Federated Learning in Pedestrian Trajectory Prediction Tasks

Claas Brüß

Master's thesis

Federated Learning in Pedestrian Trajectory Prediction Tasks

Claas Brüß

30th June 2021

Chair of Data Processing
Technische Universität München

Claas Brüß. *Federated Learning in Pedestrian Trajectory Prediction Tasks.* Master's thesis, Technische Universität München, Munich, Germany, 2021.

Supervised by Prof. Dr.-Ing. Klaus Diepold and Matthias Kissel; submitted on 30th June 2021 to the Department of Electrical and Computer Engineering of the Technische Universität München.

© 2021 Claas Brüß

Chair of Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de/`.

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit `http://creativecommons.org/licenses/by/4.0/` or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Abstract

The paradigm of federated learning is a quickly growing and diversifying field in distributed machine learning. Through it computational resources of networked edge devices can be utilized to collectively train machine learning models under coordination of a central server. Due to local on device training with exclusively local training data and only the trained model being shared with the network no local data is directly exposed the network. Pedestrian trajectory forecasting methods model the intentions of all observed pedestrians as well as the social interactions between pedestrians to accurately predict the future trajectories of all pedestrian observed in a scene. With federated trajectory forecasting being in its infancy no public attempts have been made to federate the training of pedestrian trajectory forecasting models.

This project provides a first foray into the application of federated learning to pedestrian trajectory forecasting. A test environment consisting of a combination of the state of the art frameworks FedML in federated learning and Trajnet++ in pedestrian trajectory forecasting is developed and applied to provide initial insights into the impact of system design choices on convergence behaviour in this setting. For this an implementation of the SocialLSTM model with social grid based interaction module was federated in training over 12000 scenes from the UCY/ETH/Trajnet datasets and benchmarked through Trajnet++ evaluation tools.

Experiments in 10 federated learning system settings are evaluated in Trajnet++ final performance metrics and compared to a locally retrained baseline model of the same type provided through Trajnet++. The impact of choices in the number of communication rounds, the number of participating clients per communication round as well as the choice of adaptive or stateless federated optimization is empirically investigated.

The classicly retrained baseline outperforms all federated settings. It was found that eventhough the overall volume of training data remained the same for all experiments the data volume allocated each clients in a communication round was strong indicator for performance with higher allocated data volumes settings outperforming setting with lower allocated data volumes when the same optimization method was applied. The adaptive federated optimization algorithm FedAdam consistently outperformed Federated Averaging optimization algorithm in otherwise identical settings. Larger training datasets are necessary to further improve the performance of federated pedestrian trajectory forecasting.

Contents

1 Introduction

1.1 State of Machine Learning, Edge Hardware and Data Sources

In his PyTorch Developer Day 2020 talk ”The Future of AI Tools” [1] Andrew Trask demonstrates how most of the groundbreaking milestones in the development of AI were not due to algorithmic or model innovation, but rather due to larger and cleaner data sets becoming available. In addition he goes on to show how the algorithms are improving linearly over time, while the parameter count in models and therefore the number of computation cycles necessary to train these models are growing exponentially with the larger models showing ever higher performance.

By putting more emphasis on the utilization of quickly improving edge computation capabilities a lot of these limitations can potentially be side steeped. Propelled by Moore’s law more performant chips and SoCs are being deployed as edge devices year over year allowing for ever more demanding computation tasks to be moved to the edge. Fast advances in machine learning model compression and the development of programming frameworks geared towards deploying pretrained machine learning models with very limited memory available have made edge inference a possibility, rendering the centralized aggregation of raw sensor data unnecessary in these cases. First steps towards model training at the edge have been taken, but have remained reserved to high performance edge devices such as smartphones.

While the ever higher demand in computation power is sought to be addressed by Moore’s Law, obtaining large volumes of high quality data has turned out to be extremely costly and complex. Thus most data is not being shared but siloed not only out of concern for privacy but out of economic interest. Through the dominant rise of internet and software platform businesses such as Facebook, Google, Microsoft and Amazon, data gathering capabilities increasingly concentrated and incentivized those players to invest massively in data center computing infrastructure, machine learning and algorithms research as well as the necessary talent for these. The fact that digital services can have world wide reach has created a market dynamic that made aggregating high volumes of data and high computation performance under one roof, a must for being competitive in machine learning driven services and products.

1.2 Federated Learning

In recent years multiple developments have begun to progressively challenge data siloing and the exclusively centralized machine learning paradigm. Besides broad anti-trust and privacy concerns resulting in more action from state governments against extensive data hoarding, two

[1] https://www.youtube.com/watch?v=kzLeTz_vIeQ

major technical factors are contributing to this change.

On one hand there is the adoption of software driven analytics and machine learning in more and more previously unaddressed industries and on the other hand the explosion in the number and distribution of networked edge devices. The often by necessity highly dispersed locations of these edge devices limit the ability to transmit and aggregate sensor data and other machine outputs. Data aggregation is mainly constrained by the network bandwidth available to the edge devices, high costs stemming from the transmission of large data volumes and the running cost of maintaining or renting the necessary data center infrastructure on the receiving end.

In 2016 a research team at Google led by H. Brendan McMahan [25] presented a distributed approach for the training machine learning models seeking to address the challenges arising from training on decentralized data. Inspired by distributed training schemes developed for application in large data centers they outlined how repeatedly aggregating and merging of models trained on edge devices allowed them to produce a global model with performance rivalling a model of the same network topology trained on centrally aggregated data. The resulting global model is then broadcast back to clients so it can be improved through local training in the next cycle. A singular loop in the continuous cycle of model aggregation and broadcasting is called communication round.

They termed this new paradigm **Federated Learning** and stated three problem properties for its ideal application cases that can be tackled by this new approach to model training:

- Training on real world data from mobile devices provides clear advantages over training on proxy/synthetic data in data centers
- The training data is privacy sensitive or large in volume so it is beneficial for it to remain in decentralized storage
- Should super-vised training be necessary, the labels on the data need to be inferable from the context (e.g. user interaction)

The models driving the most popular smartphone applications and services closely fit this description, which led to the early platform wide deployment of Federated Learning in Android and iOS for language related services such as real time prediction of word choice while typing in Google's Gboard or improved speech recognition in virtual assistants such as Apple's Siri. The field has since bloomed and vastly diversified into domains such as medical imaging, secure multi-party computation in finance risk and pharmaceuticals to name a few[18].

1.3 Car as a Computing Platform

Cars have become some of the most computationally powerful mobile edge devices in the last couple of years. The introduction of sophisticated entertainment systems, automated parking and a rising number of sensors have pushed this development. Sharing economies of mobility assets and the promise of autonomous driving have led to the understanding that substantially more

computational performance and new chipsets specifically geared towards the automotive market would be needed to enable these technologies. To address this need semiconductor companies such as NVIDIA introduced new automotive specific product lines [2] with in some cases staggering computational performance more reminiscent of high-end personal computers rather than cars up to that point.

In his seminal work ”The High Cost of Free Parking” [31] Prof. Donald Shoup states that cars are on average parked for 95 percent of the time. This suggests that these increasingly powerful but also increasing costly chipsets are only in operation for about 5 percent of the time due to their need for electricity. A survey by the Idaho National Laboratory [32] concluded that privately used electric vehicles are most commonly being charged over night with cars often remaining plugged in for over 10 hours. These dormant chipsets could be utilized during the charging process of the rapidly growing number of electrical vehicles worldwide. Each vehicle could act an edge computing network node providing computing performance at the mere cost of electricity. With little to no need for cooling this would severely undercut cloud computing prices for applicable computation tasks.

Progress in semi-automated as well as autonomous driving research has been closely coupled to the ability of gathering data from test vehicles on the road. The sheer volume of data being recorded through these vehicle’s on-board sensors provided challenges in transmitting the data to central data centers through mobile networks. During his CES 2021 keynote Mobileye Ceo Prof. Amnon Shashua stated that 10 kilobyte per driven kilometer would be the maximum economically viable data volume that could be uploaded from a normal vehicle in the coming years. This constraint has so far been addressed by aggregating the full data on physical storage devices from only a limited number of specialized test vehicles or in some cases by strong heuristic filtering and data upload from vehicles in normal operation. Prof. Amnon Shashua also stated that for further progress in the field of autonomous driving to be made far more complex evaluations of the traffic context would be necessary and far more data would be needed to train the models with that capability.

1.4 Federated Human Trajectory Forecasting

This is where the far lower cross device communication bandwidth needs of federated learning could potentially help overcome the current network limitations. As described above the dormant computation power of charging electric cars could be used to train a wide spectrum of models applicable in the automotive space. Autonomous driving research presents a large area of opportunity for improvement through this scheme due to its almost prohibitive demands for very large training data volumes, the necessity of multiple models of high complexity working in accordance, large variety in edge cases and suitability for highly parallel training. In particular forecasting and planning tasks constitute a perfect fit, because compared to visual classification tasks far less

[2] https://developer.nvidia.com/drive/drive-agx

computation power is needed for training to reach state-of-the-art levels and no explicit labeling is needed since groundtruth is provided through the studied sequence itself.

1.5 Project Scope

This research project intends to provide a first step towards the viability evaluation of the application of federated learning for forecasting models in automotive systems. The particular focus lies on the federated training of current state-of-the-art models for human trajectory forecasting and the comparison to a classic centralized training approach. In this comparison far more parameters are relevant for the convergence behaviour of optimizers in federated learning and thus the final model inference performance after training. Therefore a modular testing environment is built and provided to aid indepth standardized exploration of the parametric design space for the federated training of pedestrian trajectory forecasting models. Utilizing the testing environment this project seeks to address the following question through the first empirical observations:

How do the number do choices in system design parameters impact the convergence and final inference performance of pedestrian trajectory forecasting models with limited training data available to the complete network?

In particular the impact of the following parameters will be investigated:

- **Number of communication rounds**
- **Number of participating clients per communication round**
- **The volume of training data available to each client per communication round**

2 State of the Art

2.1 Pedestrian Trajectory Forecasting

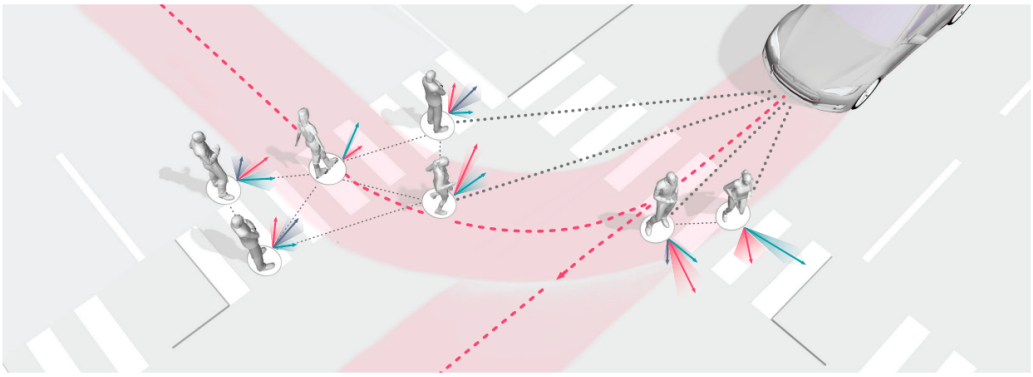

Figure 2.1: Interaction graph between pedestrians and a car in street scene [30]

Like virtually all complex sequence modeling and forecasting tasks model based on LSTMs provide the best performance. Modern approaches model each pedestrian trajectory in the observed scene by representing it by a LSTM Encoder-Decoder network.

