# Distributed Load Forecasting using Smart Meter Data: Federated Learning with Recurrent Neural Networks

Mohammad Navid Fekri[a], Katarina Grolinger [a,*] and Syed Mir[b]

[a]Department of Electrical and Computer Engineering, Western University, London, ON N6A 5B9, Canada
[b]London Hydro, London, ON, Canada N6A 4H6

ARTICLE INFO

*Keywords*:
Federated Learning
Load Forecasting
Consumption forecasting
LSTM
FedAVG
FedSGD

ABSTRACT

Load forecasting is essential for energy management, infrastructure planning, grid operation, and budgeting. Large scale smart meter deployments have resulted in ability to collect massive energy data and have created opportunities in sensor-based forecasting. Machine learning (ML) has demonstrated great successes in sensor-based load forecasting; however, when prediction is needed on a smart meter level, typically a single model is trained for each smart meter. With a large number of meters, this becomes computationally expensive or even infeasible. On the other hand, with conventional ML, training a single model for several smart meters requires participants to share their data with the central server. Consequently, this paper proposes federated learning for load forecasting with smart meter data: this strategy enables training a single model with all participating smart meters without the need to share local data. Two alternative federated learning strategies are examined: FedSGD, which performs one step of gradient descent on client before merging updates on the server, and FedAVG, which carries out several steps before the merging. Specifically, residential consumers are diverse what makes training a single model challenging as load profiles vary across consumers. The results show that the FedAVG achieves better accuracy than FedSGD while also requiring fewer communication rounds. Comparing to individual models for each meter and a single central models for all meters, FedAVG achieves comparable or better accuracy.

## 1. Introduction

Energy drives economies and societies, but it has also been the biggest contributor to global warming and accounts for about two-thirds of greenhouse gas emissions [1]. To combat the negative environment impact, countries and institutions are setting aggressive targets; for example, European Union aims for 40% reduction in emissions and 27% improvement in energy efficiency by 2030 [2]. Nevertheless, energy demand is continuously increasing making energy management a crucial factor in reducing environmental impact while ensuring that the growing demand is met [3].

Load forecasting has an essential role in energy management as it facilitates power infrastructure planning, generation scheduling, balancing demand and supply, thus alleviating the impact of renewable energy generation and assisting in energy budgeting. Moreover, financial benefits from improved load forecasting can be large: for example, reducing forecasting error from 15.7% to 12.2% saved 2.5 M$ for Xcel Energy [4].

Advanced metering infrastructure together with massive deployments of smart meters enables utility companies to measure and record energy consumption in intervals of one hour or less for individual buildings or even individual households. For example, in UK, over 15 million smart meters are operating across homes and businesses [5]. This big smart meter data have forged opportunities for new deeper insights into energy usage patterns and have made possible large-scale load forecasting on individual consumer level.

Sensor-based load forecasting approaches use historical load data collected by smart meters or similar technologies, often together with meteorological information, to train machine learning (ML). Conventionally, smart meter data are transferred to a data center or another centralized system for storage and for training ML models. There are two main categories of these centralized solutions: the first category trains a single model for each smart meter and, thus, generates individual forecasts for each meter, while the second category trains a machine learning model with the aggregated data from several smart meters and, hence, provides aggregated load forecasts [6]. Although both of these centralized solutions have shown great results, they require transferring all data to a centralized location, which results in significant network traffic [7]. Moreover, a centralized ML not only requires sharing local data with the centralized systems imposing security and privacy concerns, but also makes complying with stringent data regulations such as EU General Data Protection Regulation (GDPR) difficult [8]. As the number of smart meters grows, training an individual ML model for each smart meter becomes computationally expensive and even infeasible.

Federated Learning (FL) [9] presents a possible solution to these challenges by decoupling the ability to do train the ML model from the need to store the data on the cloud or another centralized system. In FL, a global ML model is shared across many devices: each device receives a copy of the global model and improves it by learning from local data. Then, instead of raw data, the updated parameters of the local models are sent to the server to be aggregated and incorporated into the global model. FL is a major shift from a costly central ML system to a distributed ML approach that

*Corresponding author
Email address: kgroling@uwo.ca (K. Grolinger)
ORCID(s): 0000-0001-8079-7117 (M.N. Fekri); 0000-0003-0062-8212 (K.G. )

can exploit numerous distributed computational resources. This learning technique enhances data privacy because the data stay on the local devices and reduces network traffic by only exchanging model updates as opposed to raw data [10]. However, FL assumes that a single model can capture patterns across diverse clients. In load forecasting, this approach would entail a single global model capable for generating individual load forecasts for each smart meter. As a diversity of patterns among meters is large, a single model may encounter difficulties in capturing this diversity and may lead to inferior forecasts.

Consequently, this paper proposes FL for load forecasting with smart meter data. Since in recent years Recurrent Neural Networks (RNNs) have been outperforming other load forecasting techniques [11], we employ an RNN variant, Long Short Term Memory (LSTM) Network, as a base learner. To handle diverse magnitudes of smart meter readings, the normalization is carried out individually for each smart meter data and then the sliding window technique is used to prepare data for ML. Two different FL techniques are examined: Federated Stochastic Gradient Descent (FedSGD) and Federated Averaging (FedAVG). Moreover, we examine a dynamic environment in which some devices join the federation after the training and only use the already trained model for forecasting without participating in the training. The results show that both FedSGD and FedAVG achieve better accuracy than a central model or the individual LSTM models for each meter. Between the two federated learning strategies, FedAVG not only achieves higher accuracy than FedSGD, but also needs a fewer training rounds to converge and, therefore, generates lower network traffic.

The remainder of the paper is organized as follows: Section 2 presents the related works, Section 3 introduces LSTM, the proposed approach is presented in Section 4, and Section 5 explains the experiments and corresponding results. Finally, Section 6 concludes the paper.

## 2. Related Work

This section first reviews recent FL works and then discusses ML for load forecasting including studies that employ FL for load forecasting.

### 2.1. Federated Learning

FL is a burgeoning learning paradigm that has shown promising prospects in various industrial and engineering fields such as health care [12, 13], autonomous vehicle [14], text analysis [15], and image processing [16], to name but a few [17]. Leroy *et al.* [18] proposed an approach based on federated learning for training speech-based models on mobile devices. They used an adaptive averaging strategy in place of weighted averaging to reduce the number of communication rounds required to reach the target performance. This method achieved competitive accuracy compared to the centralized approaches.

Liu *et al.* [19] introduced FedVision, a visual object detection platform enabled by Convolutional Neural Network (CNN) and FL for training a shared model through a collaboration of multiple clients. Since parameters from diverse local models can have different contributions towards the shared model performance, FedVision applies a compression technique to prune less useful weights while preserving model performance. This results in reduced neural network size and faster transmission.

To enable training personalized ML models with health information without compromising privacy, Yiqiang [20] proposed FedHealth. Knowledge from data is aggregated though federated learning and, then, the local models are personalized by transfer learning. FedHealth adopts the federated learning paradigm [21] to aggregate the local models; however, rather than just replacing the local model with the aggregated one, transfer learning is used to personalize the global model for each user.

Briggs *et al.* [22] combined FL with hierarchical clustering (FL+HC). The FL+HC training process consists of three steps. First, a typical FL model is trained with the clients local data for a few training rounds. Next, a hierarchical clustering algorithm is employed to iteratively compare and merge the clients with similar weights (grouping similar clients). Once clusters are determined, each cluster trains its specialized model independently.

Smith *et al.* introduced MOCHA [23], a federated paradigm that combines multi-task learning [24] with FL. Each node in FL may observe data with a distinct distribution, so it is intuitive to fit a separate model for each local node; however, these separate models have relationships and exhibit similarities. MOCHA applies multi-task learning techniques to fit separate weight vectors for each node.

In the presence of highly skewed non-IID (independent and identically distributed) data, the accuracy of FL can reduce significantly. To address this problem, Zhao *et al.* [25] suggested sharing a small subset of data among all nodes. Although this method has shown the accuracy increase, sharing data introduces security and privacy concerns. Agnostic Federated Learning proposed by Mohri *et al.* [26] updates the shared model using a weighted average of the clients' gradients and a new optimization method. Their study shows that the proposed approach contributes to fairness and reduces bias.

The reviewed FL works [18, 19, 20, 22, 23, 26, 25, 21] propose FL techniques and address challenges in diverse domains; however, they have not been applied for load forecasting, and their capabilities for load forecasting need to be investigated. Diversity of distributions and load patterns observed by individual smart meters imposes challenges and may degrade the accuracy of the single global model created through FL. Our study proposes federate learning for load forecasting and evaluates the performance of FedSGD and FedAVG in comparison to the central approach and individual ML models.

