Privacy-Preserving Probabilistic Voltage Forecasting in Local Energy Communities

This paper was downloaded from TechRxiv (https://www.techrxiv.org).

LICENSE

CC0

SUBMISSION DATE / POSTED DATE

17-01-2022 / 24-01-2022

CITATION

Toubeau, Jean-François; Teng, Fei; Wang, Yi; Von Krannichfeldt, Leandro; Morstyn, Thomas (2022): Privacy-Preserving Probabilistic Voltage Forecasting in Local Energy Communities. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.18535274.v1

DOI

10.36227/techrxiv.18535274.v1
privacy-preserving probabilistic voltage forecasting in local energy communities

Jean-François Toubeau, Member, IEEE, Fei Teng, Member, IEEE, Yi Wang, Member, IEEE, Leandro Von Krannichfeldt, and Thomas Morstyn, Member, IEEE

Abstract—This paper presents a new privacy-preserving framework for the short-term (multi-horizon) probabilistic forecasting of nodal voltages in local energy communities. This task is indeed becoming increasingly important for cost-effectively managing network constraints in the context of massive integration of distributed energy resources. However, traditional forecasting tasks are carried out centrally, by gathering all individual information in a single database, thus exposing private aspects of end-users. To avoid such privacy issues, this work relies on a distributed learning scheme, known as federated learning wherein individuals’ data are kept decentralized. The learning procedure is then augmented with differential privacy, which offers formal resilience guarantees that the trained model cannot be reversed-engineered to infer sensitive local information. Moreover, the problem is framed using cross-series learning, which allows to smoothly integrate any new client joining the community (i.e., cold-start forecasting) without being plagued by data scarcity. Outcomes show that the proposed approach can achieve a trade-off between privacy and model performance for different architectures of deep learning networks.

Index Terms—Differential privacy, Deep learning, Federated learning, Heterogeneous data, Voltage forecasting.

I. INTRODUCTION

The deployment of distributed energy resources (DER), such as photovoltaic (PV) generation and electric vehicles, is increasingly affecting the operation of distribution networks [1]. In order to comply with security constraints while avoiding costly infrastructure investments, one solution is to proactively manage local energy exchanges [2]. This can be efficiently achieved though renewable energy communities (RECs) which gather local end-users into organized entities, wherein energy resources are pooled and allocated to reach common (e.g., economical, or environmental) objectives [3]. However, to ensure an optimal coordination between resources, these local entities need to be informed with accurate predictions of the future system state with an adequate time granularity (e.g., 30 minutes) [4].

Our objective is thus to develop a new framework for the short-term probabilistic forecasting of nodal voltage magnitudes in RECs, by exploiting the information from smart metering devices. In contrast with state estimation that is used for identifying current voltage values in support of real-time grid management [5, 6], state forecasting is used for estimating future voltage values to enable a pro-active network scheduling (to avoid costly and suboptimal redispatch actions).

Although load and renewable energy forecasting tasks have been thoroughly studied (see, e.g., [7]–[9]), the literature on voltage forecasting is still very sparse. In [10], [11], nodal generation and consumption are firstly predicted, and then embedded into a network model to calculate nodal voltages. In [12], the relation from (predicted) nodal energy exchanges to voltage levels is build based on linear approximations of the unbalanced three-phase power flows. However, these methods rely on the perfect knowledge of the network parameters, which are usually uncertain (e.g., the phases on which each client is connected are often unknown). To bypass this limitation, data-driven approaches have been developed. The voltage levels are represented by vector autoregressive (VAR) processes in [13]. To avoid such linear models, a deep learning model is proposed in [14], where it is shown that a high accuracy can be achieved by monitoring only a few strategic buses. In [15], an ensemble approach (combining different regression models) is introduced for the deterministic voltage prediction. This work has been extended in [16] and [17] in a probabilistic framework.

All these algorithms assume that all private data can be freely accessed from a centralized location. In a competitive environment, the nodal (e.g., smart meter) data are owned by end-users, who may be reluctant to share this information (as it may reveal private aspects such as home occupancy, routines and usage of specific appliances). To address privacy concerns, an efficient solution is to rely on distributed learning. In [18], the alternating direction method of multipliers (ADMM) has been used in the context of renewable energy forecasting to a ridge linear quantile regression model. Since ADMM does not guarantee privacy (as adversaries can recover the data if they have access to the intermediate calculations), the ADMM procedure is enriched with data encryption in [19] for training autoregressive models. Alternatively to such ADMM-based techniques (tailored to linear models), federated learning (FL) has recently emerged [20] to train complex (non-linear) forecasters, such as tree-based [21] or deep learning [22] models. FL is a decentralized approach aiming at collaboratively learning a shared prediction model, through the exchange of local model updates instead of the raw training data.

However, although FL complicates data inference since no centralised server holds all the information, it is not sufficient to ensure data privacy. Indeed, it has been shown that trained models may be reverse-engineered to extract detailed input information from the end-users involved in the training phase [23], [24]. It is thus essential to provide guarantees that the trained forecasting model protects the privacy of individual databases [25]. In this paper, we therefore augment FL (which lacks rigorous privacy guarantees against inference attacks) with differential privacy (DP). The principle of DP is to inject
noise into the training procedure, which is calibrated in such a way that the privacy leakage of any sensitive information can be bounded and quantified [26]. In particular, the methodology is tailored to achieve user-level privacy, i.e., enforcing that the dataset of any client has a limited impact on the learned model, thus preventing inference of local raw information.