2.1.1 Datasets

Most publications in the field of pedestrian trajectory forecasting use the combination of the UCY [21] and ETH [27] datasets to benchmark their implementations. This was extended through the Trajnet++ benchmarking suite which in addition to UCY and ETH included the datasets WildTrack [7], L-CAS [33] and CFF [2]. Projects that not exclusively model pedestrian behavior, like the Trajectron++ project mentioned above, often also benchmark their implementations on the nuScenes dataset [5] published by Motional. For pedestrian trajectory forecasting these datasets are distilled down trough pre-processing to co-ordinates of the each pedestrian in the observed scene at each time step within the observation time. This yields deceptively small training datasets, in case of the combination UCY and ETH less than 10 MB. The complexity arises from the interdependence of the pedestrians actions, reminiscent of n-body problems in physics.

2.1.2 Interaction Modeling

Since the introduction of the first neural network based pedestrian trajectory forecasting approach with social pooling called SocialLSTM in 2015 by Alahi et al.[1] a wide range of social interaction modeling mechanisms have been published. Kothari et al.[20] have recently published a wide

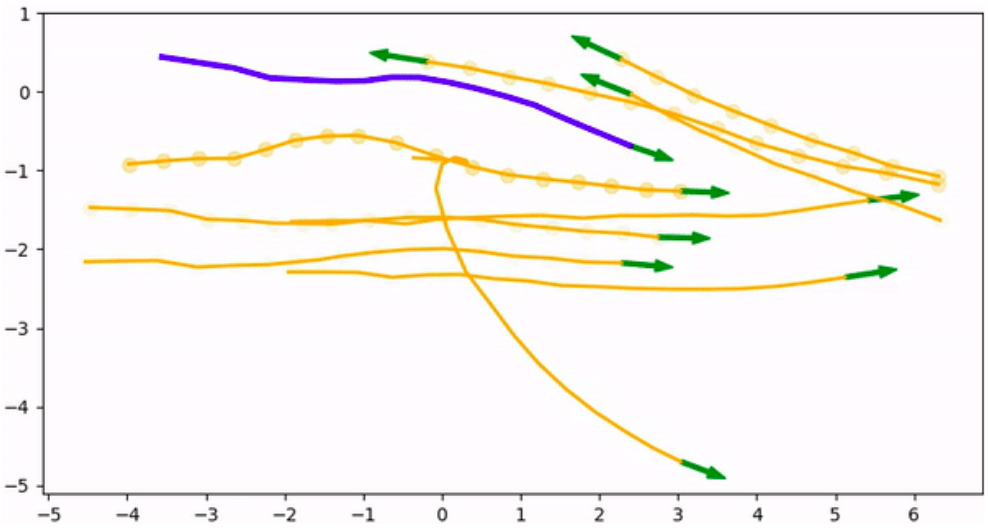

Figure 2.2: Scene with trajectories of 10 pedestrians over time [20]

survey of most of these methods. An accompanying benchmarking environment call Trajnet++ including implementations for all of the surveyed approaches as well as a comprehensive evaluation and visualization system was made publicly available.

While these publications focus on purely pedestrian settings Salzmann et al. [30] presented Trajectron++ in 2020. It abstracts the observed scene into spacio-temporal graph which models not only pedestrians and their social influence, but also includes cars as well as other modes of transportation and distinctly models the interactions of these with pedestrians over time. Through this they provide a more general approach of social interaction models for a wider selection of traffic scenarios.

More recently transformer based approaches such as AgentFormer [37] have become dominant in benchmarks for the ETH and UCY datasets and have demonstrated seemingly strong performance in benchmarks for the nuScenes dataset. The code is not publicly available as of the time of this project being submitted.

2.2 Federated Learning

The recently updated review publication ”Advances and Open Problems in Federated Learning” [18] provides the broadest overview on current topics, open problems and upcoming research areas within federated learning available. It is maintained by leading field experts Peter Kairouz and H. Brendan McMahan compiling input from over 50 renown researchers and is the natural starting point for the introduction into current topics in federated learning at large.

2.2.1 Federated Optimization

The most recent development in federated optimization is the introduction and study of adaptive federated optimization by Reddi et al. [28]. This work addresses the standing need for bringing popular adaptive optimization algorithms such as Adam [19] to enable a more effective training of modeling tasks, such as pedestrian trajectory forecasting, in which state of the art approaches utilize adaptive optimization.

Another line of active research studies the convergence behavior of federated optimization schemes on non-IID datasets published prior to the release of adaptive federated optimization. Both theoretical [23] and empirical [22] studies have been published investigating the impact of non-IID data distributions on the convergence of Federated Averaging[25], the baseline federated optimization method based on averaging. Li et al.[23] in particular quantify the tradeoff between convergence and data volume sent in communication for convex optimization.

2.2.2 Federated Learning in Trajectory Forecasting

To the knowledge of the author there are no publications specifically discussing the federated training of pedestrian trajectory forecasting models and their social interaction methods as of the submission of this thesis project.

Some work as been published on federated trajectory forecasting for multi-robot systems [24], but the field of federated trajectory forecasting is currently in its infancy.

2.2.3 Federated Learning in Vehicular Networks

A number of teams have begun to begun to explore the utility of vehicles, mostly automotive, as edge devices in networks for federated learning applications. Particular focus lies on the reduction of data transmission overhead federated learning and advanced model aggregation schemes can potentially provide compared to normal data aggregation.

Elbir et al. [9] list communications-related research challenges and discuss the impact of communication network bandwidth restrictions as well as communication delay. In addition they layout the advantages of cars acting as edge devices in federated training of models in need of frequent retraining.

Ye et al. [36] present a selective model aggregation approach in order to improve the quality of the overall aggregated model. For this design heuristics are developed under which image quality and computational capabilities provided by clients are judge locally before data is sent to central servers. To aid the information exchange between servers and vehicles a contract system is design. They provide simulation results demonstrating superior efficiency and accuracy performance of the present selective aggregation approach compared to Federated Averaging.

		TFF	FATE	PaddleFL	LEAF	PySyft	FedML
Diversified Computing Paradigms	standalone simulation	✓	✓	✓	✓	✓	✓
	distributed computing	✓	✓	✓	✗	✓	✓
	on-device training (Mobile, IoT)	✗	✗	✗	✗	✗	✓
Flexible and Generic API Design	topology customization	✗	✗	✗	✗	✓	✓
	flexible message flow	✗	✗	✗	✗	✗	✓
	exchange message customization	✗	✗	✗	✗	✓	✓
Standardized Algorithm Implementations	FedAvg	✓	✓	✓	✓	✓	✓
	decentralized FL	✗	✗	✗	✗	✗	✓
	FedNAS (beyond gradient/model)	✗	✗	✗	✗	✗	✓
	VFL (vertical federated learning)	✗	✓	✓	✗	✗	✓
	SplitNN (split learning)	✗	✗	✓	✗	✓	✓
Standardized Benchmarks	linear models (e.g., Logistic Regression)	✓	✓	✓	✓	✓	✓
	shallow NN (e.g., Bi-LSTM)	✓	✓	✓	✓	✓	✓
	Model DNN (e.g., ResNet)	✗	✗	✗	✗	✗	✓
	vertical FL	✗	✓	✗	✗	✗	✓

Figure 2.3: Comparision of FedML and a selection of the most popular federated learning libraries by the FedML authors from November 2020 [15]

2.2.4 Federated Learning Frameworks

A wide selection of federated learning libraries is available today. The focus of federated learning libraries varies strongly. On one hand industry driven libraries, such as FATE [11] and PaddleFL[**yanjun**], emphasize real world deployment on distributed devices. On the other hand simulation oriented libraries like TensorFlow-Federated [17], LEAF [6], PySyft [29] and FedML[15] emphasize certain aspects such as privacy and security while providing limited features in other aspects.

FedML presents a very research focused feature set with high modularity and arguably the most flexibility in network topologies and optimization schemes. New implementations of state of the art algorithms are added constantly, with a diversifying toolkit geared towards sub-fields like federated learning in graph neural networks. As mentioned before, the design space is vast and encompasses widely varying fields and therefore a broad range of stakeholders. For collaborative research modularity and ease of deployment of new state-of-the-art algorithm implementations are instrumental to support standardized and comparable benchmarking.

3 Methods

3.1 Human Trajectory Forecasting

In the recent work ”Human Trajectory Forecasting in Crowds: A Deep Learning Perspective” Kothari et al. [20] define the task of human trajectory forecasting as follows:

Given the past trajectories of all humans in a scene, forecast the future trajectories which conform to the social norms.

This task unpacks into sequence analysis of human trajectories in an observed scene including multiple agents, the modeling of social norms as the influence of the agents on one another and the forecasting of future trajectories by generating realistic sequences taking the other mentioned aspects into account. The following sections address the aspects in this order. Recurrent neural networks (RNN) have become the de facto standard for the modeling of complex sequences. In contrast to feed forward neural networks RNNs not only take into account the input at the current time step but also include parameters from the previous time step. This gives rise to a notion of memory which is paramount for understanding any non random sequence as the current system state s^t dependent on the input x^t at the current time step t as well as the system state in the last time step s^{t-1}. The parameterization of the function or model f is described by ω.

$$s^t = f(s^{t-1}, x^t; \omega) \tag{3.1}$$

The system state s^t can also be expressed as h^t to indicate that the system state is defined through the hidden values of the system. The embedding of h^{t-1} as well as x^t is weighted by the weight matrices W and U respectively and passed through an embedding function ϕ. The output o^t at time step t results from the hidden states at current time step weighted with V and an offset c. The activation function f then yields the predicted value $\hat{y}^t$ at every time step t. This provides an output sequence of the same length as the input sequence.

$$\begin{aligned} h^t &= \phi(Wh^{t-1} + Ux^t + b) \\ o^t &= c + Vh^t \\ \hat{y}^t &= f(o^t) \end{aligned} \tag{3.2}$$

Classic RNNs only consider the last time step besides the current one and have been shown to struggle with modeling sequence patterns with inherent long-term dependencies. This is addressed by the concept of gated RNNs such as LSTMs and GRUs.

Encoder-Decoder Architectures

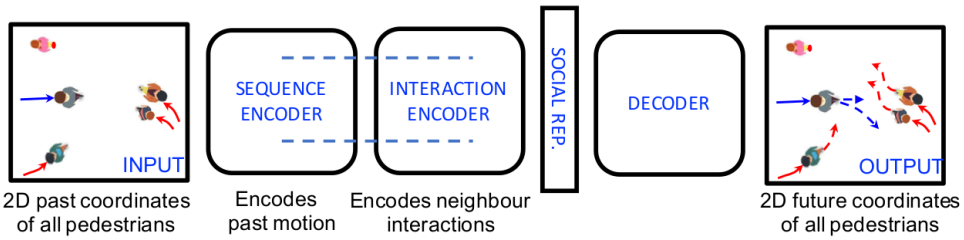

Figure 3.1: Encoder - Decoder Sequence in Social Pooling Methods [20]

In order to map input sequences of variable length to output sequences of variable length, two recurrent neural networks can be paired in encoder-decoder architectures. The encoder takes in a sequence of a preset length and outputs a tensor representation of a certain dimensionality. Following this the decoder takes in this tensor and outputs a new sequence of another preset length. This technique allows for the forecasting of future sequences by observing a past sequence. In this application a sequence of co-ordinates that form a trajectory is observed by the encoder for a set number of time steps to then pass on the resulting tensor to the decoder. The decoder then produces a predicted sequence of a predetermined number of time steps.