### 2.2. Load Forecasting

Various load forecasting techniques have been developed over the years [6], but here we focus on machine-learning

techniques as they have been dominant approaches in sensor-based forecasting [6]. Grolinger *et al.* [27] introduced an approach based on Support Vector Regression (SVR) and local learning for load forecasting with big data. Their method partitions the training set through clustering and then trains a separate SVM model for each cluster. Local learning with SVR improved the load forecasting accuracy while achieving order or magnitude shorter training time than the stand alone SVR. Zainab *et al.* [28] proposed a framework for training in parallel individual models on smart meter data to reduce training time. A parallel pre-processing stage is performed on individual smart meter data sets, and then a single model is trained for each dataset concurrently. The framework employed a variety of ML algorithms as the base learners in order to examine trade-offs between model accuracy and execution time.

In recent years, deep learning techniques, and especially RNNs, have been gaining popularity for energy load forecasting. RNNs are well suited for this task as they are capable of learning temporal patterns in energy data and, consequently, they produce accurate predictions outperforming alternative statistical and machine learning approaches [6]. Sehovac *et al.* [29] proposed Sequence to Sequence Recurrent Neural Network (S2S RNN) with attention mechanism for load forecasting. The S2S model improves time modelling by combining two RNNs, an encoder and a decoder, to map the input sequence to the output sequence. The attention mechanism helps capture longer time dependencies present in the load data by strengthening the link between the encoder and the decoder.

Tian *et al.* [30] also used S2S RNN, but they focused on scaling load forecasting for a large number of smart meter and proposed Similarity-based Chained Transfer Learning (SBCTL). In SBCTL, the model for the first meter is trained in a traditional manner while all other models employ transfer learning to take advantage of the already trained models according to similarities between energy consumption patterns in smart meters data. While transfer learning aims to improve learning in a particular domain by reusing knowledge or the model obtain from another domain, federated learning aims to train a single global ML model for a number of clients without requiring clients to share their local data.

An RNN was also used in Online Adaptive RNN proposed by Fekri *et al.* [11]. In contrast to conventional offline techniques which train the model once and then use it for inference, Online Adaptive RNN learns from new data as they arrive. This technique employs Batch-Normalized LSTM (BNLSTM) as the base learner and an online Bayesian optimizer to tune the model parameters on the fly throughout the training process. Online Adaptive RNN achieved higher accuracy than conventional LSTMs and several other adaptive learning approaches.

The reviewed works [6, 29, 30, 11] achieved good accuracy in load forecasting studies; however, they all train an individual ML model for each smart meter or a group of meters which becomes computationally very expensive when the number of smart meters grows. Also, these techniques train the ML models on a centralized system and, thus, require transferring all data from all smart meters to that server, which results in increased network traffic and latencies. These centralized solutions are also associated with security and privacy vulnerabilities due to sharing and transmitting data from smart meters to the central server.

To address these issues, Taïk *et al.* [7] examine the use of federated averaging for short-term load forecasting. The proposed method employs federated averaging architecture with the weighted averaging as the aggregation technique and LSTM as the global model. Their initial results show that FL is a promising approach for an hour ahead forecasting. Similarly, Li *et al.* [31] also took advantage of the FL with the weighted averaging; however, their study focused on the security aspect rather than the forecasting accuracy, and, as a result, the model's time complexity was high as it included encryption and decryption time.

Yuris *et al.* [32] proposed a hybrid of FL and clustering to predict electric vehicle charging station demand on the power grid. The charging stations are first clustered based on their location and then federated averaging approach is applied individually on each cluster. FL with clustering achieved the forecasting accuracy very similar to standalone FL.

Reviewed studies on FL for load forecasting [7, 32, 31] present initial attempts to introduce FL in this domain; however, there is a need for deeper understanding of FL capabilities and limitations. Yuris *et al.* [32] focus on charging stations demand while Taïk *et al.* [7] investigate FL with smart meter, but they select only the clients with similar properties. In contrast, our study investigates FL with smart meter data, but does not assume clients' similarity, and examines FL in presence of clients with different data distributions. The work of Li *et al.* [31] is primarily focused on the security aspect, while we investigate forecasting accuracy. Moreover, our work compares two FL techniques, FedSGD and FedAVG, and examine the behaviour of the FL system when some clients join the federation upon the completion of training.

## 3. Long Short-Term Memory

Recurrent Neural Networks (RNNs) have recently become vastly popular in various domains, including natural language processing and electric load forecasting, because of their ability to model temporal dependencies present in sequential data [33]. However, regular RNNs suffer from gradient vanishing and exploding problems, which impedes learning with long data sequences [11]. Long Short Term Memory (LSTM) networks overcome these issues by introducing additional gates for better control of the gradient flows with the cell. The LSTM cell is illustrated in Fig.1 while its computation at time $t$ is given as follows:

$$i_t = \sigma(W_{xi}x_t + b_{xi} + W_{hi}h_{t-1} + b_{hi}) \tag{1a}$$
$$f_t = \sigma(W_{xf}x_t + b_{xf} + W_{hf}h_{t-1} + b_{hf}) \tag{1b}$$

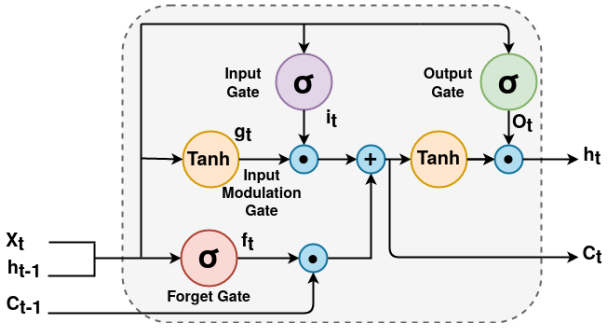


**Figure 1:** The standard LSTM cell

$$o_t = (W_{xo}x_t + b_{xo} + W_{ho}h_{t-1} + b_{ho}) \tag{1c}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xg}x_t + b_{xg} \tag{1d}$$
$$+ W_{hg}h_{t-1} + b_{hg}) \tag{1e}$$
$$h_t = o_t \odot \tanh(c_t) \tag{1f}$$

Here, $i$, $f$ and $o$ are input, forget, and output gates, $c$ is the cell state and, $h$ is the hidden state. Symbol $\odot$ represents the elementwise multiplication and $\sigma$ is the sigmoid activation function. Input-hidden and hidden-hidden weights are $W_x$'s and $W_h$'s, whereas the corresponding biases are $b_x$'s and $b_h$'s.

This memory mechanism makes LSTMs successful in load forecasting tasks [11]; therefore, our federated learning approach employs LSTM architecture to learn from sequential smart meter data.

## 4. Federated Learning for Load Forecasting

Conventional ML for load forecasting collects the readings from individual smart meters in a data center or another centralized system and then trains ML models on that centralized system. In contrast, Federated Learning (FL) trains an ML model across multiple data holders such as decentralized nodes and edge devices while keeping data local and transfers only the model updates to the central server. Consider $K$ data holders $F_1, ...F_K$ wishing to train a single ML model by consolidating their respective data $D_1, ..., D_K$. A centralized method brings all data together in the centralized location and uses $D = D_1, \cup \cdots \cup, D_K$ to train a single model. In federated learning, data holders $F_1, ...F_K$ collaboratively train a model $M_{FED}$ without data holders $F_i$ sharing their data $D_i$ with others, under the condition that the performance of federation $P_{FED}$ remains very close to the performance of the single central model $P_{SUM}$. This condition can be stated as:

$$|P_{FED} - P_{SUM}| \leq \omega \tag{2}$$

where $\omega$ is a small non-negative real number [10].

For load forecasting, we train a deep neural network with multiple smart meter data distributed over the network without explicitly exchanging data samples with the central server. None of the local data are ever transmitted between parties; only model-related parameters are shared. The federated learning process is described in the next subsection, followed by the local preprocessing and the considered FL algorithms.

### 4.1. Federated Learning Process

Fig. 2 depicts FL process for load foresting: a single model is trained collaboratively over distributed smart meter data. As described in section 3, the LSTM model has shown great successes in load forecasting; therefore, it is used here as well. While we consider two FL strategies, FedSGD and FedAVG, the overall FL process is the same with one round of training consisting of the following steps:

**Step 1:** If this is the first training round, the server initializes the global LSTM weights; otherwise, the server proceeds with the weights obtained from the previous training round. A random subset of the smart meter devices is selected for the current training round, and the server sends a copy of the global model to those selected devices. Only a subset of devices participate in each training round as this improves convergence [10].
**Step 2:** The devices receive a copy of the global model and train it using only the local data. To enable training with LSTMs, this local data is preprocessed and transformed into a suitable form: as described in Subsection 4.2, the preprocessing is the same for both algorithms FedSGD or FedAVG. On the other hand, specifics of the local training depend on the type of FL, FedSGD or FedAVG.
**Step 3:** The devices that participated in the local training send the updated model parameters to the server. As each device trained the model with different local data, the updated parameters are different among devices.
**Step 4:** The server receives the local model parameters and aggregates them to construct an improved global model. In this study, we consider two main aggregation approaches, FedAVG and FedSGD. The process repeats from step 1 until convergence.
**Step 5:** After the model converges, the server sends the trained global model to all participants.
**Step 6:** The participants replace the out-of-date local models with the updated one received from the server and are now ready to carry our load forecasting. Note that when the trained model is used for forecasting, local data is still prepared with the same preprocessing technique as the one used during the model training in Step 2.

