of CoverNet, specifically with respect to the trajectory sets, were implemented. One version used *fixed* trajectory sets to classify its predictions. These sets can be seen in Figure 3.9 and are unaffected by the momentary agent state or surrounding factors. Another version used *dynamic* trajectory sets, taking the agent's current dynamic state into account. The latter uses physical models to guarantee that all trajectories in the trajectory set are dynamically feasible. The dynamic sets have been shown to improve performance in comparison to the fixed sets. However, since this study focuses on evaluating FL, we implemented the fixed trajectory sets as this reduced the computational requirements and still provides results to benchmark.

The fixed trajectory sets are subsets of all trajectories available in the nuScenes training dataset. They are selected by the conditions stated in Equation 3.1 with acceptable error tolerance ϵ, affecting the coverage and size of the trajectory set. The maximum point-wise Euclidean distance δ between predicted trajectory $\hat{s}$ and ground truth s was used to determine the smallest fixed trajectory set κ that approximates the full set of 20 000 trajectories κ', as:

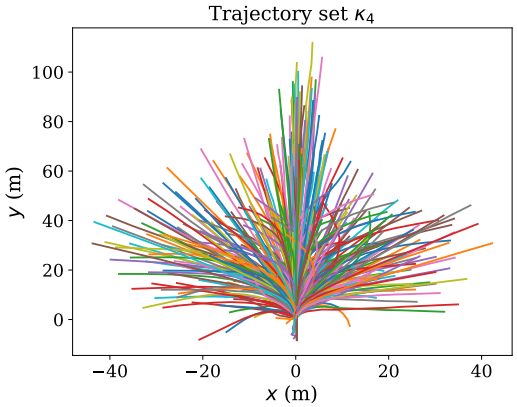
$$\begin{aligned} \underset{\kappa}{\text{argmin}} \quad & |\kappa| \\ \text{subject to} \quad & \kappa \subseteq \kappa', \\ & \forall k \in \kappa', \exists l \in \kappa, \, \delta(k, l) \leq \epsilon, \\ & \delta(s_{t:t+N}, \hat{s}_{t:t+N}) := \max_{\tau=t}^{t+N} \|s_\tau - \hat{s}_\tau\|_2 \end{aligned} \tag{3.1}$$

where:
τ = the current time steps,
t = the first time step,
N = number of time steps.

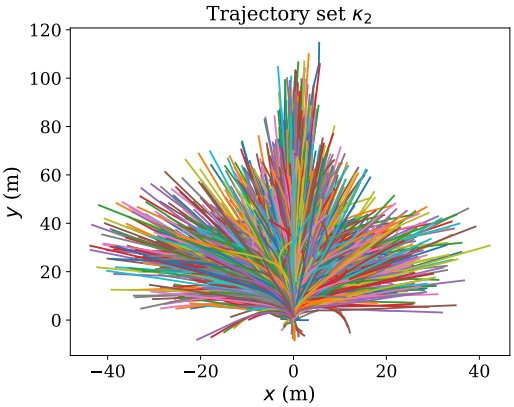
The trajectory sets κ_8, κ_4 and κ_2 corresponding to $\epsilon = 8$ m, $\epsilon = 4$ m and $\epsilon = 2$ m are presented in Figure 3.9, respectively. In simpler terms, this guarantees that all ground truths s in the dataset are no more than a maximum ϵ m from any predicted trajectory $\hat{s}$ at each time step τ. An example can be seen in Figure 3.11. The tolerance $\epsilon = 8$ results in 64 trajectories (modes), $\epsilon = 4$ in 415 modes, and $\epsilon = 2$ in 2206 modes.



(a) Trajectory set κ_8 corresponding to $\epsilon = 8$. There are 64 trajectories in total.



(b) Trajectory set κ_4 corresponding to $\epsilon = 4$. There are 415 trajectories in total.



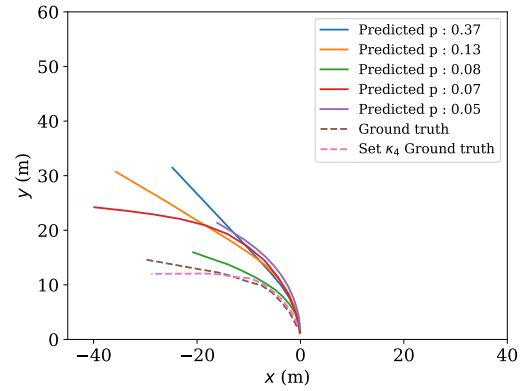
(c) Trajectory set κ_2 corresponding to $\epsilon = 2$. There are 2206 trajectories in total.

Figure 3.9: The three available fixed trajectory sets κ_i with varying $\epsilon_i = [2, 4, 8]$ m.

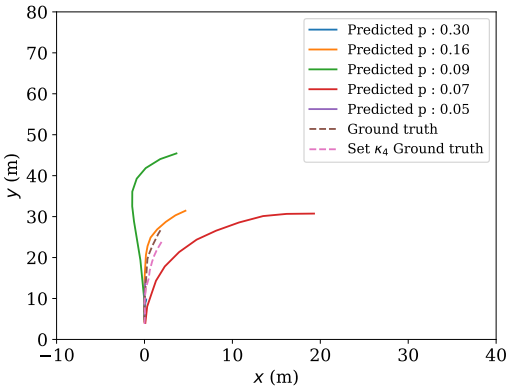
The inputs to the CoverNet model are the previously shown three-channel (RGB) scene raster BEV images and an agent state vector representing the agent's speed (m/s), acceleration (m/s^2), and yaw rate (radians/s). The corresponding agent state vector to the red vehicle in the example BEV with 1.0 seconds of history included, presented in Figure 3.5a is:

$$[4.38 \text{ m/s}, 0.31 \text{ m/s}^2, 0.03 \text{ radians/s}].$$

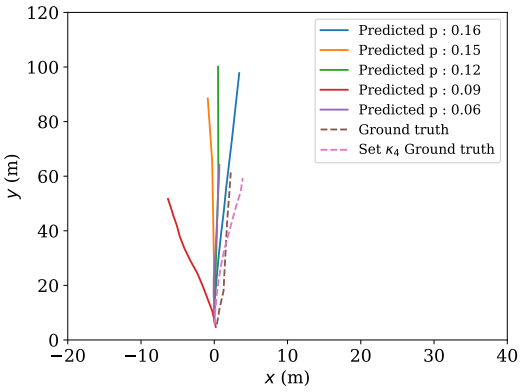
The problem of predicting the trajectory of a chosen vehicle is then transformed into a classification problem. By comparing the ground truth trajectory to all trajectories in the chosen trajectory set κ with respect to the mean point-wise L2 distance, the trajectory closest to the ground truth is selected as the κ_i-specific ground truth, as shown in Figure 3.10. This is then used for the classification where the loss is computed using the cross-entropy loss function. However, for all evaluation metrics, the real ground truth is used.



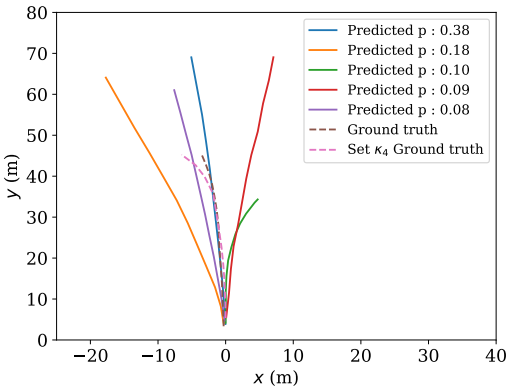
(a) Left turn.



(b) Right turn.



(c) Driving straight.



(d) Poor predictions for a slight left turn.

Figure 3.10: Examples of predicted trajectories from CoverNet. The examples include a left turn, a right turn, straight driving, and a slight left turn with poor predictions. The top 5 predicted trajectories, along with their respective probabilities, are compared to the real and κ_4-specific ground truths.

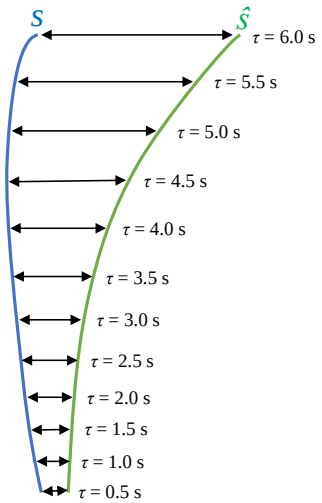


Figure 3.11: Illustration showcasing the displacement of a predicted trajectory $\hat{s}$ and the ground truth trajectory s for all time steps τ.

Several different metrics can be used for evaluation of CoverNet's performance. We use the following metrics to evaluate predicted trajectories compared to the ground truth: average displacement error (ADE_k) in different forms, final displacement error (FDE_k) and hit rate ($\text{HitRate}_{k,d}$). Figure 3.11 illustrates a predicted trajectory $\hat{s}$ and the displacement relative to the ground truth s. The minADE_k is then defined as

$$\text{minADE}_k = \min_{\hat{s} \in P} \frac{1}{N} \sum_{\tau=t}^{t+N} \|s_\tau - \hat{s}_\tau\|,$$

where P is the set of all predicted trajectories, $\hat{s}$ the k most probable (top) predicted trajectories and s the ground truth trajectory. The metric returns the minimum ADE value of the top k trajectories. To evaluate the average performance of the top k trajectories, we introduce

$$\text{aveADE}_k = \text{ave}_{\hat{s} \in P} \frac{1}{N} \sum_{\tau=t}^{t+N} \|s_\tau - \hat{s}_\tau\|,$$

returning the average ADE value of the top k trajectories. The FDE_1 is defined as

$$\text{FDE}_1 = \|s_{t+N} - \hat{s}^*_{t+N}\|,$$

where $\hat{s}^*$ is the top predicted trajectory. This metric only considers the displacement between the final predicted position, in our case when $\tau = 6.0$ s in Figure 3.11, to the corresponding ground truth value. $\text{HitRate}_{k,d}$ is calculated by inspecting the condition

$$\text{HitRate}_{k,d} = \min_{\hat{s} \in P} \max_{\tau=t}^{t+N} \|s_\tau - \hat{s}_\tau\| \leq d,$$

treated as 1 if upheld and 0 otherwise, and taking the average of all data samples. The $\text{HitRate}_{k,d}$ gives an indication of how often any of the k top predicted trajectories are within the tolerance d from the corresponding ground truth position throughout all predicted time steps τ. Lower displacement error metrics and larger HitRate correspond to higher performance.

3.2.3 Trajectory prediction experiments

Before the main experiments with FL were done on nuScenes, we did experiments using CL to choose how to process the raw data, trajectory set κ_i, and image resolution.

For the choice of a processed dataset, we considered validation accuracy and data processing requirements. The results are shown in Figure 3.12. Here, the first processed dataset was constructed by extracting one sample (input image and state vector) per vehicle and scene with 2.0 seconds of history, resulting in $\sim$ 2000 samples. The second and third were done using multiple non-overlapping samples per vehicle and scene. This method can be explained by an example. For a 20-second long scene, 40 frames are available at 2 Hz. Assuming a vehicle is tracked for all 40 frames, it is possible to extract 8 non-overlapping data samples of the vehicle with 2.0 seconds of history, since each sample includes 5 time steps. This was done using 2.0 and 1.0 seconds of history, resulting in $\sim$ 3000 and $\sim$ 6000 samples, respectively. Considering that the first processed dataset is the smallest, enabled faster training, and reaches the highest accuracy, it was chosen for all further experiments.

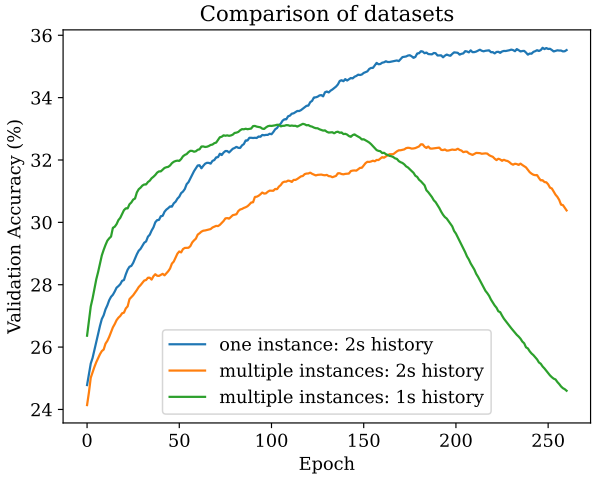


Figure 3.12: A comparison of validation accuracy using three processed datasets. These include one instance per vehicle and scene with 2.0 seconds of history, and non-overlapping multiple instances per scene with 2.0 and 1.0 seconds of history. A rolling mean of 40 epochs was used to smooth the originally noisy data for improved visualization.