Another important aspect for the REC is to smoothly accommodate end-users with different history, such as new homes with no historical data. Hence, traditional learning strategies wherein local features (e.g., past load and PV generation) from all nodes are aggregated into the same input vector should be avoided. Indeed, the sites with few historical measures inherently limit the number of samples to train the model (i.e., the longer history of other end-users cannot be used). In this paper, we therefore develop a cross-learning approach [27], which mitigates the problem of (local) data scarcity by treating each end-user as a different sample to train a single, generic model. By generalizing to all samples from all clients of the community, the model endogenously learns common patterns from neighboring nodes (thus capturing space dependencies). Moreover, this transfer of learning enables cold-start forecasting for end-users with no historical data. Overall, the contributions of the paper are three-fold.

1) We innovatively combine federated learning (to avoid data exchanges during training), differential privacy (to prevent inference of private information from the trained model) and cross-series learning (to accommodate end-users with diverse measurements histories) for the privacy-preserving probabilistic forecasting of voltage levels in RECs. The framework is developed for different deep learning (DL) models, due to their ability to capture complex high-dimensional dependencies.

2) We ensure that the training mechanism satisfies Rényi differential privacy, which offers a tight analysis of the total privacy consumed. Since privacy losses may occur at each training round of the federated gradient descent procedure, it is important to keep track of the cumulated privacy budget. In this way, based on the REC preferences, the trade-off between model performance and privacy losses can be easily adapted.

3) We leverage layer normalization as an efficient way to normalize individual input features. In contrast to traditional approaches, this solution does not rely on shared statistics among end-users (that would break data privacy), and rather internalizes the normalization into the first layer of the model. This generic procedure can thus be applied to any neural network architecture.

Outcomes reveal the added value of the proposed privacy-preserving framework with respect to fully private forecasting (where each end-user trains its own model with its own data), thus highlighting the interest for end-users to collaboratively train a joint community-wide model. In particular, recurrent neural networks exhibit high robustness to training noise, which is a key property to learn accurate private forecasters.

II. PROBLEM FORMULATION AND BACKGROUND

In this section, we firstly formulate the probabilistic voltage forecasting problem using cross-series learning. Then, we introduce the underlying deep learning-based model. Finally, we elaborate on the concepts of federated learning and differential privacy, which will be used (in section III) to enrich the forecasting model with strong privacy guarantees.

A. Model description

The objective is to generate privacy-preserving probabilistic forecasts of nodal voltages within low-voltage RECs. This information is indeed essential to support an optimized operation of the system, e.g., by providing insights on how much flexibility needs to be gathered (from different end-users) to prevent voltage violations [28].

However, nodal voltages are governed by intricate time correlations as well as space dependencies arising from network constraints (i.e., neighbouring buses are likely to exhibit similar voltage patterns). Capturing such space-time dependencies is a challenging task since the size of the REC may evolve (e.g., with new homes) and the REC may be composed of end-users with different history, including some with very few measurements. Hence, the traditional solution of jointly predicting all nodal voltages in a single instance of the forecasting model should be avoided since the total number of training samples would be limited by the end-user with the smallest database (so that many relevant data would be lost). Moreover, such an approach is not scalable since increasing the number of end-users is leading to a high-dimensional output space, which may lead to high training complexity.

Here, we tackle this issue by using cross-series learning [27]. In this setting, we learn a single forecasting model \( f_0 \) (with parameters \( \theta \)) which is shared by all end-users. Each user (using only its own private data along with publicly available information) predicts the voltage level corresponding to its node. In this way, clients with limited history will simply have fewer samples for training the joint model.

Overall, by generalizing from correlated voltage patterns learned from all individuals, the resulting model \( f_0 \) is able to accommodate clients with different history, while providing cold-start forecasts for new nodes with no historical measures. This bypasses the need to retrain the model from scratch, at the profit of dedicated recalibration techniques.

Practically, the model \( f_0 \) is used by each individual end-user \( n \in \mathcal{N} \) of the community for predicting, at the forecast creation time \( t_0 \), the conditional distribution of voltage levels \( y_{t_1:T,n} = (y_{t_1,n}, ..., y_{T,n}) \) for node \( n \) over the horizon \([t_1, T]\) (using exclusively by information \( x_{t_0,n}^{\text{all}} \) available at node \( n \) before \( t_0 \)):

\[
\begin{align*}
\text{Pr}(y_{t_1:T,n} | x_{t_0,n}, x_{t_0,n}^{(p)}, x_{t_1:T,n}^{(f)}, x_{n}^{(s)}) \forall n \in \mathcal{N} \quad (1)
\end{align*}
\]

where \( y_{t_0,n} \) are the past nodal voltages (measured before \( t_0 \)), \( x_{t_0,n}^{(p)} \) are the past time-varying covariates (before \( t_0 \)), \( x_{t_1:T,n}^{(f)} \) are the known future covariates (over the horizon \([t_1, T]\)), and \( x_{n}^{(s)} \) are the time-invariant features.

These covariates are summarized in Table I. It should be noted that the past net imports-exports of the community is public information provided by the central authority (e.g.,
distribution system operator). The imports predictions are also given, and are obtained with an autoregressive model (using only past values). Moreover, to ensure that space dependencies are captured during the training phase (via transfer of statistical learning between end-users), it is necessary to properly encode the spatial information of the nodes [29]. Here, we use a generic approach, wherein we specify the feeder of the network to which is connected the end-user, along with the distance between the end-user and the feeder’s root node.

### B. Deep Learning Model Structure and Training

The multi-horizon time-series forecasting problem can be naturally treated as a sequence-to-sequence task, wherein the goal is to convert a sequence of \( k \) past observations into a sequence containing the \( T \) predictions of interest. In this work, this task is solved using an encoder-decoder model, i.e., an advanced deep learning model, which has shown high performance in different studies [27], [30].