Conventional ML typically assumes independent and identically distributed (IID) variables [34, 31, 35]; however, data collection in the FL setup violates this assumption. The non-IID data in load forecasting are caused by the smart meters corresponding to particular users or groups of users with different preferences and behavioral patterns. Alternatively stated, smart meters typically collect data in different contexts, leading to significant differences in the data distributions and patterns among them. Although this imposes challenges for ML training, the proposed FL for load forecasting achieves better accuracy than the conventional ML approaches as will be shown in the evaluation.

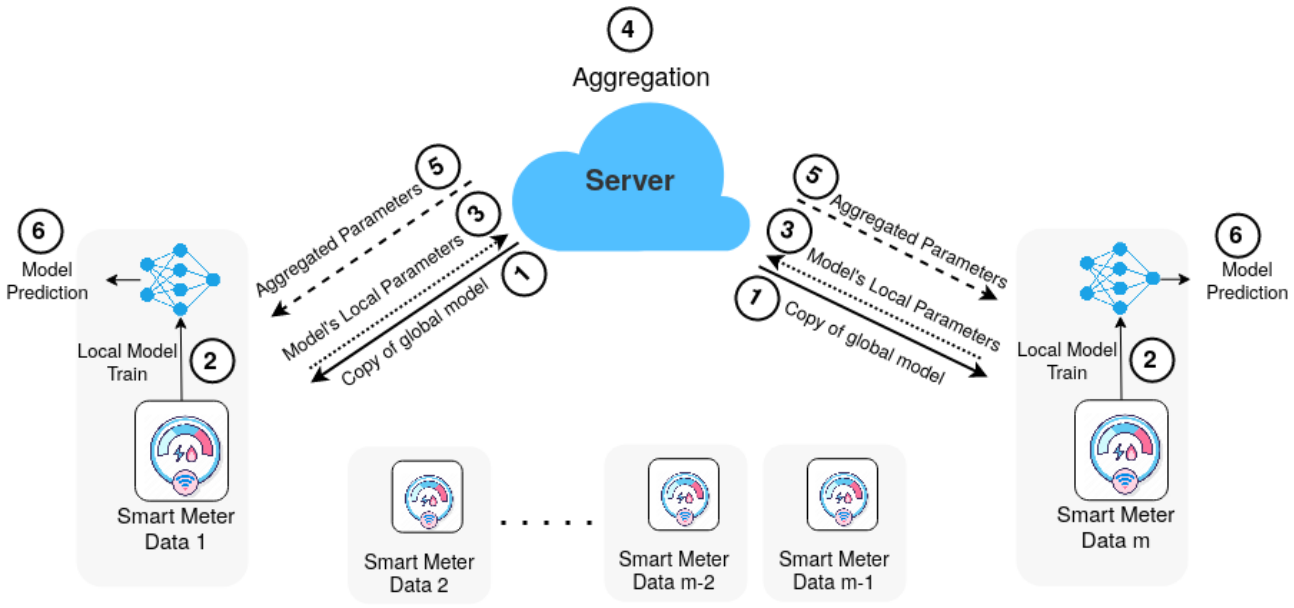


**Figure 2:** Federated Learning for Load Forecasting

## 4.2. Local Preprocessing

Steps 2 and 6 of the federated learning process, as mentioned in Subsection 4.1, both involve preprocessing. The preprocessing is significant in conventional ML, but it is even more important in federated setting as data at the local nodes may contain different distribution, patterns, and data scales, which makes the training process more difficult.

In conventional ML, data are scaled to reduce the large features dominance and improve the convergence; however, while conventional ML scales after aggregating all data, here we scale data individually on each node. If smart meters observe similar patterns but with different magnitudes, this individual scaling will make these load profiles more similar to each other and facilitate training. Specifically, Min-Max normalization is applied at each node individually. It transforms $x$ to $x'$ without distorting differences in the ranges of values as follows:

$$x' = \frac{x - Min(x)}{Max(x) - Min(x)} \tag{3}$$

where $x$ is the original feature value, $Min(x)$ and $Max(x)$ are the minimum and maximum of that feature on the considered node, and $x'$ is the normalized value in the range of 0 to 1 [11].

Next, data must be transformed into a form suitable for modeling temporal dependencies and for use with LSTMs. As common when RNNs are used with sensor data, the sliding window technique is applied [11]: it converts time-series data into windows of size $w \times f$ where $w$ is the number of time steps contained in the window and $f$ is the number of features. The first window contains the first $w$ smart meters readings. Then, the window slides for $s$ times steps, and the second window contains readings from $s + 1$ to $s + w$, and so on.

Here, features include the load data reading from smart meters and any other features generated from the reading date/time or from meteorological information. In experiments, we include features such as the day of the week, the hour of the day, and the day of the year. Although sequential models, such as LSTM, are able to extract temporal patterns, these additional features assist the model with capturing the date and time-related patterns.

## 4.3. Federated Learning Algorithms

In general, the federated learning objective for $K$ devices can be described in a form of the optimization problem:

$$\min_{w} \ell(w) = \sum_{k=1}^{K} \frac{n_k}{n} L_k(w) \tag{4a}$$

$$\text{where } L_k(w) = \frac{1}{n_k} \sum_{i \in P_k} \ell_i(w) \tag{4b}$$

where $\ell(w)$ is the global model's loss function, $L_k(x)$ is the loss of the $k^{th}$ device, and $\ell_i(w)$ is the loss for sample $i$. $P_k$, $k \in \{1, ..., K\}$ denotes a partition of data points stored on the device $k$, $n_k = |Pk|$ is the size of $P_k$, and $n = \sum_{k=1}^{K} n_k$ is the size of all data on all devices. The objective is to find $w$ which minimizes the loss $l(w)$ over the distributed data $P$ with the assumption that $P_i$ may be very different from $P_j$ for devices i≠j.

Here, we consider two ways of solving this optimization problem: Federated Stochastic Gradient Descent and Federated Averaging.

**Federated Stochastic Gradient Descent (FedSGD).** In FedSGD [36], a distributed stochastic gradient descent algorithm is applied in the federated setting to optimize the

**Algorithm 1** Federated Stochastic Gradient Descent (FedSGD)

1: **Server Execution:**
2: Initialize global model weights $w_0$, and learning rate $\eta$
3: **for** each round t=1,2,... **do**
4: $S_t \leftarrow$ random set of $m$ clients
5: Send global model to $S_t$ clients
6: **for** each client k $\in S_t$ **in parallel do**
7: $g^k_{t+1} \leftarrow$ **GradientStep**$(k, w_t)$
8: $w_{t+1} \leftarrow w_t - \eta \sum_{k \in K} \frac{n_k}{n} g^k_t$
9: Send the model to all participants

10: **Client Execution:**
11: **procedure** GRADIENTSTEP$(k, w)$
12: $g \leftarrow \bigtriangledown \ell(w)$ over $P_k$
13: return $g$ to server

**Algorithm 2** Federated Averaging (FedAVG)

1: **Server Execution:**
2: Initialize global model weights $w_0$
3: **for** each round t=1,2,... **do**
4: $S_t \leftarrow$ random set of $m$ clients
5: Send global model to $S_t$ clients
6: **for** each client k $\in S_t$ **in parallel do**
7: $w^k_{t+1} \leftarrow$ ClientUpdate$(k, w_t)$
8: $w_{t+1} \leftarrow \sum_{k \in K} \frac{n_k}{n} w^k_t$
9: Send the model to all participants

10: **Client Execution:**
11: **procedure** CLIENTUPDATE$(k, w)$
12: $B \leftarrow$ split $P_k$ into batches of size $b_s$B
13: **for** each local epoch $e < E$ **do**
14: **for** batch $b \in B$ **do**
15: $w \leftarrow w - \eta \bigtriangledown \ell(w)$
16: return $w$ to server

model collaboratively: Algorithm 1 depicts the steps of FedSGD. For each communication round (Line 3), a subset of devices $S_t$ is selected randomly (Line 4) to receive a copy of the global model (Line 5). Then, each client device $k$ from $S_t$ (Line 6) takes one step of the gradient descent $g$ locally on the current model $w_t$ using its local data (Line 7). Procedure *GradientStep* (Line 11) is executed on clients: it calculates the gradient $\bigtriangledown$ over local data $P_k$ and returns it to the server. Then, the server aggregates the received gradients by taking the weighted average of the clients gradients proportional to the number of training samples and the global model is updated using this weighted average and learning rate $\eta$, as shown in Line 8. The process repeats from Line 3 until convergence. Finally, the trained model is broadcasted to all participants (Line 9).

