Federated Learning for 5G Base Station Traffic Forecasting

Vasileios Perifanis[1], Nikolaos Pavlidis[2], Remous-Aris Koutsiamanis[3], Pavlos S. Efraimidis[4]
[1,2,4]Democritus University of Thrace, Greece, [3]IMT-Atlantique – Inria – LS2N, France

Corresponding author: Vasileios Perifanis, vperifan@ee.duth.gr

Abstract *– Mobile traffic prediction is of great importance on the path of enabling 5G mobile networks to perform smart and efficient infrastructure planning and management. However, available data are limited to base station logging information. Hence, training methods for generating high-quality predictions that can generalize to new observations on different parties are in demand. Traditional approaches require collecting measurements from different base stations and sending them to a central entity, followed by performing machine learning operations using the received data. The dissemination of local observations raises privacy, confidentiality, and performance concerns, hindering the applicability of machine learning techniques. Various distributed learning methods have been proposed to address this issue, but their application to traffic prediction has yet to be explored. In this work, we study the effectiveness of federated learning applied to raw base station aggregated LTE data for time-series forecasting. We evaluate one-step predictions using 5 different neural network architectures trained with a federated setting on non-iid data. The presented algorithms have been submitted to the Global Federated Traffic Prediction for 5G and Beyond Challenge. Our results show that the learning architectures adapted to the federated setting achieve equivalent prediction error to the centralized setting, pre-processing techniques on base stations lead to higher forecasting accuracy, while state-of-the-art aggregators do not outperform simple approaches.*

Keywords – Data Privacy, Federated Learning, Mobile Networks, Non-iid data, Traffic Forecasting

1. INTRODUCTION

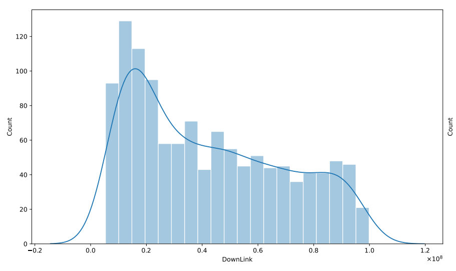
With the development and increasing deployment of fifth-generation cellular networks (5G), networking infrastructure faces the challenges of traffic handling from heterogeneous devices, load balancing, and in general, the reliability of traffic management [1]. It is necessary to employ techniques for traffic forecasting with low prediction error to improve the quality of services and provide intelligent resource management and orchestration without performance degradation. Since forecasting is based on historical data, the goal is to identify therein spatio-temporal data patterns to generate high quality predictions.

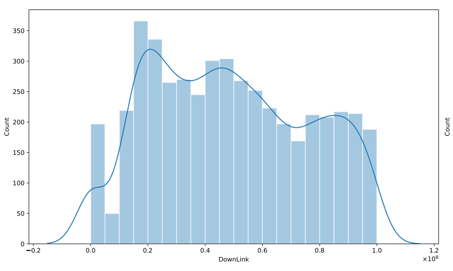
In the context of *Federated Traffic Prediction for 5G and Beyond Challenge*[1], we are given a dataset of three individual base stations in Spain, *ElBorn*, *LesCorts*, and *PobleSec*, each comprising a multivariate time-series, i.e., multiple time-dependent variables such as the Radio Network Temporary Identifiers (RNTI) count and the number of allocated resource blocks (RB). The goal of this challenge is to train a model using a federated learning approach to predict the value of five distinct measurements for the next timestep, given the measurements of the immediatelly previous 10 timesteps. Among the five measurements, the challenge focuses on the uplink and downlink transport block sizes, which are crucial for determining the volume and direction of flows.

[1]https://supercom.cttc.es/index.php/ai-challenge-2022

Most of the recent approaches in time-series forecasting concern Multi-Layer Perceptrons (MLPs), Recurrent Neural Network-based (RNN) models [2], such as Long-Short Term Memory (LSTM), Gated Linear Unit (GRU) or Convolutional Neural Networks (CNNs). These models have been shown to achieve higher predictive accuracy than traditional methods, such as AutoRegressive Integrated Moving Average (ARIMA) and linear regression [3].

Despite the recent advancements in neural networks applied to time-series forecasting, the predictive accuracy is limited by the number of observations or focused on the observations of a single party. For instance, in mobile networks, each base station (one “party” in this sense) has its own unique characteristics and hence, there are no guarantees that a model trained on a single base station will generalize to other base stations, even in the same region/territory. A straightforward solution would be that of allowing base stations to transfer their data to a single datacenter and train a machine learning model with the combined observations. However, transmitting data and training a machine learning model is fragmented by regulations and laws around the world, such as the GDPR and HIPAA [4]. In addition, business confidentiality and competition issues prevent organizations from sharing their data with a third party. To address these limitations, federated learning is proposed as a distributed machine learning paradigm that allows each party to collaborate in a model’s training process without exchanging or revealing their own data


(a) DownLink distribution in ElBorn.


(b) DownLink distribution in LesCorts.

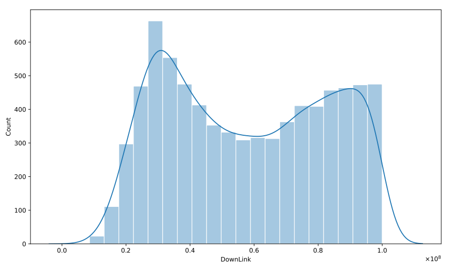

(c) DownLink distribution in PobleSec.

Fig. 1 – Distribution skew between three different base stations.

[5]. After collaborative training, the final global model has the potential to generalize, at least, to the parties that participated in the distributed training.

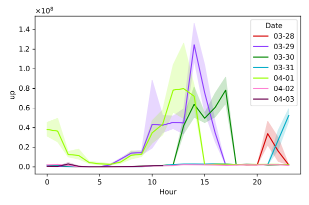
Federated learning differs significantly from the conventional/centralized learning and has its own limitations and characteristics. Except for the differences in the training process between the two approaches, in centralized learning, the processed data are assumed to be independent and identically distributed (iid). On the other hand, the iid data assumption is unlikely to hold in federated settings due to the fact that participating entities can differ significantly, not only in their data distribution but also in the number of observations [6, 7]. Figure 1 presents an example of distribution skew among the three base stations considered in this work regarding the downlink transport block size, which reflects the attribute and target skewness. Note that we consider only the values below 1×10^8 to remain consistent in the comparision of the three base stations and for visualization purposes. Despite the cut at the specified maximum, it is easily observed, that the distribution is different among base stations. For instance, the distribution in the first base station (1a) is positively skewed. In the second base station, the distribution approximates the normal and in the third base station it is bimodal. Table 1 reports the skewness and kurtosis for the uplink and downlink measurements per base station without a cut at a maximum. The skewness and kurtosis for the uplink measurements is greater than 1 and 3 per base station, respectively. Thus, the distributions for the up measurements are positively skewed and the presence of outliers is highly likely. Regarding the downlink values, in LesCorts, the skewness falls in $[-0.5, 0.5]$ and the kurtosis is < 3, which indicate that the distribution is almost symmetrical and the probability of outliers is low. The rest of the base stations follow the observations from the uplink measurements, i.e., positive skewness and presence of outliers. Note that the reported values are consistent with the visualization in Fig. 1.

Besides data and target distribution skewness, the corresponding attributes hold temporal characteristics. In other words, since we deal with time-series representations, federated learning poses another challenge,

	Skewness		**Kurtosis**	
Base Station	Up	Down	Up	Down
ElBorn	8.3262	2.3513	132.0776	5.8289
LesCorts	12.2926	0.4804	321.1801	-0.5705
PobleSec	8.0330	3.5667	101.8521	23.8463

Table 1 – Skewness and kurtosis for the UpLink and DownLink measurements per base station.

namely, the temporal skew [8]. First, the observations are collected in different time periods, i.e., March/April, 2018 - January, 2019 and February/March, 2018 per base station, respectively. Second, the data distributions change over time. An example of a changing distribution in the ElBorn base station for the up and down measurements is given in Fig. 2. It is observed that (at least) three out of seven days follow different patterns regarding the up and down measurements, which indicates local temporal skew. For instance, the pattern from the previous day(s) may not follow the pattern for subsequent days. Note that the lineplots correspond to the mean values per hour with 95% confidence interval since we performed a grouping operation by hour.


(a) UpLink changing distribution.

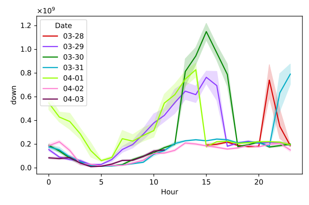

(b) DownLink changing distribution.

Fig. 2 – UpLink and DownLink distribution per day and hour in ElBorn.

The resulting unique characteristics of federated learning, i.e., distribution, quantity and temporal skew among participating entities can lead to a huge accuracy degradation [8, 9], which is attributed to weight divergence in the training process.

Inspired by the neural networks applied to the centralized setting for time-series forecasting, in this work, we study state-of-the-art learning models, including RNNs and a three dimensional CNN applied to the federated setting. One of the most crucial steps in federated learning is the aggregation function, especially when dealing with non-iid data. Most of the proposed aggregators have been evaluated on different domain tasks such as image classification, sentiment analysis and next character/word prediction [10]. Hence, in addition to the learning models, we assess the influence of state-of-art federated aggregation functions, including Federated Averaging with proximal term (FedProx) [11], Federated Averaging with Server Momentum (FedAvgM) [12], Federated Normalized Averaging (FedNova) [13] and Adaptive Federated Optimization (FedOpt) and its variations [14] in a real-world time-series forecasting task with non-iid data.

The main contributions of this paper are summarized as follows:

- We identify the challenge of training high quality federated forecasting models under three perspectives: i) the data distribution skew, ii) the quantity skew and iii) the temporal skew.
- We experimentally compare five machine learning models trained under three different settings: i) individual learning, i.e., each base station performs training and evaluation using only the local observations, ii) centralized learning and iii) federated learning.
- We conduct an extensive experimental study using nine different aggregation algorithms, some of which are specifically designed for handling non-iid data.
- We show that pre-processing techniques can lead to huge prediction error degradation and have more influence on the final prediction than state-of-the-art aggregation algorithms for non-iid data.
- We open source a framework for evaluating federated time-series forecasting[2]. Our approach is flexible and fully customizable and can serve as baseline for the federated time-series research.

Through our experiments we show that the prediction accuracy of federated models is on par with individual and centralized learning. However, a model's quality can differ significantly among learning settings, e.g., the highest quality model on centralized learning may not be the best model under the federated setting and vice versa. Finally, we experimentally show that local pre-processing techniques and specifically outliers flooring and capping can lead to huge error degradation, while state-of-the-art aggregation algorithms for non-iid data do not outperform other, simpler, approaches. To the best of our knowledge, this is the first study that applies federated learning to raw LTE data, considers attribute and target distribution, quantity and temporal skew and evaluates the prediction error by combining five machine learning models and nine aggregation functions.

The rest of this work is structured as follows. Section 2 introduces the main concepts of time-series forecasting and federated learning and reviews related work on federated traffic prediction. Section 3 defines the federated traffic prediction problem and presents the design of the federated setting as well as the federated aggregation functions. Section 4 discusses the experimental results. Finally, Section 5 concludes our work.

2. PRELIMINARIES AND RELATED WORK

2.1 Time-series Forecasting

Time-series forecasting is an essential task in a wide range of real-life problems such as weather [15], energy consumption [16], sales [17] and traffic prediction [3]. In recent years, there has been significant increase in machine learning methods applied to time-series data. Unlike classic methods such as ARIMA-based models, deep learning approaches have shown great success in modeling time-series since they can map non-linear observations. The neural architectures that can model time-series data are classified into the following three categories [18]: (i) Fully Connected Neural Networks such as MLPs, ii) Recurrent Networks such as RNNs, LSTMs and GRUs and iii) Convolution Networks such as CNNs. For more details regarding generalized time-series analysis using deep learning methods we refer our readers to [18].

In this work, we apply federated time-series analysis on raw LTE data to model the spatio-temporal dependence of traffic demand. Traffic forecasting is a crucial task in the 5G era due to the rapid traffic growth and changing patterns over time [19]. In the context of traffic prediction, Trinh et al. [3] compared LSTM to MLP and ARIMA and showed that LSTM significantly outperforms the rest models. Feng et al.[20] proposed DeepTP, an LSTM-based network which models the spatial dependence of time-series and temporal changes and showed that the proposed framework outperforms LSTM without enhancements and traditional methods such as ARIMA. Chen et al. [21] proposed a clustered LSTM-based model for multivariate time-series modeling. Sebastian et al. [22] proposed a decomposition of time-series to seasonal, trend and cycle components and fed the resulting vectors into the GRU model. Gao [23] proposed an enhancement of an LSTM network with stationarity evaluation (SLSTM), specifically designed for the measurements of a 5G network.