Next, we did experiments on the three trajectory sets κ_2 (2206 modes), κ_4 (415 modes), and κ_8 (64 modes). In Figure 3.13, aveADE$_5$ is presented as the evaluation metric for the trajectory sets, but all other metrics showed similar trends. From a pure performance perspective, κ_4 performed the best. The size of the data files saved during each run is proportional to the number of modes. Therefore, κ_4 was considered a good compromise between performance and size of files for our experiments.

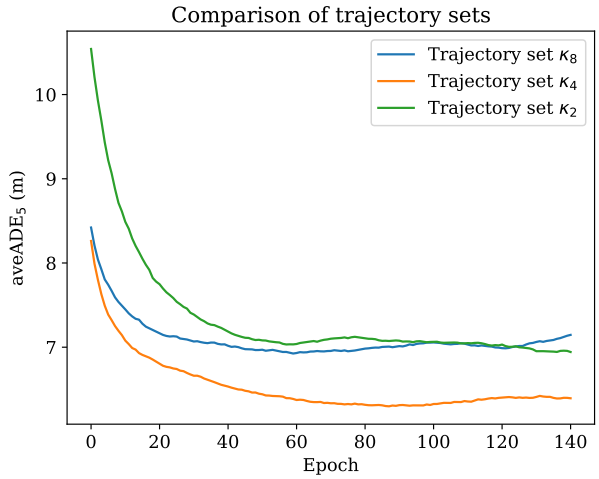


Figure 3.13: A comparison of aveADE$_5$, using the three trajectory sets κ_2 (2206 modes), κ_4 (415 modes) and κ_8 (64 modes). A rolling mean of 10 epochs was used to smooth the originally noisy data for improved visualization.

To speed up simulations and make it possible to overcome memory issues, we down-sampled the input images, originally 500 x 500 pixels, by factors of 1/2, 1/4, and 1/10 in both dimensions. As seen in Figure 3.14, 250x250 reached the same performance as 500x500, while reducing the number of pixels by a factor of 4.

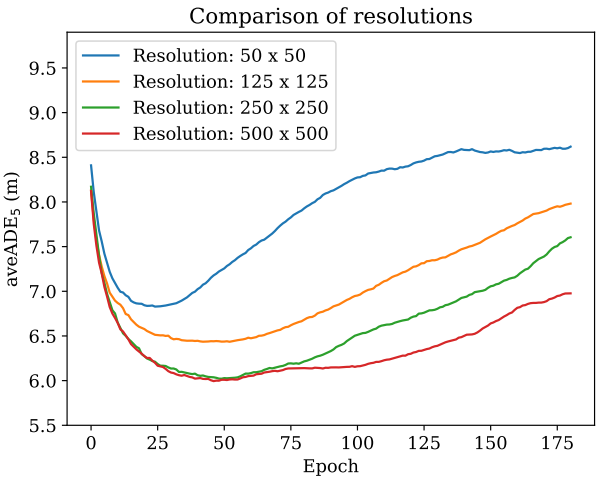


Figure 3.14: A comparison of aveADE$_5$, using three differently processed datasets where the images are generated using resolutions of 500x500 (original), 250x250, 125x125 and 50x50. A rolling mean of 20 epochs was used to smooth the originally noisy data for improved visualization.

We used the Flower framework for the main experiments, based on the setup used for the implementation validation in Section 3.1. It was modified to work with CoverNet, the nuScenes dataset, and the new metrics. We conducted experiments on batch size B and learning rate L_r to choose a suitable baseline for the upcoming

experiments. Next, we include a sweep of the federated optimizers FedAvg, FedAvgM, and FedProx, and variations of IID, non-IID, balanced, and non-balanced data. Furthermore, we evaluate the dependency on the number of clients C and client fraction F_C. Finally, we compare FL to CL for different variations of non-IID data distributions in an attempt to simulate real-life scenarios and evaluate the performance potential of FL. Table 3.2 provides an overview of the parameters tested in all experiments. All runs but one were done for $R = 1000$ rounds, where each round used $E = 1$ epoch of data. All experiments are evaluated on the same validation set consisting of 512 samples each from Boston and Singapore. The metrics used for evaluating performance are minADE_1, minADE_5, aveADE_5, FDE_1 and $\text{HitRate}_{5,2\text{m}}$. Training speed, the time needed to complete one round, and convergence rate, the number of rounds needed for convergence, were also observed and used for evaluation of all experiments.

Parameter	**Tested values**
Batch size B	4, **8**, 16, 32
Learning rate L_r	$1\text{e}-5$, $5\text{e}-5$, $\mathbf{1e}-4$, $5\text{e}-4$
Federated optimizers	**FedAvg**, FedAvgM, FedProx
Data distribution	IID balanced, IID non-balanced, non-IID balanced, non-IID non-balanced
	Full size, Half size
Number of clients C	1, 2, 4, **8**
Client fraction F_C	0.25, 0.5, 0.75, **1.0**

Table 3.2: Table listing the parameters and values tested for trajectory prediction. The values chosen for the baseline are marked as bold.

The following experiments were conducted, where "[]" indicates what parameters were tested, "n" is short for *non*, and "b" is short for *balanced*, where necessary.

3.2.3.1 Batch size

Experiment 1 evaluated the influence of batch size B and the results affected the choice of B for all following experiments. This was done using 2 clients and IID and balanced data. The following variables were tested:

1:	B	L_r	C	F_C	Optimizer	Data
	$[4, 8, 16, 32]$	$1\text{e}-4$	2	1.0	FedAvg	IID balanced

3.2.3.2 Learning rate

Experiment 2 evaluated the influence of learning rate L_r and the results affected the choice of L_r for all following experiments. This was done using 2 clients and IID and balanced data. The following variables were used:

2:	B	L_r	C	F_C	Optimizer	Data
	8	[1e-5,5e-5,1e-4,5e-4]	2	1.0	FedAvg	IID balanced

3.2.3.3 Federated optimization algorithms

Experiment 3 evaluated the influence of the federated optimizers FedAvg, FedAvgM, and FedProx. The results affected the choice of optimizer for all the following experiments. It was only run for 500 epochs, due to the maximum performance being reached after only 100 epochs. This was done using 2 and 8 clients and non-IID and balanced data. The following variables were used:

3:	B	L_r	C	F_C	Optimizer	Data
	8	$1e-4$	2	1.0	[FedAvg,FedAvgM,FedProx]	non-IID balanced

3.2.3.4 Variations of IID and balance

Experiment 4 evaluated the influence of variations of data distributions IID balanced, IID non-balanced, non-IID balanced, and non-IID non-balanced. The results affected the choice of dataset distributions for the following experiments. This was done using 2 and 8 clients, with the following variables:

4:	B	L_r	C	F_C	Optimizer	Data
	8	$1e-4$	$[2, 8]$	1.0	FedAvg	[IID b, IID n-b, n-IID b, n-IID n-b]

3.2.3.5 Number of clients

Experiment 5 evaluated the influence of number of clients C. The results affected the choice of C for the following experiments. This was done using both IID balanced and non-IID balanced data, with the aim to evaluate if the data has an influence on the performance in combination with varying C. The following variables were used:

5:	B	L_r	C	F_C	Optimizer	Data
	8	$1e-4$	$[1, 2, 4, 8]$	1.0	FedAvg	[IID b, n-IID b]

3.2.3.6 Client fraction

Experiment 6 evaluated the influence of client fraction F_C, the fraction of clients randomly selected for each round of FL. Using $C = 8$ clients synergized well with the chosen fractions [0.25, 0.50, 0.75, 1.0], corresponding to [2, 4, 6, 8] clients sampled per round. The results affected the choice of F_C for the following experiments. This was done using both IID balanced and non-IID balanced data, with the aim to evaluate if the data has an influence on the performance in combination with varying F_C. The following variables were used:

6:	B	L_r	C	F_C	Optimizer	Data
	8	$1e-4$	8	$[0.25, 0.5, 0.75, 1.0]$	FedAvg	[IID b, n-IID b]

3.2.3.7 FL versus CL

Experiment 7: compared different scenarios of FL and CL to validate the effectiveness of FL. This includes local CL runs using 864 samples from Boston (864B) and likewise for Singapore (864S). Two more CL runs were done, using half of the dataset with an equal number of samples from each region (432B+432S) and the full dataset (864B+864S). The corresponding FL setups with 2 clients, simulating two regional servers not able to share data with each other, were done for comparison. This experiment evaluates the performance benefit of making previously inaccessible data accessible through FL, by comparing 864B CL, 864S CL, and 864S+864S FL. In addition, it is also possible to draw conclusions on the difference in the performance of FL and CL when using the same data, using both the full (862B+864S) and half datasets (432B+432S).

7:	B	L_r	C	F_C	Optimizer	Data
	8	$1e-4$	$[1, 2]$	1.0	FedAvg	n-IID b [full and half datasets]

4

Results and analysis

In the Results and analysis chapter, we present the results and findings obtained from the conducted experiments. First, the FL implementation validation results on MNIST are presented alongside an initial analysis. This is followed by the main trajectory prediction using FL experiments, including a simple analysis of the results and their implications. The terms round and epoch are used interchangeably throughout the chapter as all experiments use one epoch of data per round.

4.1 Implementation validation results

In this section, we present the results of our implementation validation experiments conducted on the MNIST dataset. These tests were performed to investigate the performance implications of our FL model for various parameter configurations and settings. We considered different learning rates, batch sizes, number of clients, client fraction rates, federated optimizers, and data distribution types (IID and non-IID). Training speed refers to the time needed to complete one round (epoch) of training. The convergence rate refers to the number of rounds needed for convergence.

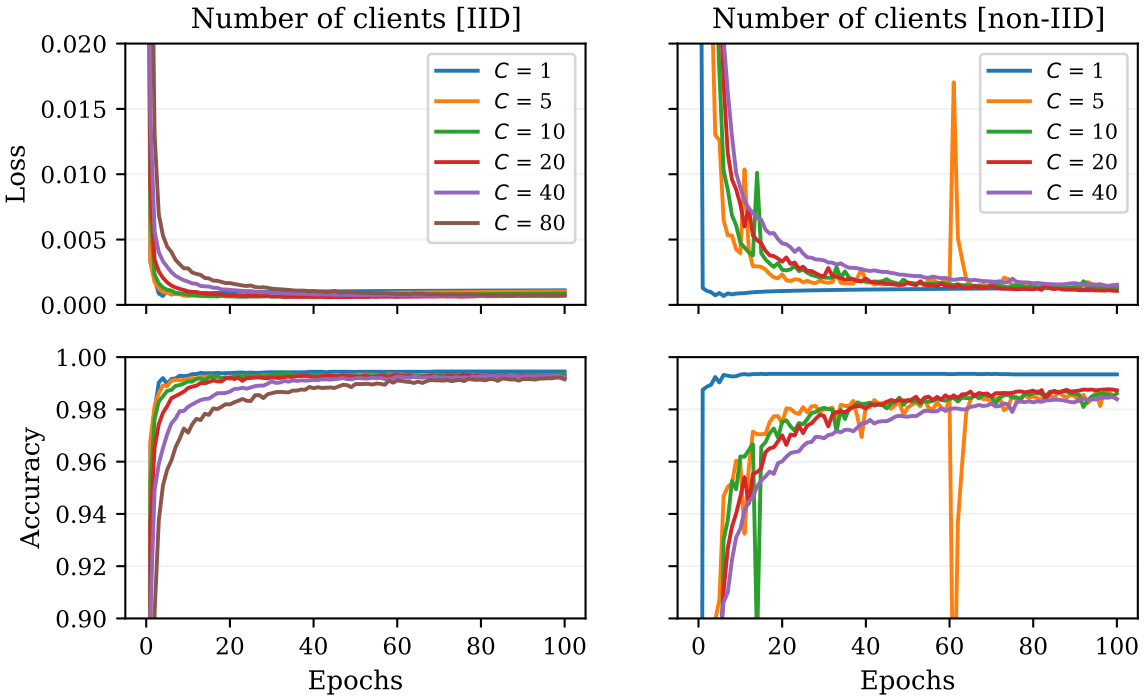


Figure 4.1: Accuracy and loss are displayed as a function of the number of epochs for different number of clients with IID (Experiment 5) and non-IID data distribution (Experiment 6) between the clients. The tests were performed with the FedAvg optimizer and all clients were involved in each round of training.