**Federated Averaging (FedAVG)**. Like FedSGD, FedAVG also solves the defined FL optimization problem. In contrast to FedSGD, in which the local clients take one step of the gradient descent and exchange the gradients without applying them to the local models, FedAVG allows the devices to update the local model by iterating through weight updates $w \leftarrow w - \eta \bigtriangledown \ell(w)$ multiple times before sending the updated model weights to the server. Algorithm 2 presents the FedAVG process. Each round of FedAvg starts the same as FedSGD by randomly selecting a subset of devices $S_t$ and broadcasting the model to the chosen devices (lines 4 and 5). The selected devices train in parallel their local models with the local data for multiple epochs (Line 7). Procedure *ClientUpdate*, Line 11, shows the local model update process: the local data $P_k$ are divided into the batches $B$ of size $s_b$ (Line 12) and the local device trains the received model for multiple epochs with created batches as shown in Lines 13 to 15. Each device from $S_t$ sends the new local weights to the server (Line 16), and the server updates the global model by calculating the weighted average of the received local weight as shown in Line 8. As in FedSGD, the process is repeated from Line 3 until convergence and, finally, the trained model is broadcasted to all participants (Line 9).

FedSGD is an efficient method that guarantees the convergence in FL settings and is barely influenced by the non-IID problem under adequate training parameters [37]; however, it requires a large number of training rounds to produce good results [36]. FedAvg is an FedSGD alternative that shows significant improvement in communication and time efficiency [37]. The basic idea behind FedSVG is that if all local devices start from the same initialization parameters, averaging the weights is strictly equivalent to averaging the gradients and, therefore, does not necessarily harm the averaged model performance. Nevertheless, it is shown that heterogeneity of data slows down the convergence of FedAVG [38]. Xiang *et al.* [38] proved that by having an adaptive learning rate, the model can converge on non-IID data; consequently, we use adaptive learning rate as well.

Heterogeneity of smart meter data is high as these data are collected in different contexts and influenced by diverse human behaviour resulting in different load patterns and distributions. FedSGD is well suited for this context as it has shown promising results in learning from heterogeneous data [37]; therefore, it is examined here with respect to load forecasting. To reduce communication rounds and time complexity, we examine FedAVG, but the standard SGD is replaced with an Adam optimizer which embraces adaptive learning rate to accelerate convergence and improve the learning capability on non-IID data [38].

## 5. Evaluation

This section first introduces the dataset and evaluation metrics. Next, the performance of the proposed FL methods is compared with central training and individual LSTM models. Then, we examine a dynamic environment in which some devices do not participate in training, but use the trained model for local load forecasting. Finally, FedSGD and FedAVG are compared in terms of convergence, and the overall

results are discussed.

### 5.1. Dataset and Evaluation Metrics

This study is performed in collaboration with London Hydro, a local electrical distribution utility. Federated learning for load forecasting will enable London Hydro to provide large-scale forecasting services to its residential consumers and, consequently, increase return on investment from the smart meter infrastructure. London Hydro provided real-world data for the evaluation of the presented approaches through Green Button Connect My Data (CDM), a platform for secured sharing of energy data with the consumer's consent. The evaluation was conducted with 19 residential consumers, each one containing hourly energy consumption for three years, resulting in 25,560 readings per households, or 485,640 readings in total. As these readings are also used for the billing purposes, they are highly reliable and have the same reading frequency and the number of samples for each household.

Additional features including the day of the year, the day of the month, the week of the year, the day of the week, and the hour of the day were devised from the load reading date/time to assist with modelling daily, monthly, and weekly patterns.

Diversity among consumers in terms of load profiles is large: for illustration, Fig. 3 depicts the load data for the three households. It can be observed that load patterns, as well as load magnitudes, vary greatly among consumers. This diversity makes it difficult for a single model to capture all patterns among consumers. Therefore, many studies train a single model per consumer [30]; however, here we examine devising a single model for all consumers through federated learning.

For the evaluation, each individual household dataset is split into 70% for training and 30% for testing. This split remains the same for federated learning experiments as well as for conventional ML experiments, centralized and individual models for each meter, conducted for the purpose of comparison.

The model performance is compared with two metrics: Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). RMSE measures the deviation of the residuals (prediction errors); in other words, it measures how far the predicted values are form the observed (actual) values. MAPE is expressed as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2} \tag{5}$$

where $y_t$ are the actual values, $\hat{y}_t$ are the corresponding predicted value, and N is the number of observations.

The MAPE metric measures the average absolute error and is calculated as follows:

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\frac{|y_t - \hat{y}_t|}{y_t} \tag{6}$$

Note that RMSE is a scale dependent error metrics: the same RMSE value has a different meaning for different data magnitudes. On the other hand, MAPE expresses errors in terms of percentages and, thus, is better suited for comparison among data sets.

### 5.2. Comparison of FedAVG and FedSGD with Individual LSTMs and Central Model

In this subsection, the proposed FedAVG and FedSGD for load forecasting are compared to the individual LSTMs and the central model. The individual LSTMs approach refers to training an individual LSTM for each household. This approach does not require any exchange of data with the central server, but the drawback is that there is a large number of models that need to be maintained individually. As individual models are trained for a specific client, they are personalized models and, thus, are good at capturing intricacies of the specific clients. A comparison of FedAVG and FedSGD with individual LSTM will examine how good the federated learning strategies are in training a single model to capture diversities among clients.

In the central model, data from all households are combined to form a coherent dataset used to train a single LSTM. In this approach, all data must be transferred to thecentral location. By comparing are FL strategies to the central model, the ability of the proposed FL models to learn from heterogeneous data will be determined.

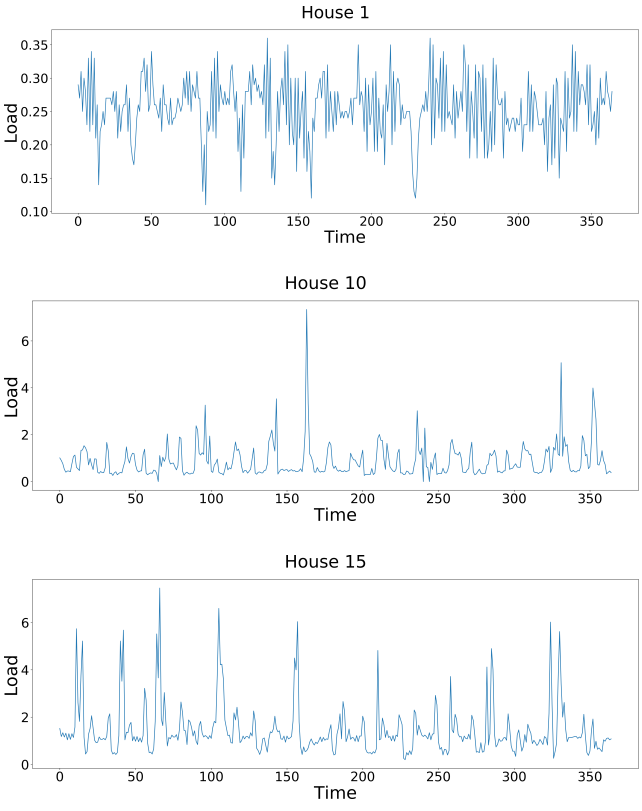


**Figure 3:** Electricity load examples for Home 1, Home 10, and Home 15

To keep the comparison fair, all models, federated, central, and individual, use the same architecture consisting of one layer LSTM with 32 hidden units. Tuning hyperparameters individually for each of the considered models has a potential to increase accuracy; however, this would result in massive computational cost. Moreover, in FL, hyperparameter tuning is still an open challenge [34] because of the distributed environment, FL-specific hyperparameters, and the network traffic associated with tuning.

For the federated learning strategies, FedAVG and FedSGD, six clients participate in each round of training. This number was selected as it allows for diversity of clients in each round while still including less then one third of clients per round. In FedAVG, one round of training on the client consists of five epoch, thus allowing clients to make reasonable learning steps before aggregation. Further tuning could improve federated learning results, but would select the parameters only for this specific combination of clients.

