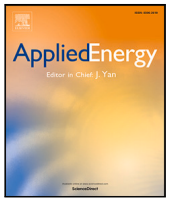
Applied Energy 326 (2022) 119915

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



Privacy-preserving federated learning for residential short-term load forecasting

Joaquín Delgado Fernández *, Sergio Potenciano Menci, Chul Min Lee, Alexander Rieger, Gilbert Fridgen

Interdisciplinary Centre for Security, Reliability and Trust (SnT), 29 Avenue John F.Kennedy, Luxembourg, 1855, Luxembourg

ARTICLE INFO

Keywords:
Deep neural networks
Differential privacy
Federated learning
Secure aggregation
Privacy-preserving federated learning
Short-term load forecasting

ABSTRACT

With high levels of intermittent power generation and dynamic demand patterns, accurate forecasts for residential loads have become essential. Smart meters can play an important role when making these forecasts as they provide detailed load data. However, using smart meter data for load forecasting is challenging due to data privacy requirements. This paper investigates how these requirements can be addressed through a combination of federated learning and privacy preserving techniques such as differential privacy and secure aggregation. For our analysis, we employ a large set of residential load data and simulate how different federated learning models and privacy preserving techniques affect performance and privacy. Our simulations reveal that combining federated learning and privacy preserving techniques can secure both high forecasting accuracy and near-complete privacy. Specifically, we find that such combinations enable a high level of information sharing while ensuring privacy of both the processed load data and forecasting models. Moreover, we identify and discuss challenges of applying federated learning, differential privacy and secure aggregation for residential short-term load forecasting.

1. Introduction

As the supply from intermittent and difficult-to-forecast renewable power sources increases, load forecasting – and especially residential short-term load forecasting (STLF) – is becoming ever more crucial for the reliability of modern power systems [1,2]. Residential STLF covers forecasting windows from a few minutes to a week ahead [2,3]. It plays an important role for many operational processes in the power system, such as planning, operating, and scheduling [4,5]. For instance, it enables energy providers to identify gaps between supply and demand in their customer portfolios. These gaps typically lead to high imbalance costs and ultimately to higher electricity prices for residential customers [6,7].

Traditionally, residential STLF has relied on aggregated load data and reference load profiles [5,8,9]. Yet, aggregation and reference profiles are often ill-suited for power systems with a high share of distributed generation and active demand-side management [5,8]. Moreover, they have become less reliable with residential heating and mobility being increasingly electric [10,11] and consumption patterns growing more dynamic, for instance, due to fluctuating levels of remote work [12]. These trends make accurate forecasting of individual residential loads an important priority.

There are various traditional methods for more granular STLF, but most build on limiting linearity assumptions (correlation between values and past values) even though residential load patterns are often highly dynamic [5]. Examples include time series models that rely on seasonal autoregressive integrated moving averages (ARIMA) [5,13], exponential smoothing, or linear transfer functions. Residential STFL is thus increasingly relying on methods that can work with non-linear dependencies, such as many Artificial Intelligence (AI) models [14–18].

A core challenge for any of these methods is the availability of granular data [19]. In many countries, this 'data scarcity' problem is tackled by pushing for advanced metering infrastructure (AMI), which substantially increases the resolution of residential load data [20]. STLF methods can make use of this data using either 'centralized' or 'de-centralized' approaches. Centralized approaches transfer smart meter data to a central forecasting system. While these forecasting systems promise very accurate results, they face a twofold problem. First, they are subject to substantial privacy challenges because smart meter data are often easily attributable to natural persons. That is, data collected from smart meters can be detailed enough to permit the identification of specific customers [21]. The transfer and aggregation of smart meter

* Corresponding author.
E-mail address: joaquin.delgadofernandez@uni.lu (J.D. Fernández).

https://doi.org/10.1016/j.apenergy.2022.119915
Received 21 April 2022; Received in revised form 6 July 2022; Accepted 28 August 2022
Available online 15 September 2022

data is thus typically subject to data privacy regulations such as the European Union's General Data Protection Regulation (GDPR) and its obligations and requirements for processing personal data [22,23]. Second, there are considerable regulatory uncertainties. In particular, it is often unclear how device ownership (who owns the smart meter) and aggregation impact data ownership. Moreover, specific regulations for smart meter data are typically absent [24,25]. These regulatory uncertainties often mean that centralized approaches such as Belgium's Atrias [26], or Norway's Elhub [27], which provide so-called *data lakes*, may not be desirable.

Decentralized approaches aim to tackle some of these issues by processing smart meter data locally. A particularly promising of these decentralized approaches is Federated Learning (FL) [28,29]. Federated Learning is a machine learning technique that offers a collaboration framework for clients. In a so-called 'federation' clients jointly train and share prediction models instead of training data. Although FL cannot guarantee privacy by itself [30,31], it can be combined with privacy-preserving techniques such as differential privacy (DP) and secure aggregation (SecAgg).

Even though such a combination could substantially benefit residential STLF, academic attention to FL has been limited so far [32–43] and the two components have mostly been considered mostly in isolation [44–46]. With this paper, we seek to close several gaps in the literature on FL-based STLF: Firstly, we aim to deepen the understanding of FL-based STLF by examining the effects of clustering based on Pearson correlation and the effects of architectural complexity. Secondly, we analyze the privacy and performance effects of adding privacy-preserving techniques (DP and SecAgg) to FL. Third, we identify key challenges associated with using a combination of FL and privacy-preserving techniques.

To do so, we conduct the following analysis: Initially, we identify promising NN architectures from a review of the recent FL literature. Subsequently, we select the most effective of these architectures and investigate six scenarios using real-world historical data. In a first scenario, we evaluate the performance of the selected architecture in a 'centralized' setting to establish a performance benchmark for the remaining five FL scenarios. In the second scenario, we investigate the performance and computational cost effects of moving from a centralized setting to a FL setting. In a third scenario, we then examine the effects of using correlated training data based on Pearson correlation and socio-economic factors. Correlation is typically avoided in non-federated ML models to increase data variability. Yet, for FL models, correlated data may increase forecasting accuracy [35] and mitigate problems with non-IID (non-independent and non-identically distributed) data. In the fourth scenario, we reflect on the trend to work with ever more complex models and explore the effects of increasing the complexity of the NN's architecture. In scenarios 5 and 6, we study how privacy-preserving techniques affect the training and performance of federated models. Specifically, we investigate the effect of different DP implementations (i.e., clipping techniques) and SecAgg on accuracy, privacy, and computational costs.

The remainder of the paper is structured as follows. Section 2 provides an overview of related work on the use of NNs for STLF, FL, and privacy-preserving techniques. Section 3 covers our evaluation method, including the simulation environment, dataset and evaluation metrics. Section 4 describes our evaluation design. It covers the selection of the baseline NN architecture, the specification of the analyzed differential privacy and secure aggregation techniques, the training process for the federated learning models, and the design of six evaluation scenarios. Section 5 presents the evaluation results for the six scenarios. Finally, Section 6 provides a synthesis of our results and points out directions for further research.

2. Related work

2.1. Federated learning

In most fields, AI-based methods have already proven their value. However, their performance is highly dependent on the quantity and quality of available training data. Generally speaking, AI-based methods are typically limited by data fragmentation and isolation — mostly due to competitive pressure and tight regulatory frameworks (related to data privacy and security). To address these challenges, McMahan et al. proposed a new technique, FL [28,29]. The main idea of FL is to collaboratively train machine learning models between multiple independent clients without moving or revealing the training data. In other words, FL allows competing participants to leverage each others' datasets without revealing their own individual datasets. In doing so, models trained with FL enable more accurate forecasts than models that were independently trained by each client. To date, there are two canonical training algorithms for FL and four different configurations for the distribution of data and errors.

The two canonical training algorithms are: federated stochastic gradient descent (Fed-SGD) and federated averaging (Fed-Avg) [28]. Fed-SGD works by averaging the client's gradients after every pass through a local data batch. More specifically, Fed-SGD clients compute gradients of their 'loss' for a sub-set of their data. The loss is a non-parametric function that penalizes bad predictions and to minimize it, the clients need to move toward the empirical minimum by taking steps in the opposite direction of the gradient. Clients subsequently send their locally computed gradients to a central server. The central server aggregates and averages them – either equally or in a weighted manner – to update the model weights. These updated weights are again sent to the clients and each client trains their local model with the updated weights. Training continues in an iterative manner until a pre-defined number of so called communication rounds have been reached or a common goal is achieved. In Fed-SGD, a communication round represents a full pass through all batches.

In Fed-Avg, the clients send their model weights instead of their gradients. Once the central server has received the weights, it aggregates and averages them to arrive at a new 'consensus' that will be sent back to the clients for the next training round. Unlike Fed-SGD, Fed-Avg does not split the training data into batches, which has two effects: the number of communication rounds is reduced substantially (only once per epoch) and an improvement in forecasting accuracy [28,39]. As in Fed-SGD, the training process continues until the pre-defined number of epochs has been reached or a common goal is achieved.

Besides different algorithms, FL applications can also differ in their configurations. These configurations depend on how the data is structured. More specifically, they depend on the configuration of the feature space $\mathcal{X}$, the label space $\mathcal{Y}$, and the space formed by the identifiers $\mathcal{I}$. Different setups of the triplet $(\mathcal{X}, \mathcal{Y}, \mathcal{I})$ can be classified as Horizontal, Vertical, Transfer and Assisted Federated Learning [47]. Take for instance two clients i and j.

- Horizontal Federated Learning is when i and j share the same feature space such that $\mathcal{X}_i = \mathcal{X}_j$ but their label spaces $\mathcal{Y}$ are different so that $\mathcal{Y}_i \neq \mathcal{Y}_j$. In our residential STLF example, Horizontal FL would be applicable when the model is to be trained on smart meter data from a range of clients with the same feature set (consumption, weather profile, etc.) and the data is held by different companies.
- Vertical Federated Learning is when $\mathcal{I}_i = \mathcal{I}_j$, but $\mathcal{X}_i \neq \mathcal{X}_j$ and $\mathcal{Y}_i \neq \mathcal{Y}_j$. This would be the case, for instance, when two companies have access to the same client but each of them holds a different feature set regarding the client.
- Federated Transfer Learning happens when $\mathcal{X}_i \neq \mathcal{X}_j$, $\mathcal{Y}_i \neq \mathcal{Y}_j$, $\mathcal{I}_i \neq \mathcal{I}_j$, $\forall D_i, D_j, i \neq j$. Federated Transfer Learning can be used, for instance, when two companies have different clients and feature sets but want to nevertheless collaboratively train a model.

- Assisted Learning (AL) is done through collided data between clients. Xian et al. [48] define collision as when clients with the same data entries of a dataset D have different feature spaces $\mathcal{I}_i = \mathcal{I}_j, \mathcal{X}_i \neq \mathcal{X}_i \; \forall D_i, D_j, i \neq j$. One client may use the errors of another for their own benefit by increasing their training performance.

Regardless of the chosen algorithm and configuration, FL is vulnerable to moral hazard [49] or so-called 'soft' attacks on the contextual integrity of the shared data. Moral hazard arises because FL is by nature collaborative [50]. Multiple clients must work together to train models iteratively using the respective data at their disposal. If one or several of these clients manipulate the joint training process, it does not work. In effect, federated learning requires trust between the clients involved.

2.2. FL-based short term Load forecasting

Short-term load forecasting is a complex, multivariate time series problem. Its complexity is high because residential load data is often replete with irregularities, missing or inaccurate values, and seasonality. Petropoulus et al. [2] provide an in-depth overview of these challenges. Yet, they also point out the increasing importance and momentum that STLF has gained over recent years. STLF is crucial because system operators require it for unit commitment and optimal power flow calculations [2,4,51]. Moreover, it enables utilities, energy suppliers, and distribution grid operators (DSOs) to optimize their customer portfolios, design tariffs, and strategically adapt flexibility offerings [2,4].

STLF typically build on three groups of methods: traditional methods, AI-based methods, and hybrid methods that integrate traditional and AI-based components [2]. Traditional methods such as ARIMA can capture seasonal trends but fall short when it comes to non-linear patterns and non-aggregated data. At the same time, they are simple to use and have light computational costs [2]. AI-based methods, in turn, are well suited to identifying non-linear patterns and work well with individual (i.e., residential level) and aggregated data (i.e., substation level) [5,52].

Within the larger group of AI-based methods, FL is a relatively new but increasingly popular method for STLF. Our following overview of these FL studies which follows is based on a search in Semantic Scholar using the following search terms: *short-term load forecasting neural networks* and *Federated Learning for Residential Short Term Load Forecasting*.

The first group of studies employ Fed-SGD [34,41]. He et al. [34] additionally use k-means clustering and compare performance between six scenarios with a different number of clusters in each scenario. Their results suggest that grouping data based on comparable load patterns substantially improves the performance of FL models. Lin et al. [41], in turn, focus on limiting the high computational cost of Fed-SGD. To this end, they introduce an asynchronous stochastic gradient descent algorithm with delay computation (ASGD-DC). Specifically, their algorithm uses a Taylor expansion to compensate for the delay of clients with lower computational power.

The second and substantially larger group of studies employ Fed-Avg. Similar to He et al. [34], Briggs et al. [32], Savi et al. [33], Afaf et al. [35], and Biswal et al. [36] investigate different forms of clustering for Fed-Avg. Their findings suggests that clustering based on k-means and socio-economic factors can also substantially improve the performance of Fed-Avg. With certain caveats, their findings also suggest that its possible to train good models with a small number of clients. Li et al. [37], in turn, use Fed-Avg to compare the effects of different federation sizes, ranging the number of clients from 2, to 4, and 6. They also vary the number of training rounds (epochs) from 5 to 15. Their results suggest performance is increased by increasing the number of clients and training rounds.

Xu et al. [38] as well as Husnoo et al. [42] investigate the effect of increasing the number of clients participating in the training rounds. Their results show a considerably drop in performance for the higher participation cases. This drop appears to be the result of non-IDD consumption data between the clients.

Khalil et al. in [43] use Fed-Avg to train a FL model for building control, replicating the use of FL for household training. They consider six floors of a seven-story building as clients. They later personalize the global FL model for the 7th floor – not used in the FL training – by running locally five additional rounds (epochs) and not sharing the data with the global model. Their results suggest that even the personalized FL model can help a smart building controller reduce total electricity consumption using FL.

In terms of relative performance, Fekri et al. [39] find that Fed-Avg provides more accurate results for STLF than Fed-SGD. Shi et al. [40],in turn, look beyond canonical FL and use a multiple kernel variant of maximum mean discrepancies (MK-MMD) to fine-tune the central server model (global). They train for several rounds using transfer learning to adapt the global model to specific customers. Their results indicate better performance than a canonical Fed-avg implementation.

The works of [32–43] provide important stepping-stones in FL-based STLF. In particular, they clearly indicate the prospect of using collaborative training to create accurate forecasting models. However, they provide only limited insights into the challenges of using FL. In particular, it is not yet clear if different but simpler clustering techniques such as Pearson correlation are also effective. Also, prior literature has not yet looked at the effect of architectural complexity. Moreover, existing studies do not or only in a very limited way account for matters of privacy. Thus, this paper aims to provide a better understanding of clustering and architectural complexity and explores the addition of different privacy preserving techniques.

2.3. NN architectures for FL-based short term load forecasting

The studies presented on FL-based STLF use a range of different NN architectures (Table 1). Overall, the architectures have become deeper (i.e., multi-layered) over time as depth is typically associated with more accurate results [52]. In terms of layer design, we found Fully Connected layers (FCL), Long Short-term Memory (LSTM) Layers [53] and Convolutional Neural Networks (CNN). LSTMs have feedback connections which understand the dependence between items in a sequence and which make them suitable for temporal pattern recognition. CNN layers emulate human retinas and can capture the spatial distribution of graphic patterns. Moreover, we found Encoder–Decoder or autoencoder architectures [54]. In these architectures, the NN is provided with a sequence (a vector) as an input and maps this sequence to another sequence. Encoder–Decoder architectures reduce the effects of outliers because they transpose the original input space into a differently encoded space [55,56]. Sehovac et al. [57] present a particular interesting example of a Seq2Seq architecture that includes an attention mechanism to help the decoder extract additional information.

Aside from different layer designs, we also identified hybrid designs. For instance, Kim et al. [58] use CNN with LSTM layers to find both spatial and temporal patterns. Building on their work, Tuong et al. [59] add a bi-directional LSTM layer to identify temporal trends both forward and backwards in time. Similarly, Zulfiqar Ahmad et al. [14] combine Seq2Seq from [54] with a CNN layer design. This combination allows for the capture of both temporal and spatial patterns and offers protection against outliers. Shi et al. [60] take a different path by clustering and pooling the training data to increase variability and reduce overfitting.

2.4. Privacy preserving techniques for federated learning

Privacy-preserving techniques can support the design of forecasting systems that comply with privacy requirements and regulations

Table 1
Neural network architectures for FL-based and non FL-based STLF.

Method	Dataset	Neural network architecture	Year
Marino et al. [54]	UCI - Individual household electric power consumption	LSTM + Repeat vector + LSTM + 2x FCL	2016
Kong et al. [61]	Australia SGDS Smart Grid Dataset	Stacked LSTM + FCL	2017
Li et al. [62]	Fremont, CA 15 min Retail building electricity load	Missing or incomplete architecture description	2017
Shi et al. [60]	Irish CBTs - Residential and SMEs	Stacked LSTM + Pooling mechanism	2018
Yan et al. [63]	UK-DALE Domestic Appliance-Level Electricity dataset	2x Conv + 1x LSTM + FCL	2018
Kim and Cho [64]	UCI - Individual household electric power consumption	Missing or incomplete architecture description	2019
Kim and Cho [58]	UCI - Individual household electric power consumption	2x Conv + LSTM + 2x FCL	2019
Le et al. [59]	UCI - Individual household electric power consumption	2x Conv + Bi + LSTM + 2x FCL	2019
Khan et al. [14]	UCI - Individual household electric power consumption	2x Conv + 2x LSTM (Encoder) + 2x LSTM (Decoder) + 2x FCL	2020
Afaf et al. [35]	Pecan Street Research Institute	2x LSTM (same size) + FCL	2020
Sehovac et al. [57]	Non-disclosed or private data	Sequence to Sequence with attention	2020
Li et al. [37]	Global Energy Forecasting Competition 2012	Missing or incomplete architecture description	2020
Xu et al. [38]	Pecan Street Research Institute	Missing or incomplete architecture description	2021
Briggs et al. [65]	Low Carbon London Dataset	2x LSTM (same size) + FCL	2021
He et al. [34]	Australia SGDS Smart Grid Dataset	2x LSTM (same size) + FCL	2021
Savi et al. [33]	Low Carbon London Dataset	LSTM (64) + LSTM (32) + FCL	2021
Zhao et al. [66]	Pecan Street Research Institute	2x LSTM (same size) + FCL	2021
Biswal et al. [36]	Commission for Energy Regulation (CER)	Missing or incomplete architecture description	2021
Khalil et al. [43]	CU-BEMS, smart building electricity consumption and indoor environmental sensor datasets	Missing or incomplete architecture description	2021
Shi et al. [40]	Low Carbon London Dataset	Missing or incomplete architecture description	2022
Lin et al. [41]	Commission for Energy Regulation (CER)	Missing or incomplete architecture description	2022
Husnoo et al. [42]	Solar Home Electricity Data from Eastern Australia	LSTM (256) + LSTM (128) + FCL	2022

[22,67,68]. From an organizational perspective, these techniques allow competing agents like energy providers to cooperate and integrate with utilities and DSOs [23,67]. Furthermore, their use might facilitate the creation of local markets that support the energy transition [69].

Privacy-preserving techniques are especially relevant for FL. Although FL offers considerable improvements over centralized ML methods, it does not guarantee privacy. Firstly, the shared data (gradients or model weights) may allow inadvertent attribution, and secondly, privacy can be compromised through the communication between clients and the central server. For instance, Zhu et al. found a way to use gradient updates to reconstruct the training data of a client [30]. This effectively means that gradient updates are to be treated as personal data and that FL requires additional measures when data privacy is required. In the following, we describe two such measures: DP as a way to anonymize training data and SecAgg as a mechanism to enable privacy-sensitive communication between clients and the central server.

Dwork [70] introduces DP as a technique to guarantee privacy when retrieving information from a dataset. As described in [71], "differential privacy addresses the paradox of knowing nothing about an individual while learning useful information about a population". DP hides individual data trends by using additive noise. In more technical terms, Dwork [70] introduced epsilon differential privacy (ϵ-DP) as follows: *"For every pair of inputs x and y that differ in one row, for every output in S, an adversary should not be able to use the output in S to distinguish between any x and y"*. The privacy budget (ϵ) determines how much of an individual's privacy a query may use, or to what extent it may increase the risk of breaching an individual's privacy. A value of $\epsilon = 0$ represents perfect privacy, which means that privacy cannot be compromised through any analysis on a dataset in question [72]. Jayaraman et al. [73] extended the concept of (ϵ-DP) to (ϵ, δ-DP) where δ is the failure probability to better control for the tails of the privacy budget.

DP is typically implemented by adding random noise to data queries. This noise is usually sampled from a Laplacian or Gaussian distribution [71]. Finding an adequate noise level is crucial but not trivial — especially for FL. Too much noise can not only hide patterns in the data but also complicate convergence of the local models due to the random updates of the patterns during training. Simply speaking, more noise means more privacy, but more noise also means less accuracy.

An alternative to adding noise to the training process or the data is using secure multi-party computation (SMPC) protocols, which enable privacy-preserving communication. One such protocol is SecAgg [74]. SecAgg uses cryptographic primitives that prevent the central server from reconstructing each client's involvement and contribution. In more technical terms, SecAgg allows a set of distributed, unknown clients to aggregate a value x without revealing the value to the other clients. The backbone of SecAgg is Shamir's *t-out-of-n* Secret Sharing. It enables a user to split a secret s into n shares [75]. To reconstruct the secret, more than $t-1$ shares are needed to retrieve the original secret s. Any allocation with less than $t-1$ shares will provide no information about the original secret. SecAgg implies two main algorithms: sharing and reconstruction. The sharing algorithm transforms a secret into a set of shares of the secret that are each associated with a client. Following [75], these shares are constructed in such a way that collusion between $t-1$ participants (t being the total number of participants) is insufficient to disclose other clients' private information. The reconstruction algorithm works in the opposite direction. It takes the mentioned shares from the clients and reconstructs the shared secret.

Of the two privacy-preserving techniques, only DP has so far been examined in the context of residential STLF. Chhachhi et all. [46], Eibl et al. [76], and Zhao et al. [77] use DP to train a 'centralized' machine learning model. More specifically, they perturb the datasets by adding noise drawn from either a Gaussian or Laplacian distribution before each training round of the model. To the best of our knowledge, Zhao et al. [66] are the first to combine FL and DP for STLF. Specifically, they include DP in the training process of a Fed-Avg model. However, they do not systematically analyze different DP parameters. Moreover, they do not look at secure multi-party computation protocols, such as SecAgg.

3. Method

3.1. Simulation environment

The evaluations in this paper are based on simulations we ran on the IRIS Cluster of the high performance computer (HPC) facilities of the

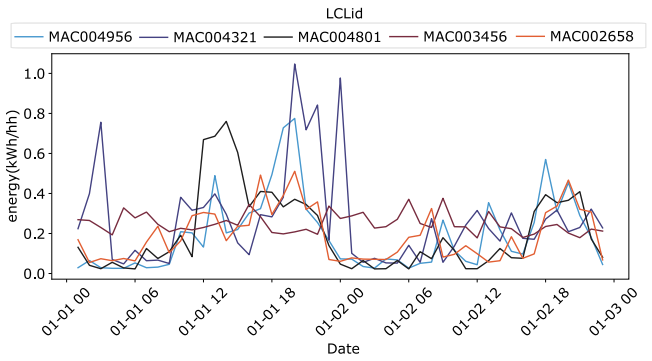


Fig. 1. Energy consumption (kWh/h) of 4 LCLIds from 01 January 2013 to 03 January 2013.

University of Luxembourg [78]. The simulations ran in an environment with 32 Intel Skylake cores and two NVIDIA Tesla V100 with 16 GB or 32GV depending on the allocation. We programmed the federation code in Python and based it on the machine learning framework provided by Tensorflow-Federated, [1] (TFF). The DL models are written in Keras [79].

3.2. Dataset

For our simulations, we used a large dataset from the Low Carbon London project, which was conducted by UK Power Networks between November 2011 and February 2014 in London, United Kingdom (herein LCL dataset) [80]. It contains the electrical consumption [kWh] data from 5567 households in a half-an hour resolution. The LCL dataset also contains a socio-technical classification of the households following the ACORN scheme [81] and is divided into individual household entries known as LCLid (Low Carbon London id).

To make the dataset ready for our simulations, we treated it in a 4-step procedure. First, we reduced the resolution of the LCL dataset to hourly values. The down-scaled values in the treated data set are the sum of two subsequent half-hour values in the original data set. This treatment significantly reduced the computational burden of our simulations. Secondly, we trimmed outliers or null values. Thirdly, we scaled all variables to have the same range using a Min–Max scaler. This re-scaling was necessary to ease the FL learning process as all values have to be in a known range, in our case: 0 to 1. Fourthly and finally, we split the dataset into a training and validation dataset. The training dataset (75%) contains electrical consumption data from January to December 2013 and the validation set (25%) covers data from January 2014 to March 2014. In Fig. 1, we provide an example of the processed data. It visualizes the electricity consumption [kWh] of 5 randomly selected households for a 2 day period using 1 h timestamps.

3.3. Evaluation metrics

Evaluation metrics offer an important means for the training and testing of forecasting models. However, the use of certain metrics can lead to undesirable results because FL models are known to converge to a *middle point* [82]. More specifically, FL models optimize the error of prediction with respect to the ground truth. In a distributed environment where there are *many such truths*, the models tend to minimize the mean of the loss across datasets. This tendency can provoke FL models to predict the average of each of the datasets and hence offer promising mean squared errors (MSE, Eq. (1)) and mean absolute errors (MAE, Eq. (2)). Such predictions, however, mean that the FL model did not learn local patterns in the data.

[1] https://github.com/tensorflow/federated

Therefore, MSE and MAE are typically not enough to evaluate the performance of a FL model and additional metrics, such as mean absolute percentage error (MAPE, Eq. (3)) and root mean square error (RMSE, Eq. (4)), are needed to quantify deviations of model predictions from the ground truths. The formal equations for these four metrics are as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - x_i\right)^2 \tag{1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - x_i\right| \tag{2}$$

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{x_i - y_i}{x_i}\right| \tag{3}$$

$$RMSE = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n}(y_i - x_i)^2} \tag{4}$$

4. Evaluation

4.1. Selection of a baseline neural network architecture

One crucial aspect for any AI method and specifically FL is the selection of the underlying NN architecture. To pick an architecture for our evaluation, we compared those in Table 1 that had a clear 'implementation guide' we could replicate. For this comparison, we used the metrics described in Section 3.3, trained the architectures with a maximum of 300 epochs on the training dataset and evaluated them on the evaluation dataset. We used the authors' codes where available and otherwise implemented the architecture ourselves. To limit computational costs, we used an early stopping mechanism for the training, that ended the training when the evaluation metrics did not improve over 10 epochs.

In Fig. 2, we illustrate the evaluation results for the twelve architectures we could replicate. Some architectures behaved worse on our dataset than on the dataset used by the respective authors. One possible reason for these differences could be scaling. Kim et al. [58,59], for instance, worked with a non-scaled dataset. This means that depending on the standard deviation of the dataset σ, the error metrics can differ substantially. For instance, the MSE scales proportionally with the standard deviation: $MSE_{scaled} = MSE_{non-scaled} * \sigma$. To avoid this scaling effect, we calculated all metrics using standardized data (Section 5).

Overall, the architectures in [33,34,42,54,61,63,66] had the lowest MAPE, from 6.7 to 7.1. From these, we selected Marino et al.'s [54] autoencoder architecture. Autoencoders are known to perform well even with non-idd data, so we selected the most performant autoencoder architecture among our shortlist of architectures. Marino et al.'s [54] architecture uses a 50-neuron encoder layer, a 12-neuron latent space, a 50-neurons decoder layer, and two final layers with 100 and 1 neurons respectively.

For our investigation of the effects of architectural complexity, we selected Khan et al.'s [14] architecture as it performed best among the more complex architectures in our sample. Khan et al.'s [14] architecture is different from Marino et al.'s [54] in that it uses convolutional layers and LSTM.

4.2. Fl, differential privacy and secure aggregation set-up

For our simulations, we selected Fed-Avg over Fed-SGD as it requires fewer communication rounds and has better performance [39,83]. Moreover, we used a horizontal FL configuration as our clients represent different LCLIds but share the same feature space.

To implement DP, we followed the steps proposed by McMahan et al. [84] rather than those of Chhachhi et al. [46] and Lu et al. [85],

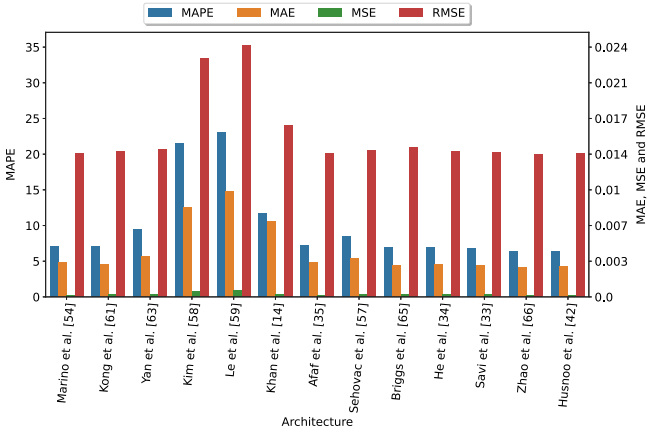


Fig. 2. RMSE, MSE, MAE, MAPE of the current literature applied to this paper's dataset.

in which noise is added to the dataset before the training. McMahan et al. [84] propose the central server to add noise after aggregating the updates of the model weights at every training round (in Fed-Avg). In other words, it differs from canonical Fed-Avg, which aggregates model weights.

The process proposed by McMahan et al. requires the definition of a query function sensitivity ($\mathbb{S}$) and a clipping strategy. The sensitivity of the query function determines the *actuation range* of the added noise. It represents the Euclidean distance between two datasets (C) differing in at most one element k: $\mathbb{S}(\tilde{f}) = max_{C,k} \left\| \tilde{f}(C \cup \{k\}) - \tilde{f}(c) \right\|_2$ [71]. Considering McMahan et al.'s first lemma [84] and assuming all clients are equally weighted, the sensitivity $\mathbb{S}$ is bounded as $\mathbb{S}(\tilde{f}(c)) \leq S/n$, with n being the number of clients. The vectors in Δ_k include the different model updates computed among the clients.

To bound the sensitivity of the query function, we needed to maintain the models' updates in a known range. One approach to ensure this range control is clipping model updates by a defined value before averaging. There are two strategies to clip the values of a neural network: 'per layer clipping', which applies clipping on a layer basis or 'flat clipping' which applies a clipping value to all the network parameters. Both clipping strategies project the values of the updates into a l2 sphere with the norm determined by the clipping value.

For both, per layer and flat clipping, there are two sub-strategies. One is to clip values using a fixed norm, known as fixed clipping. The second sub-strategy is called adaptive clipping [86]. It adapts the clipping norm based on a target quantile (i.e., 0.5) of the data distribution [86].

For the sake of simplicity, we used flat clipping as $\Delta'_k = \pi(\Delta_k, S)$ with S being the overall clipping value for the model updates. At the same time, we implemented both fixed and adaptive flat clipping strategies.

Once we had defined the query sensitivity and applied a flat clipping strategy, we evaluated how noise levels scale with the query sensitivity to obtain the minimum level of noise with a privacy guarantee. We added Gaussian noise as defined by: $N(0, \sigma^2)$ for $\sigma = z \cdot \mathbb{S}$, where z is the noise scale and $\mathbb{S}$ is the sensitivity of the query.

The addition of noise determines the overall privacy protection (ϵ) provided by DP. ϵ varies depending on the amount of noise added and the ratio of clients involved in the training (Q). Q is the ratio of clients selected out of the total which will participate in the next round of training. More noise naturally means more privacy and a lower ϵ. A higher Q, in turn, means less privacy and a higher ϵ [87].

To compute the privacy protection after a query, that is, each training round of our model, we used the privacy accountant provided by Renyi Differential Privacy (RDP) [88] as it provides a more detailed analysis of the privacy budget than the one created by [84].

For SecAgg, we used the implementation provided by Bonawitz et al. [74]. Their SecAgg implementation works as a plug-and-play algorithm that does not require any modification. We used SecAgg to ensure privacy-preserving communication between the central server and the clients. By using SecAgg in FL, clients can share their model weights without the central server or another client being able to reconstruct their weights [75].

Table 2
Scenarios considered.

Scenario	Privacy-Preserving technique	NN Architecture	Imposed correlation
0	–	Marino et al. [54]	✗
A	–	Marino et al. [54]	✗
B	–	Marino et al. [54]	✓
C	–	Khan et al. [14]	✗
D	Differential Privacy	He et al. [34]	✗
E	Secure Aggregation	Marino et al. [54]	✗

Table 3
Hyperparameters for scenarios A,B,C and E. Those marked with * the ones used in scenario 0.

Parameter	Value
Number of internal rounds before averaging	5
NN architecture	Marino et al. [54] * and Khan et al. [14]
Ratio of clients involved per round (Q)	1
Total number of clients (w)	Subject to federation size
Optimizer	Adam *
Optimizer learning rate (L_r)	0.01 *
Batch size	256 *
Number of communication rounds	300 *
Number of internal epochs after training	Not applicable

4.3. Model operation

In this subsection, we describe how we trained the FL models. For this training, we used 6 steps. We illustrate these steps as well as the additional step that FL-DP requires in Fig. 3. FL-SecAgg requires a different additional step, namely the initial sharing of public keys between the clients and central server. Fig. 3 does not illustrate this additional public key sharing.

In step 1, the central server initializes the model using Glorot initialization [89]. In step two, the central server shares the model with the participating clients. In step three, a subset of clients are selected based on the ratio (Q). Each of these clients in this sub-set then trains the received model on its data. In step four, clients send their model updates to the central server. In step five, the central server averages the aggregated updates and adds noise drawn from a Gaussian distribution in the case of DP (5' in Fig. 3). In step six, the central server returns the averaged updates to the clients. The central server and the clients repeated steps 2 to 6 until they reached 300 epochs.

4.4. Scenario design

Overall, we designed a set of six scenarios for our evaluation. Scenario 0 represents a hypothetical scenario in which all clients share their training data with the central server. This 'centralized setting' serves as a benchmark for the other scenarios. In Scenario A, we study the effects of moving from a centralized to a FL setting. In scenario B, we analyze the performance effect of clustering clients based on Pearson correlation. In scenario C, we evaluate the effect of a more complex NN architecture. Lastly, in Scenarios D and E, we study the effects of adding DP and SecAgg to the FL model. We summarize the specifications of the six scenarios in Table 2.

For scenarios 0, A, B, C and E, we ran eight simulations. These simulations evaluate the models' performance with a growing number of clients (federation size). We used the following eight federation sizes: 2, 5, 8, 11, 14, 17, 20, and 23 clients. Each of these clients worked with data from one LCLid. We had to limit the maximum number of clients

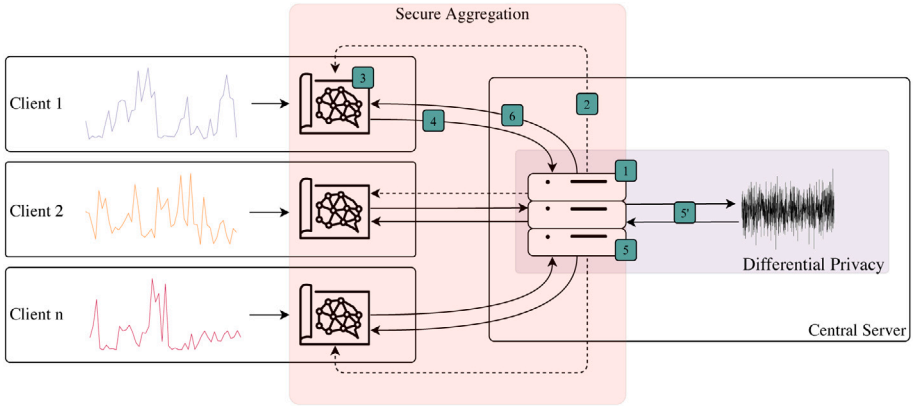


Fig. 3. Visual representation of our implementation of Federated Learning with privacy-preserving techniques.

Table 4
Hyperparameters for scenario D.

Parameter	Value
Number of internal rounds before averaging	5
NN Architecture	He et al. [34]
Ratio of clients involved per round (Q)	0.1
Total number of clients (w)	100
Optimizer	Adam
Optimizer learning rate (L_r)	0.01
Batch size	64
Number of communication rounds	100
Number of internal epochs after training	1

to 23 to control for computational cost as we simulated all clients and the communication between them in one virtual environment. In effect, every additional client did not add computational power but computational overhead.

We provide an overview of the hyperparameters for scenarios 0, A, B, C and E in Table 3. Table 4 provides the hyperparameters for the DP implementation in Scenario D.

5. Evaluation results

5.1. Scenario 0: Centralized setting

Scenario 0 analyzes the performance of a centralized setting, in which the clients send their data to a central server that trains a single model on the aggregated data. Scenario 0 uses the NN architecture presented by Marino et al. [54]. Similar to the architecture selection process, we employed an early stopper for Scenario 0 that terminated the training when there was no improvement in the validation metrics for more than 10 epochs.

In Table 5, we collect the simulation results for scenario 0. The MSEs, RMSEs and MAEs are expressed in absolute values, the MAPEs in percentage points, and the average training time per epoch in second [s]

Table 5 highlights that the overall performance of the centralized setting is very good, and that it remains almost constant for more than five clients with no evident variation in any of the metrics. The poor results in the two-client case could be the result of substantially different consumption patterns.

5.2. Scenario A: standard federated learning setting

We designed Scenario A to compare the 'centralized setting' in Scenario 0 with a FL setting, and to obtain a reference point for the other FL scenarios. Scenario A uses the NN architecture presented in [54] and does not apply privacy-preserving techniques. Furthermore, we did not impose data correlation among the clients.

Table 5
Validation error metrics and computation time for one-hour-ahead prediction: Scenario 0.

Central dataset size	MSE	RMSE	MAE	MAPE	Time per epoch [s]
2	0.00013	0.01158	0.00468	29.046	1.85
5	0.00012	0.01113	0.00308	9.068	6.01
8	0.00042	0.02067	0.00611	9.734	6.19
11	0.00028	0.01681	0.00437	8.561	8.18
14	0.00022	0.01514	0.00390	7.500	10.52
17	0.00023	0.01519	0.00383	6.850	12.56
20	0.00022	0.01498	0.00387	9.017	14.59
23	0.00019	0.01388	0.00330	7.144	16.82

Table 6
Validation error metrics and computation time for one-hour-ahead prediction: Scenario A.

Federation size	MSE	RMSE	MAE	MAPE	Time per round [s]
2	0.00015	0.01240	0.00516	30.1461	3.13
5	0.00022	0.01496	0.00468	16.2269	11.54
8	0.00058	0.02407	0.00745	11.9892	10.72
11	0.00042	0.02049	0.00538	10.1082	13.39
14	0.00035	0.01872	0.00542	10.1077	18.58
17	0.00032	0.01787	0.00469	8.5392	21.05
20	0.00031	0.01775	0.00479	11.2933	25.10
23	0.00028	0.01701	0.00478	10.8257	29.39

Table 6 presents the simulation results for Scenario A. The error metrics are expressed in absolute values and the average training time per epoch is expressed in seconds [s].

Table 6 highlights that performance of FL models varies depending on the federation size. While MSEs, MAEs and RMSEs remain almost constant, there is a clear improvement in MAPEs. These results are in line with those by Savi et al. [33] and Fekri et al. [39] and indicate that larger federation sizes lead to more accurate FL models.

To better illustrate this effect, we plot how the MAPEs evolved for the eight federation sizes along the training rounds in Fig. 4. Overall, we can observe a *quasi-exponential* decrease over the 300 rounds, approaching final values between 6.8 and 29, which indicate reasonably good forecasts [90].

In comparison to Scenario 0, we can observe an average performance decrease between 20% to 40%. FL appears to perform significantly worse than a 'centralized' setting, which is in line with other comparable studies [32,41,42].

Table 6 also highlights a trade-off between accuracy and computational time for federation size. As the number of clients increases, so does performance, but also computation time. This trade-off can present an important limitation for the use of FL.

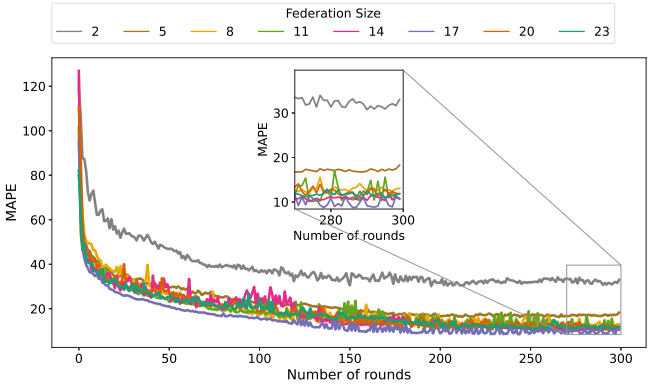


Fig. 4. Validation Mean Absolute Percentage Error (MAPE) per federation size in terms of training rounds for scenario A.

5.3. Scenario B: standard federated learning setting with imposed correlation

In scenario B, we analyzed the performance of a standard FL setting with imposed correlation among the clients in the federation. We followed Lee et al. [91] and used Pearson correlation to identify and bundle clients (or LCLids) by correlated data. This way of bundling differs from the dominant k-means approach in prior literature and offers a more direct and simple view of the correlation between clients. More specifically, we pre-filtered our dataset for specific ACORNs (H and L). For these ACORNS, we then calculated all possible non-repeated combinations and calculated their correlations. For each federation size, we selected those combinations of clients with the highest correlations.

We present the simulation results for Scenario B in Table 7. The error metrics and the correlation rate are both expressed in absolute values. We omit the computation time because it was basically the same as in scenario A 5.2.

FL with imposed correlation performed better in almost every metric than FL without imposed correlation (Scenario A). The MSEs decreased by an average 35.87%; RMSEs by 21.81%; MAEs by 25.57% and the MAPEs by 27.61%. They nevertheless still trail Scenario 0 by 6.35% on average. Moreover, these values are subject to some caveats. Our model with two clients had a correlation rate of 0.62, which led to a 75% better performance than the two-client case in Scenario A. Moreover, the performance of the model with 17 clients was worse than the same model in Scenario A, and 45% of the error metrics in Scenario B were better than those in scenario 0.

These results align well with similar studies, such as [34,36,39] or [33], where the application of k-means to cluster customers leads to performance improvements between 10% and 15%.

Overall, scenario B suggests that clustering based on Pearson correlation among the clients in a federation can substantially improve the performance of FL-based STLF. Specifically, utilities, energy providers, and DSOs could leverage simple socio-economic factors (ACORNS) and historical, individual smart meter data to cluster their residential customers into correlated groups. Each cluster can use a different FL model to reduce imbalance costs for inaccurate forecasts and offer tailored demand-side management programs.

5.4. Scenario C: standard federated learning setting with a more complex neural network architecture

In scenario C, we explore how a more complex NN architecture [14] impacts the performance of FL-based STLF. The motivation for scenario C is rooted in the trend to use ever more complex machine learning architectures in the hope of catching patterns invisible to less complex architectures. At the same time, it is unclear whether larger architectures increase performance.

Table 7
Validation error metrics and correlation rates for one-hour-ahead prediction: Scenario B.

Federation size	MSE	RMSE	MAE	MAPE	Correlation rate
2	0.00002	0.00463	0.00170	4.54	0.62
5	0.00015	0.01238	0.00373	9.77	0.51
8	0.00022	0.01513	0.00426	8.91	0.49
11	0.00021	0.01465	0.00402	8.23	0.45
14	0.00020	0.01429	0.00390	8.66	0.42
17	0.00032	0.01805	0.00465	8.22	0.37
20	0.00029	0.01726	0.00428	8.38	0.34
23	0.00026	0.01640	0.00432	9.95	0.31

Table 8
Validation error metrics and computation time for one-hour-ahead prediction: Scenario C.

Federation size	MSE	RMSE	MAE	MAPE	Time per round [s]
2	0.00024	0.01550	0.00720	31.50674	6.25
5	0.00052	0.02289	0.01282	33.42653	21.10
8	0.00117	0.03433	0.01754	20.92209	20.43
11	0.00115	0.03398	0.01495	21.93438	30.34
14	0.00087	0.02955	0.01404	18.44877	34.52
17	0.00077	0.02783	0.01080	13.80498	40.59
20	0.00081	0.02858	0.01435	24.28874	50.19
23	0.00061	0.02486	0.01059	19.02717	59.76

To account for the size of the model in [14] and its computational burden, we implemented three modifications to the set-up of our simulation environment. The first modification concerns the GPUs. For each of the Nvidia Tesla allocated on the HPC, we created two virtual cards, resulting in four cards we could use for our simulation. The second modification is related to the batch size, which we increased from 100 to 200. Increasing the batch size can help to prevent or limit overfitting since there are more data entries available to compute the loss of the model. Finally, we modified the model in [14] by transforming the initially proposed LSTM layers to CuDNNLSTM [92]. The transformation enabled the LSTMs to use the Compute Unified Device Architecture (CUDA) kernel of our Tesla GPUs.

The simulation results of scenario C are presented in Table 8. The results clearly indicate the increased computational costs of training a FL model with a complex architecture. The computational time is almost twice as high as in scenarios A and B. On the other hand, the performance of the model with the more complex architecture was worse that of the smaller model's for all federation sizes and all metrics, ranging from 50% up to 142%.

These results suggest a clear case of overfitting. Overfitting is generally defined as the lack of generalization of a model. An overfitted model crosses the line between learning tendencies or patterns and *memorizing* the data received as input.

Fig. 5 provides a visualization of this overfitting. The performance on the training subset is represented by the solid lines, while the performance on the validation subset is visualized by the dotted lines. The dotted lines begin to increase again after round 120, whereas the solid lines decrease as the model is over-fitted to the training data. .

In effect, scenario C offers a cautionary tale for utilities, energy providers, and DSOs that want to use FL for short-term load forecasting. Not only are more complex FL architectures more expensive and detrimental to the environment [93], they are also more sensitive to handle.

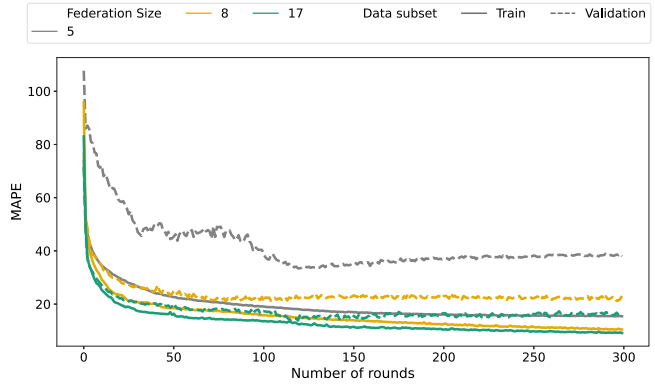


Fig. 5. Validation and Training MAPEs for federation sizes 5,8, and 17 in Scenario C.

5.5. Scenario D: privacy-preserving federated learning setting with differential privacy

Scenario D focuses on adding DP to FL and how this impacts the performance of FL-based STLF. Furthermore, we compare two *flat clipping* approaches: fixed and adaptive clipping, as described in Section 4.2.

In scenarios A and B, we used Marino et al.'s model [54] as the baseline architecture. Encoder–decoder architectures can cope well with outliers due to their capacity to abstract information into the latent space. This capacity is very beneficial for FL where different clients can have substantially different data points. However, we found that these architectures are substantially more vulnerable to noise than standard stacked LSTM networks. One reason for this vulnerability could be that they compact information from a higher dimensional space into a smaller one. Adding noise to the weights of this latent space will have a multiplicative effect on the model's output in the decoder phase. To avoid such encoder–decoder noise problems for our DP simulation, we changed the architecture in Scenario D to a two-layer LSTM with 50 neurons each, and a final dense layer as in He et al. [34].

DP offers two approaches to obtain a high privacy budget given a defined amount of noise: reduce the ratio of clients that participate in each training round (Q), retrain the model locally for several epochs on client data, find a lower δ, and/or increase the noise scale (z). For Scenario D, we employed a ratio of $Q = 0.1$. With $Q = 0.1$, a total of 100 clients and without the addition of privacy preserving techniques, our model had a MAE of 0.00300, a MSE of 0.012, a RMSE of 0.01114, and a MAPE of 8.3846, which matches results in Scenario A.

Moreover, we considered recommendations by Zhao et al. [66] and Xu et al. [38] to introduce local re-training. Specifically, they propose to conduct several local training rounds on each client between each aggregation with DP to better fit the local models. Yet, we found that these repeated rounds did not improve performance so we chose to use just one local training round. However, we did optimize the δ to $\delta = 4e^{-3}$ as proposed by Zhao et al. [66].

The first strategy we implemented was fixed clipping following the two main steps in McMahan et al. [84]. In the first step, we determined the lowest possible clipping value (S) as being too low clipping values can negatively affect the convergence rate as they clip all values bigger than S. We treated S as a hyper-parameter and used an iterative approach to find the lowest possible clipping value. Specifically, we followed McMahan et al. [84] and used iterative steps of 0.1 for S, starting with $S = 0.1$ until $S = 0.7$. We present the error metrics for the different S values in Table 9.[2]

[2] Setting a fixed value for the clipping slows the training process significantly. The values in Table 9 are the validation metrics after 2000 communication rounds. Without any clipping strategy, the models converge at an earlier rate (see Fig. 4).

Table 9
Validation error metrics for different clipping values for one-hour-ahead prediction with the sample client ratio $Q = 0.1$ and total number of clients $w = 100$: Scenario D.

S	MSE	RMSE	MAE	MAPE
0.10	0.00043	0.02094	0.00628	10.69357
0.20	0.00035	0.01884	0.00502	8.89023
0.30	0.00038	0.01969	0.00496	8.00244
0.40	0.00038	0.01963	0.00486	7.71642
0.50	0.00039	0.01978	0.00493	7.92688
0.60	0.00034	0.01869	0.00477	7.81763
0.70	0.00036	0.01915	0.00484	7.53057

Table 10
Exploration of the different noise levels, in bold the hyper-parameter **z** that defines the amount of noise.

Qw	S	$\mathbb{S} = S/Qw$	z	$\sigma = z \cdot S$
10	0.3	0.03	0.1	0.003
10	0.3	0.03	0.2	0.006
10	0.3	0.03	0.3	0.009
10	0.3	0.03	0.4	0.012
10	0.3	0.03	0.5	0.015
10	0.3	0.03	0.6	0.018
10	0.3	0.03	0.7	0.021
10	0.3	0.03	0.8	0.024
10	0.3	0.03	0.9	0.027

Based on these iterations, we selected $S \approx 0.3$ as our fixed clipping value. It is the lowest clipping value with comparatively good error metrics and the marginal increase in error metrics from lowering S increases disproportionately below ≈ 0.3.

Once we had identified the lowest possible clipping value S, the second step was to identify a tolerable level of noise. With $S = 0.3$, a total number of clients $w = 100$, and $Q = 0.1$, we applied $\mathbb{S} = S/Qw$ to calculate the standard deviation of the noise level $\sigma = z \cdot \mathbb{S}$. Similarly with the approach that we took with S, we treated z as a hyper-parameter and ranged it from 0.1 to 0.9

In Table 10, we present the performance metrics for each of the z variations. Each of the explored z values represents a different level of noise added to the federated model. Intuitively, there is a trade-off between the amount of noise and performance, whereby more noise (increase in z) reduces performance. This trade-off dynamic is clear from the error metrics in Table 11. Nevertheless, the overall error metrics for DP based on fixed clipping are generally low and indicate good forecasting performance.

Concurrently, more noise also means better privacy, as indicated by the increasing privacy guarantees in column three of Table 11. We calculated these guarantees using the Rényi Differential Privacy Accountant [88]. The highest amount of noise we examined (z=0.9) provides a privacy guarantee of (4.2, $4e^{-3}$), which is close to perfect privacy ($\epsilon = 0$). In effect, scenario D demonstrates that adding DP to FL maintains comparatively good performance and offers high privacy guarantees.

The second clipping strategy that we analyzed is adaptive clipping. With adaptive clipping, clipping value are calculated automatically. To evaluate this approach, we used Andrew et al.'s adaptive clipping implementation [86], in which the algorithm iteratively (per communication round) adjusts the norm clip, trying to approximate it to a predefined quantile (0.5 in our case).

This data quantile approximation expends privacy budget as it queries the data. To prevent this *privacy leakage* Andrew et al. [86] propose to add noise during the approximation. This noise (σ_b) is defined by 0.05 times the number of clients per round, in our case $\sigma_b = 0.5$. This addition of noise has a slight affect on the total privacy guarantee of the model. It results in increased effective noise as $z_\Delta = (z^{-2} - (2\sigma_b)^{-2})^{-1/2}$.

Fig. 6 highlights the adaptive adjustments of the clipping value over the training rounds. There is a sharp increase in the clipping norm at

Table 11
Validation error metrics with $S = 0.3$ and a varying noise scale **z** from 0.1 to 0.9 for one hour-ahead-prediction with the sample client ratio $Q = 0.1$ and total number of clients $w = 100$ after one epoch of local training.

Noise scale (**z**)	Privacy guarantee (ϵ, δ)	MSE	RMSE	MAE	MAPE	Timer per round [s]
0.1	(911, $4e^{-3}$)	0.00010	0.00946	0.00272	7.5426	86.74
0.2	(190, $4e^{-3}$)	0.00010	0.00957	0.00312	8.8930	85.11
0.3	(69.3, $4e^{-3}$)	0.00010	0.00959	0.00309	8.4391	87.48
0.4	(32.4, $4e^{-3}$)	0.00010	0.00962	0.00321	9.1156	84.66
0.5	(17.9, $4e^{-3}$)	0.00011	0.00971	0.00340	9.7164	88.52
0.6	(11.2, $4e^{-3}$)	0.00011	0.00972	0.00344	9.9693	84.28
0.7	(7.58, $4e^{-3}$)	0.00011	0.00979	0.00354	10.0378	81.46
0.8	(5.5, $4e^{-3}$)	0.00013	0.01075	0.00519	15.6755	82.08
0.9	(4.2, $4e^{-3}$)	0.00011	0.00991	0.00372	10.6031	87.48

Table 12
Validation error metrics with adaptive clipping at different noise levels from 0.1 to 0.9 using as initial clipping value $C^0 = 0.1$ and the step factor for the geometric updates $\eta C = 0.2$ for one hour ahead prediction with the sample client ratio $Q = 0.1$ and total number of clients $w = 100$ after one epoch of local training.

Noise scale (**z**)	Effective noise (z_Δ)	Privacy guarantee (ϵ, δ)	MSE	RMSE	MAE	MAPE	Time per round [s]
0.1	0.100	(910.0, $4e^{-3}$)	0.00010	0.00936	0.00276	7.9966	84.39
0.2	0.200	(189.4, $4e^{-3}$)	0.00010	0.00930	0.00260	7.3866	88.41
0.3	0.300	(68.7, $4e^{-3}$)	0.00009	0.00930	0.00257	7.0985	85.30
0.4	0.402	(31.9, $4e^{-3}$)	0.00010	0.00945	0.00292	8.2810	86.92
0.5	0.504	(17.5, $4e^{-3}$)	0.00010	0.00948	0.00301	9.0461	88.57
0.6	0.607	(10.8, $4e^{-3}$)	0.00010	0.00955	0.00302	8.8343	86.27
0.7	0.711	(7.2, $4e^{-3}$)	0.00010	0.00961	0.00317	9.4312	87.68
0.8	0.817	(5.2, $4e^{-3}$)	0.00010	0.00955	0.00325	9.6126	88.27
0.9	0.924	(3.9, $4e^{-3}$)	0.00010	0.00955	0.00319	9.2953	87.93

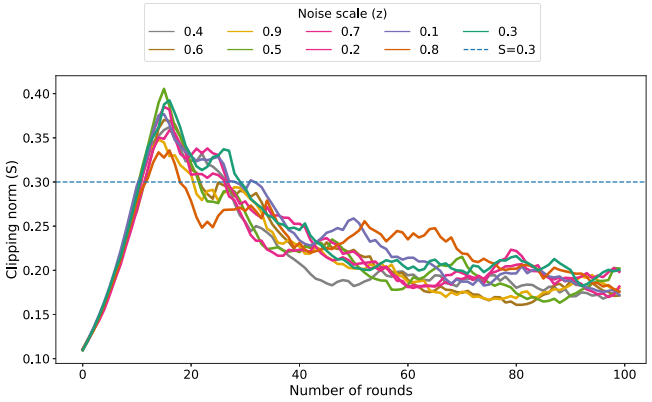


Fig. 6. Evolution of the adaptive clipping norm at different noise levels z (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9) using as initial clipping value $C^0 = 0.1$ and the step factor for the geometric updates $\eta C = 0.2$.

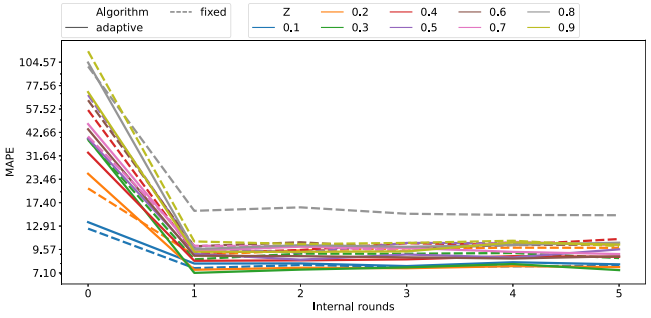


Fig. 7. Validation Mean Absolute Percentage Error (MAPE) per local training epoch for adaptive and fixed DP.

the beginning of the training rounds due to the low initial clipping value $C^0 = 0.1$. Such a low quantile allows only a few data points to participate in selecting the clipping value. The smaller the quantile, the fewer data points participate and thus, it is more difficult to estimate the optimal clipping value.

As in our case, the adaptive clipping algorithm may overshoot as a result and increase the clipping norm to higher values. After this overshot, the adaptive clipping algorithm correctly approximates the optimal clipping value $S \approx 0.2$.

We present the simulation results for adaptive clipping in Table 12. On average, adaptive clipping outperformed fixed clipping by 9%. Moreover, the privacy guarantee is close to perfect privacy (3.9, $4e^{-3}$)

Adaptive clipping appears not only more attractive from a performance and privacy perspective. It is also easier to use in terms of performance and privacy. Fixed clipping requires an initial and computationally expensive manual step to identify an appropriate clipping value, whereas, in adaptive clipping, this value is calculated automatically in the training rounds. Thus, DP with adaptive clipping presents the more convenient choice for residential STLF.

The results we present in Tables 11 and 12 are those after the local training round suggested by Zhao et al. [66]. Unlike Zhao et al. [66], who worked with five local training round, we used only one as additional rounds did not significantly improve performance (Fig. 7). Nevertheless, clients profited from local training with negligible computational overhead.

5.6. Scenario E: privacy-preserving federated learning setting with secure aggregation

In this scenario, we examine SecAgg as an alternative technique to add privacy to FL. Whereas DP adds random noise to model updates, SecAgg targets the communication and aggregation of the clients' model updates. Hence, there is no trade-off as in scenario D, where it is important to find an adequate noise level.

Similar to scenarios A, B and C, we present the simulation results for the eight federation sizes in Table 13. We express the error metrics in absolute values and the average computation time in seconds [s].

Table 13
Error metrics and computation time for one-hour-ahead prediction using SecAgg: Scenario E on test set.

Federation size	MSE	RMSE	MAE	MAPE	Time per round [s]
2	0.00017	0.01324	0.00532	31.01177	4.54
5	0.00018	0.01348	0.00431	15.60893	13.23
8	0.00060	0.02457	0.00759	12.28532	13.34
11	0.00039	0.01996	0.00523	9.65965	18.21
14	0.00034	0.01864	0.00503	9.67057	22.25
17	0.00033	0.01820	0.00466	8.25973	26.70
20	0.00033	0.01836	0.00522	12.88359	34.64
23	0.00028	0.01683	0.00453	10.19247	38.10

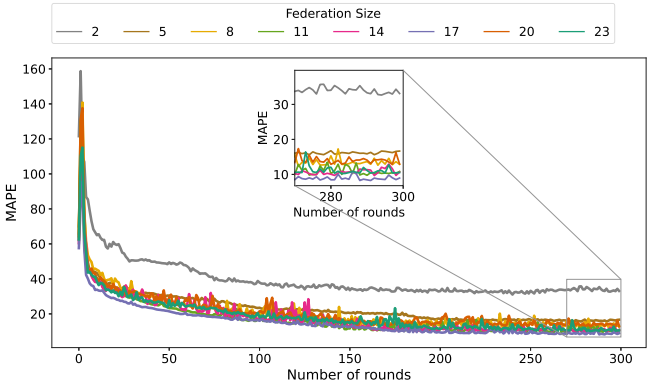


Fig. 8. Validation Mean Absolute Percentage Error (MAPE) per LCLids federation size in terms of training rounds for Scenario E.

Furthermore, we complement the results with Fig. 8. It depicts the MAPE, following a similar curve as in Scenario A.

Table 13 shows that the use of SecAgg affects computation time only marginally. As SecAgg does not add any noise, it also provides less burden than DP. Consequently, SecAgg presents a more performant alternative for residential STLF with the cost of an extra 30% of computation time. However, it is important to note that SecAgg does not provide complete privacy because latent patterns could still point toward the original data subject. More specifically, Model Inversion (MI) attacks could reconstruct the original training data from the model parameters [94].

5.7. Comparison across the scenarios

We summarize our results for scenarios 0, A, B, C, and E in Figs. 9 and 10. We omitted scenario D from these figures because in scenario D we only varied the noise scale and not the federation size.

Overall, the two figures suggest an inherent trade-off between performance and privacy in residential STLF. Yet, FL models can successfully mediate this trade-off and provide high levels of performance and privacy, especially when trained on correlated data, avoid unduly complex architectures, and employ SecAgg.

6. Conclusions

This paper analyses the use of FL and its combination with privacy preserving techniques for short-term forecasting of individual residential loads. Such a combination offers an innovative approach to accommodate both accuracy and privacy. In particular, it allows those who depend on accurate forecasts of residential loads (such as utilities, energy providers, and DSOs) to train in a collaborative fashion forecasting models with granular smart meter data without having to share this data.

Our analysis builds on historical smart meter data and consists of six scenarios. While the first two scenarios set the baseline scenarios,

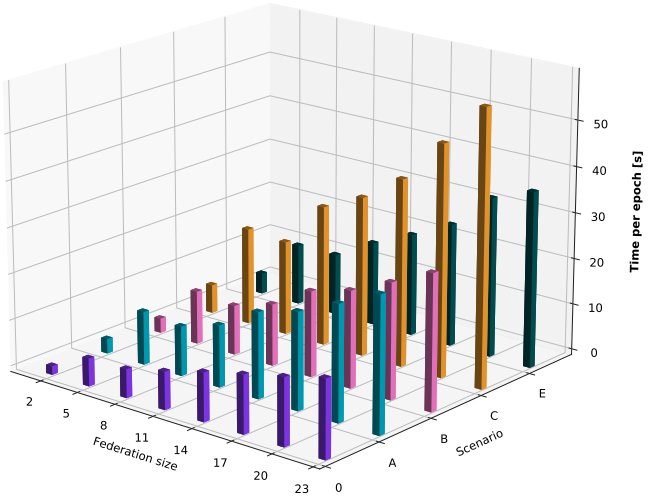
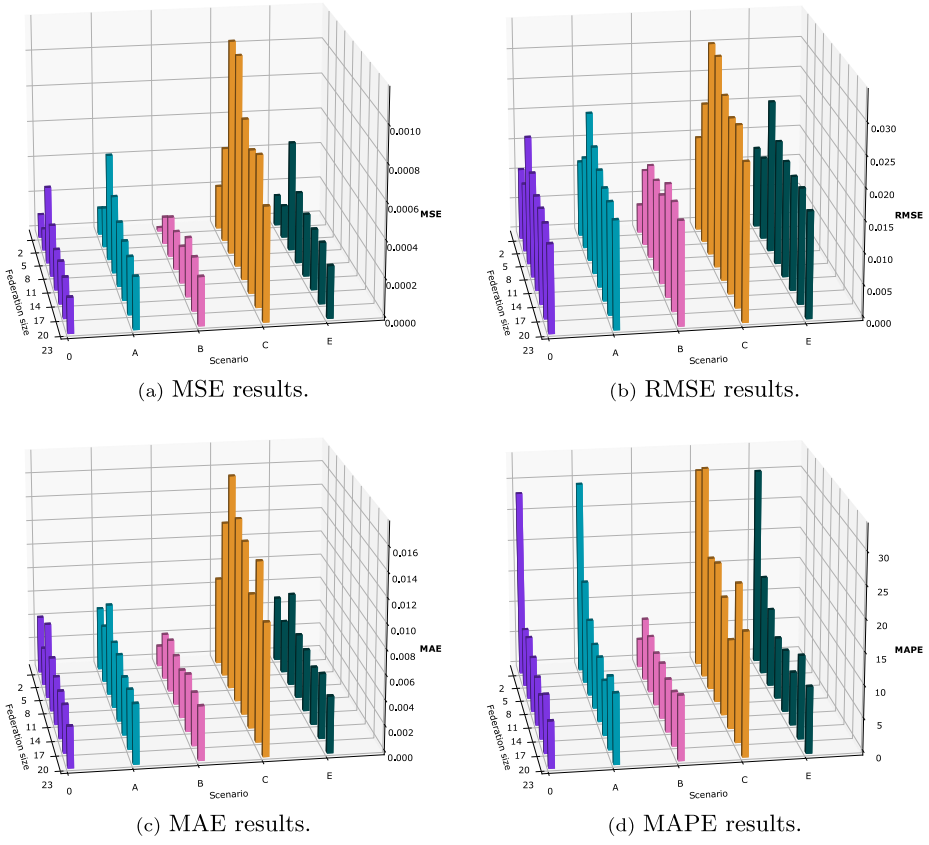


Fig. 9. Comparison of computation time across Scenarios 0, A, B, C, and E.

each of the subsequent four scenarios have a particular analytical focus. Specifically, these scenarios investigate the effects of data correlation, neural network architecture complexity, differential privacy, and secure aggregation on performance, computation time, and privacy guarantee levels. In each scenario, we also explore the effects of different federation sizes. From our analysis, we can posit the following:

1. Collaborative training of AI models with federated learning reduces forecasting accuracy as compared to a 'centralized' setting. However, it makes it easier to account for data privacy concerns through the addition of privacy-preserving techniques.
2. As the number of participating clients (smart meters) in a federation increases, forecasting accuracy tends to also increase. However, while a greater number of clients leads to greater accuracy, this also implies higher computational costs that may no always be justified.
3. Customer segmentation with Pearson correlation along socio-economic factors (e.g., with the ACORN methodology) substantially improves forecasting accuracy for FL models.
4. Complex neural network architectures imply high computational costs, difficulties in handling the architecture, and a potential risk of overfitting. It is thus important to balance accuracy and usability when selecting of model architectures.
5. Complementing federated learning with differential privacy or secure aggregation does not significantly reduce forecasting accuracy but does enable very high levels of privacy.
6. Adaptive and fixed clipping approaches to differential privacy provides similar performance. Adaptive clipping is easier to use as it does not require manual pre-selection of good clipping values, and it facilitates faster model convergence.
7. Combining autoencoder architectures with DP complicates the training of FL models. The design of these architectures magnifies the noise added by DP, which restricts the training process.
8. Secure aggregation is superior to DP in terms of usability, performance and computational burden. It can be added as a simple plug-and-play component, does not reduce performance by adding noise, and permits faster training.

Overall, our analysis suggests that a combination of federated learning with privacy-preserving techniques can be a highly promising alternative for residential short-term load forecasting. However, is not free from technical challenges. Differential privacy requires careful configuration of noise size, clipping values and client ratios to balance accuracy and privacy. Secure aggregation does not require such configuration but its cryptographic set-up can also be challenging as well.



(a) MSE results. (b) RMSE results.

(c) MAE results. (d) MAPE results.

Fig. 10. Comparison of evaluation metrics across Scenarios 0, A, B, C, and E.

Furthermore, computational costs limit the number of clients that can be used for training.

More broadly, our study contributes to a better understanding of the use of FL and privacy-preserving techniques for residential short-term load forecasting. It makes an important contribution to the growing literature on the applications of federated learning in electric power systems by testing different NN under distributed settings, examining the implications of privacy preserving techniques, and identifying technical challenges in using FL.

Naturally, our analysis is not free from limitations. In particular, computational costs have considerably limited the size of our federations. Even though larger federation sizes may result in somewhat different results, nevertheless we believe that our overall results are robust, as we have explored several settings in terms of: number of clients, baseline NN architectures, and dataset characteristics.

Further research may nevertheless want to (1) assess larger federation size settings with additional correlation indicators, such as the existence of distributed energy resources (i.e., photovoltaics, electric vehicles, or home energy management systems), (2) investigate data input disruptions produced by hostile agents or errors caused by malfunctions of a smart metering device, and (3) examine other, innovative NN architectures with attention mechanisms and multi-variate input data. After all, FL is highly collaborative and iterative and perfect data and operation may not always be possible in real-world applications.

CRediT authorship contribution statement

Joaquín Delgado Fernández: Conceptualization, Methodology, Data curation, Writing – original draft, Software, Writing – review & editing, Visualization. **Sergio Potenciano Menci:** Conceptualization, Methodology, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Chul Min Lee:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Alexander Rieger:** Writing – review & editing, Supervision. **Gilbert Fridgen:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

We would like to acknowledge Tom Josua Barberau and Orestis Papageorgiou for their valuable feedback on the first draft of this paper. Moreover, we would like to acknowledge the support of the European Union (EU) within its Horizon 2020 programme, project MDOT (Medical Device Obligations Taskforce), Grant agreement 814654. Additionally, this project was supported by the Kopernikus-project "SynErgie" of the German Federal Ministry of Education and Research (BMBF), by PayPal and the Luxembourg National Research Fund FNR (P17/IS/13342933/PayPal-FNR/Chair in DFS/Gilbert Fridgen) as well as by the Luxembourg National Research Fund (FNR) -FiReSpARX Project, ref. 14783405. All authors have read and agreed to the published version of the manuscript.

References

[1] Nti IK, Teimeh M, Nyarko-Boateng O, Adekoya AF. Electricity load forecasting: a systematic review. J Electr Syst Inf Technol 2020;7(1):13. http://dx.doi.org/10.1186/s43067-020-00021-8.

[2] Petropoulos F, Apiletti D, Assimakopoulos V, Babai MZ, Barrow DK, Taieb SB, et al. Forecasting: theory and practice. Int J Forecast 2022;38(3):705–871. http://dx.doi.org/10.1016/j.ijforecast.2021.11.001.

[3] ENTSO-E. Enhanced load forecasting. 2021, URL https://www.entsoe.eu/Technopedia/techsheets/enhanced-load-forecasting.

[4] Muñoz A, Sánchez-Úbeda EF, Cruz A, Marín J. Short-term forecasting in power systems: A guided tour. In: Handbook of power systems II. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010, p. 129–60. http://dx.doi.org/10.1007/978-3-642-12686-4_5.

[5] Lusis P, Khalilpour KR, Andrew L, Liebman A. Short-term residential load forecasting: Impact of calendar effects and forecast granularity. Appl Energy 2017;205:654–69. http://dx.doi.org/10.1016/j.apenergy.2017.07.114.

[6] Commission for Regulation of Utilities. Energy supply costs information paper. Tech. rep., (CRU17291). Commission for Regulation of Utilities - CRU; 2017, URL https://www.cru.ie/wp-content/uploads/2017/10/CRU17291-RFI-Information-paper.pdf [Accessed on 16 Feb 2022].

[7] Specht JM, Madlener R. Energy supplier 2.0: A conceptual business model for energy suppliers aggregating flexible distributed assets and policy issues raised. Energy Policy 2019;135:110911. http://dx.doi.org/10.1016/j.enpol.2019.110911.

[8] Maltais L-G, Gosselin L. Forecasting of short-term lighting and plug load electricity consumption in single residential units: Development and assessment of data-driven models for different horizons. Appl Energy 2021;118229. http://dx.doi.org/10.1016/j.apenergy.2021.118229.

[9] Wang Y, Chen Q, Hong T, Kang C. Review of smart meter data analytics: Applications, methodologies, and challenges. IEEE Trans Smart Grid 2019;10(3):3125–48. http://dx.doi.org/10.1109/TSG.2018.2818167.

[10] Keramidas K, Diaz Vazquez A, Weitzel M, Vandyck T, Tamba M, Tchung-Ming S, et al. Global energy and climate outlook 2019: Electrification for the low carbon transition. Publications Office of the European Union, Joint Research Center: Luxembourg; 2020, URL https://op.europa.eu/s/wWQS.

[11] International Energy Agency. Electricity final consumption by sector, wolrd. IEA; 2021, URL https://www.iea.org/fuels-and-technologies/electricity (Accessed on 21 June 2021).

[12] Bielecki S, Skoczkowski T, Sobczak L, Buchoski J, Maciąg L, Dukat P. Impact of the lockdown during the COVID-19 pandemic on electricity use by residential users. Energies 2021;14(4). http://dx.doi.org/10.3390/en14040980.

[13] Kaur H, Ahuja S. Time series analysis and prediction of electricity consumption of health care institution using ARIMA model. In: Proceedings of sixth international conference on soft computing for problem solving. Springer; 2017, p. 347–58. http://dx.doi.org/10.1007/978-981-10-3325-4_35.

[14] Khan ZA, Hussain T, Ullah A, Rho S, Lee M, Baik SW. Towards efficient electricity forecasting in residential and commercial buildings: A novel hybrid CNN with a LSTM-AE based framework. Sensors 2020;20(5). http://dx.doi.org/10.3390/s20051399.

[15] Kalimoldayev M, Drozdenko A, Koplyk I, Marinich T, Abdildayeva A, Zhukabayeva T. Analysis of modern approaches for the prediction of electric energy consumption. Open Eng 2020;10(1):350–61. http://dx.doi.org/10.1515/eng-2020-0028.

[16] Alfares HK, Nazeeruddin M. Electric load forecasting: Literature survey and classification of methods. Internat J Systems Sci 2002;33(1):23–34.

[17] Hippert H, Pedreira C, Souza R. Neural networks for short-term load forecasting: a review and evaluation. IEEE Trans Power Syst 2001;16(1):44–55. http://dx.doi.org/10.1109/59.910780.

[18] Ardabili S, Mosavi A, Várkonyi-Kóczy AR. Advances in machine learning modeling reviewing hybrid and ensemble methods. 2019, http://dx.doi.org/10.1007/978-3-030-36841-8_21.

[19] Negnevitsky M, Mandal P, Srivastava AK. An overview of forecasting problems and techniques in power systems. In: 2009 IEEE power energy society general meeting. 2009, p. 1–4. http://dx.doi.org/10.1109/PES.2009.5275480.

[20] Rashed Mohassel R, Fung A, Mohammadi F, Raahemifar K. A survey on advanced metering infrastructure. Int J Electr Power Energy Syst 2014;63:473–84. http://dx.doi.org/10.1016/j.ijepes.2014.06.025.

[21] Hinterstocker M, Schott P, von Roon S. Disaggregation of household load profiles. Vienna: Internationale Energiewirtschaftstagung; 2017, URL https://www.fim-rc.de/Paperbibliothek/Veroeffentlicht/788/wi-788.pdf.

[22] McKenna E, Richardson I, Thomson M. Smart meter data: Balancing consumer privacy concerns with legitimate applications. Energy Policy 2012;41:807–14. http://dx.doi.org/10.1016/j.enpol.2011.11.049.

[23] Kowarik A, Stolze P, Grondal O, Ilves M, Kirt T, Jansson I, et al. Report on data access and data handling. Tech. rep., ESSnet Big Data; 2016, URL https://ec.europa.eu/eurostat/cros/content/WP3_Report_1_1_en.

[24] European Commission, Directorate-General for Energy, Küpper G, Cavarretta M, Ehrenmann A, Naffah E, Szilagyi J, et al. Format and procedures for electricity (and gas) data access and exchange in member states. Publications Office; 2020, http://dx.doi.org/10.2833/719689.

[25] Haney A, Jamasb T, Pollitt M. Smart metering and electricity demand: Technology, economics and international experience. In: Faculty of economics. Cambridge working papers in economics, University of Cambridge; 2009, URL https://www.cambridge.org/core/books/future-of-electricity-demand/smart-metering-technology-economics-and-international-experience/82C8C2C82CFE1FE9D513CFB17F356C9C.

[26] Atrias. The central hub in providing information in the energy market. 2021, URL https://www.atrias.be/ [Accessed: 28 May 2021].

[27] Elhub. Netural data hub fir metering data and market processes.. 2021, URL https://elhub.no/en/# [Accessed: 28 May 2021].

[28] McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. PMLR; 2017, p. 1273–82, URL https://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf.

[29] Konečný J, McMahan HB, Ramage D, Richtárik P. Federated optimization: Distributed machine learning for on-device intelligence. 2016, ArXiv, arXiv:1610.02527.

[30] Zhu L, Liu Z, Han S. Deep leakage from gradients. In: Wallach H, Larochelle H, Beygelzimer A, dÁlché Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems, Vol. 32. Curran Associates, Inc.; 2019, URL https://proceedings.neurips.cc/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf.

[31] Geiping J, Bauermeister H, Dröge H, Moeller M. Inverting gradients-how easy is it to break privacy in federated learning? Adv Neural Inf Process Syst 2020;33:16937–47, URL https://proceedings.neurips.cc/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf.

[32] Briggs C, Fan Z, Andras P. Federated learning for short-term residential energy demand forecasting. 2021, ArXiv, arXiv:2105.13325.

[33] Savi M, Olivadese F. Short-term energy consumption forecasting at the edge: A federated learning approach. IEEE Access 2021;9:95949–69. http://dx.doi.org/10.1109/ACCESS.2021.3094089.

[34] He Y, Luo F, Ranzi G, Kong W. Short-term residential load forecasting based on federated learning and load clustering. In: 2021 IEEE international conference on communications, control, and computing technologies for smart grids. 2021, p. 77–82. http://dx.doi.org/10.1109/SmartGridComm51999.2021.9632314.

[35] Taïk A, Cherkaoui S. Electrical load forecasting using edge computing and federated learning. In: ICC 2020-2020 IEEE international conference on communications. IEEE; 2020, p. 1–6. http://dx.doi.org/10.1109/ICC40277.2020.9148937.

[36] Biswal MK, Tayeen ASM, Misra S. AMI-FML: A privacy-preserving federated machine learning framework for AMI. 2021, ArXiv, arXiv:2109.05666.

[37] Li J, Ren Y, Fang S, Li K, Sun M. Federated learning-based ultra-short term load forecasting in power internet of things. In: 2020 IEEE international conference on energy internet. 2020, p. 63–8. http://dx.doi.org/10.1109/ICEI49372.2020.00020.

[38] Xu Y, Jiang C, Zheng Z, Yang B, Zhu N. LSTM short-term residential load forecasting based on federated learning. In: 2021 international conference on mechanical, aerospace and automotive engineering. 2021, http://dx.doi.org/10.1145/3518781.3519181.

[39] Fekri MN, Grolinger K, Mir S. Distributed load forecasting using smart meter data: Federated learning with recurrent neural networks. Int J Electr Power Energy Syst 2021;107669. http://dx.doi.org/10.1016/j.ijepes.2021.107669.

[40] Shi Y, Xu X. Deep federated adaptation: An adaptative residential load forecasting approach with federated learning. Sensors (Basel, Switzerland) 2022;22. http://dx.doi.org/10.3390/s22093264.

[41] Lin J, Ma J, Zhu J. Privacy-preserving household characteristic identification with federated learning method. IEEE Trans Smart Grid 2022;13:1088–99. http://dx.doi.org/10.1109/TSG.2021.3125677.

[42] Husnoo MA, Anwar A, Hosseinzadeh N, Islam SN, Mahmood AN, Doss RRM. FedREP: Towards horizontal federated load forecasting for retail energy providers. 2022, ArXiv, arXiv:2203.00219.

[43] Khalil M, Esseghir M, Merghem L. Federated learning for energy-efficient thermal comfort control service in smart buildings. In: 2021 IEEE global communications conference. 2021, p. 01–6. http://dx.doi.org/10.1109/GLOBECOM46510.2021.9685286.

[44] Barbosa P, Brito A, Almeida H. A technique to provide differential privacy for appliance usage in smart metering. Inform Sci 2016;370–371:355–67. http://dx.doi.org/10.1016/j.ins.2016.08.011.

[45] Eibl G, Engel D. Differential privacy for real smart metering data. Comput Sci - Res Dev 2017;32. http://dx.doi.org/10.1007/s00450-016-0310-y.

[46] Chhachhi S, Teng F. Market value of differentially-private smart meter data. In: 2021 IEEEpower energy society innovative smart grid technologies conference. 2021, p. 1–5. http://dx.doi.org/10.1109/ISGT49243.2021.9372228.

[47] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. ACM Trans Intell Syst Technol (TIST) 2019;10(2):1–19. http://dx.doi.org/10.1145/3298981.

[48] Xian X, Wang X, Ding J, Ghanadan R. Assisted learning: A framework for multi-organization learning. Adv Neural Inf Process Syst 2020;33:14580–91, URL https://papers.nips.cc/paper/2020/file/a7b23e6eefbe6cf04b8e62a6f0915550-Paper.pdf.

[49] Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, et al. Advances and open problems in federated learning. Found Trends® Mach Learn 2021;14(1–2):1–210. http://dx.doi.org/10.1561/2200000083.

[50] McMahan HB, Moore E, Ramage D, y Arcas BA. Federated learning of deep networks using model averaging. 2016, CoRR, abs/1602.05629 arXiv:1602.05629.

[51] Li C. Designing a short-term load forecasting model in the urban smart grid system. Appl Energy 2020;266:114850. http://dx.doi.org/10.1016/j.apenergy.2020.114850.

[52] Vos M, Bender-Saebelkampf C, Albayrak S. Residential short-term load forecasting using convolutional neural networks. In: 2018 IEEE international conference on communications, control, and computing technologies for smart grids, smartgridcomm 2018. 2018, http://dx.doi.org/10.1109/SmartGridComm.2018.8587494.

[53] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[54] Marino DL, Amarasinghe K, Manic M. Building energy load forecasting using deep neural networks. In: IECON 2016-42nd annual conference of the IEEE industrial electronics society. IEEE; 2016, p. 7046–51. http://dx.doi.org/10.1109/IECON.2016.7793413.

[55] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Adv Neural Inf Process Syst 2014;27. URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

[56] Cho K, van Merriënboer B, Bahdanau D, Bengio Y. On the properties of neural machine translation: Encoder–decoder approaches. In: Proceedings of SSST-8, eighth workshop on syntax, semantics and structure in statistical translation. Doha, Qatar: Association for Computational Linguistics; 2014, p. 103–11. http://dx.doi.org/10.3115/v1/W14-4012.

[57] Sehovac L, Grolinger K. Deep learning for load forecasting: Sequence to sequence recurrent neural networks with attention. IEEE Access 2020;8:36411–26. http://dx.doi.org/10.1109/ACCESS.2020.2975738.

[58] Kim T-Y, Cho S-B. Predicting residential energy consumption using CNN-LSTM neural networks. Energy 2019;182:72–81. http://dx.doi.org/10.1016/j.energy.2019.05.230.

[59] Le T, Vo MT, Vo B, Hwang E, Rho S, Baik SW. Improving electric energy consumption prediction using CNN and Bi-LSTM. Appl Sci 2019;9(20). http://dx.doi.org/10.3390/app9204237.

[60] Shi H, Xu M, Li R. Deep learning for household load forecasting—A novel pooling deep RNN. IEEE Trans Smart Grid 2018;9(5):5271–80. http://dx.doi.org/10.1109/TSG.2017.2686012.

[61] Kong W, Dong ZY, Jia Y, Hill DJ, Xu Y, Zhang Y. Short-term residential load forecasting based on LSTM recurrent neural network. IEEE Trans Smart Grid 2019;10(1):841–51. http://dx.doi.org/10.1109/TSG.2017.2753802.

[62] Li C, Ding Z, Zhao D, Yi J, Zhang G. Building energy consumption prediction: An extreme deep learning approach. Energies 2017;10(10). http://dx.doi.org/10.3390/en10101525.

[63] Yan K, Wang X, Du Y, Jin N, Huang H, Zhou H. Multi-step short-term power consumption forecasting with a hybrid deep learning strategy. Energies 2018;11(11). http://dx.doi.org/10.3390/en11113089.

[64] Kim J-Y, Cho S-B. Electric energy consumption prediction by deep learning with state explainable autoencoder. Energies 2019;12(4). http://dx.doi.org/10.3390/en12040739.

[65] Briggs C, Fan Z, András P. Federated learning for short-term residential energy demand forecasting. 2021, ArXiv, arXiv:2105.13325.

[66] Zhao Y-M, Xiao W, Shuai L, Luo J, Yao S, Zhang M. A differential privacy-enhanced federated learning method for short-term household load forecasting in smart grid. In: 2021 7th international conference on computer and communications. 2021, p. 1399–404. http://dx.doi.org/10.1109/ICCC54389.2021.9674514.

[67] Bennett CJ. Regulating privacy. Cornell University Press; 2018, http://dx.doi.org/10.7591/9781501722134.

[68] Li Q, Wen Z, Wu Z, Hu S, Wang N, Li Y, et al. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. IEEE Trans Knowl Data Eng 2021;1. http://dx.doi.org/10.1109/TKDE.2021.3124599.

[69] Pressmair G, Kapassa E, Casado-Mansilla D, Borges CE, Themistocleous M. Overcoming barriers for the adoption of local energy and flexibility markets: A user-centric and hybrid model. J Cleaner Prod 2021;317:128323. http://dx.doi.org/10.1016/j.jclepro.2021.128323.

[70] Dwork C. Differential privacy. In: Bugliesi M, Preneel B, Sassone V, Wegener I, editors. Automata, languages and programming. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006, p. 1–12. http://dx.doi.org/10.1007/11787006_1.

[71] Dwork C, Roth A. The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 2014;9(3–4):211–407. http://dx.doi.org/10.1561/0400000042.

[72] Wood A, Altman M, Bembenek A, Bun M, Gaboardi M, Honaker J, et al. Differential privacy: A primer for a non-technical audience. Vanderbilt J Entertain Technol Law 2018;21(1):209–75, URL http://www.jetlaw.org/journal-archives/volume-21/volume-21-issue-1/differential-privacy-a-primer-for-a-non-technical-audience/.

[73] Jayaraman B, Evans D. Evaluating differentially private machine learning in practice. In: 28th {$USENIX$} security symposium ({$USENIX$} security 19). 2019, p. 1895–912, URL https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman.

[74] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, et al. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. New York, NY, USA: Association for Computing Machinery; 2017, p. 1175–91. http://dx.doi.org/10.1145/3133956.3133982.

[75] Shamir A. How to share a secret. Commun ACM 1979;22(11):612–3. http://dx.doi.org/10.1145/359168.359176.

[76] Eibl G, Engel D. Differential privacy for real smart metering data. Comput Sci Res Dev 2017;32(1–2):173–82. http://dx.doi.org/10.1007/s00450-016-0310-y.

[77] Zhao J, Jung T, Wang Y, Li X. Achieving differential privacy of data disclosure in the smart grid. In: IEEE INFOCOM 2014 - IEEE conference on computer communications. 2014, p. 504–12. http://dx.doi.org/10.1109/INFOCOM.2014.6847974.

[78] Varrette S, Bouvry P, Cartiaux H, Georgatos F. Management of an academic HPC cluster: The UL experience. In: Proc. of the 2014 intl. conf. on high performance computing & simulation. Bologna, Italy: IEEE; 2014, p. 959–67. http://dx.doi.org/10.1109/HPCSim.2014.6903792.

[79] Chollet F, et al. Keras. 2015, URL https://keras.io.

[80] D. J-M. Smart meter data from London area. 2019, URL https://www.kaggle.com/jeanmidev/smart-meters-in-london.

[81] CACI. The acorn user guide. Tech. rep., CACI; 2014, URL https://acorn.caci.co.uk/downloads/Acorn-User-guide.pdf.

[82] Li T, Sahu AK, Talwalkar A, Smith V. Federated learning: Challenges, methods, and future directions. IEEE Signal Process Mag 2020;37(3):50–60. http://dx.doi.org/10.1109/msp.2020.2975749.

[83] Yang Q, Liu Y, Cheng Y, Kang Y, Chen T, Yu H. Federated learning. Synth Lect Artif Intell Mach Learn 2019;13(3):1–207. http://dx.doi.org/10.2200/S00960ED2V01Y201910AIM043.

[84] McMahan HB, Ramage D, Talwar K, Zhang L. Learning differentially private recurrent language models. 2018, ArXiv, arXiv:1710.06963.

[85] Lu Y, Huang X, Dai Y, Maharjan S, Zhang Y. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. IEEE Trans Ind Inf 2019;16(6):4177–86. http://dx.doi.org/10.1109/TII.2019.2942190.

[86] Andrew G, Thakkar O, McMahan B, Ramaswamy S. Differentially private learning with adaptive clipping. Adv Neural Inf Process Syst 2021;34. URL https://proceedings.neurips.cc/paper/2021/file/91cff01af640a24e7f9f7a5ab407889f-Paper.pdf.

[87] Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, et al. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016, p. 308–18. http://dx.doi.org/10.1145/2976749.2978318.

[88] Mironov I. Rényi differential privacy. In: 2017 IEEE 30th computer security foundations symposium. IEEE; 2017, http://dx.doi.org/10.1109/csf.2017.11.

[89] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings; 2010, p. 249–56, URL https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf.

[90] Lewis CD. Industrial and business forecasting methods: a practical guide to exponential smoothing and curve fitting. Butterworth-Heinemann; 1982.

[91] Lee C-Y, Wu C-E. Short-term electricity price forecasting based on similar day-based neural network. Energies 2020;13(17). http://dx.doi.org/10.3390/en13174408.

[92] Appleyard J, Kocisky T, Blunsom P. Optimizing performance of recurrent neural networks on GPUs. 2016, ArXiv, arXiv:1604.01946.

[93] Hao K. Training a single AI model can emit as much carbon as five cars in their lifetimes. MIT Technol Rev 2019. URL https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/.

[94] Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. 2015, p. 1322–33. http://dx.doi.org/10.1145/2810103.2813677.