Despite the recent advancements in modeling time-series data, most works concern centralized learning, which differs significantly from federated learning. In addition, federated time-series forecasting has yet to be explored

[2]https://github.com/vperifan/Federated-Time-Series-Forecasting

in-depth, especially in the context of mobile networks. In this work, we consider five different model architectures that cover the three model categories presented in [18]. In particular, we consider: i) a simple MLP, ii) three RNN-based models and iii) a 3D-CNN. Note that all models are evaluated under the federated setting using raw base station data. To our knowledge, this is the first study that provides insights using different time-series modeling under federated learning.

2.2 Federated Learning

Federated Learning is a machine learning technique, originally proposed by McMahan et al. [5] as a Google AI project, which allows data to be trained collaboratively across multiple devices. This approach promises to revolutionize the model training process as it differs from traditional centralized machine learning techniques where all local data are uploaded to a central server, as well as from more conventional decentralized approaches which often assume that local data samples to be identically distributed [24]. The federated setting has been evaluated in a plethora of machine learning tasks, including image classification [5], next word prediction [10], customer analysis [25], collaborative filtering [26] and time-series forecasting [27].

During the federated learning process, each participating entity performs local training on its private data using the model weights shared by a central server. Then, unlike traditional machine learning, participants only share their calculated model parameters, hence avoiding the direct privacy leakage of their local data. The central server (often also called the aggregation server) will then gather these parameter updates and aggregate them using an aggregation function. The result of the aggregation corresponds to the new global model that then will be shared to the participants for the next round. An overview of one round of the federated process is given in Fig. 3

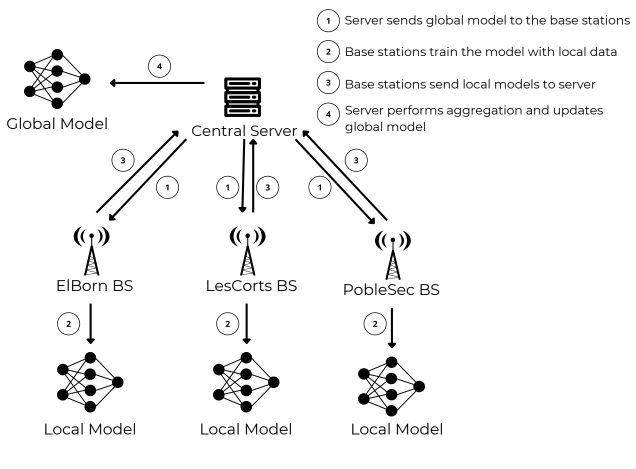

1 Server sends global model to the base stations
2 Base stations train the model with local data
3 Base stations send local models to server
4 Server performs aggregation and updates global model
Global Model
Central Server
ElBorn BS
LesCorts BS
PobleSec BS
Local Model
Local Model
Local Model

Fig. 3 – Federated Learning Process.

In this paper, we follow the common approach of federated training on raw base station time-series data, i.e., a third-party is responsible for collecting weight updates per federated round, summarizes the results using an aggregation algorithm and distributes the aggregated data to participants for local training and weight updating.

2.2.1 Federated Learning on non-iid Data

Although federated learning promises high quality privacy-preserving model generation by design, there has been extensive research regarding the effect of non-iid data on the model's accuracy [6, 8, 24]. In particular, the observations from base stations hold most categories of non-iid data, i.e., attribute, label, quantity and temporal skew [8, 24]. In our case, since we use historical data to provide one-step predictions, the attribute and label values are presented in both training features and target values. Assuming a local distribution of $P(x_i, y_i) = P(x_i|y_i)P(y_i)$, feature and label distribution skew is that $P(x_i)$ and $P(y_i)$ are different among participating entities, respectively. Quantity skew means that the number of observations vary among users. Finally, temporal skew means that the timesteps of observations can vary or the feature distribution can be changing over time (Fig. 2). Thus, we study the effectiveness of federated learning by considering a combination of non-iid data categories, which, to our knowledge has yet to be explored. The most closely related work regarding the evaluation of federated learning under non-iid data is presented in Li et al. [6], in which non-iid categories are considered independently on image and tabular datasets for the classification task.

2.3 Federated time-series Forecasting

Time-series forecasting has a wide range of applications in various industries and lots of practical applications. To comply with regulations and laws as well as to preserve user privacy, federated learning has also been explored in the context of time-series prediction.

Taik et al. [27] used a LSTM model applied to the federated setting for the household load forecasting task. After the generation of the global model, participants perform a fine-tuning step using their local data to achieve personalization. The federated LSTM is evaluated on a dataset consisting of 200 households with the FedAvg algorithm [5].

Liu et al. [28], in one of the earliest works on federated traffic flow prediction, proposed FedGRU with the FedAvg algorithm [5] to predict the number of vehicles in an area. Before launching the federated training, the participating entities join clusters using the K-Means algorithm and after grouping, participants train a global model for each cluster. The experimental part considers a dataset with 39,000 transportation sensors in major metropolitan areas of California and FedGRU is compared with centralized models such as GRU and LSTM. Based on the reported results, FedGRU outperforms centralized models when the clustering mechanism is enabled.

Liu et al. [29] proposed a federated attention-based CNN-LSTM model for anomaly detection using time-series data with the FedAvg algorithm. In addition, the communication cost of federated training is minimized using a selection-based gradient compression algorithm.

Briggs et al. [30] proposed FedLSTM applied to load forecasting similar to [27], using a dataset comprised of 100 households with the FedAvg algorithm [5]. FedLSTM integrates exogenous data such as weather information and applies a clustering step among participating entities by considering the weight similarity of local models. From the experimental results under FedLSTM, federated learning with local fine-tuning presents the best results, while the predictive accuracy is almost equivalent in individual, centralized and federated learning.

Fekri et al. [31] adapted LSTM to the federated setting and used FedAvg and FedSGD [5] for the load forecasting task, similar to [30]. Before training, the data owners performed local pre-processing, i.e., participants perform feature scaling locally. In this work, we argue that local scaling leads to inconsistencies during training and we show that global scaling leads to the lowest forecasting error.

All of the previous works in federated time-series forecasting do not raise the issue of non-iid data and evaluate the generated global models only using the FedAvg algorithm. In this work, we raise the challenge of training federated models under the presence of four categories of non-iid data, we evaluate different aggregation functions and show that local pre-processing techniques can lead to higher predictive accuracy. In addition, similar works only adapt a single neural network architecture to the federated setting, e.g., LSTM [27, 30, 31] or GRU [28], and hence, insights regarding different federated models applied to time-series forecasting have yet to be provided. To present generalized results, we evaluate five different neural networks using three different learning settings. We emphasize that in our case, the participating entities (base stations) are limited to three and hence, clustered federated learning [28, 30] cannot be applied. Finally, since federated learning requires several rounds of operations, we measure the communication cost regarding the total uplink and downlink transmission of model weights and the energy consumption for training the global model with respect to the carbon footprint (CO_2eq).

3. METHODOLOGY

3.1 Problem Formulation

For simplicity, we first describe the problem regarding individual learning, i.e., the setting where a single party holds observations and performs local training. Let $X_t = \{x_0, ..., x_d\}$ be the received measurements at time step t, where d is the number of measurements (variates). Note that in individual learning, the whole process will be executed internally without data transmission. Given a time-series of a window T at time step t, $X_t^* = \{X_{t-T+1}, ..., X_t\}$, our goal is to predict the measurements for the subsequent time step, $\hat{y}_{t+1}$ based on the T past observations. Using the whole dataset of measurements $D_{ind} = \sum_{i=1}^{m} X_t^*$, where m is the number of time-series containing the window T, the goal is to build a model that can generalize on unseen future series.

In centralized learning, the collected measurements are transmitted to a third-party or a datacenter and hence, the party that trains a machine learning model uses the combination of observed measurements. Given the combined data from n participants $D_{cen} = \sum_{i=1}^{n} D_{ind}$, the goal is to build a model that can generalize on the future series, at least, for the n participants.

Federated learning can be classified as the intermediate setting of individual and centralized learning. More precisely, there are n participants who wish to collaboratively build a forecasting model that can generalize on their future observations. Each participant $p \in n$ holds its own time-series and trains a learning model locally for a limited number of epochs. Then, participants transmit their locally-learned weights derived from their local observations to an aggregator and an averaged model is generated. The process continues until the averaged global model can generalize to the observations of the n participants. Consequently, in federated learning, each participating entity optimizes a model based on the local data, which is equivalent to the problem formulation of individual learning, while the generated global model has the capability of generalization to n entities, which is the ultimate goal of machine learning.

Unlike centralized/individual learning, a federated setting can potentially exploit the privacy/confidentiality guarantees of individual learning since participants do not share their raw data and utilizes the predictive power of n local models to build a global model that can lead to generalization. We emphasize that federated learning alone does not provide strict privacy guarantees [32, 33]. For instance, an attacker can revert the model's weights and identify observations with high probability, e.g., using gradient inversion [34] or a membership inference attack [35]. Evaluating the privacy guarantees of federated learning is out of scope of this work and is left for future research.

3.2 Federated Traffic Forecasting Design

Traditional machine learning approaches do not involve the participation of multiple clients during the training stage. For instance, a participating entity can perform training using only the local data (individual learning) or traditional approaches with multiple users require the transmission of their local data (centralized learning). In contrast, federated learning requires active participation since a global model is generated by collecting and aggregating local models per federated round.

To simulate a real-world scenario for the traffic forecasting task, we developed and open-sourced a framework that can be customized and extended to generalized time-series forecasting. Our architecture is based on the Flower federated learning framework [36] and uses PyTorch [37] as the backbone deep learning library. In addition, we have integrated several state-of-the-art aggregation algorithms that allow us to evaluate their applicability and forecasting accuracy on non-iid data using the NumPy package [38]. Hence, our code can be easily customized and extended using additional models implemented with the popular PyTorch library or/and additional aggregation algorithms using NumPy. Note that the transition from federated learning to the individual or centralized setting is also supported. Figure 4 illustrates the main components of the federated pipeline. It consists of two core entities, the *Server* and the *Client*. The Server is responsible for selecting available clients per federated round, logging historical information, such as the loss and evaluation metrics per client as well as aggregating and generating the global model parameters. A Client represents individual entities that participate in the federated computation. At a higher level, each participant is provided with methods that apply two core operations for machine learning: i) pre-processing, which is executed before federated learning and ii) a training pipeline that allows participants to perform local training and evaluate the local (global) model. The next subsections describe the main components of our approach.

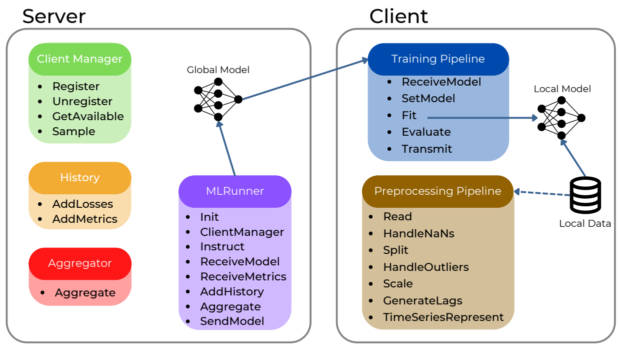


Fig. 4 – Federated time-series forecasting overview.

3.2.1 *Data Pre-Processing*

In general, time-series data are collected automatically from sensors or logging mechanisms. For instance, base stations keep logging information regarding network measurements or homes are equipped with smart meters to measure the electricity consumption. It is natural for the data to hold redundant information, errors or unusual spikes. Hence, pre-processing techniques are decisive for the subsequent learning task. Pre-processing happens before the training stage and from a machine learning perspective is indispensable since the applied transformations can lead to higher prediction accuracy. In the context of time-series we can discern four steps:

1. **Data Cleansing.** The goal of this step is to handle missing or corrupted data and identify and manage outliers. The missing data can be handled by removing them, using a transformation technique, e.g., a transformation to a constant, filling them with a representative value like the mean, or estimating them using an algorithmic approach. In this work, we adopt a simple technique and transform the missing values to zeroes. We selected *zero transformation* because a simple removal breaks the concept of continuous data. Besides, time-series are changing over time, i.e., imputing them with a constant such as the mean may not be representative during inference, and estimating them can be time and energy consuming. In the future we will assess the influence of additional data cleansing approaches.

   Outliers concern values that fall far away from most observations on a dataset and can negatively impact the learning performance. From Table 1 it is evident that all base stations and especially the uplink measurements contain extreme outliers. Similar to handling missing values, there exist several approaches to detect outliers such as statistical or learning-based models. However, those methods require several trials, which, again, is time and energy consuming. Regarding outliers, we adopt a commonly used and simple approach, which is the *flooring and capping* technique. With this technique there is no detection step, but the values that fall below and above the specified percentiles are being transformed with the values of given low and high percentiles.

2. **Train/Validation/Test Split.** This step applies a division on a given dataset to three subsets. The training set is used to fit a given model. The validation set is used to calculate useful statistics regarding the learning performance and adjust the hyper-parameters of the model. The test set is a holdout subset that is used after training to evaluate the model's accuracy on unseen data. We are provided with a training and testing dataset per

base station. The training dataset is split into 80% for training and 20% for validation, while the testing set remained intact.

3. **Feature Scaling.** The task we are trying to solve concerns multivariate time-series, which means that at each timestep there are several different attributes with different scales. To eliminate the influence of value ranges, we perform the commonly used *Min-Max normalization* and transform all variates to a value between $[0, 1]$. In particular, the transformation for each measurement in each variate is:

$$x' = \frac{x - min}{max - min},$$

where x is the observed measurement and min and max are the minimum and maximum values observed in the variate, respectively.

4. **Time-Series Representation.** The original data that concern variates per timestep should be tranformed to a time-series representation given the window T as well as to input features X and target variables Y. In other words, the provided measurements are combined to generate a dataset of sliding windows governed by T. In this work, we use $T = 10$ and transform the dataset in such a way that the targets Y correspond to the next timestep. Technically, assuming 100 observations with 11 variates and using $T = 10$, we end up with a dataset consisting of 90 instances, where each instance corresponds to an array containing 11×10 (variates $\times$ T) measurements for the input features X.

The pre-processing pipeline is an integral and indispensable step for machine learning. Data cleansing and feature scaling in a federated setting can be more involved than in individual and centralized approaches. For instance, the central server collects the data from different entities and considering high availability, it performs computations applied to the combined datasets. However, federated learning poses several challenges. First, executing trials of data transformation cannot be performed in real world settings since federated learning require active participation. Second, locally imputing missing values or outliers with a constant such as the mean lead to the following inconsistency: each participant holds different observations and consequently, the calculated statistics naturally differ among entities, which in turn, can lead to noise injection on the machine learning algorithm. Third, estimating the missing values locally leads not only to additional costs, but also to the same inconsistency with the local imputation. Simplistic approaches such as zero transformation and flooring and capping, which are selected in this work, do not involve additional communication and the transformations lead to consistent samples. Similarly, feature scaling requires a computation of the mininum and maximum (Min-Max) or other techniques like standardization require the mean and standard deviation. One approach is to locally compute the required statistics and scale the corresponding values. However, such an approach leads to the inconsistency discussed for the local imputation. For instance, consider two clients, holding a maximum of 500 and 1000, respectively. Executing the Min-Max transformation locally, leads the first client to transform the value of 500 to 1 and the second client to transform the value of 1000 to 1. Now, the machine learning algorithm during training observes the value of 1 two times, possibly with different/inconsistent targets. Unlike data cleansing, in the feature scaling step, a computation of a global variable over local values is almost inevitable. In practice, participants should only transmit their minimum and maximum values to a third-party who announces the global minimum and maximum. When confidentiality is of high importance, a privacy-preserving protocol for calculating the corresponding values should be integrated. The focus of this work is to study the effectiveness of federated learning and hence, privacy-preserving computations are left for future work.

Our pre-processing pipeline is designed in such a way to handle the three different learning settings. In centralized learning, the pipeline is owned by the central server, while in individual and federated settings, preprocessing is transferred to the client side. More precisely, after collecting some data, the owner feeds the observations to the pre-processing pipeline. Then, a handler for missing values is executed and the dataset is split to training and validation sets. After splitting, an outlier handler is applied to the training set. Outliers should not be transformed on the validation set since it concerns samples that are not provided during training. After outlier handling, the features are being scaled and finally, the data are represented as time-series using a sliding window of T. After pre-processing, the data are ready to be fed to a machine learning algorithm for training. We note that for handling missing values and outliers we selected the zero transformation and flooring and capping, respectively. In the real-world these techniques do not require any communication between participants. For feature scaling, we selected the Min-Max technique, which requires a global minimum and maximum. Here, we simply assume that participants transfer their local minimum and maximum per variate to a third-party who announces the global values.

We emphasize that pre-processing techniques heavily influence the model's performance. However, to our knowledge, there has been limited research for this stage applied to federated learning, i.e., how different entities should handle and transform their data. A similar observation is made in [31], where the authors used local pre-processing and scaled the data individually on each client. We argue that individual pre-processing leads to inconsistent transformations among participants and we experimentally show that global scaling leads to higher model quality. In addition, we experimentally show that the application of flooring and capping for outliers handling leads to a remarkable forecasting error decrease.

3.2.2 Federated Training

The training process of federated learning requires some modifications, at least for the global model generation, compared to traditional approaches. We discern two entities involved in the computation, the Server and the Client. At a high level, the server first samples available clients and transmits the current global model. Sampled clients receive the global model and perform local training using their data. After local training, the updated model is transmitted back to the server along with historical metrics such as the loss and evaluation metrics. After receiving all local models, the server aggregates them and updates the state of the global model. In our approach, all clients are represented under the Client structure.

Client Operations. After pre-processing, the local data is ready to be fed to a machine learning algorithm. The training operations under federated learning is equivalent to centralized and individual settings. More precisely, each selected participant receives the global model that was transmitted from the Server and evaluates it using its local data. Then, it sets the weights for the local model by copying the global model parameters. After initialization, the model is fit to the local data for a limited number of epochs instructed by the Server. At each epoch, the client evaluates the local model using the corresponding validation set. The final local model is the model that achieved the lowest error with the validation set, which is transmitted back to the Server.

Server Operations. The key difference between federated learning and traditional learning approaches is that a central entity is responsible for coordinating the training process and generating the global model per federated round. Fig. 5 illustrates the operations and communication between the server and clients per round. Below we describe the core operations performed on the server side.

1. **Announcement.** The Server announces the computation and clients are being registered as participants. At this stage, the server also informs participants regarding the pre-processing to follow for data transformations.

2. **Client Sampling.** After registration, the server initializes a model that is ready for local training. At each federated round, the server gets the available clients and samples a fraction of them. Our approach provides flexibility regarding those operations through the *Client Manager* interface: client registration/removal as well as sampling methods can be extended and customized to different scenarios. For instance, simulating client dropouts and different client selection and sampling criteria are supported through slight modifications. In this work, we are interested in small-scale federated learning with 3 participants and we assume that they are always available. We also set the sampling fraction to 1, i.e., the server selects all participants per federated round.

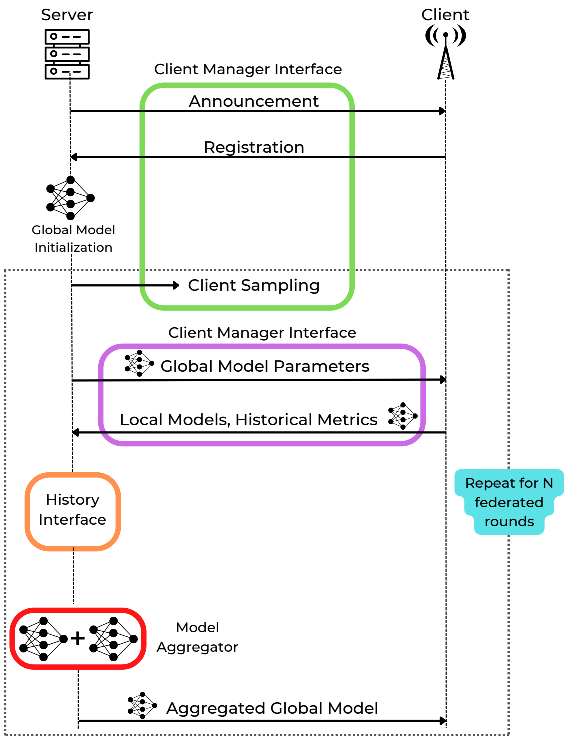


Fig. 5 – Server Client Operations during Federated Training.

3. **Instruction.** After selection, participants are being instructed to execute local training with some specified parameters. More precisely, the server communicates the global model parameters and the hyper-parameters to be used for local training such as the number of local epochs and the learning rate through the *MLRunner* interface. Again, those parameters are fully customizable and our framework allows dynamic changing, i.e., instructions can be different between clients per round. In this work, we use the most simple scenario for client instructions and choose pre-defined hyper-parameters without dynamic changing.

4. **Model and Metric Collection.** The server awaits for clients to execute local training and transmit their local models along with historical metrics. After collection, the server appends historical training information using the *History* in-

terface. Historical metrics are used to generate the final global model after federated training and for model interpretation. Specifically, the server keeps in memory the global model that achieved the lowest averaged loss. The averaged loss is computed using as weights the number of local training samples per participant. Similarly, the server keeps historical metrics using a weighted aggregation. The methods for aggregating historical information can be customized according to use-case needs.

5. **Aggregation.** After appending historical data and collecting local models, the server performs an aggregation function to generate the new global model. We consider many aggregation algorithms to evaluate their applicability and robustness to a setting with combined categories of non-iid data. The next subsection introduces the aggregation techniques used in this work. Note that after aggregation steps 2 through 5 are repeated until a specified number of rounds.

After federated training, the server transmits the final global model to participants. By design, federated learning can be executed dynamically at different time periods to capture new observations and increase the predictive accuracy. In addition, participants can also perform a local fine-tuning step to drift the model closer to their local data and provide higher quality forecasting predictions. Finally, to perform predictions on unseen data, each client should perform the pre-processing operations defined at the announcement phase and feed the transformed data to the trained model. The predictions should be transformed back to the original feature range to obtain the final prediction.

3.2.3 Federated Aggregation

One of the most important steps of federated learning is model aggregation. As we already mentioned, during this phase the central server collects and aggregates models from participants to update the state of the global model. This process shows a lot of difficulties, especially when the data are non-iid and heterogeneous among the clients. For this reason, there are many research efforts focusing on aggregation algorithms.

The most common algorithm used for federated learning operations is **FedAvg**, proposed by McMahan et. al [5]. The aggregation is performed by computing a weighted average of client models based on the data quantity, which consequently, results in clients with more samples having more influence on the newly aggregated model. However, this process does not always lead to adequate results. In some cases, FedAvg can cause objective inconsistency, that is, the global model converges to a stationary point of a mismatched objective function which can be arbitrarily different from the true (global) objective. This inconsistency is the result of non-iid and heterogeneous data presence. For this reason, alternatives of FedAvg have emerged to capture data heterogeneity.

To provide theoretical guarantees under non-iid data, Li et al. [11] proposed a re-parametrization of FedAvg. **FedProx** is a generalization of the FedAvg algorithm and it offers robust convergence, especially in highly heterogeneous settings. More precisely, a tunable term μ is introduced to control the local objective, which limits the distance between the previous and current model weights.

Another promising algorithm is **FedNova** [13], a method that correctly normalizes local model updates during averaging. FedNova's primary principle is that it averages the normalized local gradients by dividing them with the number of local steps that each client executed individually, as opposed to averaging the cumulative local gradient without any normalization step. FedNova ensures objective consistency while preserving fast error convergence.

FedAvgM [12] is a FedAvg variation that gains advantage by accumulating model updates using server momentum. Momentum is computed by iteratively multiplying previous model updates with a hyperparameter β in each epoch while concurrently adding to the result the new ones.

Finally, an attempt to create federated versions of known adaptive optimizers, namely **FedAdagrad**, **FedYogi** and **FedAdam**, is presented in [14]. These methods promise to solve cthe lient heterogeneity problem, improve performance and degrade communication cost. However, their efficiency depends heavily on hyper-parameter $(\lambda, \beta_1, \beta_2)$ optimization. Algorithm 1 summarizes the state-of-the-art aggregation methods considered in this work.

4. EXPERIMENTS

4.1 Dataset Description

We were provided with a dataset of real LTE Physical Downlink Control Channel (PDCCH) measurements, obtained from three different base stations in the city of Barcelona, Spain. The dataset is made available under the Federated Traffic Prediction for 5G and Beyond Challenge and a preliminary work using the corresponding data is presented in [3]. Both anonymity and accuracy are ensured, since the unique identifiers of users are not accessible, while individual communication can be separated within the dataset for obtaining high-definition traces. More specifically, the dataset consists of the following Base stations, that are also considered as separate FL clients:

1. **LOCATION 1, "ELBORN":** 5421 samples, collected from 2018-03-28 15:56:00 to 2018-04-04 22:36:00

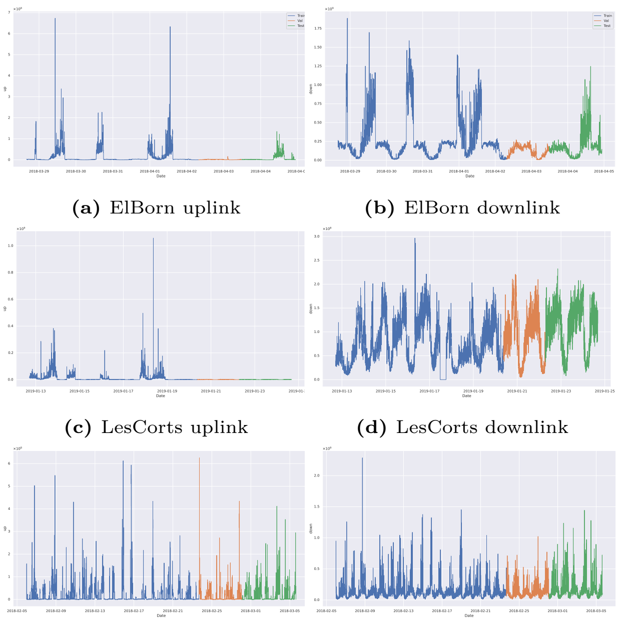
Algorithm 1: FedAvg Algorithm and it's variations FedProx, FedNova, FedAvgM, FedAdagrad, FedYogi, FedAdam

```
Input: local datasets $D^i$, number of clients $N$, number of federated rounds $T$, number of local epochs $E$, learning rate $\eta$, $beta$, $\beta_1, \beta_2 \in [0, 1)$ for FedAvgM, FedYogi & FedAdam, $\lambda$ degree of adaptivity
Output: Final global model parameters vector $w^T$
Server executes:
initialize $w^0$
for $t=0,1,...,T-1$ do
    Sample a set of parties $S_t$
    $n \leftarrow \sum_{i \in S_t} |D^i|$
    for $i \in S_t$ in parallel do
        send the global model $w^t$ to client $C_i$
        $\Delta w_i^t, \tau_i \leftarrow$ LocalTraining$(i, w^t)$
    $\Delta W \leftarrow \sum_{i \in S_t} \frac{|D^i|}{n} \Delta w_k^t$
    For FedAvg/FedProx:
    $w^{t+1} \leftarrow w^t - \eta \Delta W$
    For FedNova:
    $w^{t+1} \leftarrow w^t - \eta \frac{\sum_{i \in S_t} |D^i| \tau_i}{n} \sum_{i \in S_t} \frac{|D^i|}{n \tau_i} \Delta w_k^t$
    For FedAvgM:
    $u_t \leftarrow \beta u_{t-1} + \Delta W$
    $w^{t+1} \leftarrow w^t - u_t$
    For FedAdagrad:
    $u_t \leftarrow u_{t-1} + \Delta W^2$
    $w^{t+1} \leftarrow w^t + \eta \frac{\Delta W}{\sqrt{u_t} + \lambda}$
    For FedYogi:
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) \Delta W$
    $u_t \leftarrow u_{t-1} - (1 - \beta_2) \Delta W^2 sign(u_{t-1} - \Delta W^2)$
    $w^{t+1} \leftarrow w^t + \eta \frac{m_t}{\sqrt{u_t} + \lambda}$
    For FedAdam:
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) \Delta W$
    $u_t \leftarrow \beta_2 u_{t-1} + (1 - \beta_2) \Delta W^2$
    $w^{t+1} \leftarrow w^t + \eta \frac{m_t}{\sqrt{u_t} + \lambda}$
return $w^T$
Client executes:
For every algorithm: $L(w; \mathbf{b}) = \sum_{(x,y) \in \mathbf{b}} l(w; x; y)$
For FedProx:
    $L(w; \mathbf{b}) = \sum_{(x,y) \in \mathbf{b}} l(w; x; y) + \frac{\mu}{2} ||w - w^t||^2$
LocalTraining$(i, w^t)$ :
$w_i^t \leftarrow w^t$
$\tau_i \leftarrow 0$
for epoch $k=1,2,...,E$ do
    for each batch $\boldsymbol{b} = \{\boldsymbol{x}, y\}$ of $D^i$ do
        $w_i^t \leftarrow w_i^t - \eta \nabla L(w_i^t; \mathbf{b})$
        $\tau_i \leftarrow \tau_i + 1$
$\Delta w_i^t \leftarrow w^t - w_i^t$
return $\Delta w_i^t, \tau_i$ to the server
```

2. **LOCATION 2, "LESCORTS":** 8615 samples, collected from 2019-01-12 17:12:00 to 2019-01-24 16:20:00

3. **LOCATION 3, "POBLESEC":** 19909 samples, collected from 2018-02-05 23:40:00 to 2018-03-05 15:16:00

For each site, 11 features are provided regarding downlink and uplink. More specifically, our task is to predict **rnti_count**, the number of allocated resource blocks for the upload and download **(rb_up, rb_down)**, as well as the transport block sizes in the uplink and downlink **(up, down)** using as input the observations with a window of $T = 10$ per base station. Fig. 6 illustrates the training, validation and testing uplink and downlink measurements per base station. It is easily observed that the traffic data vary per base station, breaking the iid data assumption of traditional machine learning.



(a) ElBorn uplink **(b)** ElBorn downlink

(c) LesCorts uplink **(d)** LesCorts downlink

(e) PobleSec uplink **(f)** PobleSec downlink

Fig. 6 – Training/Validation/Test uplink and downlink series per base station.

4.2 Metrics

The metrics that we used to compute our prediction error are **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)** and **Normalized RMSE (NRMSE)**.

MAE is a very common metric used for regression problems, since it's value units correspond to those of the predicted targets. MAE is calculated as the average of the absolute error values, that is the absolute (always positive) value of the difference between the expected and predicted output. The MAE for a vector $\hat{y}$ of n predicted values and a vector y of n expected values is calculated using the following formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y_i} - y_i|$$

RMSE is a very popular measure for evaluating the quality of predictions. In essence, it represents how far predictions fall from expected values using the Euclidean distance. It is expressed as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y_i} - y_i)^2}{n}}$$

where n is the number of data points, y_i is the i-th measurement and $\hat{y_i}$ is its corresponding prediction. The presence of square root leads larger errors to have a disproportionately large effect on RMSE, therefore RMSE is sensitive to outliers. Moreover, we take into consideration that our data points, which are measurements of downlink and uplink in bytes, are very large in scale and thus they will lead to large errors. In order to facilitate the comparison between our experiments we use the **NRMSE** for the uplink and downlink measurements, which is a normalized form of RMSE:

$$NRMSE = \frac{1}{\bar{y}}\sqrt{\frac{\sum_{i=1}^{n}(\hat{y_i} - y_i)^2}{n}}.$$

Note that the MAE and RMSE metrics are used to calculate the prediction error using the five predicted values. The NRMSE considers only the uplink and downlink measurements since they are main focus of this research.

Besides metrics for evaluating the prediction accuracy of our learning models, we tried to figure out the energy and environmental cost of our proposed solution by measuring CO_{2eq} emission and power consumption of the models. We obtained these metrics using the Carbontracker Python Library [39].

4.3 Models and Learning Setting

To investigate the effectiveness of state-of-the-art deep learning models applied to the federated setting as well as the contribution of federated aggregation algorithms to the final global model quality, we conduct extensive experiments on the given dataset consisting of three different individual base stations. We compare the model architectures using *individual learning*, in which each base station uses only the local data and trains a personalized model, *centralized learning*, i.e., base stations transmit their data to a third-party and training is performed using the combined dataset and *federated learning*, i.e., base stations collaboratively train a model without exchanging their raw data. We compare the three settings with the following model architectures:

- **MLP.** It is the most simple feed-forward artificial neural network. MLPs cannot operate on matrix representations thus, the time-series are flattened to one-dimensional arrays. The selected architecture consists of 3 hidden layers, $h = \{256, 128, 64\}$.
- **RNN.** It is the most basic type of Recurrent Neural Network and can model temporal dependencies on sequential data. The selected architecture includes a layer of RNN with 128 units. The output is then fed to a MLP with one hidden layer and 128 units.
- **LSTM.** It is an improvement of RNNs, limiting the problem of exploding gradients and can model longer sequential data [18]. Similar to RNNs we select a LSTM layer with 128 units and the output is fed to a MLP with one hidden layer and 128 units to generate the final prediction.
- **GRU.** Similar to LSTMs, the GRU model addresses the problem of exploding gradients on RNNs. On a higher level, it is a simplified version of LSTMs, achieving similar results to LSTMs with lower computational parameters. For GRU, we follow the architecture of RNN and LSTM, i.e., a GRU layer with 128 units and a MLP with one layer consisting of 128 units.
- **CNN.** This kind of networks directly operate on raw data using convolution layers and are successful in computer vision tasks. In multivariate time-series, convolution operations can effectively utilize structural covariance. The selected CNN takes as input a three-dimensional matrix of size $(1, T, \#variates)$ and feeds it to four two-dimensional convolutional layers with filter sizes $\{16, 16, 32, 32\}$. After operating on convolutional layers the output is fed to a two-dimensional average pooling layer and finally to a fully-connected layer.

To keep consistency and provide fair comparisons, we keep the same model architectures on the three learning settings. For each model and learning setting, we select the Adam optimizer [40] for faster convergence with a learning rate of 0.001. The activation function among model layers is ReLU. During training, we optimize the Mean Squared Error and use a batch size of 128. For individual and centralized learning, we use a maximum of 270 epochs and for the federated setting we select 30 rounds with 3 local epochs per participant. The number of epochs for the individual and centralized setting is selected in such a way to provide fair comparison with federated learning, i.e., in federated learning the complete number of epochs is 270 (rounds $\times$ local epochs $\times$ #participants). In all settings, we select the model that achieved the lowest mean squared error on the validation set during training. For the individual and centralized approaches, we also integrate an early stopping component and with a patience of 50 epochs. The complete list of hyper-parameters is given in Table 2.

Hyper-parameter	Value
Optimizer	Adam
Learning rate	0.001
Activation	ReLU
Federated Rounds	30
Local Epochs	3
Centralized Epochs	270
Batch Size	128
Criterion	Mean Squared Error

Table 2 – Complete list of selected model hyper-parameters.

4.4 Pre-Processing Stage Influence

The experimental study is performed on a workstation running Ubuntu 20.04 with 128 GB memory and a RTX 3060 GPU. The programming language is Python. Unless stated otherwise, each experiment is repeated from scratch 150 times with different seeds to ensure accurate and generalized results. For each experiment, we report the averaged results and the stability using standard deviation.

Within this experimental evaluation, we study the influence of the pre-processing stage on final predictions using the federated setting. At this stage we perform federated training using the FedAvg algorithm [5]. Our goal here is to assess whether outlier transformation, using flooring and capping as well as using a global scaler, leads to higher quality predictions.

Base Station	Min. Percentile	Max. Percentile
ElBorn	10	90
LesCorts	10	90
PobleSec	5	95

Table 3 – Optimal minimum and maximum percentiles for flooring and capping after applying a small grid search.

We start by comparing the final predictions by training the models with and without the flooring and capping technique (FC). More precisely, we perform a small grid search to identify the optimal values for FC per base station. Using FC, the values that fall below and above the specified minimum and maximum percentiles are

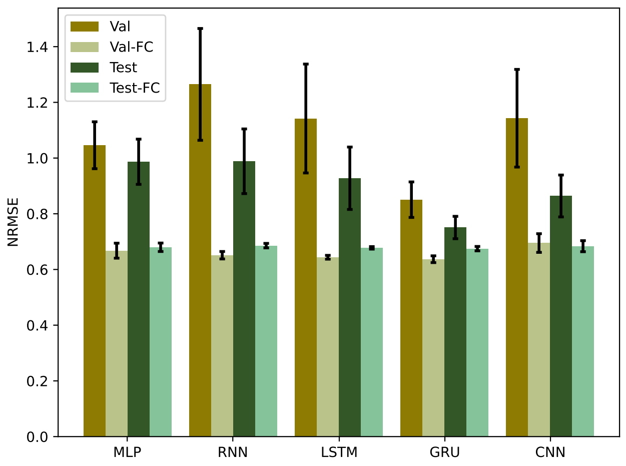
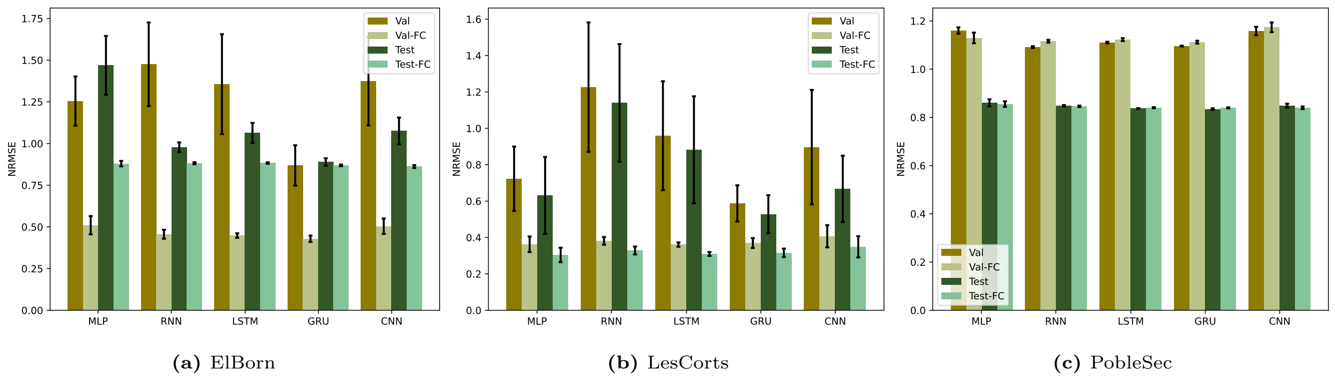


Fig. 7 – Averaged NRMSE with and without flooring and capping. FC denotes the utilization of flooring and capping.

capped and floored to the specified minimum and maximum, respectively. The optimal percentiles per base station after applying a small grid search are given in Table 3. Note that we observed that in most cases, the FC technique provides almost equivalent results. Fig. 7 presents the averaged NRMSE with and without the FC technique on the validation and testing sets, respectively. In general, FC leads to a remarkable decrease in the forecasting error as well as it is more robust with respect to the randomness introduced by the initialization. For instance, consider the results on the LSTM models. The standard deviation is negligible with the FC technique. On the other hand, training without transformations not only leads to lower forecasting accuracy, but also leads the models to be heavily dependent on the random initialization since there is huge deviation across runs.

To provide more comprehensive results, we also present the NRMSE per base station, individually. In Fig. 8, we observe that the trend of training the models using the FC technique in order to achieve lower error is also presented individually, at least for the ElBorn and LesCorts base stations. For instance, regarding the validation NRMSE reduction on ElBorn and LesCorts,



(a) ElBorn **(b)** LesCorts **(c)** PobleSec

Fig. 8 – NRMSE per base station with and without flooring and capping.

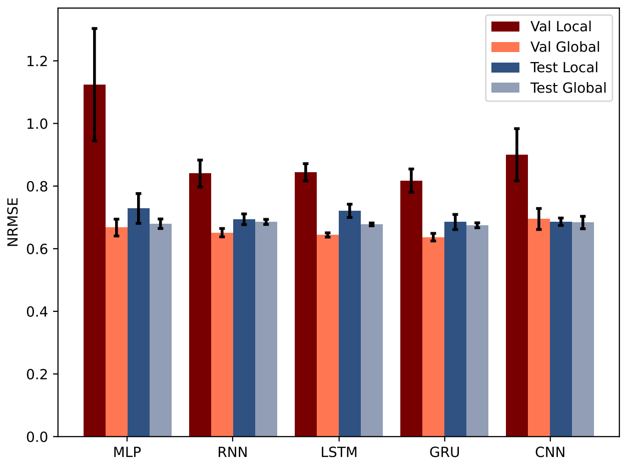


Fig. 9 – Averaged NRMSE using Local and Global Scaling.

using the LSTM model is about 1/3 of the corresponding model trained without FC. Regarding the testing error, we observe that in ElBorn there is a decrease of about 1/5 in all cases, while in LesCorts the decrease is remarkable. On the other hand, the FC technique does not improve the NRMSE in PobleSec, which is attributed to the random spikes and the heavy presence of outliers. Nevertheless, we are interested in a global model that can generalize among base stations. Hence, we continue our experiments using the FC technique.

Besides outlier handling, an integral step of preprocessing is the feature scaling technique. In federated learning we can allow each participant to locally compute the transformation or use a global variable to scale the features. In this work, we select the Min-Max normalization, which transforms each observation according to the given minimum and maximum value. Fekri et al. [31] utilized the first approach, i.e., each participant scales the values using the local minimum and maximum. However, this transformation leads to different scales per participant, which in turn can lead the model to inconsistencies between the input features and the target variables. Besides, a fair comparison between centralized and federated learning requires scaling using the global minimum and maximum. Fig. 9 illustrates the averaged validation and testing NRMSE with local [31] and global scaling. We observe that the averaged NRMSE on the validation set with global scaling leads to lower error and standard deviation compared to local scaling. On the testing set, the influence of global scaling is smaller, but in all cases, it outperforms the corresponding local scaling method. Thus, we continue the experimental study using flooring and capping as mentioned previously and scaling the local features using the global minimum and maximum. We emphasize that generating those global values requires a communication between participating entities and a privacy-preserving computation. However, such computation is lightweight since participants only share two local values and owing to higher forecasting quality, we select the global scaling approach.

4.5 Learning Setting Comparison

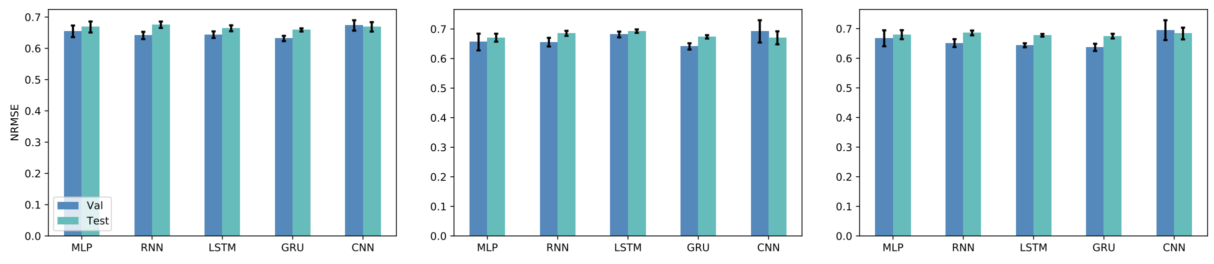
After selecting the applied transformations during preprocessing, we continue by comparing the considered learning settings. Table 4 reports the averaged MAE and RMSE using all of the five predicted outputs as well as the averaged NRMSE using the averaged results of uplink and downlink predictions. The highest quality model per learning setting is denoted with bold and the second best is highlighted with an underline. Note that we use the same learning parameters for each setting to keep consistency and provide a fair comparison. In addition, we keep FedAvg [5] as the federated aggregation algorithm.

In general, we observe that all models provide relatively similar forecasting error. In the *individual setting*, the top two models are LSTM and GRU since they present the lowest error in all the metrics. In *centralized learning*, there is no obvious way to identify the highest quality model(s). LSTM and GRU perform the best regarding MAE, GRU and CNN regarding RMSE and MLP and CNN regarding NRMSE. Nonetheless, all models result in similar prediction errors. Zooming into model stability, which provides us with a model robustness quality indicator, LSTM and GRU outperform other models. On the other hand, CNN, which is the best model regarding RMSE and the second best regarding NRMSE, has the highest standard deviation. This indicates that although CNN can provide high quality predictions, its generalization ability is susceptible to randomness. Finally, LSTM and GRU outperform the rest of the models in the *federated setting*, resulting also in lower standard deviation. Based on the above observations, LSTM and GRU are the highest quality models in each of the three learning settings.

Directly comparing the identified top models among learning settings regarding the prediction accuracy, we end up with the following observation: the individual setting results in the highest quality, followed by cen-

	Invdividual			Centralized			Federated		
Model	MAE($\times 10^6$)	RMSE($\times 10^7$)	NRMSE	MAE ($\times 10^6$)	RMSE($\times 10^7$)	NRMSE	MAE($\times 10^6$)	RMSE($\times 10^7$)	NRMSE
MLP	7.4638$\pm$8.8464($\times 10^4$)	2.8485$\pm$4.2807($\times 10^5$)	0.6755$\pm$0.01204	7.5474$\pm$1.5786($\times 10^5$)	2.8447$\pm$4.6895($\times 10^5$)	0.6705$\pm$0.013	7.6694$\pm$1.4936($\times 10^5$)	2.9655$\pm$5.7121($\times 10^5$)	0.6797$\pm$0.0158
RNN	7.3339$\pm$3.886($\times 10^4$)	2.7917$\pm$8.6265($\times 10^4$)	0.6631$\pm$0.0171	7.4373$\pm$5.6791($\times 10^4$)	2.8498$\pm$1.7813($\times 10^5$)	0.6852$\pm$0.0085	7.6031$\pm$8.7780($\times 10^4$)	2.9042$\pm$2.8667($\times 10^5$)	0.6856$\pm$0.0081
LSTM	7.2944$\pm$3.6675($\times 10^4$)	2.7699$\pm$1.0200($\times 10^5$)	**0.6417**$\pm$0.0095	7.4018$\pm$2.8927($\times 10^4$)	2.8392$\pm$9.1586($\times 10^4$)	0.6927$\pm$0.0058	7.3632$\pm$4.2940($\times 10^4$)	2.8486$\pm$1.8695($\times 10^5$)	0.6776$\pm$0.0049
GRU	**7.231**$\pm$3.4017($\times 10^4$)	**2.7652**$\pm$6.6683($\times 10^4$)	0.6589$\pm$0.0051	**7.3187**$\pm$4.1427($\times 10^4$)	**2.8103**$\pm$1.2527($\times 10^5$)	0.6734$\pm$0.0057	**7.3316**$\pm$4.7141($\times 10^4$)	**2.8275**$\pm$1.6222($\times 10^5$)	**0.6747**$\pm$0.0079
CNN	7.5244$\pm$9.6178($\times 10^4$)	2.8484$\pm$4.3976($\times 10^5$)	0.6687$\pm$0.0151	7.4774$\pm$2.0889($\times 10^5$)	2.8283$\pm$6.72082($\times 10^5$)	**0.6701**$\pm$0.0219	7.5549$\pm$9.6281($\times 10^4$)	2.9702$\pm$4.7189($\times 10^5$)	0.6836$\pm$0.0198

Table 4 – Averaged evaluation results on the test set per learning setting.



(a) Individual Learning **(b)** Centralized learning **(c)** Federated Learning

Fig. 10 – Averaged validation and test NRMSE per model and learning setting.

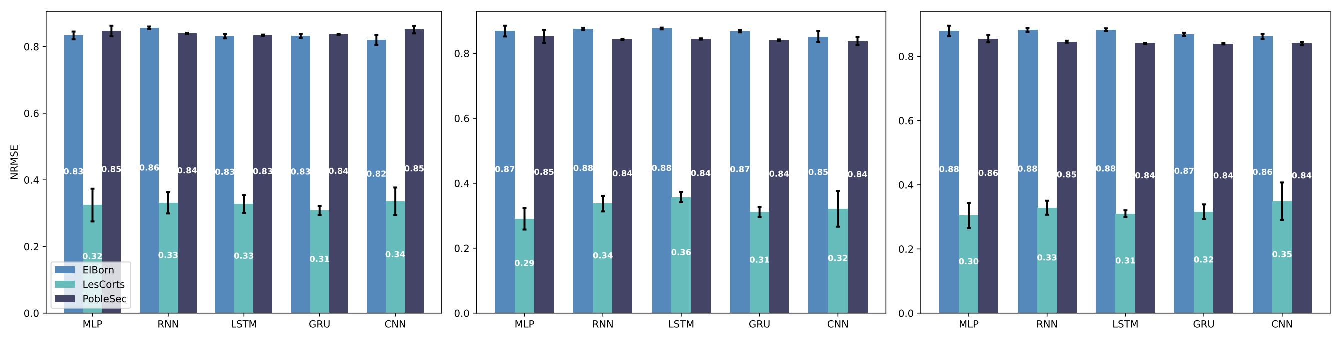
tralized and federated learning, which is consistent with [30]. Fig. 10 shows the averaged validation and testing NRMSE. In all settings, MLP and CNN have higher instability in both validation and test sets. On the other hand, LSTM and GRU provide high robustness on the validation sets, which is reflected in higher quality predictions on the test sets. Besides averaged results, we are also interested in the predictive accuracy on each base station, individually. Fig. 11 illustrates the obtained test NRMSE per site and learning setting. In general, the forecasting error is almost equivalent per model and learning setting. The predictive accuracy is higher on LesCorts, which is attributed to the fact that this site's observations does not contain extreme outliers, at least for the downlink measurements (Table 1). In addition, the employed flooring and capping technique effectively handles some outliers for the rest measurements such as the uplink values, leading to lower prediction error. On the other hand, the forecasting error on ElBorn and LesCorts is about 75% higher. This behavior is not only attributed to extreme outliers, but also to higher values for the uplink and downlink measurements.

Specifically for the federated setting, we observe that the LSTM model does not only result in high predictive accuracy, but also in high stability either individually or collectively. Fig. 12 shows the forecast of the federated LSTM model per base station regarding the downlink and uplink measurements against the ground truth values. From the visualization of forecasts, it is evident that the predictions on LesCorts are on par with the ground truth. On ElBorn and LesCorts, the model fails to capture spikes and extremely high values. Note, however, that in the latter base stations, the respective measurements are much higher than those on LesCorts, i.e., about 90% and over 97% higher for the downlink and uplink measurement, respectively. Nevertheless, the patterns of measurements are well represented and the predictive accuracy of federated learning is on par with individual and centralized settings. Fig. 13 verifies the convergence of federated learning with respect to the centralized setting regarding the MAE on the training and validation set (using the scaled data and considering the averaged MAE).

Considering the results obtained from the setting comparison, we argue that federated learning can provide high quality time-series forecasting accuracy with *generalization* ability (that cannot be ensured under individual learning), while respecting the participants *privacy* (unlike centralized learning).

4.6 Local Fine-Tuning

Although we identified that individual learning leads to lower forecasting error per base station, it concerns models operating directly on local data and hence, the gen-



(a) Individual Learning **(b)** Centralized learning **(c)** Federated Learning

Fig. 11 – Averaged test NRMSE per model, learning setting and base station.

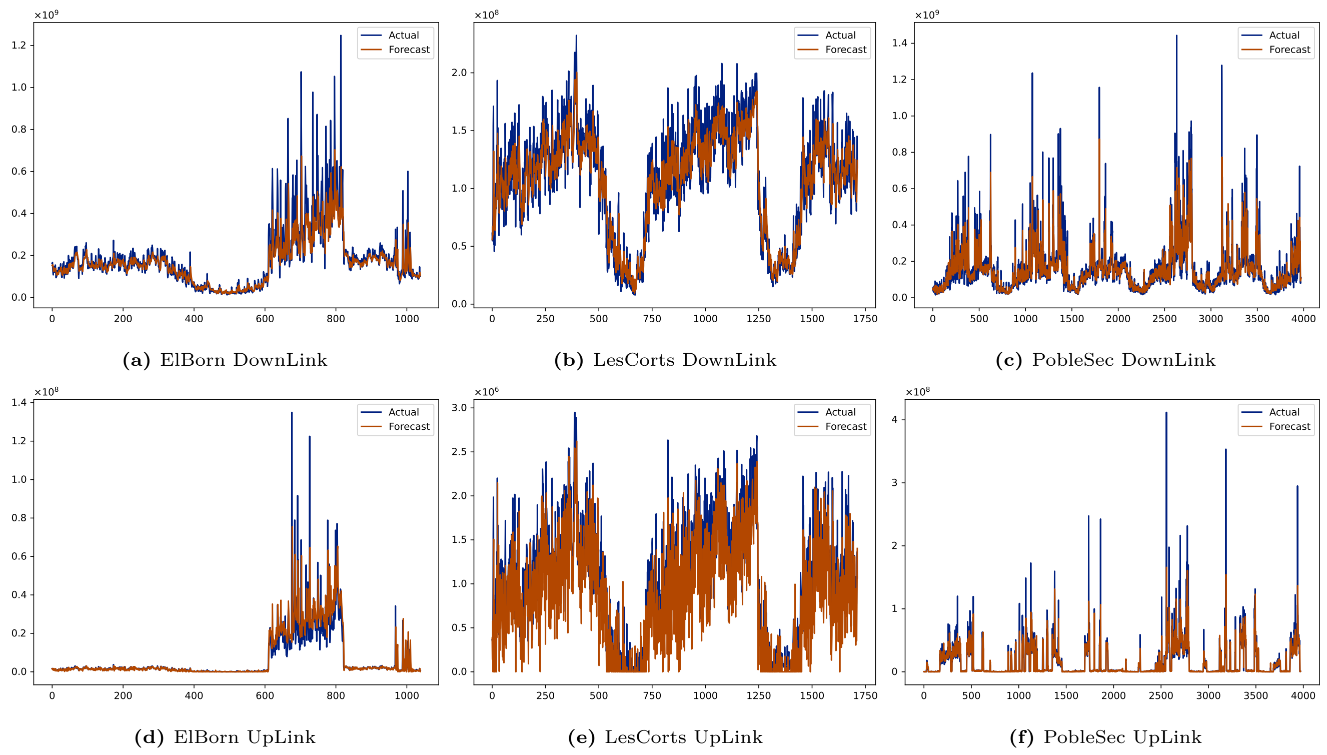


Fig. 12 – Predictions of the federated LSTM model against ground truth values per base station.

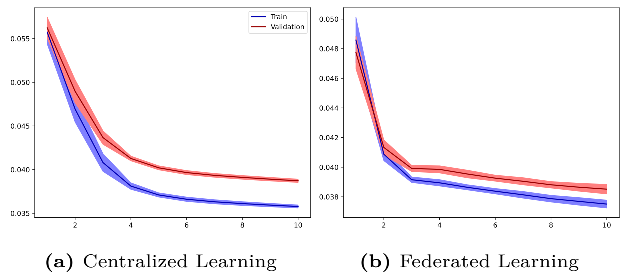


Fig. 13 – Convergence of centralized and federated settings regarding MAE.

eralization ability is limited. In other words, training a model per base station captures the spatio-temporal dynamics of that site's observations and cannot provide high quality predictions for other base stations. In addition, centralized learning requires the transmission of obtained measurements, thus, limiting the utilization of private data.

In this subsection, we perform a local fine-tuning similar to [27, 30] to observe whether additional local epochs achieve personalization. Each model is trained from scratch 10 times using different initialization seeds. Table 5 reports the obtained NRMSE per base station as well as the averaged NRMSE per learning setting on the test set. The highest quality model per learning setting considering the averaged NRMSE is denoted with bold. For ElBorn, LesCorts and PobleSec, individually, the top models are colored with orange, olive and red, respectively. Note that in individual learning there is no local fine-tuning step since the model directly learns on base station local data.

From the results, it is evident that in all cases, the local fine-tuned models are able to capture local characteristics further and result in lower prediction error. Considering the averaged NRMSE, we observe that both centralized and federated learning overcome the individual setting. Intuitively, this means that the globally-trained model captures similar dynamics from exogenous observations to that of the target base station and by per-

	Individual				Centralized				Federated			
Model	ElBorn	LesCorts	Poblesec	Avg	ElBorn	LesCorts	Poblesec	Avg	ElBorn	LesCorts	Poblesec	Avg
MLP	0.8361±0.0119	0.3411±0.0592	0.8428±0.0153	0.6733±0.0222	0.8735±0.0198	0.2791±0.0102	0.8499±0.0241	0.6675±0.0126	0.8805±0.0141	0.2909±0.0281	0.8505±0.0064	0.6832±0.0118
MLP-FN					0.8370±0.0205	0.2671±0.0119	0.8873±0.0341	0.6638±0.0171	0.8257±0.0125	0.2915±0.0069	0.8773±0.0294	0.6698±0.0132
RNN	0.8537±0.0041	0.3423±0.0299	0.8382±0.0019	0.6781±0.011	0.8729±0.0044	0.3445±0.0246	0.8411±0.0021	0.6862±0.0095	0.8785±0.0091	0.3245±0.0148	0.8455±0.0034	0.6828±0.0071
RNN-LF					0.8622±0.0049	0.3268±0.0129	0.8431±0.0028	0.6774±0.0051	0.8586±0.0041	0.3282±0.0071	0.8411±0.0028	0.6759±0.0019
LSTM	0.8314±0.0064	0.3292±0.0332	**0.8339**±0.0012	0.6648±0.0119	0.8763±0.0044	0.3501±0.0119	0.8440±0.0017	0.6901±0.0051	0.8826±0.0042	0.3068±0.0084	0.8389±0.0016	0.6761±0.0038
LSTM-LF					0.8498±0.0246	0.3033±0.0134	0.8426±0.0015	0.6652±0.0058	0.8603±0.0413	0.2929±0.0166	0.8381±0.0016	0.6638±0.0171
GRU	0.8336±0.0027	**0.3096**±0.0175	0.8357±0.0015	**0.6596**±0.0061	0.8688±0.0024	0.3188±0.0161	0.8401±0.0016	0.6759±0.0053	0.8691±0.004	0.3196±0.0208	0.8394±0.0019	0.6761±0.0073
GRU-LF					0.8305±0.0108	0.3030±0.0144	0.8396±0.0014	0.6577±0.0071	0.8380±0.0137	0.3034±0.011	**0.8379**±0.0016	0.6594±0.0047
CNN	**0.8201**±0.0054	0.3289±0.0377	0.8448±0.0057	0.6646±0.0121	0.8556±0.0183	0.3095±0.0195	**0.8383**±0.0065	0.6678±0.0109	0.8591±0.0056	0.4195±0.0933	0.8411±0.0055	0.7065±0.0307
CNN-LF					**0.8223**±0.0081	**0.2749**±0.0151	0.8395±0.0052	**0.6456**±0.0058	**0.8193**±0.0067	**0.2827**±0.0216	0.8483±0.0165	**0.6501**±0.0086

Table 5 – Averaged test NRMSE per model, learning and base station, with and without local fine-tuning. Fine-tuned models are denoted with the "-FN" suffix

forming local fine-tuning, we end up up with higher forecasting accuracy. We also emphasize that individual learning cannot provide generalization and hence, models trained under the centralized and federated settings are superior to the corresponding locally trained models. In fact, the fine-tuned CNN and GRU from both centralized and federated learning overcome the best model (GRU) on the individual setting.

Taking a closer look at the resulting error per base station, we observe a similar behavior: in most cases, the fine-tuned models overcome the corresponding global models. Some exceptions occur on PobleSec, e.g., the CNN model under both centralized and federated learning. In ElBorn, the CNN model has the best performance, while federated fine-tuned CNN slightly overcome the rest of the settings. In LesCorts, the fine-tuned centralized CNN slightly overcomes the corresponding federated model. Finally, in PobleSec, the behavior is different per learning setting since the best model architecture on one setting does not reflect the best model in the rest of the settings. This behavior is, again, attributed to arbitrary spikes and the presence of extreme outliers.

Overall, the local fine-tuning step results in higher quality predictions as well as enables generalization for the centralized and federated settings. For instance, consider a generated model under either centralized or federated learning. The global model provides generalization among base stations since it captures global dynamics, while high quality forecasting can also be provided to an external, non-participating entity. In addition, an external base station can receive the trained model and perform local fine-tuning, which cannot be easily provided through the individual setting due to privacy concerns or when the distribution between two parties are highly skewed. Besides, federated learning enables a dynamic environment, where additional participants can emerge during training. This is a unique property of federated learning and cannot be modeled on the rest of the settings. Finally, federated training can occur on demand, e.g., once per day or week, to capture the dynamics of new observations in a timely manner and potentially lead to better predictions.

4.7 Carbon Footprint Comparison

In the previous sections, we identified the potential of centralized and federated learning for generalization. However, most trained models result in almost equivalent forecasting accuracy. For instance, we observed that the global federated GRU slightly overcomes the corresponding LSTM model regarding test forecasting error, but the latter provides more stability with respect to the standard deviation across runs on the validation set.

In this subsection, we introduce another dimension to identify the top performing model and evaluate the considered models regarding energy consumption and carbon footprint (CO_{2eq}) using Carbontracker [39]. In addition, we measure the total uplink and downlink consumption for the model weights transmission during federated training.

Energy consumption and CO_{2eq} emission is not directly comparable between centralized and federated learning. In a centralized scenario, a third-party utilizes all of the available data and trains a model for a specified number of epochs. On the other hand, federated learning involves the communication between the central server and the participating entities per federated round, while participants perform a number of pre-specified number of epochs. In Section 4.3, we stated that we selected a maximum 30 rounds with 3 local epochs during federated training and 270 epochs for the centralized setting to provide fair comparisons. Although this setting can provide generalized results regarding forecasting accuracy, measuring a model's efficiency is dependent on the number of observations, iterations (epochs) and the underlying hardware. Several works like [28, 31] directly compare federated rounds to centralized epochs. However, this simplistic approach does not reflect the actual number of operations involving participants. In our case, a single federated round corresponds to 3 participants, who perform 3 local epochs. In other words, considering the total number of iterations, participants perform 9 iterations on their data per federated round. Another major difference is that centralized training involves a single entity, while real-world federated training takes place concurrently on the selected participants. Since comparing centralized and federated learning regarding energy consumption is hard to achieve, we consider the following cases:

- *Centralized-1*: Centralized learning with as many iterations as the total number of iterations considered in the federated setting. That is, a total of 270 epochs since the considered federated setting involves 30 rounds, 3 local epochs and 3 participants.
- *Centralized-2*: Centralized learning with epochs corresponding to federated rounds. This setting involves measurements considering 30 training epochs.

Table 6 shows the estimated energy consumption (kWh) and carbon footprint (g) per setting. Federated learning shows better results regarding both energy consumption and CO_{2eq} emissions. That is mainly because of the smaller amount of data that a model has to iterate over on every client in contrast to a centralized counterpart that has to iterate over all data. It should be noted, that we do not consider the communication cost here, i.e., the environmental impact of data exchange between server and clients. However, we can easily assume that in a federated learning scenario this impact is much lower, since parties only exchange model parameters rather than whole sequences of data, as in a centralized scenario. Also, we should not ignore the fact that even a slight improvement in one base station for a specific amount of data, can be extremely beneficial if we examine the problem with many more base stations in a long period of time.

Setting	Measure	MLP	RNN	LSTM	GRU	CNN
Centralized-1	Energy	0.0029	0.0032	0.0033	0.0033	0.0051
	CO_{2eq}	0.8414	0.9415	0.9788	0.9803	1.4975
Centralized-2	Energy	0.0003	0.0004	0.0004	0.0004	0.0006
	CO_{2eq}	0.0914	0.1063	0.1064	0.1127	0.1685
Federated	Energy	0.0011	0.0012	0.00135	0.0012	0.0019
	CO_{2eq}	0.3216	0.3429	0.3975	0.3645	0.5640

Table 6 – Estimated energy (kWh) and carbon footprint (g) per setting.

Taking these parameters into consideration, it can be deduced that federated learning is not only a privacy-preserving solution for sensitive data, but it can also help in minimizing the impact of running algorithms on the environment.

Besides energy consumption, we are interested in the communication cost introduced on the server and the participants. To measure it, we select the federated round that achieved the lowest error regarding the validation set and measure the cost using the model size that needs to be transmitted from the server to participants and backwards. Note that each participant (client) receives and transmits the model weights per federated round (i.e., client uplink equals to client downlink). The server transmits the model weights to the selected participants, which in our case, means that the model weights are being transmitted and received 3 times per federated round on the server side. Table 7 presents the minimum required transmissions of model weights to generate a global model. For each model, we report the corresponding size in KiloBytes (KB), the round that achieved the lowest validation error as well as the total uplink and downlink measurement at the client and server side in MegaBytes (MB). The most efficient model is RNN, but we are mainly interested in NRMSE score. Among LSTM and GRU, which are identified as the highest quality models, GRU results in more efficient computations. However, LSTM transmission size is relatively close to GRU and provides higher robustness. Considering that LSTM: i) performs the best in the validation set, ii) in the second best in the testing set, iii) it is robust against noise and iv) its efficiency is close to that of GRU, we select LSTM as our final solution. A critical future direction is to define a tunable measure that considers a model's error/accuracy, the convergence speed and its size to limit the quality-efficiency trade-off and identify the highest-performing model architecture.

Model	Size (KB)	Top Round	Client (MB)	Server (MB)
MLP	563.2	15	8.448	25.344
RNN	153.6	3	0.4608	1.3824
LSTM	586.8	4	2.3472	7.0416
GRU	442.4	4	1.7696	5.3088
CNN	262.9	20	5.258	15.774

Table 7 – Communication cost regarding the model size the federated round that achieved the lowest validation error.

4.8 Aggregation Algorithms Comparison

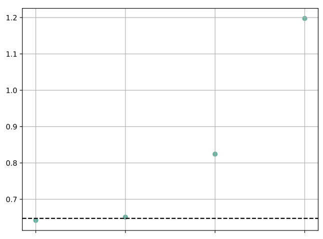
The previous experimental comparisons concern federated learning under the FedAvg aggregation algorithm [5]. To the best of our knowledge, most works, if not all, in the generalized context of federated time-series forecasting, employ FedAvg without considering other aggregators [27, 28, 29, 30, 31]. Due to the heterogeneous nature of our task under four non-iid categories we are interested in evaluating the effectiveness of different aggregation algorithms. Besides the aggregation algorithms introduced in Section 3.2.3, we also consider a simple averaging, denoted as *SimpleAvg* and aggregation based on the median of received weights, denoted as *MedianAvg*. The rest of the considered aggregators try to address the non-iid issue and introduce tunable parameters. For instance, FedProx [11] controls the weight divergence using a regularization parameter μ.

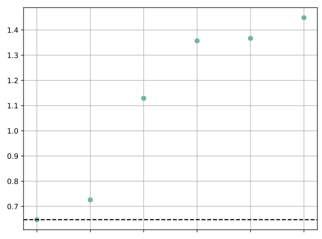
We select the federated LSTM model since it achieves high forecasting and efficiency performance. We train the model from scratch 20 times using different initialization seeds to obtain comprehensive results per aggregator. First, we tune the hyper-parameters introduced by the aggregation algorithms using a small grid search. Table 8 reports the grids per aggregator. For FedAdagrad, we keep $\beta_1 = 0.$; For FedYogi and FedAdam, we keep $\beta_1 = 0.9$ and $\beta_2 = 0.99$, according to [14].

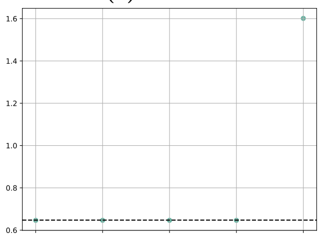
Aggregator	Grid Parameters
SimpleAvg	-
MedianAvg	-
FedAvg [5]	-
FedProx [11]	$\mu \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$
FedAvgM [12]	$\beta \in \{0., 0.7, 0.9, 0.97, 0.99, 0.997\}$
FedNova [13]	$\rho \in \{0, 10^{-3}, 10^{-2}, 10^{-1}, 0.99\}$
FedAdagrad [14] FedYogi [14] FedAdam [14]	$\eta \in \{10^{-2}, 10^{-1}, 10^{0}\}$ $\tau \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$

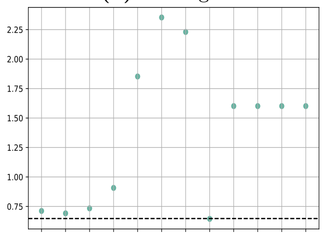
Table 8 – Hyper-parameters per aggregator in the grid-search.

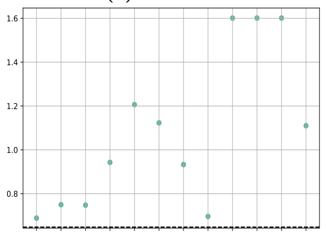
The resulting averaged NRMSE per grid in the test is illustrated on Fig. 14. The dash line corresponds to the FedAvg baseline and the points show the predictive error per grid. For *FedProx*, we observe that the lower values of μ, i.e., $\mu \in \{10^{-3}, 10^{-2}\}$, the better the predictive accuracy. Note that when $\mu = 0$, FedProx is equivalent to FedAvg. Since μ is a regularization parameter, higher values lead to slower convergence and hence, the selected


(a) FedProx


(b) FedAvgM


(c) FedNova


(d) FedAdagrad


(e) FedYogi

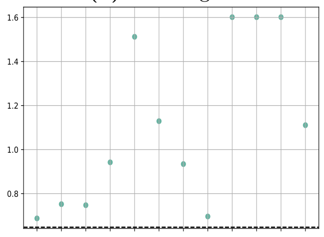
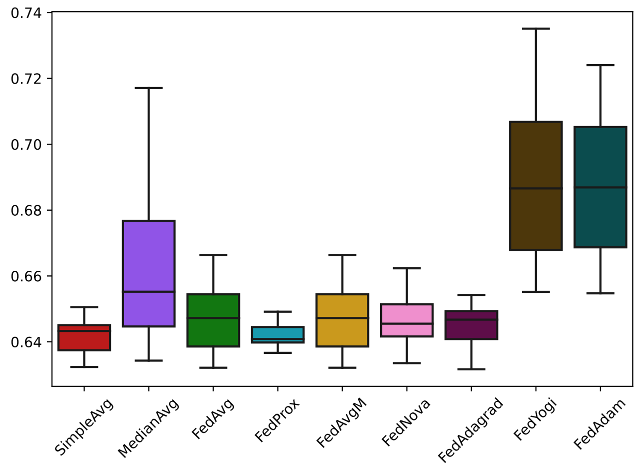

(f) FedAdam

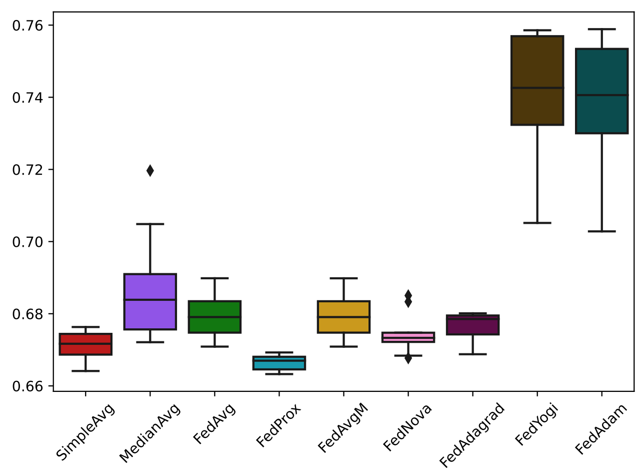
Fig. 14 – Averaged test NRMSE per grid and aggregator using the federated LSTM model. The dash line corresponds to the FedAvg baseline.

30 federated rounds do not lead to low prediction error. When μ is as low as 10^{-3}, FedProx slightly overcomes the FedAvg baseline. For the *FedAvgM* aggregator, we observe that the higher the value of β, the higher the error. Note that when $\beta = 0$, FedAvgM is equivalent to FedAvg. From the experimental results, FedAvgM does not lead to lower prediction error and cannot overcome FedAvg. Regarding *FedNova*, we observe that it follows the predictive accuracy of FedAvg when $\rho \leq 10^{-1}$ and slightly overcomes FedAvg. *FedAdagrad*, *FedYogi* and *FedAdam* require a pool of 4 hyper-parameters. In this work we optimize two of them, and the the rest is kept constant following their author observations [14]. We observe that the predictive error of these aggregators is hugely affected by the underlying parameters and in many cases, they lead to high error values. FedAdagrad overcomes FedAvg only on a single case, while FedYogi and FedAdam do not lead to higher quality predictions than FedAvg.

Overall, it is evident that FedAvg is on par with FedProx and FedNova and state-of-the-art aggregation algorithms do not outperform the simple approach of FedAvg. In fact, FedOpt algorithms [14] lead to unstable forecasting performance and the final global model is highly sensitive to the introduced hyper-parameters. The momentum parameter of FedAvgM does not improve the corresponding error, while FedProx and FedNova slightly outperform FedAvg in some cases. Fig. 15 shows the summarized NRMSE results per aggregation algorithm on the validation and testing sets. Note that we select the best hyper-parameters per aggregator based on the results on the validation set. Here, we consider two additional aggregators, i.e., SimpleAvg and MedianAvg. The NRMSE spread per aggregator clarifies the previous observations that FedProx and FedNova are on par with FedAvg. More precisely, *FedProx* provides the highest robustness against random initialization and the lower averaged NRMSE on both the validation and testing sets. Next, despite its simplicity, SimpleAvg seems to outperform the rest of the aggregators both in NRMSE and stability. We attribute this behavior to the unique patterns per base station. FedNova slightly overcomes FedAvg and shows relatively high stability. The rest of the aggregators show high



(a) Results on the validation set.



(b) Results on the testing set.

Fig. 15 – Validation and test averaged NRMSE considering the best hyper-parameters per aggregator using the federated LSTM model. The best hyper-parameters are selected based on the results on the validation set.

spread, i.e., sensitivity to randomness, while FedAdagrad requires careful tuning of its parameters.

To the best of our knowledge there is only a single work [6] that systematically evaluates the behavior of different aggregators. However, Li et al. simulated non-iid data and do not use real-world datasets. In addition, the introduced scenarios in [6] only considered two cases of fixing non-iid data, namely, feature with label distribution skew and feature with quantity skew. In our case, there are four categories of non-iid data: the feature, target, quantity and temporal skew, a mixed case that is unique and directly applies to the real world. Nevertheless, our results are consistent to [6] and specifically:

- There is no aggregator that systematically leads to higher quality models compared to FedAvg.
- FedProx is on par with FedAvg.
- Federated learning is challenging with the presence of a mixed type of non-iid data.

A key difference in our results compared to [6] is that FedNova presents high stability. We attribute this behavior to the mixed type of non-iid data of the given dataset as well as to our use case. Nonetheless, there is still room for improvements and additional experimental studies, especially in the federated time-series forecasting research area, which have been given limited attention, should be carefully investigated.

5. CONCLUSION

Generating high-quality traffic forecasting models with generalization ability is a challenging task considering the different date patterns in different base stations. In this work, we employed a federated learning approach to tackle several identified challenges owing to the non-iid data nature. Unlike centralized approaches, the proposed methods minimize privacy and business confidentiality concerns and federated learning lays the groundwork for large scale participation that can lead to building intelligent predictors. We conducted extensive experiments considering five deep learning architectures and compared three different learning settings. The results show that federated learning has the potential of generalization, does not require the transmission of private data to a third-party and provides a dynamic execution environment. Finally, local fine-tuning results in higher predictive accuracy and hence, federated learning could lead to more intelligent models than individual and centralized approaches.

A critical future direction is to focus on the *preprocessing stage*, which according to our results, heavily affects the learning performance. Although preprocessing is an integral step machine learning, there is limited conducted research on the influence of preprocessing applied to a federated setting. In addition, we showed that all learning architectures lead to equivalent forecasting accuracy. Based on our experimental study, LSTM and GRU lead to both lower errors and high robustness. Evaluating additional *novel architectures* could result in higher-quality models. We also experimentally compared several different aggregation algorithms, some of which are specifically designed to tackle the non-iid data issues. However, we observed that in most cases, the aggregators are on par with the FedAvg baseline. Investigating *additional aggregation algorithms* that can handle temporal dynamics could lead to higher predictive accuracy. Overall, we believe that our work sheds light on several unique challenges of time-series forecasting owing to the mixed skew type and can serve as a benchmark for various tasks including *larger-scale federated learning.*

ACKNOWLEDGMENTS

We thank the SUPERCOM Research Unit of CTTC for organizing the ML5G-PS-001 - "Federated Traffic Prediction for 5G and Beyond" challenge, part of the ITU AI/ML Challenge (2022) and specifically Dr. Francesc Wilhelmi for the constant support during the challenge.

REFERENCES

[1] Chafika Benzaid and Tarik Taleb. "AI-Driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions". In: *IEEE Network* 34.2 (2020), pp. 186–194. DOI: 10.1109/MNET.001.1900252.

[2] Hoang Duy Trinh, Ángel Fernández Gambín, Lorenza Giupponi, Michele Rossi, and Paolo Dini. "Mobile Traffic Classification Through Physical Control Channel Fingerprinting: A Deep Learning Approach". In: *IEEE Transactions on Network and Service Management* 18.2 (2021), pp. 1946–1961. DOI: 10.1109/TNSM.2020.3028197.

[3] Hoang Duy Trinh, Lorenza Giupponi, and Paolo Dini. "Mobile Traffic Prediction from Raw Data Using LSTM Networks". In: *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2018, pp. 1827–1832. DOI: 10.1109/PIMRC.2018.8581000.

[4] Abigail Goldsteen, Gilad Ezov, Ron Shmelkin, Micha Moffie, and Ariel Farkash. "Data minimization for GDPR compliance in machine learning models". en. In: *AI and Ethics* 2.3 (Aug. 2022), pp. 477–491. ISSN: 2730-5953, 2730-5961. DOI: 10.1007/s43681-021-00095-8. URL: https://link.springer.com/10.1007/s43681-021-00095-8 (visited on 10/06/2022).

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-Efficient Learning of Deep Net-

works from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, Apr. 2017, pp. 1273–1282. URL: `https://proceedings.mlr.press/v54/mcmahan17a.html`.

[6] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. "Federated Learning on Non-IID Data Silos: An Experimental Study". In: *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 2022, pp. 965–978. DOI: `10.1109/ICDE53745.2022.00077`.

[7] Yunyun Cai, Wei Xi, Yuhao Shen, Youcheng Peng, Shixuan Song, and Jizhong Zhao. "High-efficient hierarchical federated learning on non-IID data with progressive collaboration". In: *Future Generation Computer Systems* 137 (2022), pp. 111–128. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2022.07.010`. URL: `https://www.sciencedirect.com/science/article/pii/S0167739X22002394`.

[8] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. "Federated learning on non-IID data: A survey". In: *Neurocomputing* 465 (2021), pp. 371–390. ISSN: 0925-2312. DOI: `https://doi.org/10.1016/j.neucom.2021.07.098`. URL: `https://www.sciencedirect.com/science/article/pii/S0925231221013254`.

[9] Xiaodong Ma, Jia Zhu, Zhihao Lin, Shanxuan Chen, and Yangjie Qin. "A state-of-the-art survey on solving non-IID data in Federated Learning". In: *Future Generation Computer Systems* 135 (2022), pp. 244–258. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2022.05.003`. URL: `https://www.sciencedirect.com/science/article/pii/S0167739X22001686`.

[10] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. "LEAF: A Benchmark for Federated Settings". In: *CoRR* abs/1812.01097 (2018). arXiv: `1812.01097`. URL: `http://arxiv.org/abs/1812.01097`.

[11] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. "Federated Optimization in Heterogeneous Networks". In: *Proceedings of Machine Learning and Systems*. Ed. by I. Dhillon, D. Papailiopoulos, and V. Sze. Vol. 2. 2020, pp. 429–450. URL: `https://proceedings.mlsys.org/paper/2020/file/38af86134b65d0f10fe33d30dd76442e-Paper.pdf`.

[12] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. "Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification". In: *CoRR* abs/1909.06335 (2019). arXiv: `1909.06335`. URL: `http://arxiv.org/abs/1909.06335`.

[13] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. "Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 7611–7623. URL: `https://proceedings.neurips.cc/paper/2020/file/564127c03caab942e503ee6f810f54fd-Paper.pdf`.

[14] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. "Adaptive Federated Optimization". In: *International Conference on Learning Representations (ICLR)*. 2021. URL: `https://openreview.net/forum?id=LkFG3lB13U5`.

[15] Stephan Rasp, Peter D. Dueben, Sebastian Scher, Jonathan A. Weyn, Soukayna Mouatadid, and Nils Thuerey. "WeatherBench: A Benchmark Data Set for Data-Driven Weather Forecasting". In: *Journal of Advances in Modeling Earth Systems* 12.11 (2020). e2020MS002203 10.1029/2020MS002203, e2020MS002203. DOI: `https://doi.org/10.1029/2020MS002203`. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2020MS002203`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020MS002203`.

[16] Musaed Alhussein, Khursheed Aurangzeb, and Syed Irtaza Haider. "Hybrid CNN-LSTM Model for Short-Term Individual Household Load Forecasting". In: *IEEE Access* 8 (2020), pp. 180544–180557. DOI: `10.1109/ACCESS.2020.3028281`.

[17] A.L.D. Loureiro, V.L. Miguéis, and Lucas F.M. da Silva. "Exploring the use of deep neural networks for sales forecasting in fashion retail". In: *Decision Support Systems* 114 (2018), pp. 81–93. ISSN: 0167-9236. DOI: `https://doi.org/10.1016/j.dss.2018.08.010`. URL: `https://www.sciencedirect.com/science/article/pii/S0167923618301398`.

[18] Pedro Lara-Benítez, Manuel Carranza-García, and José C. Riquelme. "An Experimental Review on Deep Learning Architectures for Time Series Forecasting". In: *International Journal of Neural Systems* 31.03 (2021). PMID: 33588711, p. 2130001. DOI: `10.1142/S0129065721300011`. URL: `https://doi.org/10.1142/S0129065721300011`.

[19] Mikio Iwamura. “NGMN View on 5G Architecture”. In: *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. 2015, pp. 1–5. DOI: 10.1109/VTCSpring.2015.7145953.

[20] Jie Feng, Xinlei Chen, Rundong Gao, Ming Zeng, and Yong Li. “DeepTP: An End-to-End Neural Network for Mobile Cellular Traffic Prediction”. In: *IEEE Network* 32.6 (2018), pp. 108–115. DOI: 10.1109/MNET.2018.1800127.

[21] Longbiao Chen, Dingqi Yang, Daqing Zhang, Cheng Wang, Jonathan Li, and Thi-Mai-Trang Nguyen. “Deep mobile traffic forecast and complementary base station clustering for C-RAN optimization”. In: *Journal of Network and Computer Applications* 121 (2018), pp. 59–69. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2018.07.015. URL: https://www.sciencedirect.com/science/article/pii/S1084804518302455.

[22] Kristian Sebastian, Hui Gao, and Xudong Xing. “Utilizing an Ensemble STL Decomposition and GRU Model for Base Station Traffic Forecasting”. In: *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. 2020, pp. 314–319. DOI: 10.23919/SICE48898.2020.9240357.

[23] Zihang Gao. “5G Traffic Prediction Based on Deep Learning”. en. In: *Computational Intelligence and Neuroscience* 2022 (June 2022). Ed. by Yang Gu, pp. 1–5. ISSN: 1687-5273, 1687-5265. DOI: 10.1155/2022/3174530. URL: https://www.hindawi.com/journals/cin/2022/3174530/.

[24] Peter Kairouz et al. “Advances and Open Problems in Federated Learning”. In: *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210. ISSN: 1935-8237. DOI: 10.1561/2200000083. URL: http://dx.doi.org/10.1561/2200000083.

[25] Usman Ahmed, Gautam Srivastava, and Jerry Chun-Wei Lin. “Reliable customer analysis using federated learning and exploring deep-attention edge intelligence”. In: *Future Generation Computer Systems* 127 (2022), pp. 70–79. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2021.08.028. URL: https://www.sciencedirect.com/science/article/pii/S0167739X21003368.

[26] Vasileios Perifanis and Pavlos S. Efraimidis. “Federated Neural Collaborative Filtering”. In: *Know.-Based Syst.* 242.C (Apr. 2022). ISSN: 0950-7051. DOI: 10.1016/j.knosys.2022.108441. URL: https://doi.org/10.1016/j.knosys.2022.108441.

[27] Afaf Taïk and Soumaya Cherkaoui. “Electrical Load Forecasting Using Edge Computing and Federated Learning”. In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9148937.

[28] Yi Liu, James J. Q. Yu, Jiawen Kang, Dusit Niyato, and Shuyu Zhang. “Privacy-Preserving Traffic Flow Prediction: A Federated Learning Approach”. In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 7751–7763. DOI: 10.1109/JIOT.2020.2991401.

[29] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M. Shamim Hossain. “Deep Anomaly Detection for Time-Series Data in Industrial IoT: A Communication-Efficient On-Device Federated Learning Approach”. In: *IEEE Internet of Things Journal* 8.8 (2021), pp. 6348–6358. DOI: 10.1109/JIOT.2020.3011726.

[30] Christopher Briggs, Zhong Fan, and Peter Andras. “Federated Learning for Short-term Residential Load Forecasting”. In: *IEEE Open Access Journal of Power and Energy* (2022), pp. 1–1. DOI: 10.1109/OAJPE.2022.3206220.

[31] Mohammad Navid Fekri, Katarina Grolinger, and Syed Mir. “Distributed load forecasting using smart meter data: Federated learning with Recurrent Neural Networks”. In: *International Journal of Electrical Power & Energy Systems* 137 (2022), p. 107669. ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2021.107669. URL: https://www.sciencedirect.com/science/article/pii/S0142061521008991.

[32] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. “A survey on security and privacy of federated learning”. In: *Future Generation Computer Systems* 115 (2021), pp. 619–640. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.10.007. URL: https://www.sciencedirect.com/science/article/pii/S0167739X20329848.

[33] Anichur Rahman, Kamrul Hasan, Dipanjali Kundu, Md. Jahidul Islam, Tanoy Debnath, Shahab S. Band, and Neeraj Kumar. “On the ICN-IoT with federated learning integration of communication: Concepts, security-privacy issues, applications, and future perspectives”. In: *Future Generation Computer Systems* 138 (2023), pp. 61–88. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2022.08.004. URL: https://www.sciencedirect.com/science/article/pii/S0167739X22002667.

[34] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. "Inverting Gradients - How easy is it to break privacy in federated learning?" In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 16937–16947. URL: `https://proceedings.neurips.cc/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf`.

[35] Milad Nasr, Reza Shokri, and Amir Houmansadr. "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning". In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 739–753. DOI: `10.1109/SP.2019.00065`.

[36] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. "Flower: A Friendly Federated Learning Research Framework". In: *CoRR* abs/2007.14390 (2020). URL: `https://arxiv.org/abs/2007.14390`.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: `https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf`.

[38] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. "Array programming with NumPy". en. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. ISSN: 0028-0836, 1476-4687. DOI: `10.1038/s41586-020-2649-2`. URL: `https://www.nature.com/articles/s41586-020-2649-2` (visited on 11/18/2022).

[39] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. *Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models*. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems. arXiv:2007.03051. July 2020.

[40] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR (Poster)*. 2015. URL: `http://arxiv.org/abs/1412.6980`.