### *5.2.1. Forecasting one hour ahead*

Table 1 shows the average test error in terms RMSE and MAPE for FedSGD, FedAVG, individual LSTMs, and the central model for one hour ahead forecasts. The two federated learning strategies, FedSGD and FedAVG, achieved lower errors that individual LSTMS and the central model. FedSGD achieved the lowest RMSE test error while FedAVG obtained the lowest MAPE test error. As RMSE for FedAVG is very close to RMSE for FedSGD, the overall better model is FedAVG because of its low MAPE.

Table 1 examines the average accuracy for all households, but we also need to investigate how models perform for individual households. Fig. 4 depicts MAPE errors obtained by the four approaches, for each house individually. It can be observed that FedAVG outperforms others approaches for all but 5 households. For those 5 households, the central model achieves only slightly better accuracy; however, the central model performs worse in most households making FedAVG an overall better model. It is also worth noting that FedSGD performance is very similar to that of the central model, with almost the same MAPE error. This confirms the findings from Table 1 with FedSGD obtaining very close average MAPE to the central model.

While Fig. 4 compares the accuracy of the considered algorithms in terms of MAPE, Fig. 5 does so in terms of RMSE. In terms of RMSE, all algorithms achieve similar accuracy for most houses, and there is no clear winner. Nevertheless, federated algorithms achieve similar accuracy to conventional ML while not requiring data sharing.

RMSE measures the standard deviation of errors (Equation 5) and because of squaring, it imposes high penalty on larger errors and is sensitive to outliers. In contrast, MAPE calculates the average percentage error (Equation 6). Moreover, MAPE expresses the error as a percentage while RMSE is scale dependent with the same unit as the measured value. Because magnitude of the energy consumption varies among houses, MAPE is better suited when comparing among houses. In terms of MAPE, FedSGD achieves better accuracy than the remaining algorithms as observed in Table 1 and Fig. 4.

**Table 1**
Average RMSE and MAPE errors for all 19 houses: one hour ahead prediction

| Error | FedAVG | FedSGD | LSTM | Central Model |
|---|---|---|---|---|
| MAPE | 14.7522 | 16.7775 | 19.3123 | 16.8851 |
| RMSE | 0.6138 | 0.6084 | 0.6303 | 0.6200 |

An example of predicted versus actual values is shown in Fig. 6: it depicts the forecasts obtained by each of the four approaches for house 13. It can be observed that for the shown segment, the predicted values better match the actual values for FedAVG than for the other approaches. The remaining approaches, central model, individual LSTMs, and FedSGD, appear to obtain reasonable load forecasts but no distinction can be made regarding which one is better. Nevertheless, for this example, the best predictions are obtained by FedAVG, which corresponds to the observation seen with MAPE metrics shown in Table 1 or Fig. 4.

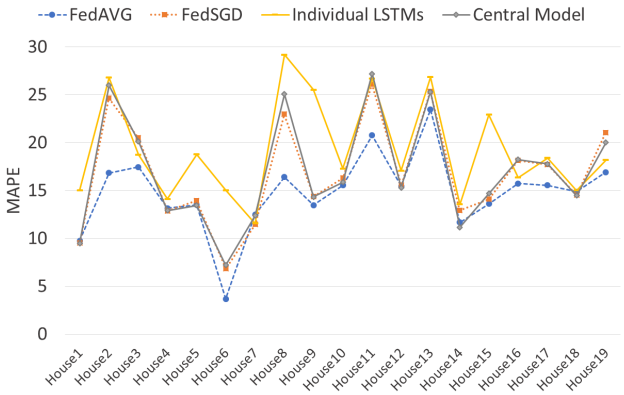


**Figure 4:** MAPE errors for FedAVG, FedSGD, LSTMs, and Central Model: one hours ahead prediction

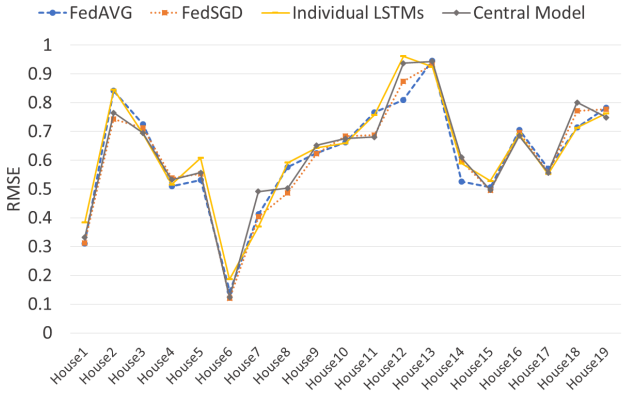


**Figure 5:** RMSE errors for FedAVG, FedSGD, LSTMs, and Central Model: one hours ahead prediction

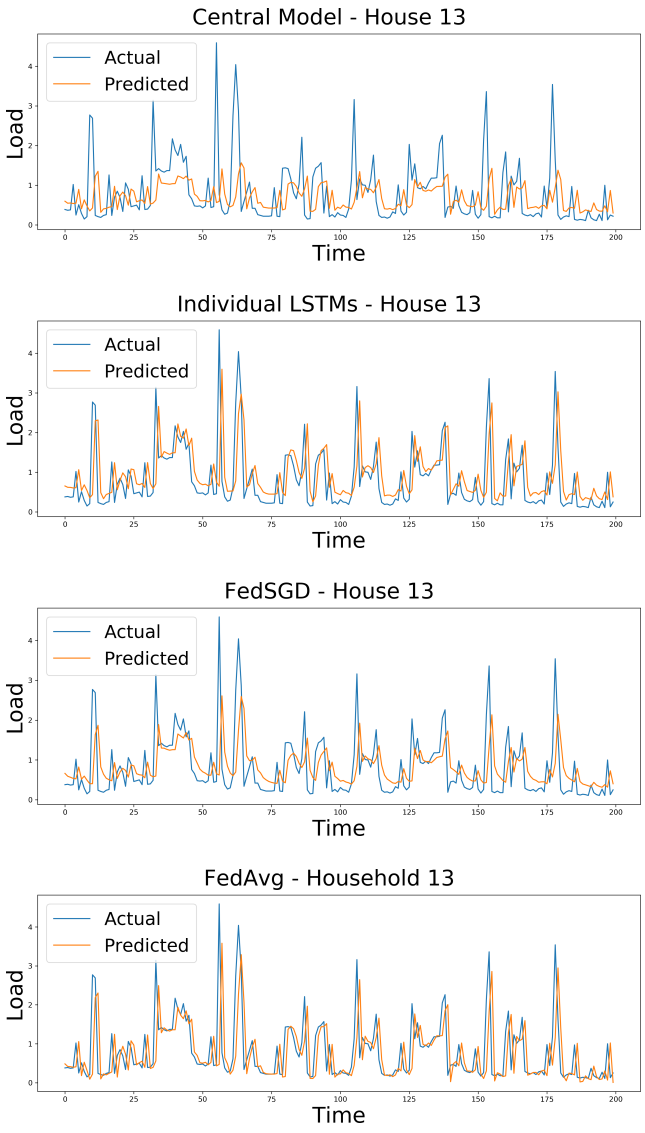


**Figure 6:** Actual versus predicted load for House 13: one hour ahead forecast

### 5.2.2. *Forecasting 24 hours ahead*

While Table 1, and figures 4 and 5 show results for one hour ahead prediction, Table 2 and figures 7 and 8 show results for 24 hours ahead forecasting. It can be observed from Table 2 that FedAVG achieved the best average accuracy in terms of MAPE while in terms of RMSE, the accuracy of FedAVG, FedSGD, LSTMs, and the central model was similar. Thus, FedAVG can be considered the best model for 24h ahead forecast. Comparing accuracy of 24h ahead forecasts, Table 2, with one hour ahead, Table 1, the average error is lower for shorter forecasting horizon, which is expected as it is, in general, easier to predict fewer hours ahead.

Fig. 7 shows MAPE values for each houses individually, for each of the four approaches. It can be observed that FedAVG and individual LSTMs achieve lower accuracy than the other two methods for most of the houses. For a few houses, individual LSTMs achieved lower errors, but overall, FedAVG is better as its average MAPE is lower than for other approaches as can be seen from Table 2.

**Table 2**
Average RMSE and MAPE errors for all 19 houses: 24 hours ahead forecast

| Error | FedAVG | FedSGD | LSTM | Central Model |
|---|---|---|---|---|
| MAPE | 17.3870 | 31.2705 | 20.5328 | 55.2167 |
| RMSE | 0.6868 | 0.6842 | 0.6439 | 0.6554 |

Comparing 24 hours ahead forecasting in terms of RMSE, Fig. 8, all four algorithms show similar accuracy for all houses. This matches the observation from Table 2, where all four algorithms have similar average RMSE. Nevertheless, as MAPE is better suited for data sets with different magnitudes, and FedAVG outperformed other approaches in terms of MAPE, FedAVG is the preferred algorithm.