To demonstrate the results, we generated a collection of graphs that illustrate the accuracy and loss metrics over the rounds. As an example, the complete results from Experiment 5 and Experiment 6, investigating the impact of the number of clients on IID and non-IID data, are presented in Figure 4.1. These results will be discussed later in this section in combination with the other findings. Displaying the loss per epoch provides a quantitative measure of the discrepancy between the predicted outputs and the actual outputs, while the accuracy provides information about the performance in terms of the percentage of correctly classified predictions. Both measurements are relevant while evaluating the model since they capture different aspects of the learning progress. Nevertheless, they tend to show similar behavior. Therefore, a combined collection of only the accuracy metrics from all Experiments 1-10, described in Section 3.1.3, are presented in Figure 4.2. The corresponding loss graphs are shown as a function of epochs in Figure A.1 in Appendix A.1.

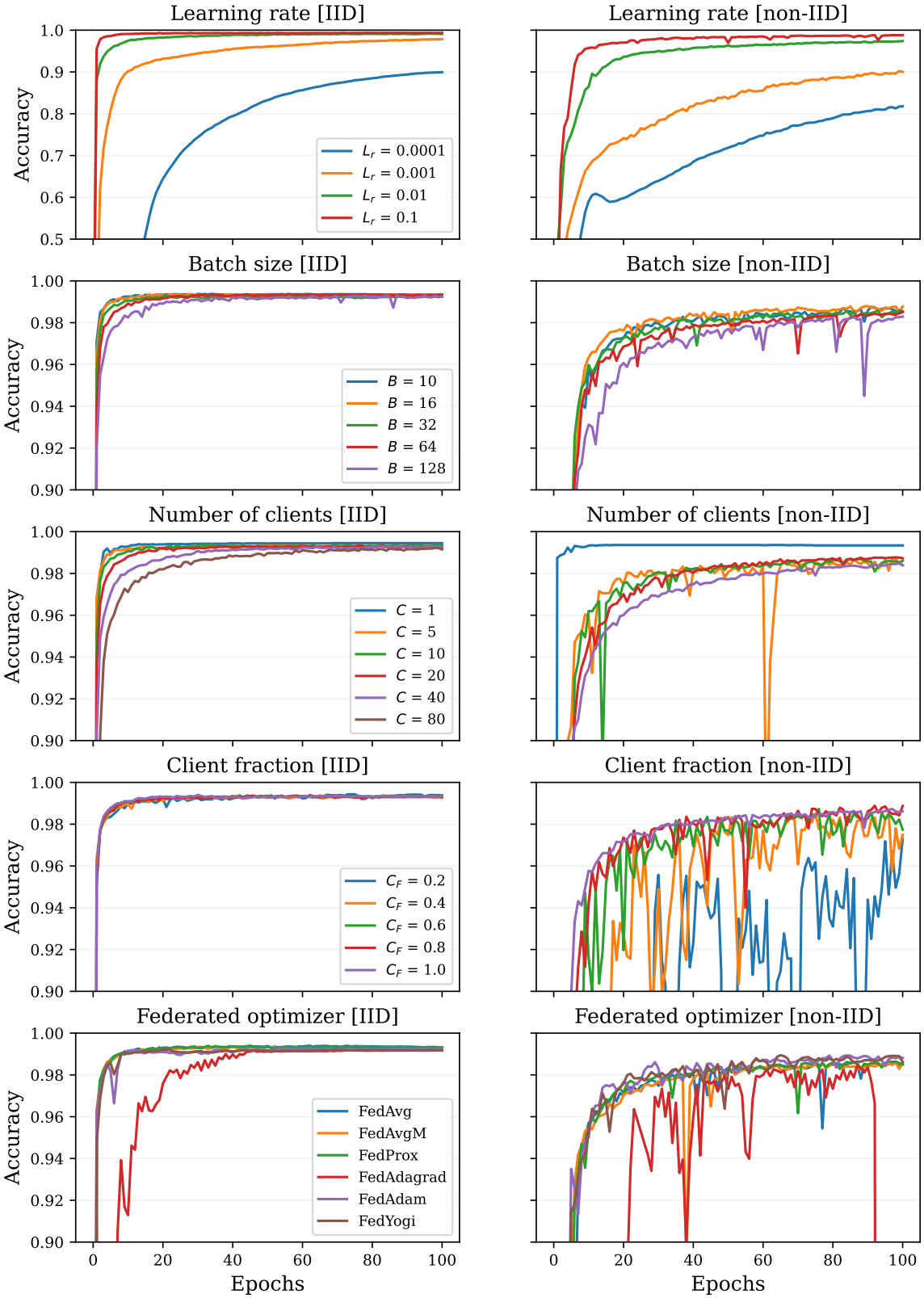


Figure 4.2: Accuracy as a function of epochs for experiments 1-10, described in Section 3.1.3. All axes, except for the graphs in the top row, display the same interval. The accuracy range for the learning rate graphs is five times larger.

Initially, we examined the influence of learning rate and batch size on accuracy and convergence rate through Experiment 1 to Experiment 4. The corresponding results are presented in the first and second rows in Figure 4.2. Comparing the outcome of training on IID data and non-IID data, we observed a decrease in both accuracy and convergence rate for the latter. Both learning rate and batch size are hyperparameters that can affect convergence and they do have a coupled effect. Modifying one value alters how the other parameter influences the model's learning process, including the frequency and magnitude of weight updates. A larger batch size utilizes the GPU more efficiently, resulting in faster training speed, but also requires more memory. These experiments indicate that 0.1 as learning rate is the best option, both for IID data and non-IID data. Further, the batch size can be set to a maximum of 32 without negatively affecting the convergence rate or performance. A batch size of 32 was also more stable than the larger values on batch size.

The impact of the number of clients used and the client fraction for each round was thereafter tested, in the execution of Experiments 5-8. For IID data, the number of clients had a direct impact on the convergence rate, with more clients resulting in a slower convergence rate. However, the final achieved accuracy was not significantly affected by the number of clients used. Notably, the tested values of client fraction did not have any clear impact on either the accuracy or the convergence rate. Note that a higher client fraction leads to training on more data samples for each round, since additional local datasets are used. For the non-IID data, where each client got the majority of its data from one specific class (digit), all tests with multiple clients were outperformed by CL (one client). Remember that with only one client, the data being IID or non-IID has no practical impact since the client receives all the data. As previously seen in the experiments altering batch size, both accuracy and convergence rate was negatively impacted by non-IID data for all tests on client fraction, but to a much larger extent. There was also a larger difference between the different client fraction values, quickly performing worse and more unstable with lower client fractions.

The results from Experiments 9 and 10 are illustrated in the last row in Figure 4.2, showing the impact of the different federated optimizers. On the IID data, all algorithms except FedAdagrad displayed almost identical learning. Although FedAdagrad was more unstable and had a lower convergence rate on the IID data, it reached the same level of accuracy as the other optimizers shortly after 40 epochs. On non-IID data, however, it was increasingly unstable. FedAvgM was also more unstable on the non-IID data, although not as severe as FedAdagrad. The other optimizers (FedAvg, FedProx, FedAdam, and FedYogi) showed more stable behavior and better performance overall.

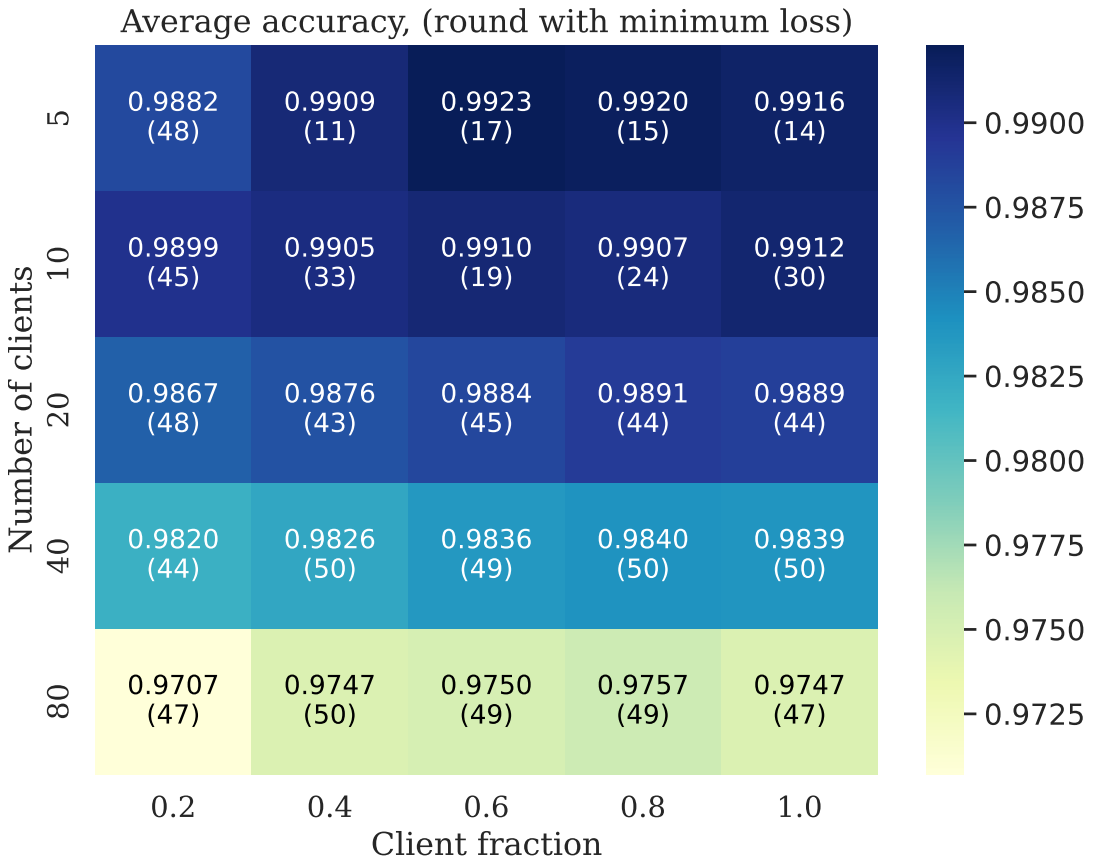


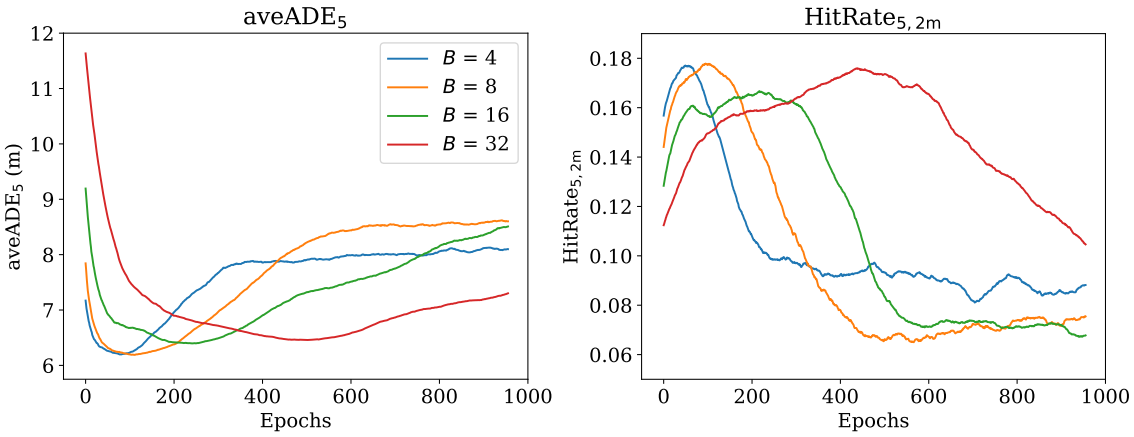
Figure 4.3: Experiment 11, altering both the number of clients and the client fraction. Each square displays the average accuracy over rounds 1-100 (and the round where the minimum loss value was registered, in parenthesis). This experiment was conducted on IID data.

Figure 4.3 presents the results from Experiment 11, where both the number of clients and the client fraction are altered for IID data. For each pair of values for these parameters, the average accuracy over the 100 rounds is listed, together with the round in which they achieved their minimal loss value, in parenthesis. The color of the background is based on the value for the average accuracy. The idea of the Figure is to account both for the final achieved accuracy and the convergence rate since both of these value affects the average accuracy. Note again that a higher client fraction corresponds to training on more data sampled for each round compared to a lower client fraction. As previously seen, fewer total clients lead to faster convergence rates. In addition, the client fraction has only a small effect on the overall results, with a very slight decrease in average accuracy and slightly slower convergence rates for smaller client fractions. This is consistent with the previous results using IID data.