As illustrated in 3.1 the decoder input can be designed to consider more information than just the tensor stemming from the sequence encoder. This information provided through a second separate encoder could also be sequence based or simple conditioning. A more in depth discussion of how this approach can be applied in pedestrian trajectory forecasting to model social interaction through interaction encoders is provided in the section 3.1.1.

Long Short-Term Memory

As discussed above RNNs are limited in their capability to model long-term dependencies in sequences due to their low memory persistence. To address these limitations Long Short-Term Memory (LSTM) was introduced by Hochreiter and Schmidhuber in 1997 [16]. They presented an approach to extend and control the memory persistence of the recurrent links in order to successfully model sequences with long-term dependencies. As depicted in figure 3.2 LSTMs are instantiated in network cell units. The cells are internally recurrent.

In addition to the classic input to output path 3 additional sigmoidal gates, namely input gate, forget gate and output gate as well as a time step delayed state self-loop are part of the cell. The state of the current time step influences the behaviour of the there gates in the next time step. These gates control the flow of information through the cell and can be connected to other neurons within the network that should inform cell behaviour.

The forget gate controls the weights of the state self-loop and can therefore be used to dy-

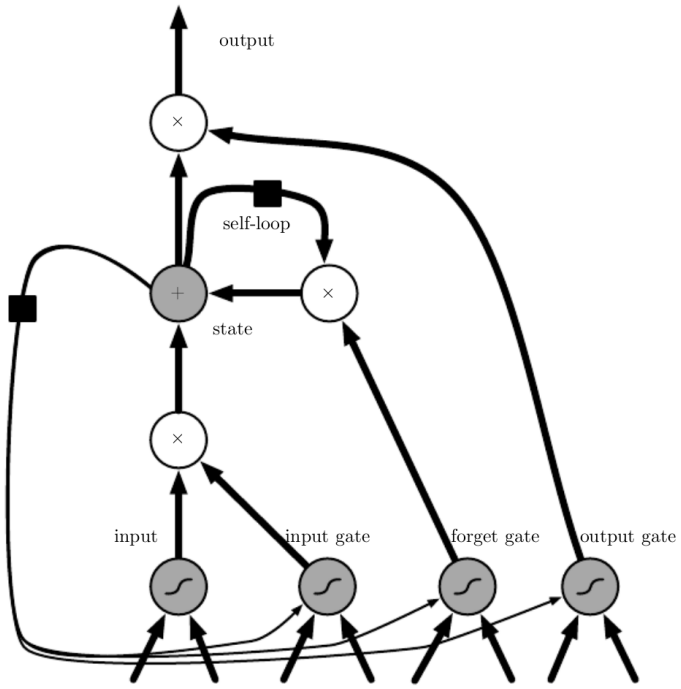

Figure 3.2: LSTM Cell Blockdiagram [10]

namically change the time scale of memory persistance. The input gate controls the weight given to the input at the current time step. These two gates in conjunction control the balance of influence of past input information accumulated through the delayed state self loop and the new input. The output gate weighs the output information and can also deny any output from leaving the cell at a particular time step. The cells replace the hidden units in classic RNNs and are therefore also connected to each other in a recurrent fashion. This type of recurrence is classically referenced as outer recurrence.

3.1.1 Social LSTM

In order to model the behavior of a pedestrian crowd agents in the scene cannot be considered individually. Social norms, such as collision avoidance and abstaining to invade personal space by keeping a certain distance, give rise to a more collective behaviour by agents influencing each other. Therefore modeling each pedestrian trajectory individually through LSTMs is not sufficient. Information about the behavior of the other pedestrians besides the primarily observed agent in the scene needs to be considered as LSTM input besides the recurrent input of the agent itself to allow for the modeling of social interaction and physically acceptable outputs. With Social LSTM Alahi et al.[1] presented the first neural network model for pedestrian trajectory forecasting that takes these factors into account .

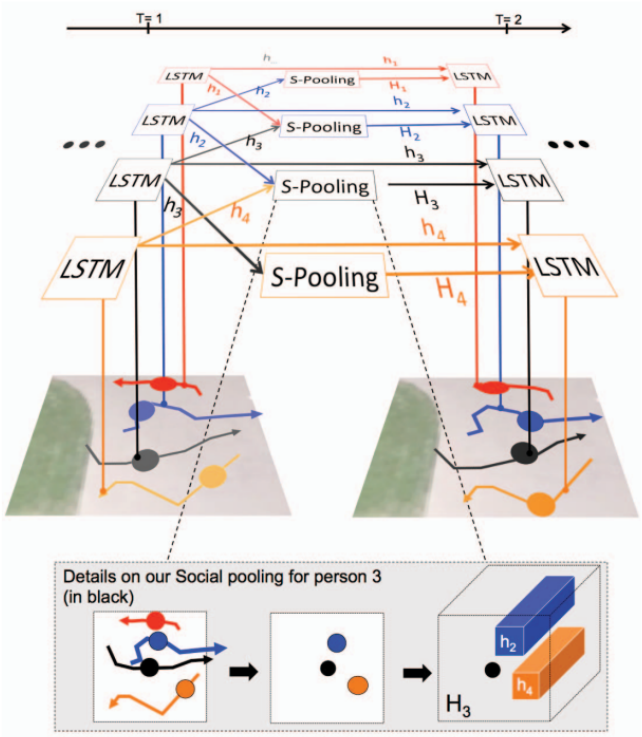

Figure 3.3: Social LSTM Method with LSTMs for each pedestrian in the scene and social pooling layer modeling the social influence and interaction between the individuals within in the scene [1]

The information exchange between agents in the scene that is necessary to model the mentioned social influences was implemented in form of a pooling layer of hidden states between neighbouring agents. This approach was dubbed Social Pooling. Various approaches for social pooling have been introduced since the introduction of Social LSTM. The spacial distance between two agents is the determining factor whether the social interaction is considered or not. The consideration threshold is set by the particular implementation.

Following the notation of Kothari et al. [20], in a scene with n agents $\mathbf{X} = X_1, X_2, ..., X_n$ denotes all trajectories and $\mathbf{Y} = Y_1, Y_2, ..., Y_n$ all corresponding future trajectories with $\hat{\mathbf{Y}}$ denoting predicted future trajectories. The position of the i-th pedestrian at time step t is denoted by $\mathbf{x}_i^t = (x_i^t, y_i^t)$ and the predicted position denoted by $\hat{\mathbf{x}}_i^t = (\hat{x}_i^t, \hat{y}_i^t)$. All positions of all pedestrians are observed for the time steps $t = 1, ..., T_{obs}$ and are predicted for the time steps $t = T_{obs} + 1, ..., T_{pred}$ with overall number of time steps being L. A scene is denoted through $\mathrm{s}(\mathbf{X}, \mathbf{Y})$ with the predicted outcome for the whole scene $\hat{\mathrm{s}}(\mathbf{X}, \hat{\mathbf{Y}})$. The state of the i-th agent at time step t is generally denoted by s_i^t and can in principle include any relevant information, such as velocity or body posture, considered in the implementation.

As illustrated in figure 3.3 the hidden states of neighbouring agents are aggregated in the social pooling layer and given as LSTM input at the next time step. Alahi et al. [1] applied a grid-based pooling approach which is discussed in the following paragraphs.

Grid-based Pooling

As illustrated in figure 3.4 each of the neighbours in the vicinity of the central primarily observed pedestrian provide their hidden state. Here h_j denotes the hidden state provided by the j-th neighbour. The number of neighbours is unknown a priori and could be very high in dense scenes. In order to retain a compact representation of neighbour states the whole scene is then divided into a grid centered around the primarily observed pedestrian and the grid cell associated with each neighbour is determined. Finally the hidden states of all neighbours occupying the same cell are combined into one. The hidden-state dimension is denoted as D. The number of cells within in the grid is determined by the neighbourhood size N_0 with the grid consisting N_0 x N_0 cells.

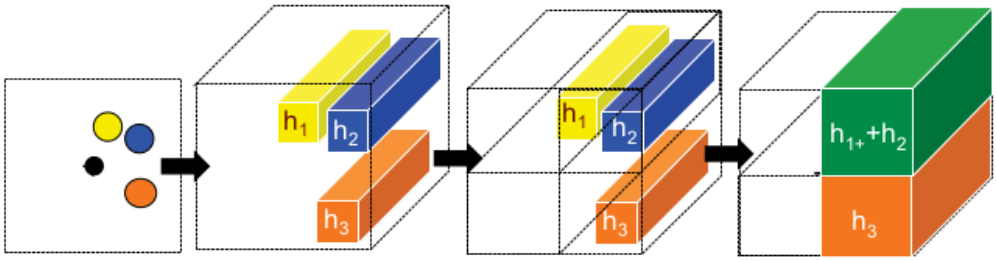

Figure 3.4: Grid-based pooling around the primarily observed pedestrian shown as a black dot [1]

The neighbourhood hidden state tensor H_i^t is a N_0 x N_0 x D tensor for the trajectory $i-th$ pedestrian.

$$H_i^t(p,q,:) = \sum_{j \in \mathcal{N}_i} \mathbf{1}_{pq}[x_j^t - x_i^t, y_j^t - y_i^t] h_j^{t-1} \tag{3.3}$$

The neighbourhood $\mathcal{N}_i$ of the i-th pedestrian is an aggregate of all pedestrian indices besides i. The function $\mathbf{1}_{mn}$ determines the cell to which the j-th neighbour is associated. It is imported to note that implementations might choose not to sum over the whole grid, but sub-sample only a smaller window centred around the primarily observed pedestrian.
The hidden state h_i^t for the i-th pedestrian at time step t then follows:

$$\begin{aligned} e_i^t &= \phi(x_i^t, y_i^t; W_e) \\ a_i^t &= \phi(H_i^t; W_a) \\ h_i^t &= \mathbf{LSTM}(h_i^{t-1}, e_i^t, a_i^t; W_l) \end{aligned} \tag{3.4}$$

W_e , W_a and W_l are the weight tensors for the input embedding, social pooling information embedding and LSTM gates respectively. The embedding function ϕ is of ReLU non-linearity.

Minibatch Stochastic Gradient Descent

Classic gradient descent calculates the gradient of a summed loss function $\mathcal{L}$ by calculating the gradient of the loss function for each i-th sample in a dataset of size M for each optimizer step. This is also called full batch optimization.

$$\nabla_\omega J(\omega) = \frac{1}{M} \sum_{i=1}^{M} \nabla_\omega \mathcal{L}(\mathbf{x_i}, \mathbf{y_i}, \omega) \tag{3.5}$$

With the large training datasets found in deep learning applications this becomes computationally prohibitive. The approach of Minibatch Stochastic Gradient Descent (SGD) sidesteps this issue through approximating the gradient by calculating the scene losses $\mathcal{L}_s$over a small subset of the dataset. This subset is called minibatch and has a size of B samples with B staying constant for the whole training process.

$$g = \frac{1}{B} \nabla_\omega \sum_{i=1}^{B} \mathcal{L}_s(\hat{\mathbf{s}}_\mathbf{i}, \mathbf{s_i}, \omega) \tag{3.6}$$

The complete training dataset is split into minibatches and each minibatch is followed by an optimizer step updating the parameters ω with a constant learning rate η.

$$\omega \leftarrow \omega - \eta g \tag{3.7}$$

Adam

Keeping the learning constant throughout the whole training process can severely limit the speed of convergence towards the optimum. Adapting effective step sizes based on previous optimization steps can yield far more stable convergence behavior. This due to the reduced risk of overshooting the optimum, making descent from the opposite side of slope necessary less often. The concept of former optimizer steps informing the current one is called momentum.