Fig. 9 shows an example for 24 hours ahead forecast. For this house and for the shown forecasting segment, FedAVG gives predictions closer to actual values. This confirms the findings from MAPE comparison in Table 2: FedAVG achieves better accuracy then the other approaches for 24 hours ahead forecasts.

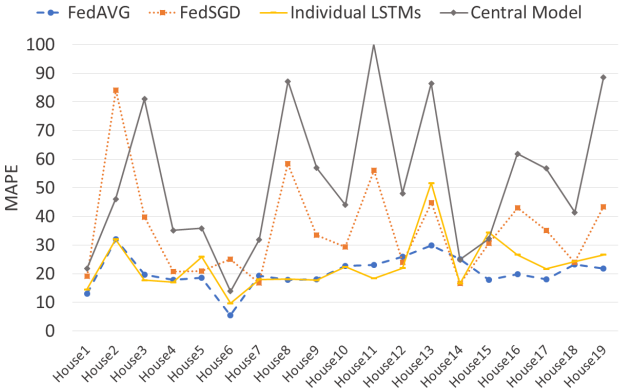


**Figure 7:** MAPE errors for FedAVG, FedSGD, LSTMs, and Central Model: 24 hours ahead forecast

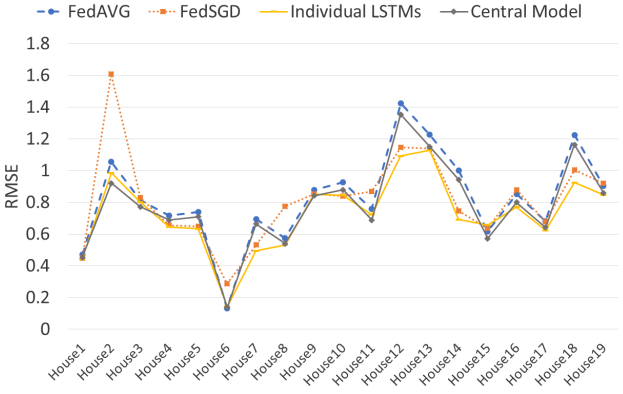


**Figure 8:** RMSE errors for FedAVG, FedSGD, LSTMs, and Central Model: 24 hours ahead forecast

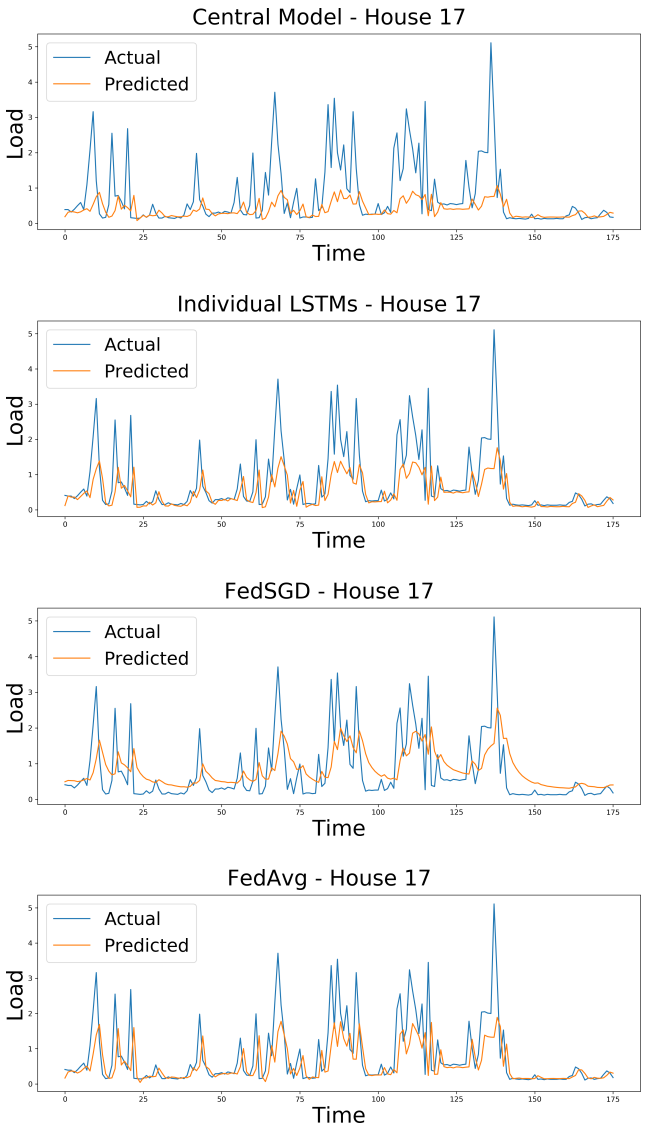


**Figure 9:** Actual versus predicted load for House 17: 24 hours ahead forecast

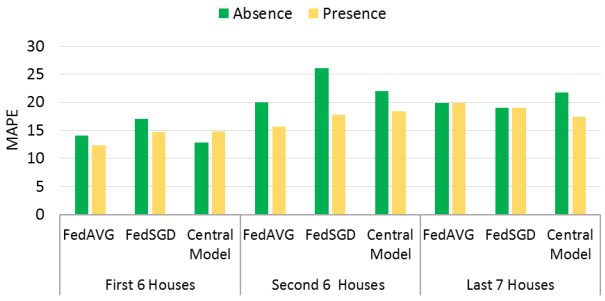


**Figure 10:** Dynamic environment: MAPE errors for FedAVG, FedSGD, and Central models: one hour ahead forecast

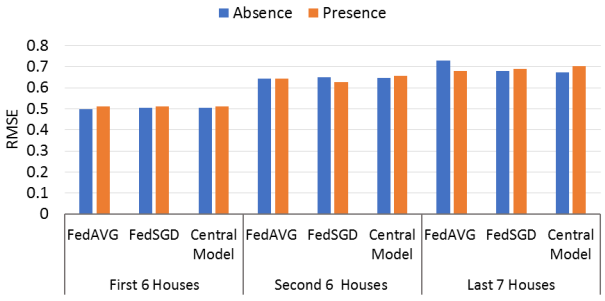


**Figure 11:** Dynamic environment: RMSE errors for FedAVG, FedSGD, and Central models: one hour ahead forecast

## 5.3. Evaluation in Dynamic Environment

This subsection examines if the model trained with federated learning can be used for the smart meters that did not participate in training. This represents a dynamic environment where some smart meters join the federation after the training is complete and only use an already trained model for forecasting.

For these experiments, the houses are divided into three groups: first 6 houses, second 6, and remaining 7 houses. The ML model is first trained without the first group of houses. Then this model is evaluated of the first group of houses and results are compared to the accuracy achieved when all houses participating in training. The same is repeated for the second and the third group of houses. Figures 10 and 11 show the results in terms of MAPE and RMSE for one hour ahead forecasts. 'Absence' indicates that the specific group of houses did not participate in training, while presence signifies that all houses participated in training. Note that an individual LSTM cannot be considered here as it requires the use of the target house data for training. Also, for each group of houses, the evaluation is always performed only on that group of houses, although other houses participated in training. This is somewhat similar cross-validation, but instead of randomly selecting the validation set sampled, a group of houses is assigned to the validation set.

In terms of MAPE, Fig. 10, all algorithms achieved better accuracy when all data are used for training, which is to be expected. However, even when a group of houses did not participate in training, the model was able to achieve the accuracy close to that of the model trained with all data. For the last 7 houses, FedSGD and FedAVG achieved almost exactly the same accuracy when those houses participated and did not participate in the training.

In terms of RMSE, Fig. 11, there was hardly any difference if the group of houses participated in the training or not. This demonstrates that for one hour ahead forecasting, federated learning strategies are successfully even for smart meters that did not participate in training. Figures 12 and 13 examine federated learning for 24 hours ahead forecasting in a dynamic environment: Fig. 12 shows MAPE while 13 shows RMSE metrics. As before, federated learning strategies do not exhibit overall performance degradation when some groups of houses do not participate in training.