4.2 Trajectory prediction results

This section presents the results of the trajectory prediction experiments on the nuScenes dataset. The metrics used for evaluation are minADE_1, minADE_5, aveADE_5, FDE_1 and $\text{HitRate}_{5,2\text{m}}$, all presented in tables. Only aveADE_5 and $\text{HitRate}_{5,2\text{m}}$ are visualized in the plots, as they were considered to be adequate since all metrics showed similar relative behavior for all experiments. All experiments are evaluated on the same validation set consisting of 512 samples each from Boston and Singapore. A higher HitRate↑ equals better performance, while the opposite is true for the displacement error metrics ADE↓ and FDE↓. This is included in all tables where our best results are marked in bold text. A rolling mean over 40 epochs has been applied to all data presented in the figures. This was done to make the results easier to analyze since the raw data is noisy, a consequence of the stochastic nature of the training algorithm and data sampling. In addition, small differences can not be assigned to the evaluated parameters with certainty, as they can in many cases be explained by randomness. All experiments were run locally for one epoch E of data for each selected client per round, meaning that the terms epoch and round are used interchangeably. Performance refers to the minimum (or maximum for HitRate) of a given metric. Training speed refers to the time needed to complete one round (epoch) of training. Convergence rate refers to the number of rounds needed for convergence.

4.2.1 Batch size



(a) aveADE$_5$ **(b)** HitRate$_{5,2m}$

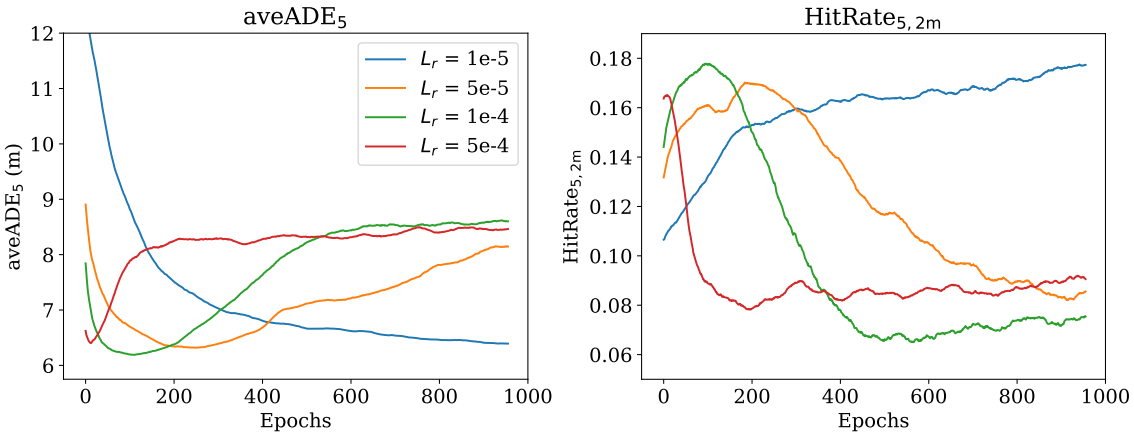
Figure 4.4: Metrics with varying batch sizes B. Fixed parameters: $C = 2$, $F_C = 1.0$, optimizer = FedAvg, $L_r = 1e-4$, and IID balanced dataset.

The first experiment follows Experiment 1 from Section 3.2.3.1 and establishes the effects of batch size B for the baseline model. Figure 4.4 highlights that a lower batch size results in a faster convergence rate, with no clear drop in performance. Instead, improved performance can be seen for the smaller batch sizes 4 or 8, seen for all metrics in Table 4.1. The faster convergence rate is especially important to keep in mind when scaling to more clients where the convergence rate gets slower. Larger batch sizes utilize the GPU more efficiently, providing faster training speeds. However, this only works up to a certain point where the memory requirements are too large and the training fails. This happened at $B = 64$. Batch size $B = 8$ was considered a good compromise of the convergence rate and training speed, keeping an overhead of the convergence rate for scaling up the number of clients. The results from CoverNet [25] using $B = 64$ are also included in the table, where the performance is better for all metrics except for HitRate. Still, our results are close enough to show that our implementation works as intended.

Table 4.1: All metrics with varying batch sizes B and results from CoverNet [25].

	minADE$_1$ ↓	minADE$_5$ ↓	aveADE$_5$ ↓	FDE$_1$ ↓	HitRate$_{5,2m}$ ↑
$B = 4$	5.66	**2.67**	6.20	12.09	0.177
$B = 8$	**5.60**	2.77	**6.19**	**11.82**	**0.178**
$B = 16$	5.87	2.77	6.40	12.27	0.167
$B = 32$	5.99	2.78	6.46	12.12	0.176
CoverNet	5.07	2.31	-	10.62	0.17

4.2.2 Learning rate



(a) aveADE$_5$ **(b)** HitRate$_{5,2m}$

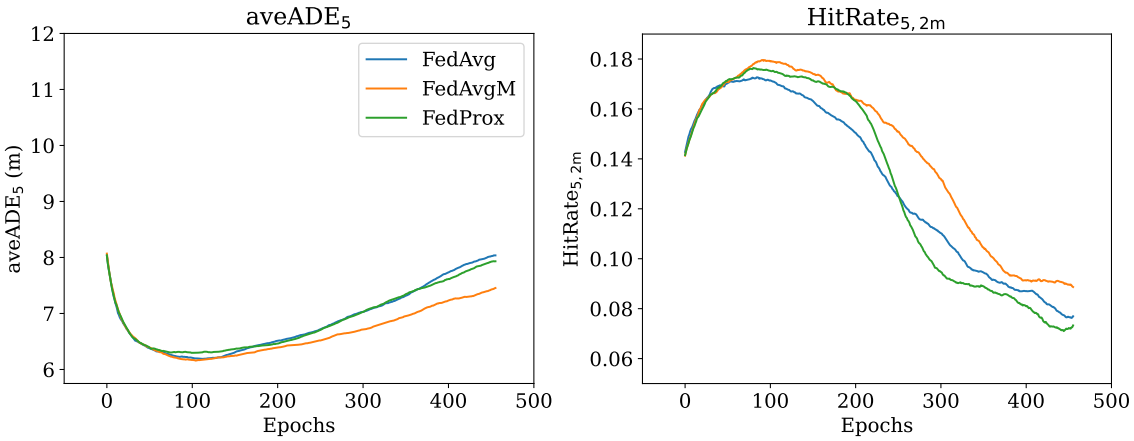
Figure 4.5: Metrics with varying learning rates L_r. Fixed parameters: $C = 2$, $F_C = 1.0$, optimizer = FedAvg, $B = 8$, and IID balanced dataset.

The succeeding experiment follows Experiment 2 from Section 3.2.3.2 and was done to choose a suitable learning rate L_r. Naturally, smaller learning rates slow training. Too large learning rates can cause divergence, a limit that is almost reached by the largest learning rate trialed here, $L_r = 5\text{e}{-4}$. The maximum performance is reached after only ten epochs, seen in Figure 4.5, while the performance does not reach as high compared to using smaller learning rates. Increasing the learning rate further caused divergence. Table 4.2 indicated that the performance was similar for all other learning rates. $L_r = 1\text{e} - 4$ was chosen to maintain good performance and a fast convergence rate, while minimizing the risk for divergence when scaling to more clients or changing other parameters. No learning rate scheduler was considered as this was not recommended by the original CoverNet publication [25].

Table 4.2: All metrics with varying learning rates L_r.

	minADE$_1$ ↓	minADE$_5$ ↓	aveADE$_5$ ↓	FDE$_1$ ↓	HitRate$_{5,2m}$ ↑
$L_r = 1\text{e} - 5$	5.70	**2.73**	6.39	12.02	**0.178**
$L_r = 5\text{e} - 5$	5.72	2.75	6.32	12.11	0.170
$L_r = 1\text{e} - 4$	**5.60**	2.77	**6.19**	**11.82**	**0.178**
$L_r = 5\text{e} - 4$	5.89	2.76	6.40	12.62	0.165

4.2.3 Federated optimization algorithms



(a) aveADE$_5$ **(b)** HitRate$_{5,2m}$

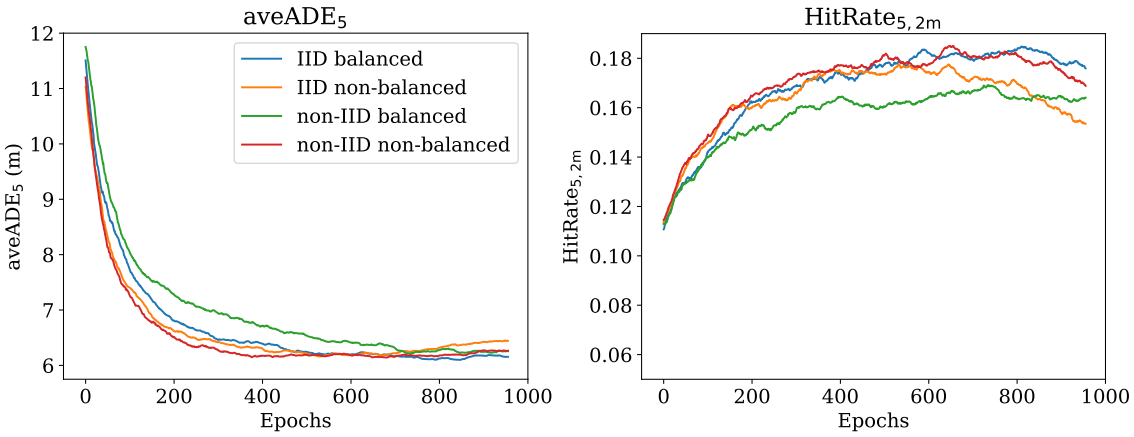
Figure 4.6: Metrics with FedAvg, FedAvgM and FedProx, all using 2 clients. Fixed parameters: $C = 2$, $F_C = 1.0$, $L_r = 1\text{e} - 4$, $B = 8$, and non-IID balanced dataset.

By following Experiment 3 from Section 3.2.3.3, we evaluated the three federated optimizers FedAvg, FedAvgM, and FedProx with the number of clients $C = 2$ and $C = 8$. This was only done for 500 epochs, due to the maximum performance being reached after only 100 epochs. The results when using 2 clients are presented in Figure 4.6 and Table 4.3. All results for 8 clients are included in Appendix A.2. The FedOpt optimizers FedAdam, FedAdagrad and FedYogi are not included as they diverged for all our tested parameters. Since the federated optimizer FedProx is designed to cope with non-IID data more efficiently than FedAvg or FedAvgM, with no significant difference for IID data, the optimizers are evaluated on non-IID and balanced data. With 2 clients, FedProx reached the best performance in FDE$_1$. However, there are no other apparent benefits indicating a performance benefit for this task as the performance is similar to the other optimizers for all other metrics. There were no differences in training speed or convergence rate either, despite FedAvgM being designed to converge faster than FedAvg. Only minor differences could be seen when running the same experiment on non-balanced or IID data. For 8 clients, FedAvgM performs worse, specifically for FDE$_1$. Again, the difference is not apparent for all metrics, indicating no real advantage for either optimizer.

Table 4.3: Metrics with FedAvg, FedAvgM and FedProx using 2 clients.

	minADE$_1$ ↓	minADE$_5$ ↓	aveADE$_5$ ↓	FDE$_1$ ↓	HitRate$_{5,2m}$ ↑
Avg, $C = 2$	5.56	2.71	6.19	11.76	0.173
AvgM, $C = 2$	5.63	2.71	**6.16**	11.93	**0.180**
Prox, $C = 2$	**5.48**	**2.70**	6.30	**11.62**	0.176

4.2.4 Variations of IID and balance



(a) aveADE$_5$ **(b)** HitRate$_{5,2m}$

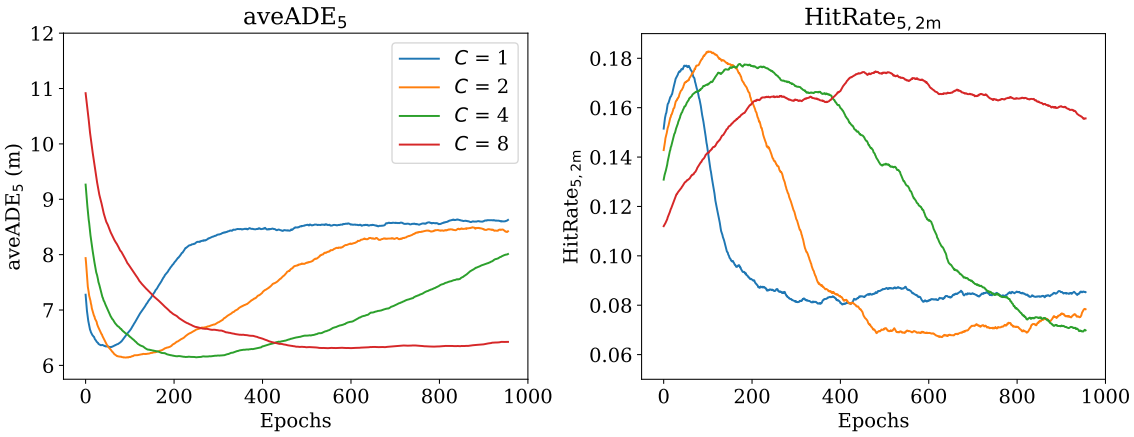
Figure 4.7: Comparison of different datasets using all combinations of IID, non-IID, balanced, and non-balanced datasets. Fixed parameters: $C = 8$, $F_C = 1.0$, optimizer = FedAvg, $L_r = 1e-4$, and $B = 8$.