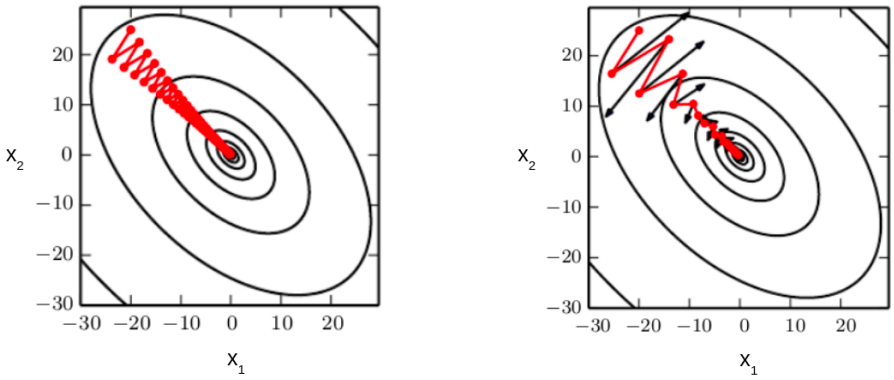

Figure 3.5: Comparison of Optimization Steps in without (left) and with (right) momentum applied. The bottom is at (0,0) with concentric equality lines around it. Adapted from [10]

As illustrated in figure 3.5 a classic gradient descent optimizer without regard for momentum requires far more steps to converge towards the bottom at (0,0) than an optimizer with regard for momentum. The momentum causes the the steps to quickly far more aligned along the axis in the direction of (1,-1) pointing towards the bottom, which is less steep than gradients in the (1,1) direction. Classic gradient descent demonstrates a persistently stronger step alignment in the (1,1) direction due to the steeper slopes and therefore larger gradients.

Orignially published in 2015, the Adam [19] algorithm has become the standard optimizer for many deep learning applications. It utilizes 2 types of momentum to provide computationally efficient optimization with low memory demand. The adaptive nature of Adam and the approach of element-wise application of the operation de-emphazise the need for deliberate and manual tuning.

Algorithm 1 ***Adam*** mini-batched

Initialization: ω_0, η, β_1 and $\beta_2 \in [0,1)$, τ, B
Initialize 1st and 2nd moment variables **m = 0**, **v = 0**
for $b \in \mathcal{B}$ **do**
$g = \frac{1}{B}\nabla_\omega \sum_{i=1}^{B} L_s(\hat{\mathbf{s}}_\mathbf{i}, \mathbf{s}_\mathbf{i}, \omega)$
$m_t = \beta_1 m_{t-1} + (1+\beta_1)g$
$v_t = \beta_2 v_{t-1} + (1-\beta_2)g \odot g$
$\Delta\omega = -\eta \frac{m_t}{\tau+\sqrt{v_t}}$ (operations applied element-wise)
$\omega \leftarrow \omega + \Delta\omega$
end for

Losses and Training in Social LSTM

Two types of losses can be considered during training of Social LSTMs. The first is a classic mean square error or L2 loss with a definite point $\hat{\mathbf{x}}^t$ being predicted:

$$\mathcal{L}_i(\hat{\mathbf{x}}^t, \mathbf{x}^t) = \frac{1}{L} \sum_{t=1}^{T_{pred}} (\hat{\mathbf{x}}_i^t - \mathbf{x}_i^t)^2 \tag{3.8}$$

The prediction of the position in the next time step $(\hat{x}, \hat{y})_i^{t+1}$ can not only be provided as definite point, but also as a estimated position through a bi-variate Gaussian distribution. This distribution is parameterized through the mean $\mu_i^{t+1} = (\mu_x, \mu_y)_i^{t+1}$, the standard deviation $\sigma_i^{t+1} = (\sigma_x, \sigma_y)_i^{t+1}$ as well as the correlation coefficient ρ_i^{t+1} . These 5 parameters are predicted through a 5 x D tensor W_p. Depending on the implementation W_p can be learned or partially set constant in σ_i and ρ_i. At time step $t + 1$ the predicted co-ordinates are therefore

$$(\hat{x}, \hat{y})_i^{t+1} \sim \mathcal{N}(\mu_i^{t+1}, \sigma_i^{t+1}, \rho_i^{t+1}) \tag{3.9}$$

The second type is the negative log-Likelihood loss $\mathcal{L}_{i,g}$ in regards to the Gaussian distribution prediction for the i-th trajectory and the trained or, depending on the implementation, preset embedding function ϕ_p.

$$[\sigma_i^t, \mu_i^t, \rho_i^t] = \phi_p(W_p, h_i^{t-1}) \tag{3.10}$$

$$\mathcal{L}_{i,g}(\omega) = -\sum_{t=T_{obs}+1}^{T_{pred}} log(\mathbb{P}(x_i^t, y_i^t | \sigma_i^t, \mu_i^t, \rho_i^t)) \tag{3.11}$$

Training in conducted through teacher forcing. This means that inputs given to the model at every time step t are ground truth. The predictions are used to obtain the necessary losses and to compute gradients but are not applied as input in the next time step. Due to the coupling of all LSTMs hidden states for the individual trajectories within the scene through the social pooling layer training is conducted joint back-propagation through time.

Trajectory Prediction and Performance Metrics

Generally there is more than one set of plausible pedestrian trajectories per scene that fulfill the modeled social norms, are physically acceptable and free of collisions. This means that even the ground truth provided through the training data is single plausible trajectory set possible and a rerecording of a specific scene with same initial state at $t = 0$ might yield slightly different trajectories.

This makes human motion in social context inherently multimodal. While Alahi et al. focussed on unimodal predictions and didn't explicitly address this in with SocialLSTM[1] initially, Alahi's team at EPFL proposed to evaluate multimodal prediction[20]. Multiple predictions are produced at each time step and are then evaluated as a set to gain more insight into the prediction performance in application.

In testing ground truth is provided to model for each time step t until time-step T_{obs}. The remaining time from $T_{obs} + 1$ up to the last time step T_{pred} the position predicted by the model itself is used as input in the next time step.

The Trajnet++ provides the an evaluation toolkit for the calculation of multiple performance metrics for unimodal and multimodal prediction as well as adherence to physical acceptability. Quoted below are the definitions for the proposed metrics as stated by Kothatri et al.[20].

Unimodel-Performance :

- **Average Displacement Error (ADE)**: Average L_2 distance between ground truth and prediction overall predicted time steps.
- **Final Displacement Error - (FDE)**: The distance between the predicted final destination and the ground truth final destination at the end of the prediction period T_{pred}.
- **Collision 1 - Prediction Collision (COL-1)**: This metric calculates the percentage of collision between the primary pedestrian and the neighbors in the forecast future scene. This metric indicates whether the predicted model trajectories collide, i.e., whether the model learns the notion of collision avoidance.
- **Collision 2 - Groundtruth collision (COL-2)**: This metric calculates the percentage of collision between the primary pedestrian's prediction and the neighbors in the groundtruth future scene.

Multimodal-Performance:

- **Top-k ADE**: Given k output predictions for an observed scene, this metric calculate the ADE of the prediction closest to the groundtruth trajectory.
- **Top-k FDE**: Given k output predictions for an observed scene, this metric calculate the FDE of the prediction closest to the groundtruth trajectory.

ADE and FDE are measured in meters and the collision metrics are output as percentages.

3.2 Federated Learning

In their field defining survey paper "Advances and Open Problems in Federated Learning" [18] Kariouz et al. define the paradigm of Federated Learning as follows:

Federated Learning *is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; instead,focused updates intended for immediate aggregation are used to achieve the learning objective.*

The reasoning behind focused updates or minimal data collection is twofold with one being privacy and other being the cost of centralised compute infrastructure as well as the transmission of large data volumes through mobile networks or other forms of large scale data aggregation. To describe this the term communication efficiency is used to evaluate and compare the overall volume of data being transmitted during a training campaign.

Even though this project puts no further emphasis on security or privacy as such the mere fact that raw data never leaves the edge device but rather undergoes what could be considered an intelligent compression and abstraction into gradients and models reduces the risks and viability of systemic surveillance at scale. These risks might stem from the service provider or in the adversarial entity. This characteristic is helpful for the adoption of any widely distributed network sharing data with a centralised entity or service.

As federated learning offers an extremely broad design space with a plethora of architectural parameters to be chosen it is important to be transparent in communication what data is sent where and what computational tasks are carried out where. Transparency in design choices and their impacts is not only important due to the intrinsic interdisciplinarity of federated learning with fields such as cryptography, distributed optimisation, network security, embedded systems, information theory and machine learning contributing to this paradigm, but also for collaboration and discussion between research groups that will mostly be limited to stand-alone simulation and entities with large numbers in edge network nodes at their disposal such as large internet companies or in our case mobility providers and car manufacturers.

As in any large network certain behaviours and patterns will only begin to emerge if the network reaches a certain size. In addition, new challenges due to potentially unbalanced non-identically and independently distributed (non-IID) datasets will challenge the assumptions in purely academic research bench marked on a small number of prepared datasets.

The following sections will discuss the general types of federated learning network architectures, the most important design choices for federated training of models such as the selection of the optimizer or the aggregation strategy, the impact of the lack in guarantees that can be given for the dataset involved as well as general network modalities in the context of federated pedestrian trajectory forecasting with networked fleets of cars acting as edge devices.

3.2.1 The Federated Training Process

The classic training process in federated learning settings is a looping 5 stage process. A full loop is called a communication round.

1. **Client selection**: Out of all clients participating in the network a subset is selected by a central server as needed. Certain requirements might have to be fulfilled for clients to participate.

2. **Broadcast**: Current global model and training instructions are distributed to the clients by the central server.

3. **Client Computation**: Each client performs computation on local data based on the broadcast instructions. This usually conducted in a stateless manner, since no guarantees can be given for participation in future communication rounds.

4. **Aggregation**: Clients send there locally trained models back to the central server. Models might not be collected from all participating clients due to time outs or perceived adversarial behaviour.

5. **Model update**: The central server merges the the collected models and computes the update for the global model.

Many variations that deviate from this common pattern are currently being explored in research as the field permeates a growing number of fields of application.

3.2.2 Distribution Regimes and Topologies

Federated Learning is most commonly split into two major regimes. Cross-silo Federated Learning assumes a comparatively small number of participating clients with high reliability and availability. These clients might Low restrictions on transmission bandwidth can assumed for most applications in this regime. The go-to application area is collaborating hospitals seeking to improve diagnostic models without infringing on patient privacy. Usually these scenarios will include clients with substantial data volumes and high security. This research area also sees strong overlap with secure multi party computation research.

Cross-device Federated Learning assumes scenarios of high or sometimes vast numbers of clients with low reliability and low availability. The mentioned initially smartphone focused deployments are the go-to examples for this regime. Clients might be mobile, geographically widely dispersed with strong restrictions on power consumption, transmission bandwidth and locally deployed hardware resources. Clients might demonstrate only cyclical availability due to the day-night cycle. A lot more consideration has to be given to the distributed aspects in optimization and adversarial attack vectors need to be understood due to the far loser federation network of edge nodes with low barriers for participation and some devices only participating in a singular communication round. Therefore clients are commonly considered to be stateless. For a more detailed distinction between these regimes please refer to the table in the appendix.

The federation of training with mobility assets, such as cars, acting as the edge nodes clearly falls within the realm of Cross-device Federated Learning. In general these settings are non-trivial to replicate in academic research due to lack of access to device networks of sizes comparable to the final deployment scenarios. The chapter experiments discusses in more detail how this was considered for the experiments of this project.

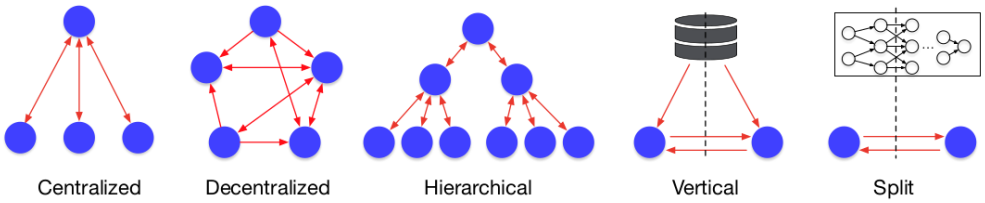

Figure 3.6: Definition of Topology Types in Federated Learning [15]

Various topologies have been investigated in the context of Federated Learning for message exchange between clients. The messages are not limited to the exchange of gradients or models, but also might be used for synchronisation or of other instructional nature. The choice of messaging topology is closely linked to the choice of optimizer and therefore closely related to the model and learning task at hand.

Centralized In this topology clients are not directly aware of other clients since there is direct communication between them. All information flow, such as the distribution of a new global model, runs through a central communication node. Topology allows for high control over the federation network operations from a singular point.

Decentralized This topology removes the idea of any definitively central entity in message exchange. Clients communicate directly with one another through a protocol of their choice. Reminiscent of peer to peer or mesh networks some clients might be temporary special status for coordination purposes.

Hierarchical In certain cases [4] clustering clients into subgroups can yield better performance, training robustness or security. This clustering in communication can in principle be applied for the exchange of partial training datasets, models, gradients, validation metrics or any other messages.

Vertical In cases were different clients have access to datasets with different features but ultimately describing the same scenarios vertical federated learning schemes can be deployed. In the context of the automotive sector this could be different sensor outfits of different product lines providing different data. Should both product lines be tasked to solve the same problems, in our case the prediction of pedestrian behavior, but using input data with differing features and therefore different models a vertical federated learning scheme would be needed. Due to the high complexity and challenges similar to the field of transfer learning this remains fairly unattractive and under investigated for the cross-device regime. Cross-silo scenarios for vertical topologies have explored in more detail [35].

Split Initially proposed by the Camera Culture Lab at MIT Medialab in 2018 [13] Split Learning basically splits a neural network between two layers and distributes these partial networks between clients. Clients exchange messages containing the values that arise at the split if needed. This way labels can potentially be kept secret from the client given the feature side neural network part.

For this project a centralized message topology was chosen. It was deemed the most realistic for the initial deployment of federated learning in the automotive space due to it's clarity and interest of centralized control of the service architecture through the central node by the automotive company itself. It also allows to study the efficacy of models of interest whilst retaining more secrecy of experimental conduct and therefore commercial interest behind all experiments.

3.2.3 Non-IID Data

Formally the conditions for n independent and identically distributed random variables can be expressed as follows:

$$F_{X_1}(x) = F_{X_k}(x)\ \forall k \in N = \{1, ..., n\} \text{ and } \forall x \in I \tag{3.12}$$

$$F_{X_1,...,X_n}(x_1, ..., x_n) = F_{X_1}(x_1) \cdot ... \cdot F_{X_n}(x_n) \tag{3.13}$$

Little is known about the client side datasets a priori. The client-partitioned data sets will likely differ greatly in distribution, contradicting equation 1, and could have been influenced by the same external factors or context leading to correlation, contradicting equation 2. Thus neither identicality nor independence in distributions can be assumed for real world federated learning applications.

Non-IID distributions can emerge at the level of client selection with $F(N) = Q$ being the distribution over available clients and at the data sampling level $F(x) = \mathcal{P}_k(x)$ for the distribution over the local subset of the client-partitioned dataset at client k. Q and $\mathcal{P}_k(x)$ might also not be constant over time therefore breaking IID conditions. This can lead to dataset shift by introducing differences in distribution between train and test datasets.

Kairouz, et al.[18] present a survey of 5 common ways how dataset can deviate in non-identical client distributions ($P_i \neq P_j$), which are discussed in the follow paragraphs in relation to the pedestrian behavior forecasting in this project.

Feature Distribution Skew This type of identicality violation is also referred to as co-variate shift with $\mathcal{P}_i(x)$ varying across clients while $\mathcal{P}(y|x)$ is the identical for all clients. This dependent on what types of features are heeded and how they are encoded. Features like co-ordinates might not be as prone to this shift as body posture might be. As most pedestrian trajectory forecasting models primarily focus on co-ordinates this shift should be comparatively mild.

Label Distribution Skew How often certain outcomes or behaviours are represented in the data might vary based on the geo-location and time context it was recorded in. Data recorded at road crossings in busy business districts at rush hour might show far more evasion maneuvers by individuals walking towards one another, while data recorded at a crossroads close to a rural

school around noon might include movement of groups of children more often. Since publicly available dataset predominantly used in academic research are often collected in a limited number of specific observation settings certain biases of this kind are inevitable due choice of setting.

Concept Drift Even tough the intent and the goal position might be similar pedestrians might choose different routes, planning strategies and behaviours in the process of plotting their trajectory through the scene. This shift is likely to occur between observation settings and therefore between the datasets collected client-side.

Concept Shift Since social norms may vary in different cultural settings the same observed behavior up to a point might result in different future outcomes. This is especially true if a social interaction, like collision avoidance, is bound to happen after the observation time span.

Quantity Skew Data Volume available to individual clients might vary strongly. Specifically in cross-device federation settings the data volume is highly dependent on edge device user behavior.

The more general umbrella term **Data Shift** refers to a difference in distributions between training and testing datasets. The shift might on one hand stem from temporal changes in those distributions but will on the other hand inevitably occur in federated learning schemes as a mix encompassing the identicality violations discussed above. As multiple clients, each potentially being exposed to very different scenarios thus recording different data, contribute to common global models, Data Shift might be apparent initially in lower performance in client-side inference if these models are tested locally. High diversity in clients will likely exacerbate this shift.

Kairouz et al. [18] stated that the presence of non-IID data presents one the fundamental challenges of the field. For state of the art centralized methods, like the pedestrian trajectory forecasting models used in this project, to be federated in real world application all of the listed factors need to be considered by investigating the underlying characteristics of the datasets involved in the training campaign. It is important to be cognizant of this when using synthetic and/or academic datasets which might not necessarily show the same diversity in distributions compared to datasets in real world application. Emulating the scenario of clients collecting their own local data in various contexts by splitting open-access academic datasets, which were captured under clear economic constraints, is non trivial.

3.2.4 Federated Optimization

Federated Averaging Originally published in 2017 [25] by a team at Google the Federated Averaging (FedAvg) algorithm is considered to be the baseline aggregation and training algorithm in Federated Learning. On the client-side SGD is applied as the optimizer.

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w) \tag{3.14}$$

with $f_i(w) = L(x_i, y_i, w)$ typically being the prediction loss on (x_i, y_i) produced with model parameters ω. Similiar to mini-batch SGD Federated Averaging can be applied in the mini-batched variant as illustrated in Algorithm 2. Clients are indexed by i; B denotes the local minibatch size, E is the number of local epochs with e being the current local epoch and η is the learning rate.

Algorithm 2 ***FedAvg*** mini-batched

Initialization: $\omega_0, ClientOpt$
for $t = 0, ..., T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $e = 0, ..., E$ **do**
      for $b \in \mathcal{B}_i$ **do**
        $g_i^t = \nabla f_i(\omega_i^t, b)$
        $\omega_i^t = ClientOpt(\omega_i^t, g_i^t, \eta, t)$
      end for
    end for
  end for
  $n = \sum_{i \in \mathcal{S}} n_i$
  $\omega_{t+1} = \sum_{i \in \mathcal{S}} \frac{n_i}{n} \omega_i^t$
end for

More recent experimental studies investigated the convergence behaviour and the limitations of FedAvg in various non-IID data distribution settings. It has become clear that FedAvg does not converge reliably towards the optimum without a decay in learning [23] and that even newer non-stateful approaches only outperform FedAvg if certain types of non-IID skews are prevalent [22]. Learning rate scheduling could be applied, but would hard to design optimally for real world applications due to the lack of information on the concise distributions in local datasets.

Adaptive Federated Optimization As discussed above in cross-device federated learning settings clients are not considered to be stateful. As a consequence adaptive optimization algorithms like Adam (3.1.1) that keep track of momentum cannot easily be deployed meaningfully on the client-side, since the optimizer state is lost during the aggregation during communication rounds and clients are not likely to be selected for two consecutive communication rounds in real world applications. Even including the momentum values in aggregation wouldn't make sense since the distributions in the data of another client might look very different.

To improve upon the limitations of the FedAvg algorithm and enable the application of adaptive optimization methods in federated optimization Reddi et al. [28] devised an algorithmic setup they called FedOpt that accommodates a server-side as well as a client-side optimizer algorithm. As the server-side itself is stateful adaptive optimization algorithms can be applied in a meaningful way while on client-side non-stateful optimizers, such as SGD are applied. For mini-batched training the client-side optimizer takes one step after each mini-batch b out of the set of mini-batches $\mathcal{B}_i$ exclusive to the client i out of the set of contributing clients $\mathcal{S}$ in communication round t. As different clients may have different training data volume available to them, Δ_t is calculated

through a weighted sum that takes the share of the number of samples n_i available to the client i in the overall number of samples n processed in this particular communication round into account. Aggregated model difference is denoted by Δ_t. The server-side learning rate and client-side local learning rate are denoted by η and η_l respectively. The number of epochs is denoted by E with current epoch e.

Algorithm 3 ***FedOpt*** mini-batched

Initialization: $\omega_0, ClientOpt, ServerOpt$
for $t = 0, ..., T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $\omega_i^t = \omega_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $e = 0, ..., E$ **do**
      for $b \in \mathcal{B}_i$ **do**
        $g_i^t = \nabla f_i(\omega_i^t, b)$
        $\omega_i^t = ClientOpt(\omega_i^t, g_i^t, \eta_l, t)$
      end for
    end for
    $\Delta_i^t = \omega_i^t - \omega^t$
  end for
  $n = \sum_{i \in \mathcal{S}} n_i$
  $\Delta_t = \sum_{i \in \mathcal{S}} \frac{n_i}{n} \Delta_i^t$
  $\omega_{t+1} = ServerOpt(\omega_t, -\Delta_t, \eta, t)$
end for

FedOpt proceeds to use $-\Delta_t$ as a pseudo-gradient in the server-side optimizer updates. Reddi et al.[28] provide proofs for convergence guarantees for the server-side application of adaptive optimizers, such as Adagrad [8], Adam[19] and Yogi [38], and thus enable the meaningful application of these well established adaptive optimization methods in cross-device federated settings.

Building on FedOpt, Federated Adam or FedAdam[28] is introduced. For this the Adam algorithm is selected for server-side optimization and SGD is choosen for client-side optimization. In contrast to the classic Adam algorithm the gradient which is then used to calculate the momentum parameters m_t and v_t is obtained through a weighted sum of the differences Δ_i^t between local model state ω_i^t of client i and of the current global model state ω^t. The weight is determined by the quotient of the number of samples n_i available to client i for training during the communication round and the over all number of all samples n across all clients during the communication round. Thus client that incorporated more samples in training than other clients participating in this communication round will be given more weight in relation to other clients. The mini-batch version of FedAdam is illustrated below. As with Adam 3.1.1 the decay parameter are denoted by β_1 and β_2. The parameter τ expresses the degree of adaptivity with behaviour being more adaptive for smaller values of τ. m_t and v_t denote the first and second momentum respectively.

Algorithm 4 ***FedAdam*** mini-batched

Initialization: $\omega_0, v_{-1} \geq \tau^2$, decay parameters $\beta_1, \beta_2 \in [0, 1)$
for $t = 0, ..., T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $\omega_i^t = \omega_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $e = 0, ..., E$ **do**
      for $b \in \mathcal{B}_i$ **do**
        $g_i^t = \nabla f_i(\omega_i^t, b)$
        $\omega_i^t = \omega_i^t - \eta_l g_i^t$
      end for
    end for
    $\Delta_i^t = \omega_i^t - \omega^t$
  end for
  $n = \sum_{i \in \mathcal{S}} n_i$
  $\Delta^t = \sum_{i \in \mathcal{S}} \frac{n_i}{n} \Delta_i^t$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\Delta^t$
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) m_t^2$
  $\omega_{t+1} = \omega_t + \eta \frac{m_t}{\tau + \sqrt{v_t}}$
end for

In order to federate the training for SocialLSTM, which is trained through Adam in modern benchmark implementations[20] for classic centralized settings FedAdam would be the natural pick, as it seeks to provide the adaptive qualities of Adam in federated learning settings.

Since the adaptive optimization only takes place once at the end of each communication round, trade-offs informed by transmission cost and hardware resource availability will likely have to be made between convergence rate and communication efficiency.

4 Experiments

4.1 Programming Frameworks

The programming language for the implementation was chosen as Python in version 3.7.10. All tensor and neural network related code is implemented with PyTorch 1.8.1 [26]

All forecasting related code was adapted from the Trajnet++ [20] repository. This includes the implementations of models, data pre-processing and the evaluation system. This evaluation system was adopted to enable clear comparison in performance between classically trained models and models trained in various federated learning training configurations.

The federation framework FedML[15] was chosen over the competition (figure 2.3) due to it's modularity and research focused feature set. The standalone category of federation experiments allows for the simulation of federated training schemes on a single machine or server without the need for multiple separate networked machines. Due to the modularity moving the setup to actually distributed machines is designed to be comparatively low hassle. It also provides a template setups for various state of the art federation algorithms and federated optimization [25][28] enabling exploration of more methods without additional deployment overhead.

4.2 Datasets

All experiments are conducted on the three publicly available datasets UCY [21], ETH [27] and the original Trajnet [20] were selected due to wide availability and use in academic research. The datasets are provided by and pre-processed through tools of the Trajnet ++ repository [20]. UCY, ETH and Trajnet Combined include 14804 scenes of which 12337 scenes constitute the available training dataset and 2467 scenes remained reserved exclusively for validation utilized only in final performance evaluation. Through pre-processing to only coordinates for the trajectories the actual overall data volume is reduced to 5.6 MB.

Each scene consists of the trajectories of all pedestrian in the scene recorded over a time period of 8 seconds with position being sampled every 0.4 seconds yielding 20 frames per scene.

In order to simulate a federated learning setting the complete training dataset is split up to guarantee that data given to the clients for the experiments is exclusive to the client and that communication round during the complete training process. This is done to emulate a setting in which each client could represent in a separate car continuously gathering new data during operation.

An intermittent communication model [34] is assumed. The distribution is client selection over all contributing clients $Q(k)$ is deliberately chosen to be even with all clients participating in each communication round. Real world application scenarios would likely include far more clients with only a small subset participating in a communication round. The impact of client selection in real world scenarios will unlikely be understood in such a limited standalone simulation setting with less than a dozen clients. The influence of client choice was therefore eliminated all together.

4.3 Training System Parameters and Hyperparameters

Experiments with 9 different parameter settings were conducted. The training data was reduced to 12000 scenes to accommodate different parameter choices whilst still providing the same number of training scenes to all experiments. The following table outlines the essential parameter settings for the experiments and numbers them. In the following specific experiments will be referenced by their number.

Experiment	Clients	Optimizer	Communication Rounds	Mini-Batches	Loss
1	1	Adam	None	1500	L2
2	5	FedAvg	15	20	L2
3	10	FedAvg	15	10	L2
4	5	FedAdam	15	20	L2
5	10	FedAdam	15	10	L2
6	5	FedAvg	30	10	L2
7	10	FedAvg	30	5	L2
8	5	FedAdam	30	10	L2
9	10	FedAdam	30	5	L2
10	5	FedAdam	15	20	Gaussian
11	10	FedAdam	15	10	Gaussian

Table 4.1: Experiments conducted with essential training parameters

As surveys have shown[12] reproducing the same results in performance metrics is non-trivial for most publications. This project tries to compare centralized with federated training in the same performance metrics. Therefore is the published benchmark implementation provided by the Trajnet++ repository[20] is trained on the local system in experiment 1 to provide a realistic baseline for comparison.

A comparison between adaptive as well as non adaptive federated optimization schemes is conducted to investigate the level of improvements provided by the state-of-the-art adaptive algorithms[28] over the more common averaging schemes[25]. Clear improvements are expected as

benchmark implementations of Social LSTM utilize Adam[19]. The parameters β_1 and β_2 were set to 0.9 and 0.999 respectively. For the gaussian loss σ_x and σ_y were set to 3.

Inspired by the Trajnet++ implementation of Social LSTM all experiments are conducted with a mini-batch size of 8 over 25 epochs. The epoch number appears to be very low, but no detailed information on epoch number or its motivation was given in the relevant publication these experiments are based on [1][20]. It is important to note that in federated settings the term epoch or local epoch refers to passes over the exclusive subset of the complete training data that is located at a specific client for the current communication round. This way it can be ensured that the overall system has processed each scene 25 times regardless whether it was trained in a centralized or federated fashion.

The experiments for federated settings encompass a selection of choices for the number of clients, the number of communication rounds, the number of mini-batches as well as 2 optimization schemes. All other system design and hyperparameters remain constant throughout.

As shown by Reddi et al.[28], FedAvg is a special case of FedOpt by selecting the server-side optimizer as SGD with server-side learning rate $\eta = 1$ while the client-side optimizer is set as SGD with client-side local learning rate η_l being set to a value of choice. The experiments utilize the built implementation of FedOpt in this fashion.

In federated optimization adaptive optimization steps are only taken on the server-side after a communication round potentially adjusting momentum. This means that for these experiments here only 15 or 30 adaptive optimizer steps are being taken. This is drastically less than the centralized training scheme with adaptive optimization steps taking place after each of the 1500 mini-batches.

4.4 Evaluation

During the training of all federated learning the platform Weights and Biases [3] is used to track and log various outputs, such as epoch loss. This is done mainly for efficiency reasons, since the experiments could potentially be run from multiple machine at the same time and plots of the tracked variables and outputs can generated automatically.

Real model performance is evaluated through the evaluator functions provided in the Trajnet++ repository, which were also used to evaluate all experiments in corresponding publication [20]. As discussed in the methods chapter 5 performance metrics are considered initially.

- ADE
- FDE
- COL-II with weights
- Topk-ADE with $k = 3$
- Topk-FDE with $k = 3$

For all experiments in federated settings the global model state is saved for every communi-

cation round. By running the whole set of these states through the evaluation pipeline these 5 performance metrics are obtained for every communication round yielding the convergence behaviour in performance metrics over time.

4.5 Hardware Setup

All experiments were run with following hardware components:

Processor	Intel XEON E5-1650 v3	3.50 GHz
Mainboard	GigaByte MW50 SV0	
Ram	Kingston ValueRAM DIMM 8 GB DDR4-2133 E	4 x 8 Gb

Table 4.2: Hardware setup for experiments

5 Results

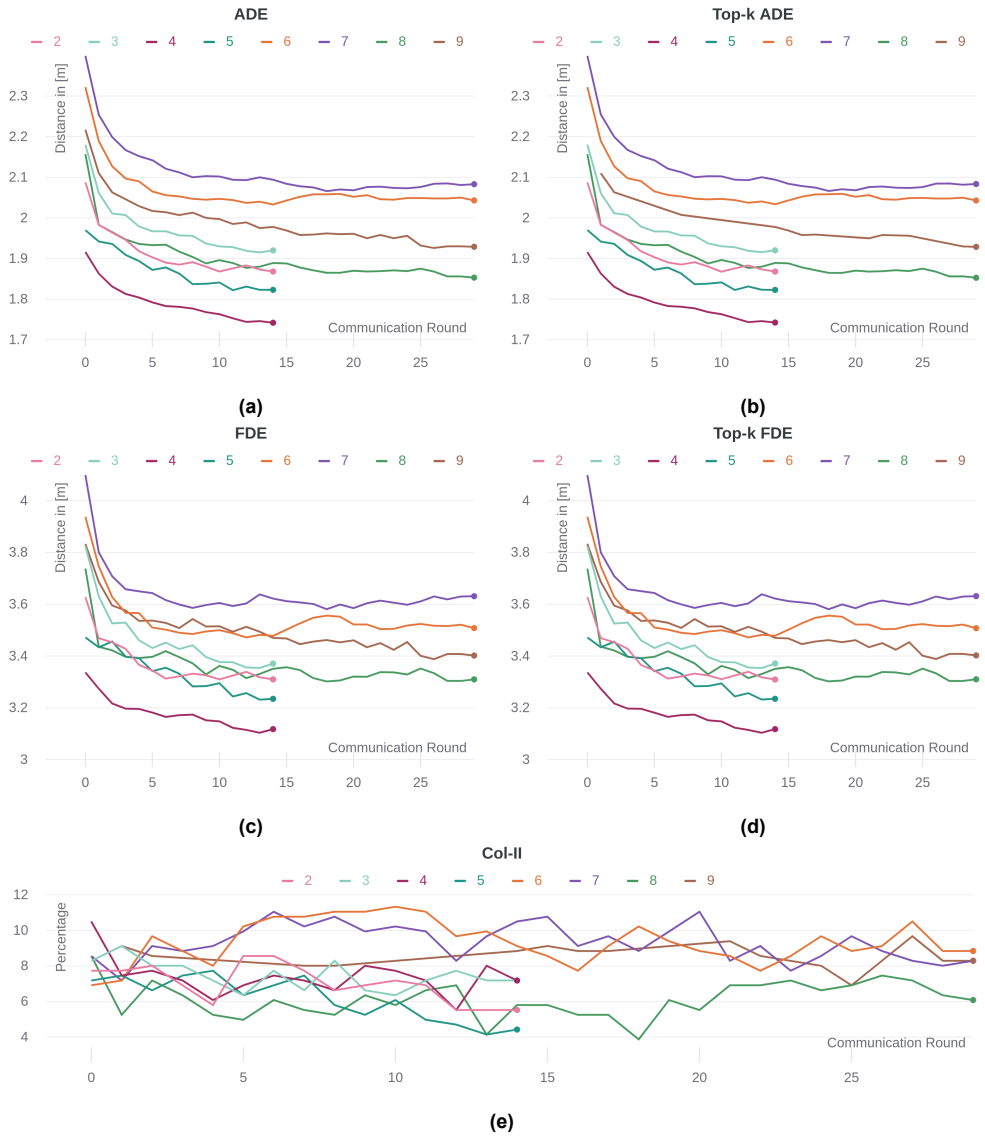

Figure 5.1: Performance metrics for the 8 experiments in federated settings trained with L2 loss over communication rounds

Initially the overall performance of this projects implementation of federated pedestrian trajectory forecasting is presented, discussed and compared to published benchmark values as

well as the locally retrained centralized baseline. A more detailed discussion on the impact of design choices in our experiments follows. Finally results are put into context of a scene and the application field of pedestrian behaviour forecasting.

The final results after 12000 scenes for experiments run under L2 loss training show far lower performance for the federated learning experiments than the locally trained centralized training baseline as well as the published results in the reference work of Kothari et al.[20]. It is also clear that the trajectory found in the unimodal prediction scenario under L2-loss remains the the top choice throughout almost all communication rounds with Top-k ADE in experiment 9 being the only exception. As depicted in figure 5.1 with sub-figures (a) and (b) being near identical as well as sub-figures (c) and (d) being identical.

No clear reduction in the groundtruth collision percentage Col-II are visible for experiments 6 through 9 with 30 communication rounds as well as experiments 3 and 4 with 15 communication rounds while mild reductions are visible for experiments 2 and 5 after 15 communication rounds.

	1	2	3	4	5	6	7	8	9
ADE	1.053	1.868	1.920	1.742	1.823	2.043	2.083	1.853	1.929
FDE	2.143	3.310	3.371	3.118	3.235	3.508	3.631	3.31	3.402
Top k ADE	1.053	1.868	1.920	1.742	1.823	2.043	2.083	1.853	1.929
Top k FDE	2.143	3.310	3.371	3.118	3.235	2.043	3.631	3.31	3.402
Col-II	4.42	5.52	7.18	7.18	4.42	8.84	8.29	6.08	8.29

Table 5.1: Final performance metrics results for retrained baseline (1) and all experiments (2-9) under L2-loss training

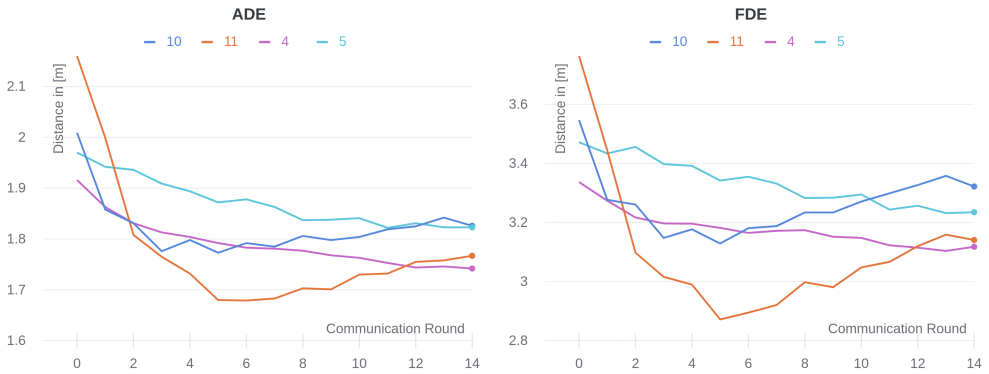

Figure 5.2: Performance metrics for the 2 top performing L2-loss experiments and their corresponding experiments with Gaussian loss over communication rounds

During the experiments design choices in 3 parameters with 2 options being tested. To gain insight into the impact of these design choices comparisons between the different choices in

	4	5	10	11
ADE	1.742	1.823	1.826	1.767
FDE	3.118	3.235	3.322	3.141
Top k ADE	1.742	1.823	1.826	1.767
Top k FDE	3.118	3.235	3.322	3.141
Col-II	7.18	4.42	8.29	9.67

Table 5.2: Final performance metrics results of the 2 top performing L2-loss experiments and their corresponding experiments with Gaussian loss

the optimization scheme, number of clients and number of communication rounds are presented and discussed. As disclaimed in the previous section the overall number of scenes available is limited to 12000. This means that the number of clients and number of communication rounds dictate the number of scenes available to each client for training during each communication round.

In figure 5.2 the 2 best performing experiments from the L2-loss experiment series are compared to their corresponding experiments trained under Gaussian loss. Both experiment trained under Gaussian loss don't exhibit their best performance in the last communication round. for experiment 11 the best performance is shown after communication round 5. Experiment 10 exhibits the best FDE performance after communication round 5 with ADE performance reaching the best performance after communication round 3 and 5. As shown in table 5.2 neither L2-loos nor Gaussian loss clearly yield better results with experiment 4 outperforming its corresponding experiment 10 while experiment 11 is outperforming experiment 5.

The same graphs shown in figure 5.1 (a) and (c) are color contrasted with corresponding experiments only differing in the specific parameter discussed depicted in the same line strength for immediate comparison.

5.1 Optimization Scheme

Even in experiments with clients being exposed to far less data than a centralized model the value of server-side optimization becomes clear. Every experiment run performed conclusively better in both ADE and FDE metrics with FedAdam applied instead of FedAvg. The performance is consistently superior from the communication round 0 onward pointing to the first communications rounds being critical as all future training is based on the these initial global models.

This raises the question whether pre-trained models stemming from classic centralized training could be of use as initial global models to potentially ensure higher performance from the start. This hybrid approach with centralized pre-training would potentially suffer from non-IID impacts on performance due to shift between distributions observed in pre-training and federated training. This would likely be an issue if purely academic data, like the datasets used in this project, would be used for pre-training and federated training would then take place in a variety of real world training.

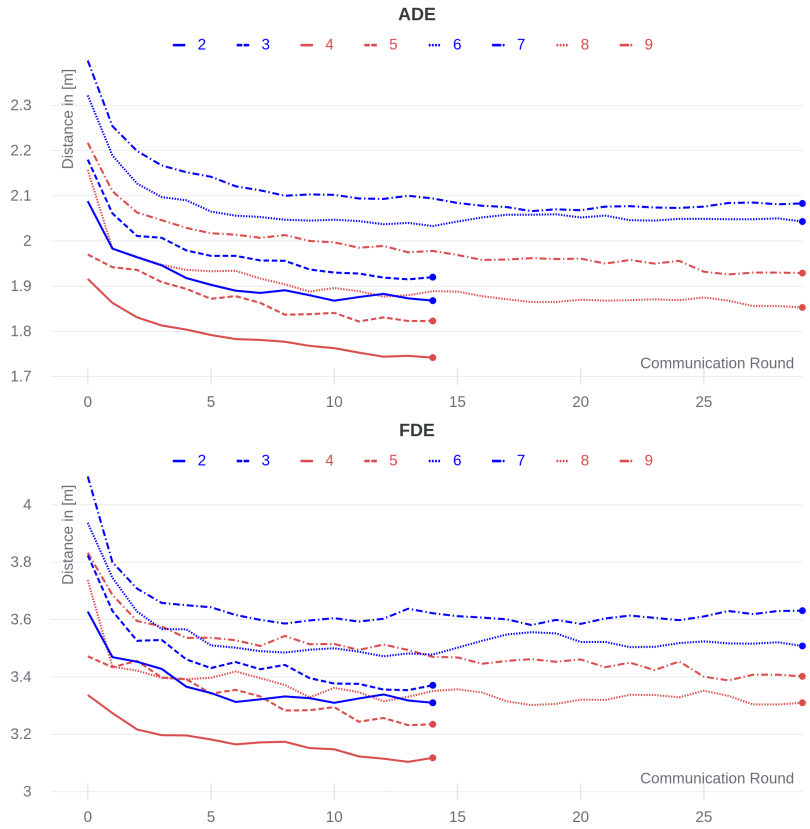

Figure 5.3: Experiments run with FedAdam in red, Experiment run with FedAvg in blue

5.2 Number of Clients and Communication Rounds

As shown in figure 5.4 experiments with the lower number of 5 clients consistently outperform their counterparts with 10 clients. This is also clearly the case for the lower number of 15 communication rounds compared to 30 communication rounds. Corresponding experiments have an offset four in the experiment number (e.g. experiment 2 corresponding to experiment 6). This indicates that the number of mini-batches available to clients for training in each communication round is a very strong indicator for final performance. As shown in figure 5.5 this is actually the case with higher number of mini-batches available to clients for each communication round resulting in higher performance. The best performance out of each of the 3 groups always applies the FedAdam optimizations scheme.

This is notable since, with the collective network seen as singular learning system including all participating clients as well as the central server, all tested learning systems were provided the same dataset. Each scene is exposed through 25 epochs on the corresponding client exclusively regardless whether the whole dataset is given to a single client as in the centralized baseline case or the dataset being distributed to 10 clients. Thus it is likely that the distribution amongst clients and the following impact on the overall optimization process predominantly contributed to

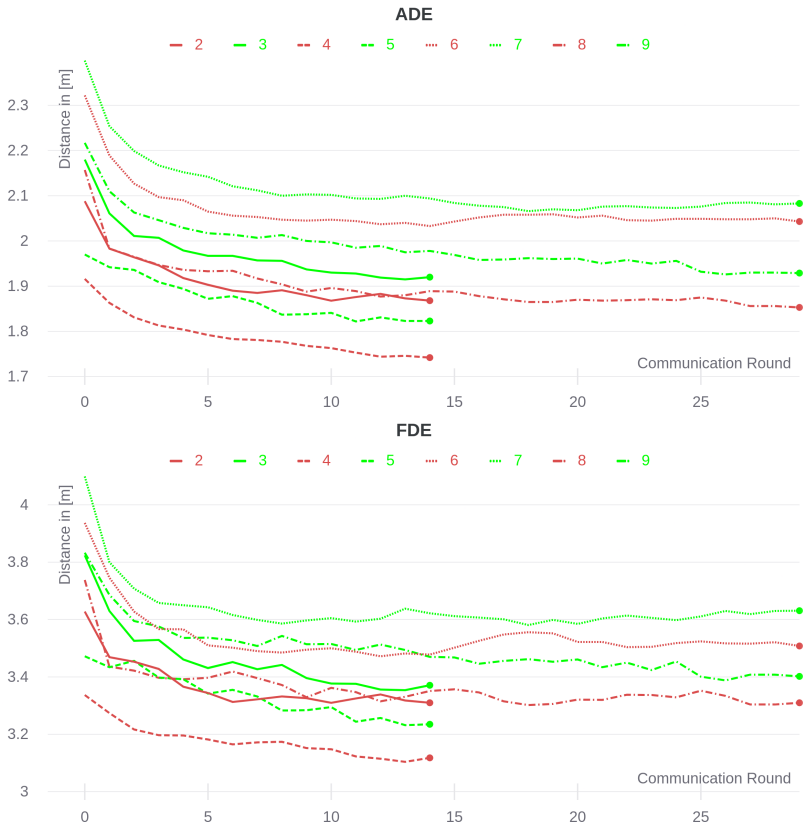

Figure 5.4: Even experiment numbers with 5 clients in red, Odd experiment numbers with 10 clients in green

the decrease in final performance.

For this academic research settings such as this project the aspect of splitting of standard academic data set with limited size is important to consider. Publicly available academic datasets commonly found in machine learning research are usually fully used for benchmarking performance. For federated learning more training data is necessary to achieve the same overall performance as classic centrally trained models.

In order not limit the convergence of optimizers severely due to too little data being available per communication round and to still be able to observe emergent behaviours in distributed training fittingly large datasets are necessary. Certain mathematical convergence guarantees and bounds for state of the art federated optimization algorithms such as FedAdam were provided by Reddi et al. [28], but will unlikely be sufficient to solely determine the necessary training data volumes to yield on par performance with classic training for many fields of application and their intricacies of distributions in the underlying datasets.

Further testing regarding the heuristics of how centralized datasets should be split is needed

Figure 5.5: Experiments with 30 mini-batches available to clients in each communication round shown in orange, 20 mini-batches in light blue and 10 mini-batches in purple

to understand the impact of non-IID distributions between the resulting subsets. Academic datasets would provide a good basis for this as the circumstances of recording are well documented and various post-processing steps to enrich the dataset can be provided through the broader research community. An example for this is the categorisation of trajectories into social interaction types, which is provided by Kothari et al. through the Trajnet++ benchmarking framework[20] and could be used to deliberately design non-IID distributions for the data subsets necessary for simulated federated learning experiments.

6 Conclusion

This project was motivated through the timely alignment in innovation uptake in the field of federated learning as well as the rise of software defined and connected automobiles. The personal realization that large volumes of highly diverse trajectory samples were likely necessary for pedestrian behavior forecasting models to be applicable in diverse cultural settings and traffic contexts prompted the research into whether the training of these models could be federated.

As the first contribution of the project a functional testing environment was setup by combining the state of the art frameworks FedML from the field of federated learning and Trajnet++ from the field of pedestrian trajectory forecasting. The second contribution and to knowledge of the author the first foray into the federation of pedestrian trajectory forecasting was made through standalone simulation of federated training of Social LSTM models with grid-based social pooling on the standard datasets provided through the Trajnet++ benchmark. Recently developed methods for adaptive federated optimization were utilized and compared in performance to previous standard methods in federated optimization.

All of the 10 conducted experiments in federated learning settings demonstrated considerably lower performance that the locally retrained baseline or the published benchmark. The performance was measured through the evaluation tools provided by the Trajnet++ framework. To gain a deeper insight into the impact of investigated design parameters on final performance experiments were grouped and compared based on the choices in the design parameters number of clients, number of communication rounds and choice of optimization method. It was found that FedAdam optimization consistently outperformed FedAvg optimization in otherwise identical settings. A small sample comparison of two the top performing experiments trained under L2-loss with their corresponding experiments trained under Gaussian loss did not yield a clear indication for better performance after training under one loss over the other.

Due to the limitation in available training data the volume of data available to each client for training during every communication round was determined through the choice for the number communication rounds as well as the number of participating clients. It was found that even though the overall volume of training data remained the same for all experiments the data volume allocated each clients in a communication round was strong indicator for performance with higher allocated data volumes settings outperforming setting with lower allocated data volumes when the same optimization method was applied.

As more sophisticated pedestrian trajectory forecasting models have been benchmarked and studied in centralized settings and adaptive federated optimization has recently become available establishing the field of federated pedestrian trajectory forecasting appears challenging

but promising. The testing environment developed for this project provides the option for the federation of various additional reference implementations of pedestrian trajectory forecasting models besides social pooling in Social LSTM which are included in the Trajnet++ framework but are outside the scope of the empirical observations of this project. Specialized extensions of the FedML framework such as FedGraphNN[14] could be used to federate scene graph based trajectory forecasting approaches such as Trajectron++ [30].

Prior to a empirical survey of the current range of state of the art models in this field a toolkit for the splitting of academic datasets for standalone simulation should be developed. The experiments of this project have indicated that the number and size of training data subsets is clear factor contributing the final performance. Being able to analyse the distributions within these subset to not only identify but also to quantify the identicality breaking skew, shift and drift phenomenons could prove instrumental in informing more performant system designs. Academic could provide a good basis for this as they are widely studied, in some cases distributions well understood and often enriched through categorisation or other meta data provided through the broader research community.

Bibliography

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. “Social LSTM: Human Trajectory Prediction in Crowded Spaces”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[2] A. Alahi, V. Ramanathan, and L. Fei-Fei. “Socially-aware Large-scale Crowd Forecasting”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[3] L. Biewald. *Experiment Tracking with Weights and Biases*. 2020. URL: `https://www.wandb.com/`.

[4] C. Briggs, Z. Fan, and P. Andras. “Federated learning with hierarchical clustering of local updates to improve training on non-IID data”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. June 2020.

[5] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. “nuScenes: A Multimodal Dataset for Autonomous Driving”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[6] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar. *LEAF: A Benchmark for Federated Settings*. 2019. arXiv: `1812.01097`.

[7] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret. “WILDTRACK: A Multi-Camera HD Dataset for Dense Unscripted Pedestrian Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[8] J. Duchi, E. Hazan, and Y. Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* (2011).

[9] A. M. Elbir, B. Soner, and S. Coleri. *Federated Learning in Vehicular Networks*. 2020. arXiv: `2006.01412`.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[11] W. A. Group. *FATE - An Industrial Grade Federated Learning Framework*. 2020. URL: `https://fate.fedai.org/`.

[12] O. E. Gundersen and S. Kjensmo. “State of the Art: Reproducibility in Artificial Intelligence”. In: *AAAI*. 2018.

[13] O. Gupta and R. Raskar. “Distributed learning of deep neural network over multiple agents”. In: *Journal of Network and Computer Applications* 116 (2018).

[14] C. He, K. Balasubramanian, E. Ceyani, Y. Rong, P. Zhao, J. Huang, M. Annavaram, and S. Avestimehr. *FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks*. 2021. arXiv: 2104.07145.

[15] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, X. Zhu, J. Wang, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr. *FedML: A Research Library and Benchmark for Federated Machine Learning*. 2020. arXiv: 2007.13518.

[16] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* (Nov. 1997).

[17] A. Ingerman and K. Ostrowski. *"Introducing TensorFlow Federated," Mar. 6, 2019*. URL: https://blog.tensorflow.org/2019/03/introducing-tensorflow-federated.html.

[18] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. "Advances and Open Problems in Federated Learning". In: *Foundations and Trends® in Machine Learning* 14 (2021).

[19] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*. Jan. 2017.

[20] P. Kothari, S. Kreiss, and A. Alahi. "Human Trajectory Forecasting in Crowds: A Deep Learning Perspective". In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–15. DOI: 10.1109/TITS.2021.3069362.

[21] A. Lerner, Y. Chrysanthou, and D. Lischinski. "Crowds by Example". In: *Computer Graphics Forum* 26 (2007).

[22] Q. Li, Y. Diao, Q. Chen, and B. He. *Federated Learning on Non-IID Data Silos: An Experimental Study*. 2021. arXiv: 2102.02079.

[23] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. "On the Convergence of FedAvg on Non-IID Data". In: *International Conference on Learning Representations*. 2020.

[24] N. Majcherczyk, N. Srishankar, and C. Pinciroli. *Flow-FL: Data-Driven Federated Learning for Spatio-Temporal Predictions in Multi-Robot Systems*. 2020. arXiv: 2010.08595.

[25] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Proceedings of Machine Learning Research. PMLR, 2017.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

[27] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. "You'll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, Sept. 2009.

[28] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan. *Adaptive Federated Optimization*. 2020.

[29] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach. *A generic framework for privacy preserving deep learning*. 2018. arXiv: 1811.04017.

[30] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. "Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data". In: *European Conference on Computer Vision (ECCV)* (2020).

[31] D. C. Shoup. *The High Cost of Free Parking (1st ed.)* Routledge, 2016.

[32] J. G. Smart and S. D. Salisbury. "Plugged In: How Americans Charge Their Electric Vehicles". In: (July 2015).

[33] L. Sun, Z. Yan, S. M. Mellado, M. Hanheide, and T. Duckett. "3DOF Pedestrian Trajectory Prediction Learned from Long-Term Autonomous Mobile Robot Deployment Data". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 5942–5948.

[34] B. E. Woodworth, J. Wang, A. Smith, B. McMahan, and N. Srebro. "Graph Oracle Models, Lower Bounds, and Gaps for Parallel Stochastic Optimization". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[35] Q. Yang, Y. Liu, T. Chen, and Y. Tong. *Federated Machine Learning: Concept and Applications*. 2019. arXiv: 1902.04885.

[36] D. Ye, R. Yu, M. Pan, and Z. Han. "Federated Learning in Vehicular Edge Computing: A Selective Model Aggregation Approach". en. In: *IEEE Access* (2020).

[37] Y. Yuan, X. Weng, Y. Ou, and K. Kitani. *AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting*. 2021. arXiv: 2103.14023.

[38] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. "Adaptive Methods for Nonconvex Optimization". In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

List of Figures

List of Tables

Appendix

	Datacenter distributed learning	**Cross-silo federated learning**	**Cross-device federated learning**
Setting	Training a model on a large but "flat" dataset. Clients are compute nodes in a single cluster or datacenter.	Training a model on siloed data. Clients are different organizations (e.g. medical or financial) or geo-distributed datacenters.	The clients are a very large number of mobile or IoT devices.
Data distribution	Data is centrally stored and can be shuffled and balanced across clients. Any client can read any part of the dataset.	**Data is generated locally and remains decentralized.** Each client stores its own data and cannot read the data of other clients. Data is not independently or identically distributed.	
Orchestration	Centrally orchestrated.	**A central orchestration server/service organizes the training**, but never sees raw data.	
Wide-area communication	None (fully connected clients in one datacenter/cluster).	Typically a hub-and-spoke topology, with the hub representing a coordinating service provider (typically without data) and the spokes connecting to clients.	
Data availability	———— All clients are almost always available. ————		Only a fraction of clients are available at any one time, often with diurnal or other variations.
Distribution scale	Typically 1 - 1000 clients.	Typically 2 - 100 clients.	Massively parallel, up to 10^{10} clients.
Primary bottleneck	Computation is more often the bottleneck in the datacenter, where very fast networks can be assumed.	Might be computation or communication.	Communication is often the primary bottleneck, though it depends on the task. Generally, cross-device federated computations use wi-fi or slower connections.
Addressability	Each client has an identity or name that allows the system to access it specifically.		Clients cannot be indexed directly (i.e., no use of client identifiers).
Client statefulness	Stateful — each client may participate in each round of the computation, carrying state from round to round.		Stateless — each client will likely participate only once in a task, so generally a fresh sample of never-before-seen clients in each round of computation is assumed.
Client reliability	———————— Relatively few failures. ————————		Highly unreliable — 5% or more of the clients participating in a round of computation are expected to fail or drop out (e.g. because the device becomes ineligible when battery, network, or idleness requirements are violated).
Data partition axis	Data can be partitioned / re-partitioned arbitrarily across clients.	Partition is fixed. Could be example-partitioned (horizontal) or feature-partitioned (vertical).	Fixed partitioning by example (horizontal).

Figure .1: Typical characteristics of federated learning settings vs. distributed learning in the datacenter. Cross-device and cross-silo federated learning are two examples of FL domains, but are not intended to be exhaustive. The primary defining characteristics of FL are highlighted in bold, but the other characteristics are also critical in determining which techniques are applicable. [18]