## 5.4. Convergence and Computation Cost

In this subsection, the proposed FedAVG and FedSGD algorithms are assessed in terms of convergence. In general, the system converges if further training does not significantly

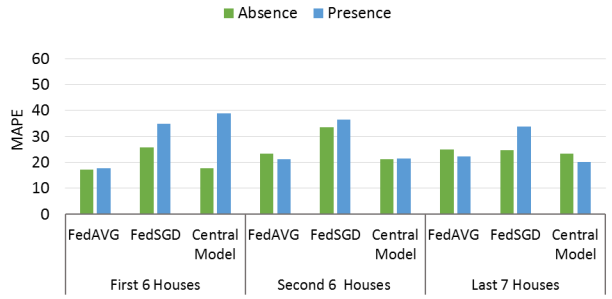


**Figure 12:** Dynamic environment: MAPE errors for FedAVG, FedSGD, and Central models: 24 hours ahead forecast

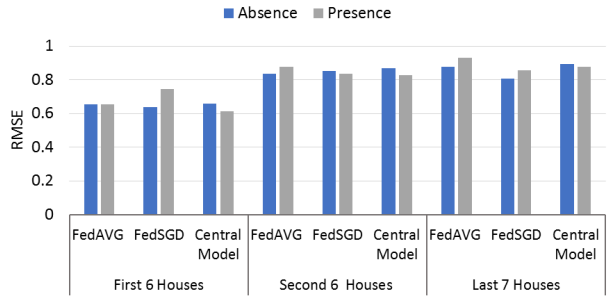


**Figure 13:** Dynamic environment: RMSE errors for FedAVG, FedSGD, and Central models: 24 hours ahead forecast

improve the model performance. For neural networks, convergence is examined in respect to epochs; however, in FL we are interested in the training rounds as they drive the communication between clients and the server. In FedSGD, each client performs only a single step of gradient descent in one training round. In contrast, FedAVG carries out several gradient descent steps on the client before communicating the updates back to the server.

To examine convergence, the same setup has been used as described in Subsection 5.2. The training errors for up to 150 rounds for the two algorithms, FedSGD and FedAVG, are shown in Fig. 17. As expected, both converge to similar errors, but FedSGD takes many more rounds to converge than FedAVG as FedAVG performs multiple steps in a single round. These training rounds are indicators of the communication and, therefore, it can be concluded that the network traffic in FedAVG is much lower than in FedSGD.

Furthermore, FedAVG computation is compared to that of the central model and individual LSTMs. With five epochs in each training round, FedAVG converged after the fifth epoch as can be seen from Fig. 14. The centralized model converged around the eleventh epoch, Fig. 15, while the convergence of individual models varied among different houses, Fig. 16, with majority converging before 15th epoch; thus, 15 epochs are considered for computation analysis. Note that convergence for the central model and individual LSTMs is shown in respect to epochs while for FedAVG, training rounds are used. This is because in FL, training rounds represent the steps of the training process.

Table 3 compares the training computation of FedAVG, the central model, and individual LSTMs. Each approach

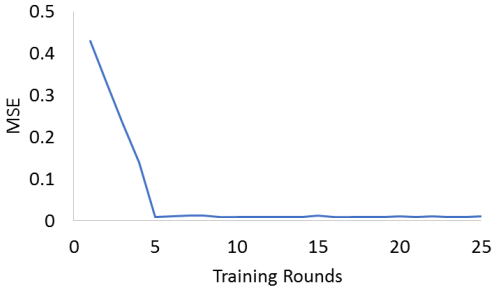


**Figure 14:** MSE for each FedAVG training round


**Figure 15:** MSE for each epoch: central model

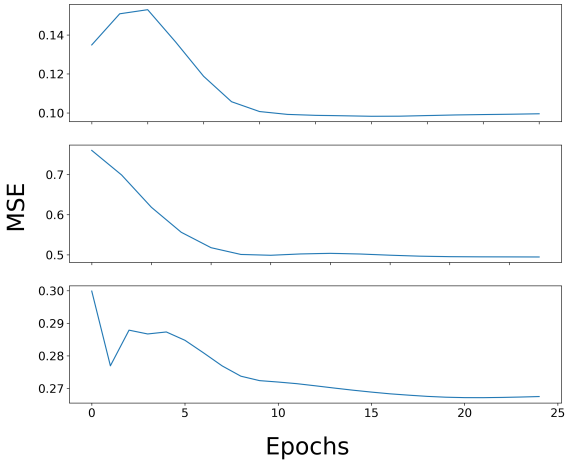


**Figure 16:** MSE for each epoch: three examples of individual LSTMs

has the same quantity of data available and uses the same batch size of 250. Therefore, each one processes 80 batches per epoch. For FedAVG, five rounds are needed, each one with six clients executing five epochs. This results in 80 $batches \times 5\ rounds \times 6\ clients \times 5\ epochs = 12,000$ runs over batches. For the central model and individual LSTM the concept of training rounds does not apply and we assume that 15 epochs are sufficient as the central model and most individual LSTMs converge with 15 epochs (figures 15 and 16). Each of these two approaches has to process data from 19 houses resulting in a total of 80 $batches \times 19\ houses \times 15\ epochs = 22,800$ runs over batches.

Architectures for all models have the same structure:

**Table 3**
Computation Comparison between FedAVG, Central Model, and Individual LSTMs

| Algorithm | Batches/epoch | Rounds | Clients | Epochs | Total | Time(sec) |
|---|---|---|---|---|---|---|
| FedAVG | 80 | 5 | 6 | 5 | 12000 | 61.98 |
| Central Model | 80 | N/A | 19 | 15 | 22800 | 117.77 |
| Individual LSTMs | 80 | N/A | 19 | 15 | 22800 | 119.99 |

LSTM with the same number of layers and the same number of parameters. Therefore, the time to process a single batch is similar across all morels. As FedAVG needs 12,000 runs while the other two approaches need approximately 22,800 runs, FedAVG will be significantly faster. Note that this ignores the fact that in FedAVG, in each training rounds, clients train in parallel reducing the computation time. With individual LSTMs, parallelization is possible while the central model requires sequential processing.

However, the main benefit of FL is that it achieves better accuracy than the other approaches while not requiring the clients to share their local data. Moreover, the clients can join the federation after the training is complete and FedAVG still achieves good forecasting accuracy as shown in Section 5.3.

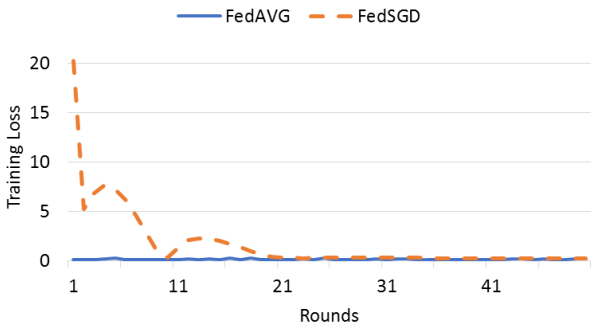


**Figure 17:** The training loss for FedAVG and FedSGD for 50 training rounds.

### 5.5. Discussion

With increasing concerns regarding security and privacy, it is becoming more important to develop ML techniques capable of training ML models without requiring participants to share their local model. Federated learning is a step in this direction although it is still in its early stages, and it requires further improvements and examinations in different contexts. This study investigates the abilities of the FedSGD and FedAVG approaches in load forecasting.

Both algorithms, FedSGD and FedAVG, achieved comparable or better accuracy than a single central model or individual local models for one hour ahead forecasts, as shown in Table 1. For 24 hours ahead, FedAVG outperformed other algorithms in terms of MAPE while in terms of RMSE there was very little difference among algorithms. Overall, FedAVG was the best algorithm as it is able to achieve high accuracy without requiring the clients to share their local data.

In addition to examining the overall error, it is important to consider performance on individual houses. It terms of MAPE, FedAVG performed better than FedSGD for one hour and 24 hour ahead forecasting, as observed from figures 4 and 7. Moreover, FedAVG required fewer training rounds than FedSGD (Fig. 17).

Our experiments from Subsection 5.3 also show that the trained model can be used with a good success for smart meters that did not participate in the training. This is important in scenarios when new smart meters are added to the federation or when there are very little data from some meters.

RNN-based models, including LSTMs, have been outperforming other architectures for load forecasting on individual smart meters [6, 29] and this study demonstrated that FedAVG achieves similar or better results than individual LSTMs. Moreover, with individual models, there must be sufficient data from each individual household to train the model. In contrast, FedAVG is even successful for households that did not participate in training as shown in Subsection 5.3.

To further examine the performance of FedAVG, we compare it to simple shifting and the FL approach proposed by Taïk *et al.* [7] in Table 4. Simple shifting for one hour ahead forecasting uses the consumption at the current time step $t$ as the forecast for the next time step $t + 1$. For 24 hours ahead forecasting, values from last 24 hours serve as the forecasts for the next 24 hours. The work of Taïk *et al.* [7] is the most related work to ours, as it also employs federated learning for load forecasting. Similar to our work, they use the FedAVG aggregating strategy, but, while we employ an adaptive learning rate, they use a fixed learning rate. Moreover, their study only considers one hour ahead forecasting.

As seen from Table 4, our FedAVG technique achieved better results than Taïk *et al.* [7] in terms of MAPE and RMSE for both one hour ahead and 24 hours ahead forecasting. Simple shifting archived better accuracy than FedAVG for one hour ahead forecast; however, for 24 hours ahead, simple shifting achieved very poor results in comparison to our FedAVG or the work of Taïk *et al.* [7]. Consequently, FedAVG is an overall better approach.