Since no large differences could be observed for the federated optimizers and due to time restrictions, only FedAvg was used to evaluate the four dataset variations IID balanced, IID non-balanced, non-IID balanced, and non-IID non-balanced. It was done following Experiment 4 from Section 3.2.3.4. We observe that IID balanced and non-IID non-balanced perform equally well in all metrics, while the performance is worse when using the two other datasets. This is seen in Figure 4.7 and Table 4.4. The non-IID balanced dataset is consistently the worst performing in all metrics. Worth noting is that the non-balanced datasets contain more data, $\sim 17\%$. This could explain the difference between the two non-IID datasets.

Table 4.4: Results from different datasets using all combinations of IID, non-IID (n-IID), balanced (B), and non-balanced (n-B) datasets.

	minADE$_1$ ↓	minADE$_5$ ↓	aveADE$_5$ ↓	FDE$_1$ ↓	HitRate$_{5,2m}$ ↑
IID B	**5.41**	2.57	**6.10**	**11.54**	**0.185**
IID n-B	**5.41**	2.55	6.16	11.79	0.178
n-IID B	5.57	2.65	6.21	11.83	0.169
n-IID n-B	5.51	**2.54**	6.14	**11.54**	**0.185**

4.2.5 Number of clients



(a) aveADE$_5$ **(b)** HitRate$_{5,2m}$

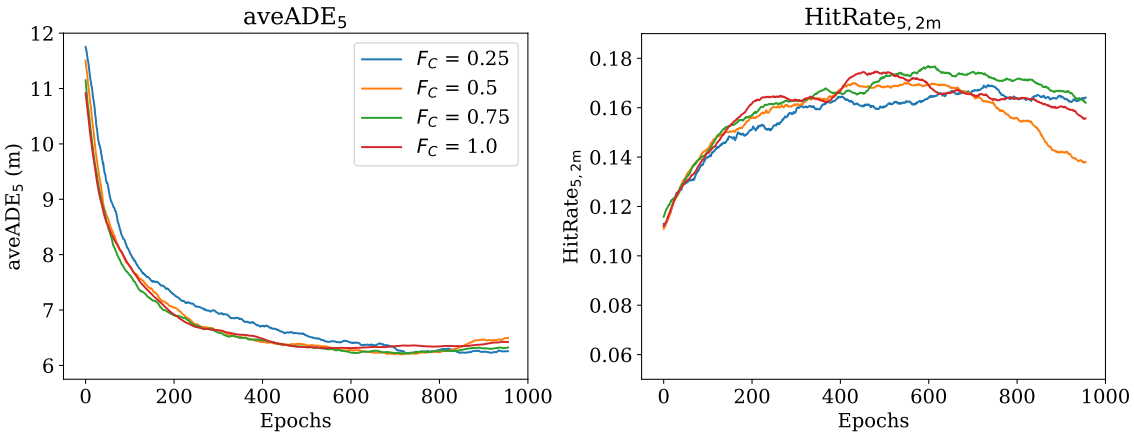
Figure 4.8: Metrics with varying number of clients C. Fixed parameters: $F_C = 1.0$, optimizer = FedAvg, $L_r = 1\text{e} - 4$, $B = 8$, and non-IID balanced dataset.

This experiment follows Experiment 5 from Section 3.2.3.5. Increasing the number of clients C slows down both the training speed (due to limitations of memory usage) and convergence rate, seen in Figure 4.8. However, there is no clear performance deficit when using more clients, as previously seen for the MNIST experiments in section 4.1. This holds true for non-IID and IID data, as seen in Table 4.5. The plots for IID data can be found in Appendix A.3. 8 clients were the maximum possible to use in combination with the nuScenes dataset, due to limitations of memory usage in combination with the available hardware.

Table 4.5: All metrics with varying number of clients C using IID and non-IID (n-IID) data.

	minADE$_1$ ↓	minADE$_5$ ↓	aveADE$_5$ ↓	FDE$_1$ ↓	HitRate$_{5,2m}$ ↑
$C = 1$ IID	5.55	2.64	6.17	11.84	0.173
$C = 2$ IID	5.60	2.77	6.19	11.82	0.178
$C = 4$ IID	5.59	2.68	6.24	11.74	0.172
$C = 8$ IID	**5.38**	2.66	6.24	11.41	0.178
$C = 1$ n-IID	5.75	2.70	6.33	12.15	0.177
$C = 2$ n-IID	5.69	**2.61**	**6.14**	**11.61**	**0.183**
$C = 4$ n-IID	5.53	2.66	6.15	11.92	0.178
$C = 8$ n-IID	5.70	2.56	6.31	11.62	0.175

4.2.6 Client fraction



(a) aveADE$_5$ **(b)** HitRate$_{5,2m}$

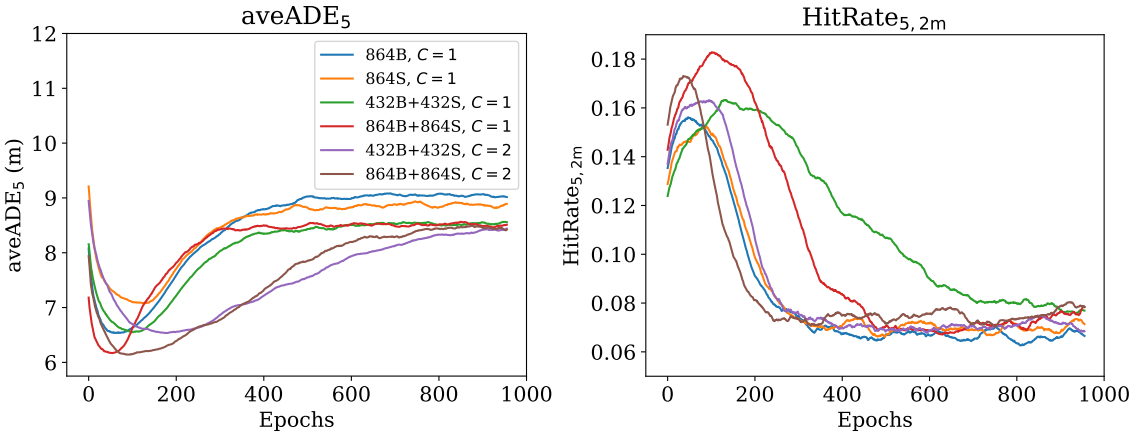
Figure 4.9: Metrics with varying client fraction F_C. Fixed parameters: $C = 8$, optimizer = FedAvg, $L_r = 1e-4$, $B = 8$, and non-IID balanced dataset.

Following Experiment 6 from Section 3.2.3.6, the client fraction experiments were done with $C = 8$, synergizing well with the chosen fractions [0.25, 0.50, 0.75, 1.0], corresponding to [2, 4, 6, 8] clients sampled per round. Changing the client fraction F_C does not substantially change the results in Figure 4.9 and Table 4.6. Plots for IID can be found in Appendix A.4. A lower client fraction utilizes fewer data samples per round, resulting in fewer computations and therefore faster training speed. For IID data, client fraction 0.25 performs the best for some metrics, indicating that smaller client fractions can improve both speed and at least maintain performance in IID scenarios. For the non-IID setup, the larger client fractions tend to perform slightly better.

Table 4.6: All metrics with varying client fraction F_C for IID and non-IID (n-IID) balanced data.

	minADE$_1$ ↓	minADE$_5$ ↓	aveADE$_5$ ↓	FDE$_1$ ↓	HitRate$_{5,2m}$ ↑
$F_C = 0.25$ IID	5.41	**2.57**	**6.10**	11.54	**0.185**
$F_C = 0.5$ IID	5.71	2.66	6.23	11.96	0.176
$F_C = 0.75$ IID	**5.36**	2.62	6.17	**11.35**	0.181
$F_C = 1.0$ IID	5.38	2.66	6.24	11.41	0.178
$F_C = 0.25$ n-IID	5.54	2.65	6.21	11.83	0.169
$F_C = 0.5$ n-IID	5.51	2.70	6.20	11.70	0.170
$F_C = 0.75$ n-IID	5.53	**2.57**	6.22	11.70	0.177
$F_C = 1.0$ n-IID	5.63	2.56	6.31	11.62	0.175

4.2.7 FL versus CL



(a) aveADE$_5$ **(b)** HitRate$_{5,2m}$

Figure 4.10: Validation of the performance benefits of FL ($C = 2$) compared to local CL ($C = 1$) in Boston (B) and Singapore (S). Fixed parameters: $F_C = 1.0$, optimizer = FedAvg, $L_r = 1e-4$, and $B = 8$.

In Figure 4.10 and Table 4.7, we compare FL to CL following Experiment 7 from Section 3.2.3.7. This includes two local CL runs using 864B and 864S, respectively, two CL runs using full (864B+864S) and half-sized (432B+432S) mixed datasets from both regions, and corresponding non-IID FL setups with 2 clients. The FL setup with 864B+864S performs significantly better than local 864B and 864S CL runs, hence showcasing the potential performance benefit of accessing more data with FL when CL is not feasible. CL using only 864S is consistently the worst performing. CL with 864B and CL and FL with 432B+432S all perform equally better. This might indicate that the Boston data represents the validation dataset better than the Singapore data, despite the balanced validation data. CL achieves comparable performance to FL when trained with the same datasets. However, all CL runs are faster than FL both in terms of training speed and convergence rates.

Table 4.7: Validation of the performance benefits of FL ($C = 2$) compared to local CL ($C = 1$) in Boston (B) and Singapore (S).

	minADE$_1$ ↓	minADE$_5$ ↓	aveADE$_5$ ↓	FDE$_1$ ↓	HitRate$_{5,2m}$ ↑
864B, $C = 1$	5.61	2.72	6.53	12.16	0.156
864S, $C = 1$	5.73	2.94	7.08	13.14	0.152
432B+432S $C = 1$	5.57	2.77	6.55	12.06	0.163
864B+864S $C = 1$	**5.45**	2.64	6.17	11.84	**0.183**
432B+432S $C = 2$	5.70	2.90	6.53	12.13	0.163
864B+864S $C = 2$	5.47	**2.61**	**6.14**	**11.61**	0.173

5
Discussion

In this section, we discuss the key findings from our experiments and the limitations of the results. The implications and subsequent challenges are also presented. Lastly, ethical considerations related to this study are discussed.

5.1 Implementation validation discussion

Besides verifying that the FL setup worked as intended, we also performed a set of FL experiments on the MNIST dataset that serve as a comparison baseline for the trajectory prediction experiments. This needs to be approached with caution, as the results can not be transferred between contexts. Yet, they are useful when interpreting the trajectory prediction result.

The FL MNIST experiments in Section 4.1 highlight that FL converges slower than CL in terms of speed, but eventually reaches a similar performance level (loss and accuracy). The learning rate had no impact other than the convergence rate, hence the largest learning rate was used for all experiments for maximized training speed. Smaller batch sizes induced a faster convergence rate and improved stability for non-IID data.

The performance of a central model (one client) was used as the benchmark and achieved a validation accuracy of approximately 99 %. The results show that FL with increasing numbers of clients takes longer to converge but eventually saturates at a similar level of performance when using IID data. This is true for all tested numbers of clients up to $C = 80$. Extending the client base would have been interesting, as some applications of FL require more than 80 clients [46]. Limitations of memory, using our specific hardware and FL framework, hindered such experiments in this study.