As shown in Fig. 1, the encoder processes past data over a look-back window of \( k \) time steps \([t_{k-1}, t_0]\), with the goal of extracting the relevant dynamics into a context vector \( c_{\text{enc}} \). Then, at time \( t_0 \), the decoder leverages this vector \( c_{\text{enc}} \), along with the known future data \( x_{t_1:T,n}^{(f)} \) and static features \( x_n^{(s)} \), to generate the multi-horizon predictions \( \hat{y}_{t_1:T,n} \). Both encoder and decoder blocks are modeled by recurrent neural networks due to their ability to represent inter-temporal dependencies. Indeed, these models have an internal memory \( h_t \) that provides an internal representation of past events, which is used to propagate relevant information through time.

In this work, we are interested in probabilistic forecasts that return the full conditional distribution \( p(y_{t_1:T,n}|x_{t_0:n}^{\text{all}}) \). This target distribution is here approximated with Quantile Regression (QR) [31], which provides the conditional quantiles \( \hat{y}_{t_1:T,n} = \{ \hat{y}_{t_1:T,n}^{(q)} \}_{q \in Q} = f_{\theta}(x_{t_0:n}^{\text{all}}) \ \forall n \in N \) [32]. For each node \( n \), the QR model \( f_{\theta} \) thus yields:

\[
\hat{y}_{t_1:T,n}^{(q)} = \{ \hat{y}_{t_1:T,n}^{(q)} \}_{q \in Q} = f_{\theta}(x_{t_0:n}^{\text{all}}) \ \forall n \in N \] (2)

where the output corresponding to each time step \( t \in [t_1, T] \) is thus \( |Q|\)-dimensional.

Practically, the model \( f_{\theta} \) is trained on historical observations to learn the (unknown) relationship between inputs \( x_{t_0:n}^{\text{all}} \) and the outputs of interest \( \hat{y}_{t_1:T,n}^{(q)} \ \forall t \in [t_1, T], n \in N, q \in Q \). This is achieved by adapting the model parameters \( \theta \) so as to minimize a user-defined loss function \( \mathcal{L}(\theta) \), which penalizes discrepancies between predictions and actual observations (in the training data). This learning phase is carried out with a stochastic gradient descent (SGD) algorithm. In this iterative optimization process, one forms (at each round), a batch \( b \in B \) of \( z_i = (x_{t_0:n}^{\text{all}}, y_{t_1:T,n}) \), samples from the historical database, and estimates (over these \( z_i \) samples) the gradient of the loss function \( \mathcal{L} \) with respect to the \( \theta \)-parameters as follows:

\[
\nabla_{\theta} \mathcal{L}(\theta, b) = \frac{1}{|b|} \sum_{i=1}^{|b|} \nabla_{\theta} \mathcal{L}(\theta, z_i) \] (3)

where \( \nabla_{\theta} \mathcal{L}(\theta, z_i) = [\partial \mathcal{L}(\theta, z_i)/\partial \theta_1, \ldots, \partial \mathcal{L}(\theta, z_i)/\partial \theta_{|\theta|}] \). Then, the weight vector \( \theta \) is updated following the direction of the batch-averaged gradient \( -\nabla_{\theta} \mathcal{L}(\theta, b) \) towards a local minimum, i.e.,

\[
\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}(\theta, b) \] (4)

where \( \eta \) is the learning rate. It should be noted that SGD variants have been developed to reach better performances. One popular example is the Adam algorithm [33] that relies on adaptive learning rates to improve convergence properties.

### C. Federated learning

Federated learning (FL) is a distributed approach where a federation of clients \( c \in C \) is coordinated by a central server to learn a global model \( f_0 \), without sharing any raw client data.

In [20], an innovative algorithm, called Federated Averaging, is developed. In this setting, the server initializes the parameters \( \theta_0 \). At each round \( r \in R \) of the federated training, a sample of clients \( C_r \subseteq C \) is selected, to whom the server broadcasts the global model \( \theta_{r-1} \). Then, each selected client \( c \in C_r \) performs local computations (e.g., SGD or Adam) on its private dataset and computes the difference \( \Delta_{c,r} \) between the new (locally) optimized model \( \theta_{c,r} \) and the global model \( \theta_{r-1} \). The local updates \( \Delta_{c,r} \) are then uploaded to the server, which calculates the global average (5), i.e.,

\[
\Delta_r = \frac{1}{|C_r|} \sum_{c \in C_r} \Delta_{c,r} \] (5)

where \( |C_r| \) is the number of clients in training round \( r \). The new global model is then computed as \( \theta_r \leftarrow \theta_{r-1} + \eta_r \Delta_r \), with \( \eta_r \) the learning rate at the server.

It should be noted that the Federated Averaging algorithm enables to perform multiple local updates, e.g., multiple steps of the SGD (3)-(4), before the averaging step (5). This reduces...
the number of communication iterations between the server and the clients. As further discussed in subsection III-C, by limiting these interactions (during which an adversary can potentially access the parameters \( \theta_i \)) stronger privacy guarantees can be achieved by the final model.

D. Differential privacy

Differential privacy (DP) is a standard methodology to bound and quantify the privacy leakage of sensitive information when performing learning tasks [26]. DP offers provable guarantees of protection against adversaries that have full knowledge of the training procedure along with an access to the model’s parameters. To that end, DP is based on the notion of adjacent databases \( D \) and \( D' \), which differ by the addition or removal of a single element. In this work, we consider on user-level DP, which focuses on the largest possible difference that one client can have on the trained model [34].

In the context of deep learning, \( M \) refers to the learning procedure (e.g., gradient descent), and DP is enforced on the weights \( \theta \) of the trained model. Since the learning process \( M \) is stochastic, there is an inherent uncertainty around the weights \( \theta \) obtained at the end of the learning. Formally, a randomized learning algorithm \( M \) is said to be \((\epsilon, \delta)\)-differentially private if, for any adjacent datasets \( D \) and \( D' \), and any subset \( S \) in the weights distribution, we have:

\[
Pr(M(D) \in S) \leq e^\epsilon Pr(M(D') \in S) + \delta
\]

where \( \epsilon \geq 0 \) is the privacy loss upper-bounding how much the probability of converging to a particular set of weights \( \theta \) is affected by including (or removing) a single client during training \( M \), while \( \delta \in [0,1] \) is the failure probability which caps any long tail of the \( M(\cdot) \)-distribution where pure \( \epsilon \)-DP guarantees do not hold. Hence, smaller values of \( \epsilon \) and \( \delta \) yield stronger privacy protection.

Overall, a training process \( M \) is differentially private if the probability of \( \theta_1 = M(D) \) and \( \theta_2 = M(D') \) are close for every choice of \( D \) and \( D' \), i.e., the data of any client do not significantly affect the weights distribution of the algorithm. In this work, the goal is to convert the federated learning of neural networks into a differentially private distributed training \( M \) that is associated with formal (i.e., provable) privacy guarantees (by bounding \( \epsilon \) and \( \delta \)) values.

III. DIFFERENTIALLY PRIVATE FEDERATED FORECASTER

In this section, we present how to train deep learning models with user-level DP. First, in subsection III-A, we introduce a privacy-compliant procedure that internalizes the data normalization (of individual end-users) into the prediction model. Then, in subsection III-B, we explain how to adapt the model training to offer privacy guarantees, which is achieved by augmenting FL algorithms with DP. Finally, we compute in subsection III-C the total privacy cost of the trained model.

A. Features normalization

In general, end-users have different distributions of data. In a centralized learning, this poses no problem as a common normalization of features is applied over the entire database. However, when local data are private, such a shared computation cannot be performed. This prevents the use, e.g., of traditional batch normalization [35] wherein local statistics (i.e., mean and variance values) are aggregated over the whole training data (including all clients) to generate the predictions.

To address this issue, we use layer normalization (LN), i.e., a data normalization that does not require to share user-level statistics [36]. For the input vector \( x_i \), the normalized inputs \( x_i^{\text{norm}} \) are given by:

\[
x_i^{\text{norm}} = f \left( \frac{g}{\sigma_i} \odot (x_i - \mu_i) + b \right)
\]

where \( \odot \) is the element-wise multiplication of vectors, while \( \mu_i \) and \( \sigma_i \) are the mean and standard deviation of the elements in the input vector \( x_i \) (which are computed and used locally). The bias \( b \) and gain \( g \) vectors are the internal parameters of the model that need to be learned during training. These vectors \( b \) and \( g \) are thus common to all clients, such that the normalization procedure (involving all clients) is directly internalized into the privacy-preserving learning.

B. Differentially private deep federated learning

Different approaches have recently been proposed for the differentially private training of learning models [37], [38]. As discussed in [26], an effective solution to ensure that a learning function \( f \) achieves \((\epsilon, \delta)\)-DP is to add noise proportional to the sensitivity \( S_f \) of that function:

\[
M(D) = f(D) + \text{Noise}(S_f)
\]

where \( S_f \) is defined as the maximum of the \( l_2 \)-distance \( \|f(D) - f(D')\|_2 \) for any adjacent input datasets \( D \) and \( D' \). In particular, it is shown in [39] that a function \( f \) with sensitivity \( S_f \) can achieve \((\epsilon, \delta)\)-DP by adding Gaussian noise \( \mathcal{N}(0, S_f^2\sigma^2) \), with \( \epsilon \leq 1 \) and \( \delta \geq 0.8 \cdot \exp(-\sigma^2/2). \) The resulting learning process is usually referred to as Gaussian mechanism, i.e., \( M(D) \triangleq f(D) + \mathcal{N}(0, S_f^2\sigma^2) \), where \( \sigma \) is the noise multiplier controlling the trade-off between privacy and model performance.

In the context of (centrally trained) neural networks, DP can thus be achieved by applying the Gaussian mechanism to the weight update function (4) of the stochastic gradient descent (SGD) algorithm [40], i.e.,

\[
\theta \leftarrow \theta - \eta \frac{1}{|B|} \left( \sum_{i=1}^{|B|} \nabla \theta \mathcal{L} (\theta, z_i) + \mathcal{N} (0, I^{(\theta)} S_f^2 \sigma^2) \right)
\]

where \(|B|\) is the number of samples \( z_i \) in the SGD mini-batch \( b \in B \), and \( I^{(\theta)} \) is an identity matrix of size \(|\theta|\) such that \( \mathcal{N}(0, I^{(\theta)} S_f^2 \sigma^2) \) is a \(|\theta|\)-dimensional Gaussian noise.

This principle can be further extended to FL [34], by using a DP version of the global average update (5), i.e., where
per-client updates $\Delta_{c,r}$ are aggregated (on the server) at each round $r$ of the training:

$$\theta \leftarrow \theta + \eta_s \frac{1}{|C_r|} \left( \sum_{c \in C_r} \Delta_{c,r} + N \left( 0, I^{(\theta)} |S_f|^2 \sigma^2 \right) \right)$$  \hspace{1cm} (10)

However, in traditional SGD algorithms, the sensitivity $S_f$ of the update function $f$ is a priori unknown. A solution is proposed in [40], wherein the gradient of each sample is clipped at a threshold $\overline{S}$ before the update step, such that the maximum influence (i.e., sensitivity) of a sample on the final average is bounded, i.e., $S_f \leq \overline{S}$.

However, this technique only offers privacy for single samples. In order to extend sample-level DP to user-level DP, we need to bound the sensitivity of the training function (10) with respect to the addition or removal of any client from the dataset. This is accomplished by clipping the local weights updates $\Delta_{c,r}$ (at the end of the local training round $r$ of each end-user $c$). In particular, for each user, we bound the $(l_2$-norm of) local updates $\Delta_{c,r}$ by $\overline{S}$, i.e., $\|\Delta_{c,r}\|_2 \leq \overline{S}$, as follows:

$$\Delta_{c,r} \leftarrow \Delta_{c,r} \cdot \min \left( 1, \frac{\overline{S}}{\|\Delta_{c,r}\|_2} \right)$$  \hspace{1cm} (11)

such that if $\|\Delta_{c,r}\|_2 > \overline{S}$, per-user updates $\Delta_{c,r}$ are reduced to $\overline{S}$, while $\Delta_{c,r}$ are preserved otherwise. It should be noted that bounding the influence of any user is also beneficial for training stability since it prevents the model to overfit to a particular subset of data.

When considering privacy in the federated learning of deep networks, it is important to notice that privacy breaches may occur at each round $r$ between clients and the server, through the information contained in the weight updates. To mitigate this issue, we exploit the randomness associated with subsampling. Indeed, if $\mathcal{M}$ is $(\epsilon, \delta)$-DP, then drawing a random subset of end-users (from all $|C|$ clients) before applying $\mathcal{M}$ follows $(O(\gamma |C|), \gamma \delta)$-DP, with $\gamma < 1$ (which depends on the sampling strategy) [41].

Here, end-users are randomly and independently sampled with probability $q \in (0, 1)$ at each round $r \in \mathcal{R}$ of the training mechanism $\mathcal{M}$. Hence, the number of end-users at each round $|C_r|$ is variable and unknown. It is thus replaced by its expected value $E[|C_r|] = q|C|$, such that the global model update (on the server) is:

$$\theta \leftarrow \theta + \eta_s \frac{1}{q|C|} \sum_{c \in C_r} \Delta_{c,r} + N \left( 0, I^{(\theta)} \left( \frac{\overline{S}}{q|C|} \right)^2 \sigma^2 \right)$$  \hspace{1cm} (12)

where the sensitivity of the update function (12) is bounded by $S_f \leq \frac{\overline{S}}{q|C|}$, and $\sigma$ is the noise multiplier for training the neural network.

The resulting differentially private federated learning for deep networks (DP-FDL) is given in Algorithm 1. By combining sampling and additive Gaussian noise to the update function, the training procedure $\mathcal{M}$ follows a Sampled Gaussian mechanism (SGM), which is sufficient for achieving a central differential privacy (section III-C).

As shown in Algorithm 1, federated averaging is augmented with server momentum $\beta$, due to its ability to dampen oscillations in the learning, and thus to mitigate that local models diverge from the globally optimal solution.

### Algorithm 1 Differentially private federated learning for deep networks (DP-FDL)

```python
def ServerTrain(q, \sigma, \eta_s, \eta, S):
    Initialize model $\theta_0$
    for each round $r \in \mathcal{R}$ do
        $C_r \leftarrow$ (users sampled with probability $q$)
        for each client $c \in C_r$ do
            $(\Delta_{c,r}) \leftarrow$ LocalTrain($c$, $\theta_{r-1}$, $\eta_c$, $\overline{S}$)
        end for
        $\Delta = \beta \Delta_{r-1} + \Delta_r$
        $\theta_{r+1} \leftarrow \theta_r + \eta_s \Delta$
    end for
end function

function LocalTrain($c$, $\theta_{r-1}$, $\eta_c$, $\overline{S}$)
    $\theta \leftarrow \theta_{r-1}$
    split local data into batches $b \in \mathcal{B}$
    for each batch $b \in \mathcal{B}$ do
        $\theta \leftarrow \theta - \eta_c \nabla_{\theta} \mathcal{L}(\theta, b)$
    end for
    return $\left( \Delta \cdot \min \left( 1, \frac{\overline{S}}{\|\Delta_{c,r}\|_2} \right) \right)$
end function
```

### Privacy budget

The DP-FDL algorithm consists of $|\mathcal{R}|$ successive queries of the learning function $\mathcal{M}$ in Eq. (12). An adversary with access to intermediate models $\theta_r$ may leverage this additional information to infer raw (private) data, and it is thus necessary to keep track of the total privacy loss, which is accumulating during the training ($\mathcal{M}_1, \ldots, \mathcal{M}_{|\mathcal{R}|}$).

A solution is offered by advanced composition theorems [42], which give an upper bound of the accumulated privacy loss (by assuming that the worst-case scenario wherein the same amount of leakage occurs at each query to the data). For instance, applying $|\mathcal{R}|$ times consecutively the same $(\epsilon, \delta)$-DP algorithm gives an $O\left( \epsilon \sqrt{T \log(1/\delta)} \right)$ bound [43]. However, these compositions are generic (they do not account for the specific noise distribution used in the learning), such that they tend to strongly exaggerate privacy losses.

This has motivated the development of another approach, called the moments accountant [40], which, instead of directly dealing with $(\epsilon, \delta)$-DP, relies on the notion of Rényi-DP (RDP) [44]. It is a natural relaxation of DP based on the Rényi divergence (13) of order $\alpha > 1$ between distributions $P$ and $Q$ (defined over the same probability space):

$$D_\alpha(P \| Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{z \sim Q} \left( \frac{P(z)}{Q(z)} \right)^{\alpha}$$  \hspace{1cm} (13)

where $\log$ is the natural logarithm, and $P(z)$ is the density of $P$ at $z$. By definition, $\mathcal{M}$ satisfies $(\alpha, \gamma)$-RDP if for any adjacent inputs $D, D'$, it holds that $D_\alpha(\mathcal{M}(D) \| \mathcal{M}(D')) \leq \gamma$. 
The main advantage of RDP is its simple linear composition form, i.e., if \( M \) obeys \((\alpha, \gamma)\)-RDP, then the composition \( M_{|R|} \) is \((\alpha, |R|\gamma)\)-RDP [44]. In particular, a numerically stable computational procedure for estimating the Rényi parameters for a sampled Gaussian mechanism is presented in [45], and has shown to provide strengthened privacy bounds.

However, in contrast with \((\epsilon, \delta)\)-DP, Rényi parameters are more difficult to interpret, and we are thus interested in converting the privacy budget expressed in terms of \((\alpha, \gamma)\) in the more interpretable notion of \((\epsilon, \delta)\)-DP. In Proposition 3 in [44], it is shown that an \((\alpha, \gamma)\)-RDP algorithm provides \((\gamma + \log(1/\delta), \delta)\)-DP guarantees. However, follow-up works have improved this conversion, and bounds provided by [46] are therefore used in our experiments to tighten the privacy guarantees of the learning procedure.

IV. BASELINES

In this section, we introduce the different forecasting models, along with the performance metric, used in the case study.

A. Baseline models

First, the sequence-to-sequence model (of section II-B) is tested with Long Short Term Memory (LSTM) recurrent networks for both encoder and decoder blocks. A second variant (BLSTM) relies on bidirectional LSTM networks to improve the representation of time dependencies [32].

Then, we use a traditional deep feedforward neural network (DFFNN), wherein hidden layers are composed of neurons using rectifier linear units (ReLUs) as activation function.

In complement, we build a BLSTM model wherein the information of all end-users is aggregated into a single input vector to jointly predict the voltage levels at all nodes of the grid. This model (A-BLSTM) serves as a benchmark to quantify the value of the cross-learning strategy.

Finally, to have a global overview of the forecasting accuracy, naive and state-of-the-art techniques for multi-horizon forecasting are also implemented.

- a naive probabilistic averaging model (Prob-Avg), where the voltage distribution at each time step is computed based on the average of all past observations corresponding to this specific period.
- a naive probabilistic persistence (Prob-Persistence), where the last nodal voltage value \( y_{t-1,n} \) is propagated over the prediction horizon \([t_1, t_f]\) as the mean value, while the variance is computed on the look-back window.
- a quantile regression forest (QRF), i.e., a tree-based ensemble model, in which the outputs of independent regression trees are merged for estimating the conditional distribution. The forest is set to 500 trees.
- a gradient boosting regression tree (QGBRT) trained with the quantile loss. In this ensemble model, new regression trees are sequentially generated to forecast the residuals of the previous models. The number of boosting stages is set to 100 with an early stopping criterion.

For each forecasting model (excepting the parameter-free naive methods), an hyperparameter optimization is carried out through an extensive random search to identify the optimal model complexity. The same number of iterations is used across all benchmarks.

Moreover, calendar information (day of the week, hour of the day) is processed by a dedicated embedding layer for all deep learning models. This layer is transforming the calendar information into a 3-dimensional continuous space, wherein time steps with similar characteristics are put close to each other. For tree-based models, a traditional one-hot encoding method is used (e.g., such that 24 binary variables are needed to encode all hours of the day).

B. Performance metrics

When assessing the performance of a probabilistic forecast, two complementary aspects need to be jointly analyzed, i.e., reliability and sharpness.

Reliability measures how closely the predicted intervals correspond to the actual data frequencies, while sharpness measures the width of prediction intervals. To evaluate the trade-off between both concepts, we use the quantile loss. It has indeed been shown the quantile loss yields consistent outcomes with other metrics such as the Winkler score, and the continuous ranked probability score (CRPS) [47].

The quantile loss \( QL_{\tau,n} \) for node \( n \) at time step \( \tau \) of the test set is given by:

\[
QL_{\tau,n} = \sum_{q \in \mathcal{Q}} q \max \left( y_{\tau,n} - \hat{y}^{(q)}_{\tau,n}, 0 \right) + (1 - q) \max \left( \hat{y}^{(q)}_{\tau,n} - y_{\tau,n}, 0 \right)
\]

where \( \hat{y}^{(q)}_{\tau,n} \) are the quantiles predicted by the forecaster, while \( y_{\tau,n} \) are the actual voltage observations. In this paper, we compute the quantiles for \( q = 1, 10, 25, 50, 75, 90 \) and 99 %.

Practically, we compute the total pinball loss \( QL^{tot} \), i.e. the average value of all pinball losses (14) over all points \((\tau, n)\) of the space-time domain of the test set. Smaller values of \( QL^{tot} \) correspond to better forecasting outcomes.

V. CASE STUDY

The proposed privacy-preserving voltage forecasting strategy is tested on the IEEE European Low Voltage Test Feeder, shown in Fig. 2, which has a radial structure with 6 line ramifications (used to encode the spacial information of each node). The nodal voltages are predicted over a multi-horizon of \( T = 8 \) intervals of 30 minutes (i.e., 4-hour ahead) for the \( N = 57 \) nodes. A look-back window of \( k = 12 \) intervals (i.e., 6 hours) is selected to capture past dynamics. Each client has a different history of recordings, ranging from 120 to 478 days. To have comparable performance indices among end-users, the test set is composed of the same days for all clients (corresponding to the last 96 days of the horizon), and the rest of the database is used for training and validation.

A. Comparison between fully private and centralized models

To quantify the practical interest for end-users to participate in collaborative learning strategies, we compare the forecasting performance of centralized and fully private models.
First, we study the prediction performance of (non-private) centralized models, i.e., an ideal (but unrealistic) case where each end-user shares its private data to form a single database, from which the community manager trains a single model.

Second, we implement the fully private counterpart, where a different model is trained by each end-user based only on its own private data. It should be noted that this approach does not scale well in real-life power systems (with many communities) since the number of models to locally train (and store) is then equal to the number of end-users, which may be impractical.

In deep learning-based models, the weights are initialized using a Glorot uniform distribution, while the batch size is set to 96 (as a good trade-off between training time and final performance). The optimization is performed with the Adam algorithm with a learning rate of 0.001. Then, in all tree-based and deep learning models, early stopping is implemented for avoiding overfitting in the learning procedure.

Table II presents the performance and training time of the different probabilistic forecasters. For deep learning model, the number of epochs is added in parentheses. It should be noted that, in the private setting, the performance of local models is averaged over all end-users of the community.

### Table II

<table>
<thead>
<tr>
<th>Model</th>
<th>Centralized</th>
<th>Local (private)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QL(_{\text{tot}}) [pu]</td>
<td>time [min] (epochs)</td>
</tr>
<tr>
<td>Prob-Persist</td>
<td>0.319</td>
<td>-</td>
</tr>
<tr>
<td>Prob-Avg</td>
<td>0.048</td>
<td>148</td>
</tr>
<tr>
<td>QRF</td>
<td>0.041</td>
<td>64</td>
</tr>
<tr>
<td>QGBDT</td>
<td>0.043</td>
<td>10 (33)</td>
</tr>
<tr>
<td>DFFNN</td>
<td>0.034</td>
<td>52 (18)</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.032</td>
<td>88 (16)</td>
</tr>
<tr>
<td>BLSTM</td>
<td>0.192</td>
<td>54 (11)</td>
</tr>
<tr>
<td>A-BLSTM</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The experiments show that machine learning models (deep learning and ensemble methods) significantly outperform naive benchmarks (in both centralized and local settings), which stresses the interest of using advanced models to properly represent the dynamics of nodal voltage levels. In particular, centralized BLSTM-based networks (with 1 hidden layer composed of 50 neurons for both encoder and decoder blocks) achieve the best accuracy with QL\(_{\text{tot}}\) = 0.0032 pu, due to their ability to capture the non-linear inter-temporal dependencies within data. The convergence is obtained in 16 rounds, with a per-round training time of around 5.5 minutes.

Moreover, we observe high differences in performance between centralized and local models. Specifically, the local (private) probabilistic averaging (Prob-Avg) wherein the voltage distributions are differentiated between nodes, reduces the total quantile loss of the centralized version from 0.319 pu down to 0.192 pu. This difference reflects the importance of capturing nodal dependencies between voltage levels.

Surprisingly, when training fully private models (using only local data), sequence-to-sequence networks exhibit poor convergence, and are even surpassed by tree-based and feedforward networks. This arises from the difficulty to train recurrent neural networks from scratch with a limited amount of data. In this way, the hyper-parameter optimization has led to very compact LSTM (and BLSTM) local models (with 1 hidden layer of 8 neurons). In that regard, the A-BLSTM (where all nodal voltage levels are jointly predicted) only yields QL\(_{\text{tot}}\) = 0.192 pu. This clearly highlights the added value of the cross-learning approach which, in addition to capture space dependencies (via the transfer of learning between nodes), enables enlarging the dataset (with the aggregation of samples from all end-users \(n \in \mathcal{N}\)). The augmented training set helps reducing the uncertainty space of prediction, thus increasing the sharpness of predictions. In this way, the best centralized model (BLSTM) has a total quantile loss of 0.0032 pu over the test set, while the best local model (DFFNN) performs no better than QL\(_{\text{tot}}\) = 0.0144 pu.

Overall, the resulting performance gap between centralized and local models is a clear indication of the added value for end-users to collaboratively train a community-wide probabilistic forecaster. This need is exacerbated for end-users with data paucity, as they are particularly exposed to poor convergence and weak accuracy.

### B. Utility – privacy trade-off

There is an inherent trade-off between the performance of the prediction model and the privacy of the local data. However, few studies focus on how selecting an adequate privacy budget \(\epsilon\), although it is well-established that higher privacy budgets can be allocated to more complex learning tasks [38]. In that regard, it has been shown that DP-enhanced neural networks are resistant to inference attacks (with state-of-the-art attack frameworks) for \(\epsilon\) up to 100 [48]. In complement, studies on memorization attacks (which exploit the fact that highly parametrized models may memorize some patterns in training data) demonstrated that attacks have been blocked by DP mechanisms with very little noise (\(\epsilon = 10^5\)) [49].

However, from (6), we see that \((\epsilon, \delta)\)-DP allows for potentially large privacy losses (no bound on \(\epsilon\)) with probability \(\delta\). In the worst-case scenario wherein this \(\delta\)-fraction exclusively
relates to a single client, this may prove detrimental. Hence, to ensure privacy for each end-user, a solution is to have \( \delta < 1/|\mathcal{N}| \). In the following of the paper, the \( \epsilon \) values are thus reported for \( \delta = 10^{-2} \).

First, we quantitatively analyze the utility-privacy trade-off for the three different deep learning models, along with the BLSTM without layer normalization (called I-BLSTM), wherein the inputs of each end-user are normalized independently (based only on local statistics). The models are trained during \( |\mathcal{R}| = 60 \) rounds. In Table III, we give the optimal number of rounds \( r^{\text{opt}} \) and the corresponding privacy loss \( \epsilon^{\text{opt}} \), along with the total quantile loss (over the test set) for two noise levels (\( \sigma = 0.25 \) and 0.75) for \( |\mathcal{C}| = 10 \) users per round. The Adam algorithm with \( \eta_c = 0.001 \) is used for training local models, while SGD with a learning rate \( \eta_s = 1 \) is used at the server side. The momentum \( \beta \) is set at 0.6 in Algorithm 1.

From Table III, we see that (LSTM and BLSTM) sequence-to-sequence models are more robust to the noise added, while the DFFNN struggles in achieving decent outcomes in the DP framework. In this way, while the BLSTM is still able to reach \( QL^{\text{tot}} = 0.112 \) pu for \( \sigma = 0.75 \), the DFFNN only performs 0.140 pu (which is roughly equivalent to the private models). Also, outcomes stress the added value of using layer normalization in the federated learning since the I-BLSTM model is consistently outperformed by the reference BLSTM.

Overall, raising \( \sigma \) from 0.25 to 0.75 significantly reduces the worst-case privacy loss, but this comes at the expense of forecasting performance (for all models). It should be noted that using a noise multiplier higher than 0.8 even leads to catastrophic failures. This effect is clearly observed with the performance gaps between \( QL^{\text{tot}} = 5 \) and 10 (e.g., \( QL^{\text{tot}} \) increases from 0.112 pu up to 0.132 pu for \( \sigma = 0.75 \)). In our case, further increasing \( q|C| \) to 12 or 15 end-users do not improve the results, and we conclude that the best solution is to use \( q|C| \) equal to 10. For illustrating the quality of results obtained using the BLSTM network with \( q|C| = 10 \) and \( \sigma = 0.25 \), the probabilistic voltage forecasts of 4 nodes (A, B, C and D in Fig. 2) during a summer day are shown in Fig. 3. The gray areas represent the forecasted quantiles while the red line stands for the actual voltage time series.

In Table IV, we see that models trained with traditional federated learning (which is not augmented with differential privacy, i.e., \( \sigma = 0 \)) cannot reach the performance of centralized models, but they are still significantly better than their private counterpart. In particular, the federated BLSTM with \( q|C| = 10 \) end-users per round converges towards \( QL^{\text{tot}} = 0.054 \) pu, while the centralized and private equivalents ultimately able to converge towards better solutions, i.e., the BLSTM outperforms the DFFNN for all noise levels \( \sigma \). Also,

![Fig. 3. Probabilistic voltage forecasts for four different nodes of the low-voltage community.](image)

| Model   | \( q|C| \) | \( \sigma \) | \( r^{\text{opt}} \) | \( \epsilon^{\text{opt}} \) | \( QL^{\text{tot}} \) [pu] |
|---------|-----------|-------------|----------------|----------------|-------------------|
| DFFNN   | 10        | 0.25        | 55             | 173.6          | 0.096             |
| LSTM    | 10        | 0.25        | 56             | 182.6          | 0.089             |
| BLSTM   | 10        | 0.25        | 54             | 167.7          | 0.090             |
| I-BLSTM | 10        | 0.25        | 54             | 167.7          | 0.115             |
| DFFNN   | 10        | 0.75        | 47             | 9.1            | 0.140             |
| LSTM    | 10        | 0.75        | 56             | 10.0           | 0.117             |
| BLSTM   | 10        | 0.75        | 54             | 7.3            | 0.112             |
| I-BLSTM | 10        | 0.75        | 48             | 9.2            | 0.135             |

In Table IV, we see that models trained with traditional federated learning (which is not augmented with differential privacy, i.e., \( \sigma = 0 \)) cannot reach the performance of centralized models, but they are still significantly better than their private counterpart. In particular, the federated BLSTM with \( q|C| = 10 \) end-users per round converges towards \( QL^{\text{tot}} = 0.054 \) pu, while the centralized and private equivalents ultimately able to converge towards better solutions, i.e., the BLSTM outperforms the DFFNN for all noise levels \( \sigma \). Also,
C. Sensitivity analysis on hyper-parameters

In addition to the parameters specific to privacy, the prediction accuracy of the forecasters is also affected by other hyper-parameters, which need to be carefully tuned. In Table V, we evaluate the influence of the number of hidden units (# neurons), batch size (|b|) and clipping threshold (S) on the BLSTM test performance and per-round training time for σ = 0.25 and q|C| = 10, thus deriving valuable insights for applying differentially private federated learning in smart grid forecasting applications.

<table>
<thead>
<tr>
<th># neurons</th>
<th></th>
<th>b</th>
<th>S</th>
<th>QL* [μu]</th>
<th>time [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>96</td>
<td>0.175</td>
<td>0.090</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>96</td>
<td>0.175</td>
<td>0.095</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>96</td>
<td>0.175</td>
<td>0.150</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.175</td>
<td>0.091</td>
<td>123</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>200</td>
<td>0.175</td>
<td>0.099</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>96</td>
<td>0.1</td>
<td>0.111</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>96</td>
<td>0.3</td>
<td>0.163</td>
<td>42</td>
<td></td>
</tr>
</tbody>
</table>

In contrast with centralized training wherein more complex models have a higher modeling power, using more hidden units in the federated DP framework strongly decreases the model quality. In practice, it is thus preferable to use more compact models, which are less sensitive to the noise added in the learning procedure. This has also a positive effect on the communication burden since it requires transferring a smaller weight vector (and thus less bandwidth) between the server and end-users.

Then, the batch size has a very small effect on the final performance but it significantly influences the computation time. In this way, by decreasing the batch size from 96 to 10, the per-round training time increases from 41 to 123 seconds. Finally, identifying the optimal value of the clipping threshold S̅ is not an easy task. Indeed, low values (S̅ = 0.1) may discard valuable information from the magnitude of local gradients, thus jeopardizing the gradient descent search of optimal parameters. Conversely, when S̅ = 0.3, more noise is added in the global model update (12), which complicates the training. Here, we therefore use S̅ = 0.175.

VI. Conclusions

This paper has presented a new privacy-preserving framework, applied to the probabilistic forecasting of nodal voltage magnitudes. The proposed framework enables to distribute the computations among the parties, and to derive a trade-off between utility and privacy by embedding the learning procedure into a differentially private mechanism. Outcomes show that compact recurrent models are inherently more robust to noise, which makes them natural candidates for the development of privacy-enhancing techniques in renewable-dominated smart grids.

As a perspective, one may be interested in tracking the privacy spent by each client, which is highly challenging since the set of clients participating in each round is private [50]. Also, it may be useful to develop state-of-the-art attack frameworks to have an empirical evaluation of how much information an adversary can actually infer from trained models.

REFERENCES