## 6. Conclusion

Traditional machine learning for load forecasting involves the central server which carries out ML training. The drawback of this approach is that all the data collected by different devices must be sent to the central server, what introduces privacy and security risks, puts strain on communication networks and requires large centralized computational

**Table 4**
MAPE and RMSE for FedAVG, Simple Shifting, and Taïk *et al.* [7]: one hour ahead and 24 hours ahead prediction

| | MAPE(%) | | RMSE | |
|---|---|---|---|---|
| Algorithm | one hour ahead | 24 hours ahead | one hour ahead | 24 hours ahead |
| FedAVG | 14.7522 | 17.3870 | 0.6138 | 0.6868 |
| Taïk *et al.* [7] | 19.7516 | 21.9798 | 0.6569 | 0.7358 |
| Simple Shifting | 13.0784 | 42.5700 | 0.5555 | 1.6719 |

resources.

Consequently, this paper proposes a federated learning (FL) approach for load forecasting with smart meters capable of training a machine learning model in a distributed manner, without requiring the participant to share their local data. In the proposed FL approach, a global ML model is shared across independent devices corresponding to individual smart meters and each device updates its local copy of the shared model using local data. Then, these local updates are sent to the server to be aggregated and merged into the global model. As recurrent neural networks can capture temporal dependencies and have been very successfully in load forecasting, the proposed approach employs LSTM, a variant of RNN, as the base learner. Two strategies, FedSGD and FedAVG, have been examined: they differ in the way they train the local model and in the frequency of sending the model updates to the server.

The experiments show that both, FedAVG and FedSGD approaches achieve higher accuracy than the individual LSTMs and central models for one hour forecasting horizon. For this horizon, FedAVG achieved slightly better accuracy than FedSGD. For 24 hours ahead, FedAVG outperformed all other approaches while FedSGD experienced conversion difficulties and exhibited higher errors than individual LSTMS. Also, the proposed approach was evaluated in a dynamic environment where some smart meters join the federation after the training is complete and use the already trained model for load forecasting. The results demonstrate that even in this scenario, the FedAVG and FedSGD achieve high accuracy.

The future work will evaluate the proposed approach with a large number of homes and examine the impact of various RNN architectures. Moreover, we will investigate customizing the global model by fine-tuning it on each client with local data.

## Acknowledgements

This research has been supported by London Hydro and Ontario Centres of Excellence under Grant OCI #: 33066. The authors would like to thank London Hydro for supplying industry knowledge and data used in this study.

## References

[1] UN Environment Programme, Energy, https://www.unep.org/explore-topics/energy, 2020.

[2] European Environment Agency, Energy and climate change, https://www.eea.europa.eu/signals/signals-2017/articles/energy-and-climate-change, 2017.

[3] I. K. Nti, M. Teimeh, O. Nyarko-Boateng, A. F. Adekoya, Electricity load forecasting: a systematic review, Journal of Electrical Systems and Information Technology 7 (2020) 1–19.

[4] G. Notton, C. Voyant, Forecasting of intermittent solar energy resource, Advances in Renewable Energies and Power Technologies 1 (2018) 77–114.

[5] W. Hurst, C. A. C. Montañez, N. Shone, Time-pattern profiling from smart meter data to detect outliers in energy consumption, Internet of Things 1 (2020) 92–108.

[6] A. Arif, N. Javaid, M. Anwar, A. Naeem, H. Gul, S. Fareed, Electricity load and price forecasting using machine learning algorithms in smart grid: A survey, in: International Conference on Advanced Information Networking and Applications, 2020, pp. 471–483.

[7] A. Taïk, S. Cherkaoui, Electrical load forecasting using edge computing and federated learning, in: EEE International Conference on Communications, 2020, pp. 1–6.

[8] N. Truong, K. Sun, S. Wang, F. Guitton, Y. Guo, Privacy preservation in federated learning: Insights from the gdpr perspective, arXiv preprint arXiv:2011.05411 (2020).

[9] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan, et al., Towards federated learning at scale: System design, arXiv preprint arXiv:1902.01046 (2019).

[10] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Transactions on Intelligent Systems and Technology 10 (2019) 1–19.

[11] M. N. Fekri, H. Patel, K. Grolinger, V. Sharma, Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network, Applied Energy 282 (2021) 116177.

[12] X. Wu, Z. Liang, J. Wang, Fedmed: A federated learning framework for language modeling, Sensors 20 (2020) 4048.

[13] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, W. Shi, Federated learning of predictive models from federated electronic health records, International journal of medical informatics 112 (2018) 59–67.

[14] S. R. Pokhrel, J. Choi, Federated learning with blockchain for autonomous vehicles: Analysis and design challenges, IEEE Transactions on Communications 68 (2020) 4734–4746.

[15] Y. Wang, Y. Tong, D. Shi, Federated latent dirichlet allocation: A local differential privacy based framework, in: AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 6283–6290.

[16] W. Liu, L. Chen, Y. Chen, W. Zhang, Accelerating federated learning via momentum gradient descent, IEEE Transactions on Parallel and Distributed Systems 31 (2020) 1754–1766.

[17] L. Li, Y. Fan, M. Tse, K.-Y. Lin, A review of applications in federated learning, Computers & Industrial Engineering (2020) 106854.

[18] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, J. Dureau, Federated learning for keyword spotting, in: IEEE International Conference on Acoustics, Speech and Signal pages=6341–6345, year=2019, ????

[19] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, Q. Yang, Fedvision: An online visual object detection platform powered by federated learning, in: AAAI Conference

on Artificial Intelligence, volume 34, 2020, pp. 13172–13179.
[20] Y. Chen, X. Qin, J. Wang, C. Yu, W. Gao, Fedhealth: A federated transfer learning framework for wearable healthcare, IEEE Intelligent Systems 35 (2020) 83–93.
[21] J. Konečnỳ, B. McMahan, D. Ramage, Federated optimization: Distributed optimization beyond the datacenter, arXiv preprint arXiv:1511.03575 (2015).
[22] C. Briggs, Z. Fan, P. Andras, Federated learning with hierarchical clustering of local updates to improve training on non-iid data, arXiv preprint arXiv:2004.11791 (2020).
[23] V. Smith, C.-K. Chiang, M. Sanjabi, A. S. Talwalkar, Federated multi-task learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4424–4434.
[24] S. Ruder, An overview of multi-task learning in deep neural networks, arXiv preprint arXiv:1706.05098 (2017).
[25] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, arXiv preprint arXiv:1806.00582 (2018).
[26] M. Mohri, G. Sivek, A. T. Suresh, Agnostic federated learning, in: 36th International Conference on Machine Learning, 2019, pp. 8114–8124.
[27] K. Grolinger, M. A. Capretz, L. Seewald, Energy consumption prediction with big data: Balancing prediction accuracy and computational resources, in: IEEE International Congress on Big Data, 2016, pp. 157–164.
[28] A. Zainab, D. Syed, A. Ghrayeb, H. Abu-Rub, S. S. Refaat, M. Houchati, O. Bouhali, S. B. Lopez, A multiprocessing-based sensitivity analysis of machine learning algorithms for load forecasting of electric power distribution system, IEEE Access 9 (2021) 31684–31694.
[29] L. Sehovac, K. Grolinger, Deep learning for load forecasting: Sequence to sequence recurrent neural networks with attention, IEEE Access 8 (2020) 36411–36426.
[30] Y. Tian, L. Sehovac, K. Grolinger, Similarity-based chained transfer learning for energy forecasting with big data, IEEE Access 7 (2019) 139895–139908.
[31] J. Li, Y. Ren, S. Fang, K. Li, M. Sun, Federated learning-based ultra-short term load forecasting in power internet of things, in: IEEE International Conference on Energy Internet), 2020, pp. 63–68.
[32] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, S. Srikanteswara, Energy demand prediction with federated learning for electric vehicle networks, in: IEEE Global Communications Conference, 2019, pp. 1–6.
[33] A. Almalaq, G. Edwards, A review of deep learning methods applied on load forecasting, in: 16th IEEE international conference on machine learning and applications, 2017, pp. 511–516.
[34] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, arXiv preprint arXiv:1912.04977 (2019).
[35] A. L'Heureux, K. Grolinger, H. F. Elyamany, M. A. Capretz, Machine learning with big data: Challenges and approaches, IEEE Access 5 (2017) 7776–7797.
[36] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, 2017, pp. 1273–1282.
[37] D. Chai, L. Wang, K. Chen, Q. Yang, Fedeval: A benchmark system with a comprehensive evaluation model for federated learning, arXiv preprint arXiv:2011.09655 (2020).
[38] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of fedavg on non-iid data, arXiv preprint arXiv:1907.02189 (2019).