Using IID data yielded better performance than non-IID when evaluating FL. The non-IID data distribution introduced more noise and overall lower maximum performance in most experiments. CL performed better than FL using the non-IID data distribution. The IID and non-IID distribution methods only affect FL as explained in Section 3.1 and have no effect when applied to a single client (CL). As 10 clients and 10 classes were used in the FL experiments, each local client dataset mostly contained only one class and hence tried to learn different digits, affecting the convergence negatively. Decreasing the client fraction F_C for each round of FL

to less than 1.0 corresponded to lower performance, but increased training speed since less data is processed each round. It also introduced large fluctuations of loss and accuracy when using non-IID data. As explained for the number of clients experiments, each client tries to learn different digits. Choosing a fraction of the clients each round means that the model trains on only a few digits each round, resulting in large fluctuations in the trainable parameters. This could potentially be mitigated by using a smaller learning rate but was not done in this study due to time limitations and subsequent priorities. Learning is still somewhat hampered for larger F_C, but not substantially since more classes are represented in each round. Using no overlap of the classes resulted in no learning when tested, showing the importance of overlap for the MNIST classes when using FL.

The three optimizers in FedOpt were unstable for all our tested parameters, only converging after several tries with luck involved. Unfortunately, we were not able to find a set of parameters that improved this behavior. Nevertheless, when they converged, FedAdam and FedYogi performed better with smaller fluctuations than the other optimizers on non-IID data, following the advantages for heterogeneous data outlined in [40]. Overall, there were no substantial performance benefits of either federated optimizer except for FedAdagrad which performed poorly in all experiments. FedAvg was reliable in all scenarios with no major drawbacks and was therefore used in most of the experiments.

In summary of the MNIST experiments, we learned that for FL (compared to CL) convergence is slower, performance is similar for multiple clients and different client fractions using IID data, and our non-IID data distribution affects the stability negatively with slightly lower performance for both lower client fractions and more clients.

5.2 Trajectory prediction discussion

The trajectory prediction experiments on nuScenes follow the same general trend as the implementation validation MNIST results. The performance when using FL is similar to CL, while the training speed and convergence rate are slower and scale with the number of clients C. This increases our confidence in the potential performance benefits of applying FL in practice. However, many more aspects need to be taken into account for real-life applications. These include challenges of privacy protection when sending model updates (Section 2.2.2), potentially reduced communication costs, systems heterogeneity, and unreliable clients [46]. The learning rate and batch size experiments only affected convergence rates and were therefore chosen to minimize the run time for all experiments.

The results should be interpreted carefully due to our experimental setup. The dataset was small compared to the MNIST dataset, containing only 2016 data samples at maximum size. Additionally, all scenes were extracted from urban environments with no cross-validation testing, i.e., training on Boston data and validating on Singapore data. No cross-fold validation was done either, i.e., changing data

partitioning of the training and validation sets to explore unintentional data biases. The same validation set of 512 samples from each city was always used. For a car manufacturer, it is essential to validate any system in all relevant driving environments before making it available in the vehicles, hence why we chose to validate on all available geographical locations.

Using multiple clients resulted in slower training speeds and convergence rates, with a similar maximum performance to the benchmark CL setup. This time it was possible to compare to a benchmark set by the original CoverNet paper for this task [25], as done in Section 4.2.1. Here, our performance in all metrics is relatively close to the ones in the original paper, indicating that our model works as intended. With this result, we have enough confidence to draw conclusions about FL and its effectiveness relative to CL.

FL performed equally to CL on both IID and non-IID data, with slower convergence rates. However, reducing the client fraction F_C resulted in improved training speed. This time, a smaller F_C did barely have any negative effect on the performance of IID or non-IID data. The real-life problem of unreliable clients could therefore be argued to be mitigated in this case, as a lower F_C corresponds to simulating unreliability where only a fraction of randomly selected clients is chosen for each round. Furthermore, the three tested federated optimizers FedAvg, FedAvgM and FedProx did not have any noteworthy effects on the results on either dataset or distribution. The differences in the data samples from Boston and Singapore are not extreme, hence probably why the benefits of the FedProx optimizer do not show, and FedAvg was considered suitable to use in most experiments.

The arguably most important result from this study was presented in Section 4.2.7. We simulated the case where FL enables data from both Boston and Singapore to be used, compared to a scenario where only Boston or Singapore data is available in a CL scenario. It showed substantially improved performance when FL enables more data to be used. The performance scaled with more available and diverse data. Training using only 864 Singapore samples consistently performed the worst, while training on 864 Boston samples achieved almost the same performance as when using 432 samples from each city. This indicates that the data diversity matters in at least one of our two cases, not only the number of data samples available. Previously, we have shown that FL does not necessarily reduce performance in most scenarios. This result implies that FL can improve the generalization and possibly robustness of a trajectory prediction model, by unlocking additional or more diverse data samples. Overall, the results suggest that FL and CL learn equally well on the same dataset, with FL performing better when unlocking larger quantities of or more diverse data samples.

5.3 Ethical considerations

As ADAS/ADS technology, usually relying on ML, continues to advance, there are numerous ethical, societal and ecological issues that need to be addressed. Such

issues include transportation safety, concerns about liability, ethical dilemmas, environmental sustainability, privacy and security. The following section addresses these issues further.

5.3.1 Safety and ethical dilemmas

Some of the most commonly discussed issues in literature and media are the potential safety improvements and how they relate to liability and ethical dilemmas [47]. Implementing ADAS/ADS in vehicles has been shown to reduce the probability of accidents [48]. However, a side effect of these functionalities is that the driver's focus can be reduced, potentially making the safety improvements redundant in certain situations where human input is necessary. The level of safety required before full implementation of ADS can be done, with no assumption of driver focus or input, is subject to dispute [49]. Safety standards (such as ISO26262, the standard for road vehicles' functional safety) and future safety legislation will define the required safety level.

One common ethical dilemma often discussed in relation to autonomous vehicles is the trolley problem [47]. In this problem, a train heads down a track, and a choice has to be made between hitting one or several people. If no action is taken, five people are hit. If a lever is pulled to shift the train to a different lane, only one person will be hit. Should the lever be pulled? It is easy to think of similar examples more relevant to autonomous vehicles. What action should be taken if a vehicle needs to decide between hitting a person on a pedestrian crossing, turning left and risking hitting an oncoming car, or turning right to end up in the trench? How should the ADAS/ADS of a vehicle be trained on such scenarios, and what should the preferred outcome be? If minimizing injuries is the assumed criterion, is it realistic for the system to know how to act for a particular outcome? To calculate all involved objects' trajectories and make an informed decision based on the predictions of future paths and potential damages in each scenario? One of the goals of developing FL for automotive applications is to enable learning from corner cases [50], such as the situation described in this paragraph. It is established that to enable intelligent decision-making, the decision maker has to be well informed, in this case, about the vehicle's surroundings and trajectories of all objects. Continuous research of training methods for ADAS/ADS, including FL, can potentially contribute to improved decision-making and, consequently, road safety [48].

Several other aspects of ethical dilemmas can be discussed. Should the age of the involved people affect the choice, i.e., should the training include preferences of the attributes of people? Should the system be trained to mimic what a human driver would do? In order to minimize total injuries, it can be argued that the least amount of people or vehicles should be hit, weighed against the severity of each injury. However, this assumes that risk analysis for each possible scenario can be done in a fraction of a second. Biases affecting such choices can unintentionally be introduced during training, i.e., if the dataset is unbalanced. One prominent example of bias in an automotive setting is the weather [51]. Training on data

containing different weather and lighting scenarios is important to ensure the performance is acceptable across all conditions, including different geographical regions and climates. The same argument can be made about traffic scenarios where ethical dilemmas arise; variety is essential to minimize unintentional biases and improve generalization. This further motivates research of FL and other methods that can potentially improve the variety of the available data [46].

From the scope of this study, no active choices have to be made, as only trajectory prediction will be studied. Trajectory planning and control systems, trained to actively minimize risk on the road, are outside the scope. However, the performance of the trajectory prediction model (including object detection) affects the performance of the overall ADAS/ADS functionality. A famous example of the functionality breaking down was when a Tesla car hit a stationary white truck on a highway in 2016 without attempting to avoid the collision [52]. It is claimed that neither the driver nor ADAS/ADS could distinguish the white trailer from the brightly lit sky, illustrating the consequences if not all systems work as intended.

While discussing the performance influence of FL on ADAS/ADS, a comparison to other training methods is needed. Previous studies have shown the potential of FL to perform on a similar level to CL [53] [54]. If FL performs worse than a corresponding CL method, how much worse is acceptable? How is this weighed against the privacy and potential cost benefits of FL? Can these benefits ever be measured against the deterioration of the safety system's performance? In practice, the performance of FL probably needs to be on par with or better than CL before any full-scale implementation in a fleet of vehicles is considered. Since road safety is one of the main concerns for automotive manufacturers and their customers, caution is to be expected [3].

Another common topic relevant to ADS specifically is the question of liability [47]. Who is to blame in an accident caused by an autonomous vehicle? Should the vehicle always prioritize its passengers? Volvo Cars, among other large entities within the automotive industry, has promised to take full liability when their vehicles are in autonomous mode and at fault in an accident, according to a press release [55]. Representatives from Mercedes and Google have made similar claims in interviews [56]. This provides an answer for now, but questions remain regarding how this will be implemented in practice. Will full liability really be taken, or is it an attempt to influence the perception of the public who is not read up on the fine print?

5.3.2 Ecological aspects

One of the main discussions regarding the future of transportation deals with lowering emissions. How autonomous vehicles can affect emissions is a topic not discussed as frequently [47]. Developed to increase efficiency, both regarding eco-driving and route planning, the average vehicle emits less greenhouse gases (GHG) compared to the average vehicle controlled by a human [57]. However, a potential drawback could be an increase in the number of passenger cars on the road, due to enabling easier

travel compared to manual driving. This could replace public transport alternatives, which by their design are more efficient for a given choice of propulsion method. One option to deal with this issue is to use ADS capabilities to replace passenger cars with a fleet of autonomous buses or taxis. Such a taxi service is currently offered by Waymo in certain parts of the US [58]. Depending on how the distribution of vehicles with ADS functionality changes in the future, one might argue both positive and negative effects of emissions of GHG. Assuming equal ADS development in all transportation sectors, a net positive effect of emission reduction should be expected [57].

Furthermore, traffic congestion contributes to the waste of energy and consequently negative environmental impact [59]. A traffic system where all vehicles are autonomous should in theory be able to reduce congestion and improve the flow of the traffic system. A critical penetration rate of autonomous vehicles for a high potential of traffic flow improvements is 40% [60]. Therefore, introducing autonomous vehicles in the current system dominated by manual driving might initially negatively impact the traffic flow, but will eventually have positive effects. Trajectory prediction models, performing well in a wide range of possible traffic scenarios and environments, are essential for such an improvement.

FL applied at the client level can be defined as edge computing. By processing data closer to the data source, the vehicle's sensors in this instance, overall efficiency can be improved as there is no need for costly data transfer. Additionally, data traffic is reduced and a larger amount of the available processing power at the vehicle level is utilized [29]. This can reduce the need for energy-demanding cloud computing, i.e., data centers, which have been estimated to be responsible for roughly 1 % to 1.5 % of the world's total energy consumption [61] [62]. Increasing the number of computations done by the vehicle's computers will naturally increase local power demand, but it is not known if the total energy consumption would reduce by changing from cloud computing to FL for a vehicle fleet. One estimation claims that the required computing hardware development for the energy consumption to break even in 2050 in such a scenario is not probable to happen, based on the development rate today [63]. This is however based on many assumptions and the prediction is yet to be verified in practice.

5.3.3 Privacy

As discussed earlier in this study, one of the main advantages of FL is the potential for strengthened data privacy and security. This is due to the advantages of FL where no data collection is required, while the most common data leakage occurs during collection and processing. Nonetheless, measures are still needed as FL is not inherently secure by default, since for example reversed deduction of model updates can be done to produce the training data [46]. With such measures applied, improved privacy is possible to achieve. Another potential benefit of FL is that data does not need to be shared between regions, for example, countries with different views on data protection like Sweden and China [46]. This would potentially allow

for greater global collaboration without breaching the local data privacy legislation of different nations.

5.3.4 Economical aspects

Another issue concerning the broad topic of vehicle safety is the premiumization of advanced safety features. With time, it is natural for vehicle safety to improve. Consequently, prices increase as new innovations are introduced. Throughout history, passive crash protection has been the main focus, but in recent years ADAS/ADS functionalities have been given increasing amounts of attention and research resources, with forecasts showing the global market for ADAS tripling from 2021 to 2030 [64]. To actively avoid an accident is naturally a desirable outcome, but these new and improved safety features are usually offered at premium prices. This comes down to the substantial costs of adding hardware (sensors), software platforms, and the development associated with adding new technology. From an ethical point of view, is it ethical to charge premium prices for safety? Is it ethical to only make these systems available to the wealthy? Over time, new and desirable features trickle down to cheaper vehicle segments, resulting in a wider spread of safety technology. Does this increase the base price of vehicles? The average price of passenger cars in Sweden increased from 34 000€ to 39 000€ from 2015 to 2020 (adjusted for inflation), with some countries in Europe experiencing even higher price hikes [65], partly due to raising the baseline for safety standards. Similarly to premium passive safety features of the past, active safety (including ADAS/ADS) can be expected to be common in a majority of sold vehicles around the world in the near future. In the EU in particular, all newly sold road vehicles from 2024 will be required to be equipped with certain ADAS features [66], depending on the vehicle type. A few examples; all road vehicles will need to be equipped with adaptive cruise control and driver attention warning systems, cars and vans need lane-keeping assistance and emergency braking, while buses and trucks need various warning systems. These changes are inevitable, meaning that it will get more difficult for the common family to afford a new, but safer, car.

6
Conclusion

Driving automation has the ability to improve safety and efficiency in traffic. Active safety systems, such as ADAS and ADS, face challenges related to robustness and the need to adhere to stringent requirements, including data privacy regulations. Robustness can be improved through continuous learning from diverse driving scenarios, including edge cases, and therefore requires large quantities of data. Collecting this data from a fleet of customer cars (clients) is not feasible due to large communication costs and the data privacy regulations. FL is a possible solution that addresses these challenges as it enables learning while keeping client data local. This can allow for accessing otherwise unavailable data and increases the chances of observing edge cases.

Although previous studies have investigated the effects of FL and have covered some applications within an automotive setting, there has been no evaluation of the influence of FL on trajectory prediction performance for a geographically diverse dataset. Therefore, this study includes an implementation of FL to train a trajectory prediction model on a dataset collected in different geographical regions, namely Boston and Singapore. Limitations of our experiments include no practical testing on several hardware devices, size limitations of the used dataset, and a restricted model search.

The performance influence of FL was tested on two datasets, MNIST and nuScenes, through a set of similar experiments. The results indicate that a larger number of clients in FL yields slower convergence rates. Still, a similar final accuracy is eventually reached. Choosing a smaller client fraction, simulating unreliable clients, reduces the computational resources needed for each round. However, using a lower fraction with non-IID data reduced the performance in the MNIST experiments, while performance remained the same for the nuScenes experiments. Most notably, the results show that overall model performance can be improved through the use of the data FL unlocks compared to CL in the imagined scenario. In this scenario, FL enables data samples from both Boston and Singapore to be used for learning, compared to only having data from one city available for CL.

Based on our results and limitations, there are multiple research paths that could be further explored. These include investigating FL for other experiments, using more clients, using larger datasets, or with more complex or diverse datasets or models. The privacy methods outlined in Section 2.2.2 can be explored and evaluated in a suitable application. Out-of-distribution performance, for example, training a model on datasets A and B and testing it on dataset C, can be researched. Another

suggestion is to evaluate FL for a model architecture that is able to handle sensor data from varying sensor setups, i.e., vehicles with different combinations of cameras, LiDARs, and RADARs. Unsupervised representation learning may be used to achieve this [67] and can eventually be done by modeling similar feature representations from different sensors and merging the outputs into a single feature vector. In combination with FL, this could demonstrate a positive effect by unlocking data from a more diverse set of vehicles in various geographical regions. Another example of important future work is to implement FL in practice on hardware devices acting as clients and servers. This can be done to evaluate real communication and reliability challenges and is arguably one of the most important building blocks needed to realize FL.

Bibliography

[1] S. Pouyanfar, S. Sadiq, Y. Yan, *et al.*, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, Sep. 2018, ISSN: 0360-0300. DOI: 10.1145/3234150. [Online]. Available: https://doi.org/10.1145/3234150.

[2] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, "Deep learning in the automotive industry: Applications and tools," in *2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 3759–3768. DOI: 10.1109/BigData.2016.7841045.

[3] M. Van Ratingen, A. Williams, L. Anders, *et al.*, "The european new car assessment programme: A historical review," *Chinese journal of traumatology*, vol. 19, no. 02, pp. 63–69, 2016.

[4] Synopsys, *The 6 levels of vehicle autonomy explained | synopsys automotive*, Synopsys.com, 2022. [Online]. Available: https://www.synopsys.com/automotive/autonomous-driving-levels.html (visited on 02/11/2023).

[5] D. Hendrycks, N. Carlini, J. Schulman, and J. Steinhardt, *Unsolved problems in ml safety*, 2021. DOI: 10.48550/ARXIV.2109.13916. [Online]. Available: https://arxiv.org/abs/2109.13916.

[6] E. Union, *Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts*, Europa.eu, 2021. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%5C%3A52021PC0206.

[7] A. Bird, *Ai regulation in the u.s.: What's coming, and what companies need to do in 2023 | news insights | alston bird*, www.alston.com, Dec. 2022. [Online]. Available: https://www.alston.com/en/insights/publications/2022/12/ai-regulation-in-the-us (visited on 02/11/2023).

[8] S. Wang, C. Li, D. W. K. Ng, *et al.*, *Federated deep learning meets autonomous vehicle perception: Design and verification*, 2022. DOI: 10.48550/ARXIV.2206.01748. [Online]. Available: https://arxiv.org/abs/2206.01748.

[9] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information Processing Management*, vol. 59, no. 6, p. 103 061, 2022, ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2022.103061. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457322001649.

[10] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, *Federated learning: Strategies for improving communication efficiency*,

2016. DOI: 10.48550/ARXIV.1610.05492. [Online]. Available: https://arxiv.org/abs/1610.05492.

[11] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, *Federated optimization: Distributed machine learning for on-device intelligence*, 2016. DOI: 10.48550/ARXIV.1610.02527. [Online]. Available: https://arxiv.org/abs/1610.02527.

[12] H. Zhang, J. Bosch, and H. H. Olsson, "Real-time end-to-end federated learning: An automotive case study," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2021, pp. 459–468. DOI: 10.1109/COMPSAC51774.2021.00070.

[13] P. Yu and Y. Liu, "Federated object detection: Optimizing object detection model with federated learning," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, 2019, pp. 1–6.

[14] A. Géron, *Neural networks and deep learning*. O'Reilly, 2018.

[15] Y. Bengio, "Learning deep architectures for ai," *Foundations*, vol. 2, pp. 1–55, Jan. 2009. DOI: 10.1561/2200000006.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, Dec. 2014.

[17] T. Tieleman, G. Hinton, *et al.*, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[21] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, "Learning to predict vehicle trajectories with model-based planning," in *Conference on Robot Learning*, PMLR, 2022, pp. 1035–1045.

[22] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, "Vehicle trajectory prediction by integrating physics- and maneuver-based approaches using interactive multiple models," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5999–6008, 2018. DOI: 10.1109/TIE.2017.2782236.

[23] J. Liu, X. Mao, Y. Fang, D. Zhu, and M. Q.-H. Meng, "A survey on deep-learning approaches for vehicle trajectory prediction in autonomous driving," *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 978–985, 2021.

[24] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 353–359. DOI: 10.1109/ITSC.2017.8317913.

[25] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14 062–14 071, 2019.

[26] H. Cui, V. Radosavljevic, F. Chou, *et al.*, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," *CoRR*, vol. abs/1809.10732, 2018. arXiv: 1809.10732. [Online]. Available: http://arxiv.org/abs/1809.10732.

[27] H. Zhao, J. Gao, T. Lan, *et al.*, "Tnt: Target-driven trajectory prediction," in *Conference on Robot Learning*, 2020.

[28] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Agile application-aware adaptation for mobility," *SIGOPS Oper. Syst. Rev.*, vol. 31, no. 5, pp. 276–287, Oct. 1997, ISSN: 0163-5980. DOI: 10.1145/269005.266708. [Online]. Available: https://doi.org/10.1145/269005.266708.

[29] F. Wang, M. Zhang, X. Wang, X. Ma, and J. Liu, "Deep learning for edge computing applications: A state-of-the-art survey," *IEEE Access*, vol. 8, pp. 58 322–58 336, 2020. DOI: 10.1109/ACCESS.2020.2982411.

[30] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *European Symposium on Research in Computer Security*, 2020.

[31] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, ser. STOC '09, Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 371–380, ISBN: 9781605585062. DOI: 10.1145/1536414.1536466. [Online]. Available: https://doi.org/10.1145/1536414.1536466.

[32] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Application of Cryptographic Techniques*, 1999.

[33] P. Bogetoft, D. Lund, I. Damgård, *et al.*, "Secure multiparty computation goes live," vol. 5628, Feb. 2009, pp. 325–343, ISBN: 978-3-642-03548-7. DOI: 10.1007/978-3-642-03549-4_20.

[34] P. Kairouz, H. B. McMahan, B. Avent, *et al.*, "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019. arXiv: 1912.04977. [Online]. Available: http://arxiv.org/abs/1912.04977.

[35] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, *On the convergence of fedavg on non-iid data*, 2019. DOI: 10.48550/ARXIV.1907.02189. [Online]. Available: https://arxiv.org/abs/1907.02189.

[36] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *CoRR*, vol. abs/2106.06843, 2021. arXiv: 2106.06843. [Online]. Available: https://arxiv.org/abs/2106.06843.

[37] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. arXiv: 1602.05629. [Online]. Available: http://arxiv.org/abs/1602.05629.

[38] T. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *CoRR*, vol. abs/1909.06335,

2019. arXiv: 1909.06335. [Online]. Available: http://arxiv.org/abs/1909.06335.

[39] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, 2018. arXiv: 1812.06127. [Online]. Available: http://arxiv.org/abs/1812.06127.

[40] S. J. Reddi, Z. Charles, M. Zaheer, *et al.*, "Adaptive federated optimization," *CoRR*, vol. abs/2003.00295, 2020. arXiv: 2003.00295. [Online]. Available: https://arxiv.org/abs/2003.00295.

[41] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016. arXiv: 1610.05492. [Online]. Available: http://arxiv.org/abs/1610.05492.

[42] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," 2005.

[43] D. J. Beutel, T. Topal, A. Mathur, *et al.*, "FLOWER: A FRIENDLY FEDERATED LEARNING FRAMEWORK," Open-Source, mobile-friendly Federated Learning framework, Mar. 2022. [Online]. Available: https://hal.science/hal-03601230.

[44] D. J. Beutel, T. Topal, A. Mathur, *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[45] H. Caesar, V. Bankiti, A. H. Lang, *et al.*, "Nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[46] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106 775, 2021, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.106775. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121000381.

[47] A. Martinho, N. Herber, M. Kroesen, and C. Chorus, "Ethical issues in focus by the autonomous vehicles industry," *Transport Reviews*, vol. 41, no. 5, pp. 556–577, 2021. DOI: 10.1080/01441647.2020.1862355. eprint: https://doi.org/10.1080/01441647.2020.1862355. [Online]. Available: https://doi.org/10.1080/01441647.2020.1862355.

[48] "Real-world benefits of crash avoidance technologies," *IIHS/HLDI*, Mar. 2022. [Online]. Available: https://www.iihs.org/media/290e24fd-a8ab-4f07-9d92-737b909a4b5e/4GauQQ/Topics/ADVANCED%5C%20DRIVER%5C%20ASSISTANCE/IIHS-HLDI-CA-benefits.pdf.

[49] P. Liu, R. Yang, and Z. Xu, "How safe is safe enough for self-driving vehicles?" *Risk Analysis*, vol. 39, no. 2, pp. 315–325, 2019. DOI: https://doi.org/10.1111/risa.13116. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/risa.13116. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/risa.13116.

[50] M. Nakanoya, J. Im, H. Qiu, S. Katti, M. Pavone, and S. Chinchali, "Personalized federated learning of driver prediction models for autonomous driving," *CoRR*, vol. abs/2112.00956, 2021. arXiv: 2112.00956. [Online]. Available: https://arxiv.org/abs/2112.00956.

[51] A. Marathe, R. Walambe, and K. Kotecha, *In rain or shine: Understanding and overcoming dataset bias for improving robustness against weather corruptions for autonomous vehicles*, 2022. DOI: 10.48550/ARXIV.2204.01062. [Online]. Available: https://arxiv.org/abs/2204.01062.

[52] N. Paul and C. Chung, "Application of hdr algorithms to solve direct sunlight problems when autonomous vehicles using machine vision systems are driving into sun," *Computers in Industry*, vol. 98, pp. 192–196, 2018, ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2018.03.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361517305237.

[53] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, ser. DIDL '18, Rennes, France: Association for Computing Machinery, 2018, pp. 1–8, ISBN: 9781450361194. DOI: 10.1145/3286490.3286559. [Online]. Available: https://doi.org/10.1145/3286490.3286559.

[54] G. H. Lee and S.-Y. Shin, "Federated learning on clinical benchmark data: Performance assessment," *J Med Internet Res*, vol. 22, no. 10, e20891, Oct. 2020, ISSN: 1438-8871. DOI: 10.2196/20891. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/33104011.

[55] "Us urged to establish nationwide federal guidelines for autonomous driving," *Volvo Cars Global Newsroom*, Oct. 2015. [Online]. Available: https://www.media.volvocars.com/global/en-gb/media/pressreleases/167975/us-urged-to-establish-nationwide-federal-guidelines-for-autonomous-driving.

[56] B. Whitaker, *Hands off the wheel*. [Online]. Available: https://www.cbsnews.com/news/self-driving-cars-google-mercedes-benz-60-minutes/.

[57] M. Massar, I. Reza, S. M. Rahman, S. M. H. Abdullah, A. Jamal, and F. S. Al-Ismail, "Impacts of autonomous vehicles on greenhouse gas emissions—positive or negative?" *International Journal of Environmental Research and Public Health*, vol. 18, no. 11, 2021, ISSN: 1660-4601. DOI: 10.3390/ijerph18115567. [Online]. Available: https://www.mdpi.com/1660-4601/18/11/5567.

[58] S. Gibbs, "Google sibling waymo launches fully autonomous ride-hailing service," *The Guardian*, vol. 7, 2017.

[59] P. A. Mandhare, V. Kharat, and C. Patil, "Intelligent road traffic control system for traffic congestion: A perspective," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 07, p. 2018, 2018.

[60] M. Al-Turki, N. T. Ratrout, S. M. Rahman, and I. Reza, "Impacts of autonomous vehicles on traffic flow characteristics under mixed traffic environment: Future perspectives," *Sustainability*, vol. 13, no. 19, 2021, ISSN: 2071-1050. DOI: 10.3390/su131911052. [Online]. Available: https://www.mdpi.com/2071-1050/13/19/11052.

[61] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, and A. V. Vasilakos, "Cloud computing: Survey on energy efficiency," *ACM Comput. Surv.*, vol. 47, no. 2, Dec. 2014, ISSN: 0360-0300. DOI: 10.1145/2656204. [Online]. Available: https://doi.org/10.1145/2656204.

[62] Z. Marshall and J. Duquette, "A techno-economic evaluation of low global warming potential heat pump assisted organic rankine cycle systems for data center waste heat recovery," *Energy*, vol. 242, p. 122 528, 2022.

[63] S. Sudhakar, V. Sze, and S. Karaman, "Data centers on wheels: Emissions from computing onboard autonomous vehicles," *IEEE Micro*, vol. 43, no. 1, pp. 29–39, 2023. DOI: 10.1109/MM.2022.3219803.

[64] M. Placek, *Global advanced driver assistance system (adas) market size in 2021, with a forecast through 2030*, Statista, Jan. 2023. [Online]. Available: https://www.statista.com/statistics/1359700/global-adas-market-volume-forecast/ (visited on 02/10/2023).

[65] M. Carlier, *Eu car prices by country*, Statista, Jun. 2022. [Online]. Available: https://www.statista.com/statistics/425095/eu-car-sales-average-prices-in-by-country/ (visited on 02/10/2023).

[66] *New rules to improve road safety and enable fully driverless vehicles in the eu*, Europa.eu, 2022. [Online]. Available: https://ec.europa.eu/commission/presscorner/detail/en/IP_22_4312 (visited on 02/10/2023).

[67] A. Cheerla and O. Gevaert, "Deep learning with multimodal representation for pancancer prognosis prediction," *Bioinformatics*, vol. 35, no. 14, pp. i446–i454, Jul. 2019, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz342. eprint: https://academic.oup.com/bioinformatics/article-pdf/35/14/i446/28913346/btz342.pdf. [Online]. Available: https://doi.org/10.1093/bioinformatics/btz342.

A

Appendix 1

A.1 MNIST - All results loss

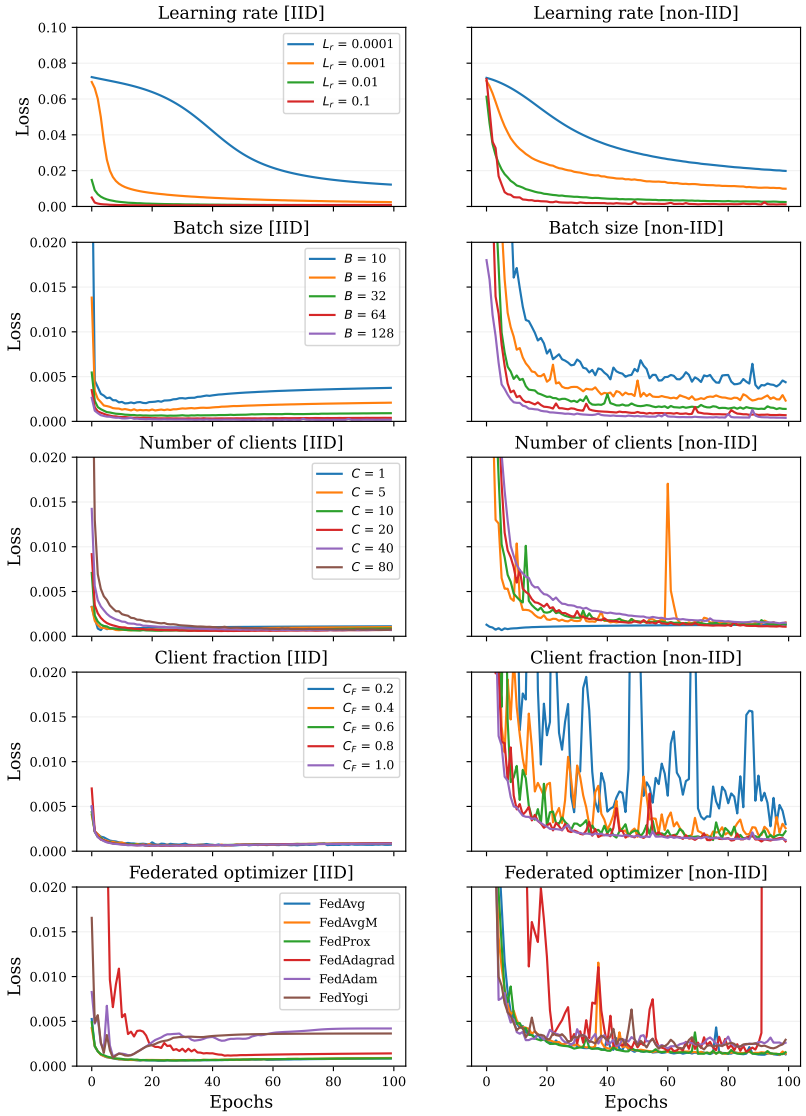
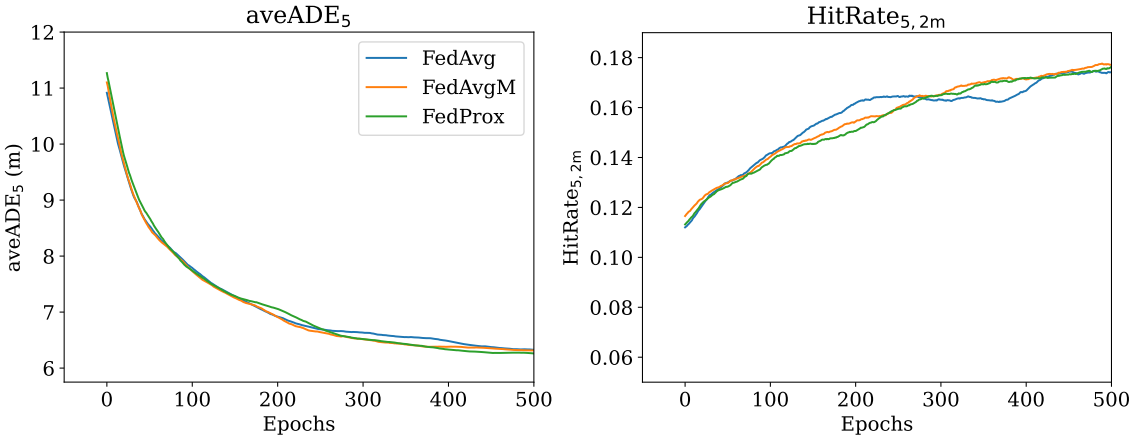


Figure A.1: Loss as a function of epochs for the previously described experiments. All axes, except for the graphs in the top row, display the same interval. The accuracy range for the learning rate graphs is five times larger.

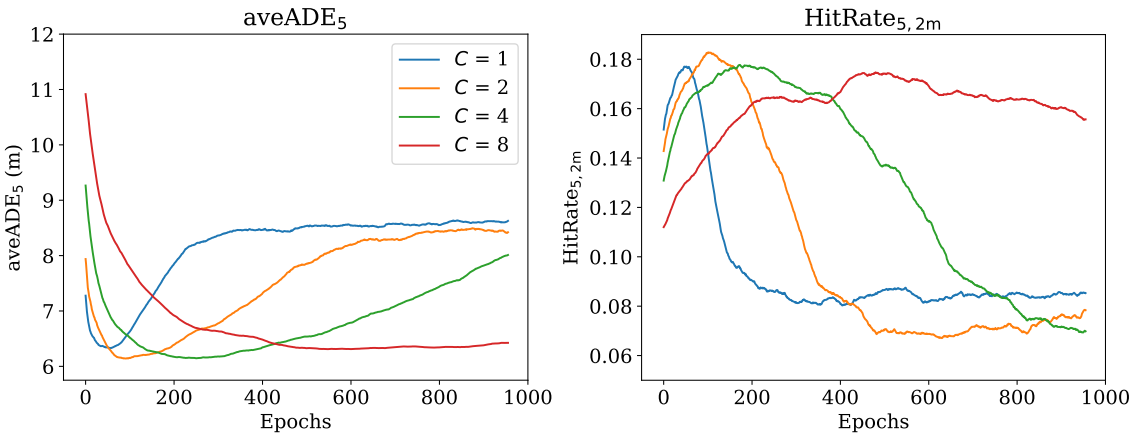
A.2 nuScenes - Federated optimizer 8 clients



(a) aveADE$_5$ **(b)** HitRate$_{5,2m}$

Figure A.2: Metrics with FedAvg, FedAvgM and FedProx, all using 8 clients. Fixed parameters: $C = 8$, $F_C = 1.0$, $L_r = 1\text{e} - 4$, $B = 8$, and non-IID balanced dataset.

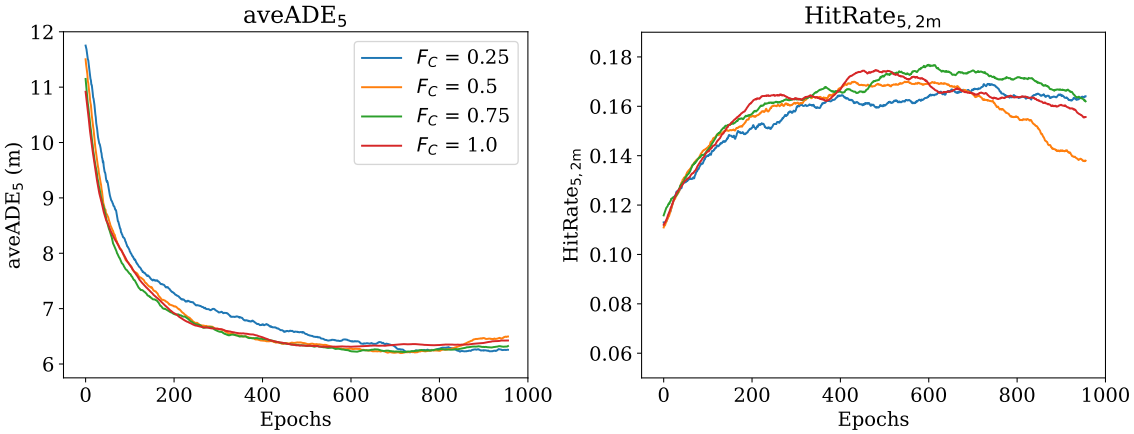
A.3 nuScenes - Number of clients IID balanced



(a) aveADE$_5$ **(b)** HitRate$_{5,2m}$

Figure A.3: Metrics with varying number of clients C. Fixed parameters: $F_C = 1.0$, optimizer = FedAvg, $L_r = 1\text{e} - 4$, $B = 8$, and IID balanced dataset.

A.4 nuScenes - Client fraction IID balanced



(a) aveADE$_5$ **(b)** HitRate$_{5,2\text{m}}$

Figure A.4: Metrics with varying client fraction F_C. Fixed parameters: $C = 8$, optimizer = FedAvg, $L_r = 1\text{e} - 4$, $B = 8$, and IID balanced dataset.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY