IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 17, 2021, accepted June 28, 2021, date of publication July 1, 2021, date of current version July 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3094089

Short-Term Energy Consumption Forecasting at the Edge: A Federated Learning Approach

MARCO SAVI AND FABRIZIO OLIVADESE
Department of Informatics, Systems and Communication, University of Milano-Bicocca, 20126 Milan, Italy
Corresponding author: Marco Savi (marco.savi@unimib.it)

ABSTRACT Residential short-term energy consumption forecasting plays an essential role in modern decentralized power systems. The rise of innovative prediction methods able to handle the high volatility of users' electrical load has posed the basis to accomplish this task. However these methods, which mostly rely on Artificial Neural Networks, require that a huge amount of users' fine-grained sensitive consumption data are centrally collected to train a generalized forecasting model, with implications on privacy and scalability. This paper proposes an innovative architecture specifically designed to overcome this need. By exploiting Federated Learning and Edge Computing capabilities, many Long Short-Term Memory (LSTM) models are locally trained by different users based on their own historical energy consumption samples. Such models are then aggregated by a specific-purpose node to build a generalized model that is re-distributed for improved forecasting at the edge. For better forecasting, our proposed local training procedure takes as input relevant features related to calendar (i.e., hour, weekday and average consumption of previous days) and weather conditions (i.e., clustered apparent temperature), and the architecture can group users according to consumption similarities (using K-means) or socioeconomic affinities. We thoroughly evaluate the approach through simulations, showing that it can lead to similar forecasting performance than a state-of-the-art centralized solution in terms of Root Mean Square Error (RMSE), but with up to an order of magnitude lower training time and up to 50 times less exchanged data when samples are recorded at finer granularity than one hour. Nonetheless, it keeps sensitive data local and therefore guarantees users' privacy.

INDEX TERMS Energy consumption forecasting, federated learning, edge computing, LSTM.

I. INTRODUCTION

Energy consumption (or load) forecasting is a fundamental task for a sustainable planning, production and transport of electricity in modern power systems [1]. While long-term load forecasting is an important means to support the long-term planning and evolution of the power system, short-term forecasting is essential to facilitate its operations [2]. Especially, short-term load forecasting at a residential customer level is becoming more and more attractive with the ever increasing decentralization of renewable electricity generation (e.g. by photovoltaic systems) and market, being a powerful tool that can be exploited by households to maximize their self-sufficiency. However, residential demand is very volatile and quickly fluctuates over time; for this reason, this task is much more challenging than long-term forecasting or aggregated short-term forecasting [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Varuna De Silva.

In the recent years, many works have faced this challenge by proposing Artificial Intelligence (AI) methods with the explicit goal of enhancing short-term residential load forecasting as much as possible [4]. However, most of the proposed AI methods naturally require significant amounts of fine-grained historical data [5] that have to be collected and stored by the power system, in centralized locations, for an accurate model training. This operation is nowadays eased by the Advanced Metering Infrastructure (AMI), current being deployed in many countries all over the world [6]: the AMI relies on devices called *smart meters*, which are deployed at customers' premises and enable the recording of fine-grained energy consumption measurements with sampling rates of 15 minutes or more.

Even though such technological advancement unlocks new services to both users and energy suppliers [7], many privacy concerns have been raised [8] by both regulation authorities and users. In fact, such collected information can be used to disclose customers' habits [9] and, in some

countries, users can refuse the installation of a smart meter. Various privacy-preserving solutions relying on data aggregation and/or obfuscation have been proposed to ensure privacy while guaranteeing the collection of some valuable information by the energy suppliers or third parties [8], but unfortunately they are incompatible with any proposed solutions for residential short-term load forecasting, which all need households' fine-grained measurements as input data. Not less important, existing AI-based solutions are computationally intensive in the model training phase, and their adoption is far from being scalable when data coming from millions of smart meters are expected to be used in that phase. A possible alternative is training the AI model on a data subset, but this unavoidably impacts on model generalization abilities.

Edge Computing [10] is a distributed paradigm that has been introduced to offload computation closer to the end users. It extends Cloud Computing towards the edge of the network, with tremendous advantages in terms of speed, efficiency, reliability, privacy, security and scalability. Edge Computing has driven the emergence of numberless innovative applications, especially in the Internet of Things (IoT) domain [11], and appears to be the most natural choice to overcome the privacy and scalability issues that occur when an AMI is used to collect and process sampled energy consumption measurements. However, some critical aspect must be taken into account to exploit this paradigm for short-term *distributed* load forecasting, mainly concerning the inherent nature of existing AI methods, as they assure model generalization only if centrally trained by data collected from many different users. This is in contrast with the most basic Edge Computing best practice, according to which sensitive data should be kept distributed and locally processed.

Recently, Federated Learning (FL) [12] has been proposed to bridge the gap between AI and Edge Computing. Federated Learning is a distributed machine learning approach where a shared global model is trained, under the coordination of a central entity, by a federation of participating devices. The peculiarity of the approach is that each device trains a local model using its own data, which never leave the edge: only model parameters are sent to the central entity for updating the shared global model. Federated Learning unlocks the full potential of Edge Computing for machine learning purposes, and has been already successfully adopted in some application domains [13], such as human-computer interaction [14], language modeling [15], healthcare [16]–[18], transportation [19], [20] and Industry 4.0 [21], where privacy and/or scalability aspects are fundamental.

In this paper we propose a short-term distributed load forecasting architecture based on Edge Computing and Federated Learning. According to our architecture, data collected by the smart meters are kept at the edge and used to collaboratively train a global model that, once re-distributed to the customers, enhances their forecasting capabilities, since it is able to identify unknown local occurring patterns that have however been experienced by other users. The architecture relies on a Long Short-Term Memory (LSTM) [22] neural network trained following the Federated Learning schema. The neural network takes as input, in addition to energy consumption samples, other features related to calendar and to local weather conditions, namely weekday, hour, average consumption in the previous days and clustered apparent temperature, which are proven to enhance the overall training procedure. The architecture also envisions the possibility to cluster similar users (e.g. according to socioeconomic aspects or to consumption similarities) with the goal of further reducing the prediction errors.

We thoroughly evaluate our approach and show that it can drive to forecasting capabilities that are as good as those of an analogous AI-based strategy requiring data centralization, but outperforms the latter in terms of model training time, offering a far more scalable and inherently privacy-aware alternative. In addition, our strategy leads to reduced communication overhead when energy consumption measurements are recorded at fine granularity (i.e., with resolution higher than one hour). We also show that users' clustering is effective, and that our proposal is more effective than other trivial privacy-preserving schemes, as it is more adaptable to rapidly-changing consumption patterns.

A. KEY CONTRIBUTIONS

The key contributions of this paper are the following:

- We propose a novel LSTM-based architecture for short-term load forecasting based on Federated Learning and Edge Computing.
- We define two approaches for users' clustering, based on socioeconomic and consumption similarities, to improve forecasting. The former is strictly related to the dataset adopted in this study, while the latter exploits the K-Means algorithm.
- We analyze an existing dataset to extrapolate relevant calendar- and weather-related attributes to be used during local training to disclose seasonality patterns.
- We evaluate our proposed strategy in terms of forecasting performance, scalability and communication overhead against a centralized state-of-the-art solution.
- We preliminarily investigate the benefits of users' clustering and of calendar- and weather-related features on local training.
- We evaluate the impact of some tuning parameters (i.e., traces used for pre-training, number of selected nodes per round, amount of training data) on our strategy's performance.
- We compare our strategy with other two possible and trivial privacy-preserving schemes.

B. ORGANIZATION OF THE PAPER

The remainder of the paper is structured as follows. Section II recalls the related work, while Section III introduces some background notions. Section IV reports the system architecture, Section V focuses on data pre-processing and on feature extraction, while Section VI provides details

on LSTM model configuration. The performance of our proposal is evaluated in Section VII, while Section VIII concludes the paper and highlights the future work. Figure 1 reports a schematic overview of the paper: the arrows indicate inter-dependencies among (sub)sections.

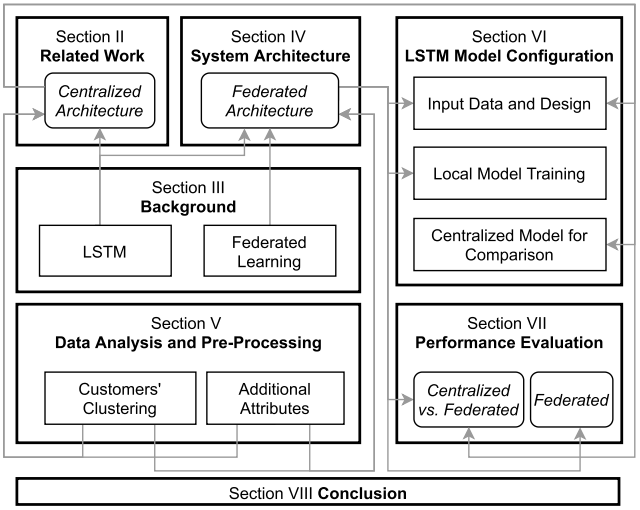


FIGURE 1. Schematic overview of the paper.

II. RELATED WORK

Short-term energy consumption forecasting has been a hot topic for many years and different techniques have been proposed to enhance prediction performance, ranging from leveraging spatial correlation [23] to exploiting enriched temporal data [4]. Another fundamental aspect is the right choice of the most suitable methodologies and models. Surveys [24]–[26] review the most relevant papers on energy consumption forecasting, with focus on data-driven models. From an analysis of the reported works, it emerges that most of them focus on non-residential scenarios and that the most widely adopted models (or methodologies) are Artificial Neural Networks (ANNs) [27]–[30], Deep Learning [31]–[35], Autoregressive Integrated Moving Average (ARIMA) and Support Vector Machine (SVM) [36], or other regression methods [37], [38].

Paper [39] compares seven different techniques for energy consumption forecasting, including ANN and Least Squares Support Vector Machine (LS-SVM). The models are trained with energy consumption measurements collected every 15 minutes and energy consumption is predicted for the next considered hour. The reported results show that ANN generally leads to better accuracy than LS-SVM in the case of load forecasting for commercial buildings, but not for residential ones. The main reason, as shown in [28], is that ANN forecasting performance is highly dependent on how parameters and hyperparameters are set, with the risk of falling in local minima during the training process. However, exploiting energy consumption related information such as activity [26] or behavioral [40] patterns can help mitigate the aforementioned issue.

Deep Learning has also been widely adopted for load forecasting. One of the most relevant works in this context is [31], which proposes an approach based on Conditional Restricted Boltzmann Machine (CRBM). The authors show that their method outperforms state-of-the-art ANN and SVM techniques. Similarly, in [32] a polling-based Deep Recurrent Neural Network is proposed, with the goal of preventing (or strongly limiting) overfitting. The approach outperforms traditional solutions such as ARIMA, Support Vector Regression and Recurrent Neural Network (RNN) approaches. Both these works show very promising results, but Deep Learning methods are usually too computationally intensive to be adopted at the edge by resource-constrained devices. Additionally, in [41] the authors propose an online adaptive method, able to continuously learn from newly arriving data and adapt to new patterns.

Recently, Long Short-Term Memory [22], an advanced and more flexible RNN architecture, has gained momentum. Being very well-suited for time-series prediction, it is the most natural candidate method to be adopted in the context of short-term load forecasting. One of the first works that adopts and assesses LSTM in this domain is [42]. The paper reveals that LSTM outperforms conventional backpropagation ANNs, since it is much more able to learn long-term temporal correlations. Another recent approach based on LSTM is [43]. The proposed solution is very flexible: it is able to forecast energy consumption with good accuracy, also when the LSTM model is used for load prediction of residential houses whose historical samples have not been included in the training set. However, such a flexibility can be guaranteed only if a huge amount of training data are used, which makes the approach demanding from a computational perspective. Some other papers propose advanced LSTM-based solutions: paper [44] applies wavelet decomposition to input data to remove unnecessary details, paper [45] proposes a hybrid LSTM and Convolutional Neural Network (CNN) model, while paper [46] focuses on joint load and price forecasting.

All the works recalled above have, as main goal, an improvement of existing prediction models to get the best possible forecasting performance, mainly considering the smart meters' energy consumption measurements as sole input feature in the training phase. However, they do not investigate how to further reduce prediction errors by exploiting (i) clustering and/or (ii) additional relevant features, which is instead the goal of the following works.

Clustering as a means to improve load forecasting has been investigated in [47]–[49]. These works show that grouping residential customers according to similarity on their energy consumption patterns is beneficial. Especially, [47] evaluates clustering when adopted in combination to different prediction models. It is shown that RNNs (and in particular LSTMs) guarantee significantly improved performance if they are trained by only using data from customers that belong to the same cluster. Instead, paper [49] groups energy customers into two clusters (related to low energy customers and high energy customers) and shows that this positively

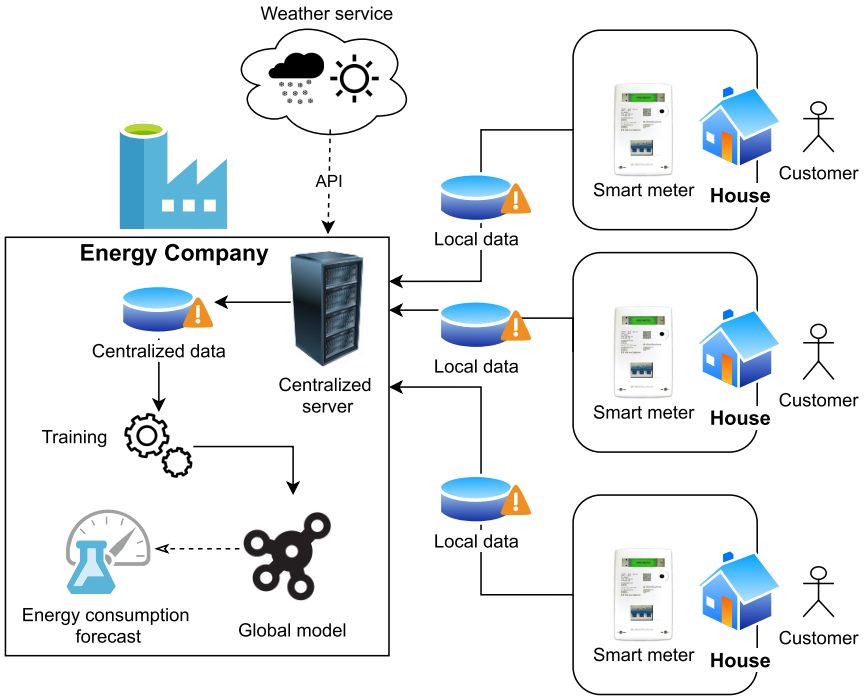


FIGURE 2. Residential short-term energy consumption forecasting: ***centralized*** architecture.

impacts on forecasting performance for both clusters. Some additional relevant features can also be used as input in the training phase. In literature, the most relevant identified features are *weather*-related [48], [50]–[52] and *calendar*-related [48], [53]. Features such as temperature, humidity (weather-related), day of the week or month of the year (calendar-related), if given as input in the training phase, can drive it to output more generalized models.

Although the machine-learning-based solutions recalled so far are different from many points of view, they all have a common architectural aspect: they require a centralized entity that collects energy consumption measurements from the customers to centrally train a global model. In other words, all the works implicitly or explicitly adopt an architecture as the one depicted in Fig. 2 that, from now on, we will call *centralized architecture* and consider as benchmark.

Only one very recent work can be found in literature that, similarly to this paper, adopts Federated Learning for load forecasting at the edge [54]. As that work, we also rely on LSTM and envision the possibility to distribute the training burden among multiple edge nodes. However, paper [54] considers energy consumption as the sole input feature of the training process, while we include additional features related to weather and calendar, and we propose a strategy that relies on customers' clustering. These two advancements are demonstrated to enhance the forecasting precision. Additionally, for the first time, we carry on an extensive performance comparison with the centralized approach shown in Fig. 2, adopted in all the other previous works.

III. BACKGROUND

In this Section we briefly recall the most important concepts related to LSTM and Federated Learning.

A. LONG SHORT-TERM MEMORY

A Long Short-Term Memory is a machine learning model belonging to the family of ANNs or, more specifically, RNNs. It is widely adopted to solve sequence classification problems and is very suitable for time series prediction. Before describing more in detail LSTM, we briefly describe the structure of ANNs and RNNs, highlighting the differences.

An ANN includes different neurons organized in sequential layers. A neuron is the atomic unit of ANNs: it applies a function on the input data (usually a weighted sum) and, later, it passes the obtained value through an activation function (that is, a threshold function). The result is then forwarded to other neurons. Each neuron is associated to multiple *weights* w_j (being j the index of each weight) that are adopted in the weighted sums and related to neurons' interconnections. The goal of the training phase of an ANNs is deciding the most appropriate model's weights w_j to maximize the performance of the network to accomplish its specific task. In an ANN, the output of a neuron in a layer is always used as input to one or more neurons in the next layer. This feeding mechanism is called *feed-forward*. An ANN includes three types of layers: an *input layer*, which receives the input data and performs a first elaboration step; one or more *hidden layers*, whose neurons elaborate input from previous layers and forward the result to the next layers; an *output Layer*, whose duty is to produce the final result.

An RNN is a generalized ANN where neurons in a layer can also be interconnected with neurons of previous layers, of the same layer or with themselves in a loop. An important feature of RRNs is that they maintain and use an internal state to capture dynamics over time t. However, typical RRNs have only short-term memory capabilities.

An LSTM [22] is a specific and more complex type of RNN, whose elementary unit is called *LSTM cell* (instead of *neuron*) and that has long-term memory capabilities. In an LSTM cell the internal state is modelled by two vectors: $h(t)$ is the short-term state, always equal to the cell output $y(t)$ in any instant t, while $c(t)$ is the long-term cell state, which is maintained and updated over time. Each LSTM cell includes three gates, whose duty is to add/remove information to/from the cell state $c(t)$ and to compute the cell output $y(t)$: the *input gate* decides what information should be kept from current input $x(t)$ to compute the current state $c(t)$; the *forget gate* decides what information should be kept and what information should be thrown away from the previous state $c(t-1)$ to compute $c(t)$; the *output gate* finally decides the output $y(t)$ and the current state $c(t)$, which will be given as input to the cell in the next time instant $t+1$.

B. FEDERATED LEARNING

Federated Learning is used to train a machine learning model in a distributed way, where different remote devices use their own collected data to carry on a local training procedure and a centralized server is then in charge of aggregating the trained local models into a global model, which is in turn re-distributed to the remote devices for a further training round. Following this iterative process, an arbitrarily high number of devices can concur to model training without the need of transferring collected data to a centralized location, since only locally-trained models need to be sent. Federated Learning has been proven to work well also when the remote devices train the model using non independent and identically distributed (non-iid) data. Moreover, it has been demonstrated to bring benefits in terms of amount of exchanged data with respect to a solution requiring data transmission to a single location for centralized training [12]. In the following, we will provide some information on the training workflow and on the local model aggregation procedure.

1) TRAINING WORKFLOW

While a detailed description of a typical FL training workflow is reported in [55], here we just recall the main aspects. Time is slotted in *rounds* and two entities are involved in the workflow: the *end devices* and the *centralized server*. Specifically, the following steps are carried out (see Fig. 3):

1) The centralized server initializes the machine learning model to be trained by randomly choosing the model's weights, and sends the chosen model type (such as ANN, RNN, LSTM, etc.) and characterizing hyperparameters (e.g. number of layers, number of neurons, etc.) to the end devices.

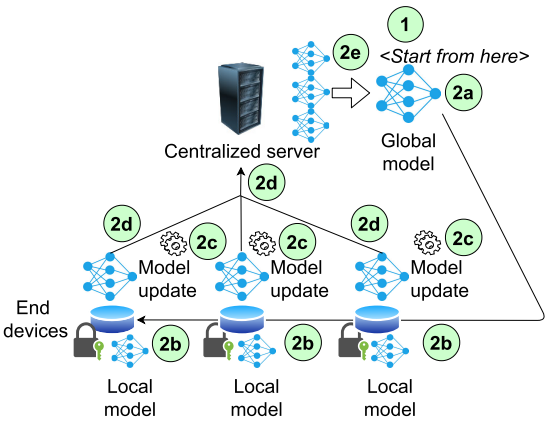


FIGURE 3. Federated learning: training workflow.

2) The training phase starts. For each round, the following steps are executed:
   a) The server selects a fraction C of end devices.
   b) The server sends to the selected end devices his global model (i.e., weights) as trained so far.
   c) The edge devices train the received model using their local data.
   d) The edge devices send the trained local model back to the server, in the form of updated weights.
   e) The server aggregates all the received local models (i.e., it computes the new weights from the updated weights received by any end device) and generates an updated global model.

How end devices are selected in each round depends on the specific application, as thoroughly investigated in [56]. Another key aspect is how *aggregation* at the centralized server occurs. In the following, we will briefly recall the most widely-adopted aggregation procedure, called Federated Averaging (FedAVG).

2) FEDERATED AVERAGING

FedAVG, introduced in [12], aims at minimizing the global loss function. FedAVG is the generalization of another aggregation procedure, called FedSGD [12]. In FedSGD, the selected end devices, in each round, perform a single step of Stochastic Gradient Descent (SDG) and send the obtained model weights to the server. The server then averages the received weights proportionally to the number of locally-used training samples: this operation can be seen as a gradient descent step on the global model. One of the drawbacks of such an approach is that, by executing only a single SDG step per round, global model training slowly converges.

FedAVG overcomes this limitation by introducing the concepts of *local epoch* and of *batch*. In each round, multiple local epochs are executed on a batch (i.e., a subset of local data): for each local epoch and batch an SDG step is done and, in this way, less server-device interaction is needed. Then, the server averages the received weights proportionally to the number of locally-used training samples as done by FedSDG.

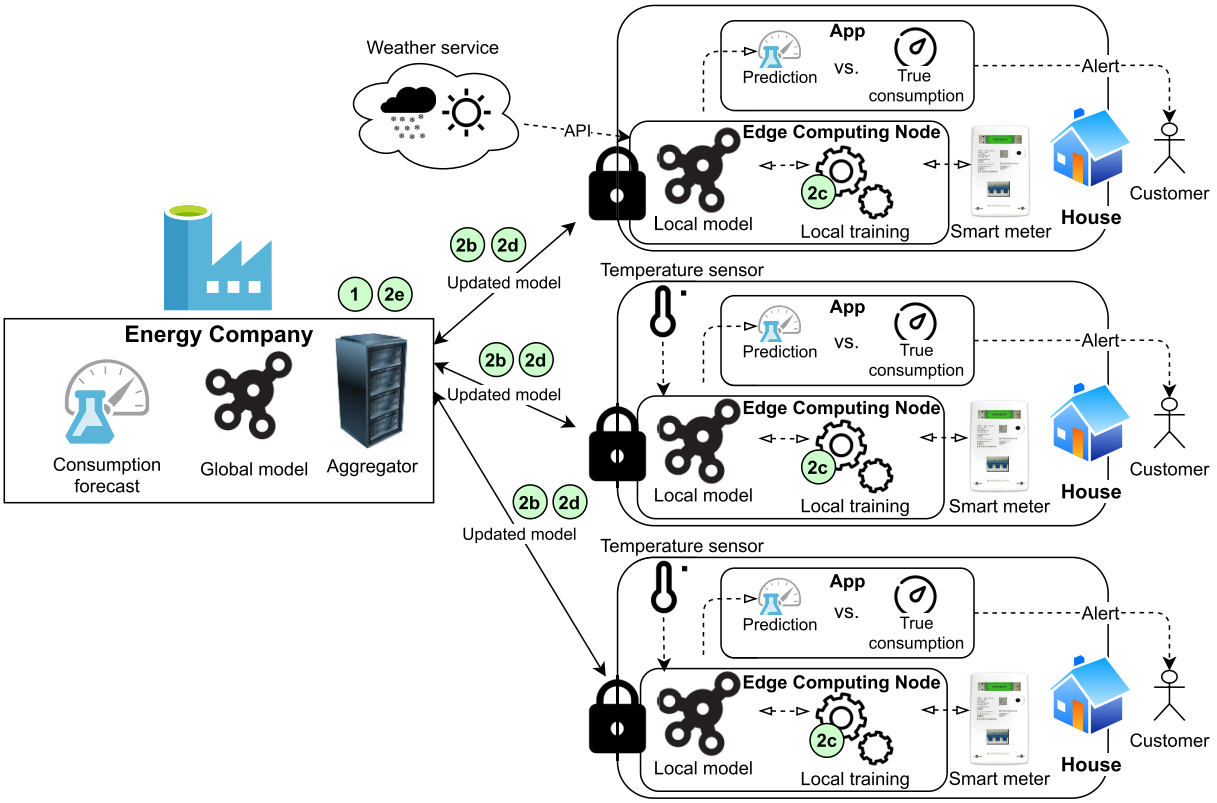


FIGURE 4. Residential short-term energy consumption forecasting: *federated* architecture.

Three parameters need thus to be carefully tuned to achieve the best possible FedAVG performance:

- N_{round}: overall number of rounds to be executed;
- N_{epoch}: number of local epochs to be executed by each selected remote device in a round;
- B_{size}: amount of local samples that are used for training in each round/epoch (batch size).

The reader should refer to Tab. 3 for a recap of all the parameters and symbols adopted in this paper.

IV. SYSTEM ARCHITECTURE

In this section we introduce our federated short-term energy consumption forecasting architecture (in brief, *federated architecture*, Fig. 4). The proposed architecture enables the possibility to accurately forecast households' energy consumption by collaboratively training an LSTM network. Local LSTM networks are trained by a multitude of edge devices through locally-generated energy consumption measurements, while centralized model aggregation and re-distribution procedures, as envisioned by Federated Learning, are adopted to get a trained global LSTM network. Two entities are considered in our architecture:

- *Energy Company:* it supplies electricity to residential customers. It may rely on a Smart Grid [57] for enhanced energy supply.
- *Residential Customers:* they consume electricity in their houses and are billed by the energy company.

Three functional nodes, defined by taking inspiration from already-existing functional nodes of Federated Learning (see Section III-B), are included and interact:

- *Smart Meter:* it is owned by the energy company and installed in any user premises. It records energy consumption measurements (i.e., samples) at very fine granularity (up to one per minute) [58].
- *Edge Computing Node:* it is either a specific-purpose (e.g. a GPU board) or a generic-purpose (e.g. a PC) computing node that can be owned by the energy company (and installed at the customer's premises) or by the customer herself. It is connected to the Smart Meter through the most appropriate wireless (e.g. Zigbee) or wired (e.g. Ethernet) network connection and is responsible for storing energy consumption measurements as captured by the Smart Meter. Additionally, it is also either connected to outdoor sensors able to capture *weather* variables (e.g. relative humidity, temperature, wind speed, etc.) or it is able to retrieve less accurate data from external weather data repositories. The Edge Computing Node is in charge of local model training, using the collected data as recalled above and, additionally, *calendar* information (more details on considered input features will be given in Section V). It is also connected to the Aggregator (see below) for model exchange, and can interact with customer's end devices (e.g. a smartphone) by means of specific-purpose

Algorithm 1 FL-Based Model Training Procedure

n_{nodes}: number of Edge Computing Nodes (indexed by i)
$n_{pretrain}$: number of Edge Computing Nodes used for pre-training
N_{round}: number of rounds (indexed by t)
$w_{j,t}$: weight j of the global LSTM model at round t
$w^i_{j,t}$: weight j of the local LSTM model as computed by i at round t
C: fraction of Edge Computing Nodes chosen in each round t
n^i_t: overall number of samples used in round t to locally train the LSTM model by i
N_{epoch}: number of local epochs
B_{size}: local batch size
LR_{edge}: learning rate of Edge Computing Nodes

Aggregator:
$w_{j,0} \leftarrow$ Initialize random $\forall j$
$w_{j,0} \leftarrow$ Pre-train with $n_{pretrain}$ Edge Computing Nodes $\forall j$ * *Optional* *\
for *each round* $t = 1, 2, \ldots, N_{round}$ **do**
  $m \leftarrow \max(C \cdot n_{nodes}, 1)$
  $S_t \leftarrow$ Random set of m Edge Computing Nodes
  Send $w_{j,t}$ $\forall j$ to Edge Computing Nodes $i \in S_t$

  Edge Computing Nodes: * *FedAVG* *\
  for *each Edge Computing Node* $i \in S_t$ **do**
    for *each local epoch* $l = 1, \ldots, N_{epoch}$ **do**
      for *each batch* $b_i = 1, \ldots, \left\lceil \frac{n^i_t}{B_{size}} \right\rceil$ **do**
        $\nabla^i_l \leftarrow$ Gradient computed by SDG
        $w^i_{j,t} \leftarrow w^i_{j,t} - LR_{edge} \cdot \nabla^i_l \quad \forall j$
    Send $w^i_{j,t}$ $\forall j$ to Aggregator

  Aggregator: * *FedAVG* *\
  $w_{j,t+1} \leftarrow \sum_{i=1}^{m} \frac{n^i_t}{\sum_{i=1}^{m} n^i_t} \cdot w^i_{j,t} \quad \forall j$

applications used for data visualization and as an end point for the *alarms* generated by the Edge Computing Node (see Section IV-B).

- *Aggregator:* it is a centralized server owned by the energy company. Its main responsibility is to gather local models at the end of each round, as trained by multiple the Edge Computing Nodes, and aggregate them. The output of this operation is a global model, which is then re-distributed to Edge Computing Nodes. The Aggregator communicates with the Edge Computing Nodes through encrypted end-to-end connections.

Note that the proposed architecture could be adopted also if other entities (e.g. a statistical institute) instead of the energy company are involved. In this case, the customer should be equipped with an energy meter sensor connected to its magnetothermic switch, as she could not directly access the data collected by the Smart Meter.

A. MODEL TRAINING PROCEDURE

The proposed training procedure adapts the workflow shown in Section III-B1: Fig. 4 reports the steps as numbered and described in that Section, while Algorithm 1 details the training operations. FedAVG is used for model update and aggregation (see [12]). With respect to the workflow reported in Section III-B1 we envision the possibility, once the LSTM model is initialized and before starting the FL-based training procedure, to centrally *pre-train* LSTM with some available samples collected by the energy company, e.g. from users that, in change of discounted energy bills, are keen to share them. Pre-training will be evaluated in Section VII.

The training workflow could potentially be executed continuously and indefinitely, so that the obtained model can be constantly refined (i.e., re-trained) with new collected data from the Smart Meters. A sliding window (e.g. of 365 days) can be implemented, so that not too old historical data is considered while re-training. Given the potentially high number of Edge Computing Nodes involved in the process (i.e., millions), if a low fraction C of them is randomly chosen in each round (e.g. $C = 10^{-4}$), the computational burden on each Edge Computing Node can be kept low. To reduce even further the computational effort, the training procedure can be executed periodically (e.g. every few weeks) for a limited amount of rounds N_{round}. In Section VII we will show that this is good enough to have an always-updated model for forecasting.

It is also important to specify that the energy company or any other entity that ensures weights aggregation and training coordination through its Aggregator, has at its disposal an always-updated model, which could be offered as-a-service to third parties or used to finely adapt energy transport and/or production strategies.

B. REAL-TIME ENERGY CONSUMPTION FORECASTING

Once the LSTM network has been trained following the procedure shown in Alg. 1, it can be used for real-time and short-term energy consumption forecasting at the edge. The ability to forecast the load in the near future (e.g. in the next hour) makes it also possible to locally notify customers about any possible consumption *anomaly* [59]. If $\hat{Y}_t$ is the forecast value at time t and Y_t the real recorded value, an *alert* is generated to the customer's application if $Y_t - \hat{Y}_t > T_h$, where T_h is a given threshold. Such an alert, occurring when an excessively high consumption has been experienced, warns the customer about this abnormal behavior. Conversely, if $\hat{Y}_t - Y_t > T_l$, a *notification of good behavior* is generated: this kind of notifications make the customer aware of a positive trend in her actual consumption, which is significantly lower than expected.

Such a local notification system can be used as a tool to increase customers' awareness on their good and bad habits so that they can consequently adapt their behavior. Ideally,

a customer should try to minimize the number of received alerts, while instead maximizing the number of notifications of good behavior. Clearly, setting the proper value for T_h and T_l is of paramount importance to avoid the generation of unneeded notifications or to spot exceptional anomalous behaviors. For instance, the value of T_h and T_l may be changed through the customer's application during vacation times. T_l can be lowered to avoid false notifications of good behavior, while T_h can be lowered to spot anomalous activities in the house, that should not be considered anomalous outside vacation times. How to adjust the thresholds is however out of the scope of this paper.

C. CUSTOMERS' CLUSTERING

Forecasting can be improved by exploiting similarities between time series through appropriate clustering. In our architecture we foresee this possibility: for instance, by means of aggregate energy consumption data (e.g. those used for billing purposes) or by means of socioeconomic aspects (if available), the energy company can group customers in k different clusters. If this is done, the proposed architecture just needs to be replicated k times, with k different LSTM networks that have to be concurrently trained and with the Aggregator that is in charge of customers' clustering and of initializing and correctly updating the k LSTM networks.

V. DATA ANALYSIS AND PRE-PROCESSING

In this Section we describe and analyze how data are pre-processed before giving them as input to the training process. We also show how energy consumption measurements can be enriched with additional attributes, with the final goal of improving forecasting performance. To make analysis and pre-processing tasks concrete, in this work we consider a set of fine-grained energy consumption measurements as taken from a large dataset collected and published by the energy company UK Power Networks [60]. The dataset includes data collected by 5,567 Smart Meters in London between November 2012 and February 2014, expressed in kWhh (kW per half-hour) and with a granularity of 30 minutes.

Two peculiarities of the dataset mainly motivate its adoption in this work. First, each *trace* (i.e., time series collected by a Smart Meter) is associated to a category as specified by the Classification Of Residential Neighborhoods' (ACORN) standard [61]. ACORN is a consumer classification that segments the UK population into different demographic types according to social factors and population behaviors; demographic types are then grouped into 18 different *macro-groups*. Second, the dataset has already been enriched with different meteorological variables recorded in London [62] and collected through DarkSky API [63] for the whole covered period. This greatly simplifies the extraction of weather-related features as required by our architecture.

Even though the remainder of the Section is tailored on the UK Power Networks dataset, the proposed procedures can be straightforwardly applied to other available and more recent datasets, such as Dataport [64].

A. DATA SELECTION

We focused on data collected from January 1st, 2013 to December 31th, 2013. Among all the 5,567 Smart Meters, only 4,968 where active in the reference period. We then removed from the set some outliers with abnormal energy consumption patterns: we considered as outliers all those traces that have an average consumption that is either lower than 0.6 kWhh or higher than 3.6 kWhh. These outliers indicate potential errors in the metering activity or are related to empty houses. The remaining Smart Meters are 4,329.

We decided to reduce even further the number of selected Smart Meters to ensure tractability, given the limited amount of computational resources available for our tests. We randomly selected 35% of the traces for each one of the 18 ACORN macro-groups, so that the final dataset preserves original ACORN proportions. The number of considered traces is then reduced to 1,507, distributed among ACORN macro-groups as indicated in the left-hand side of Tab. 1.

TABLE 1. Clustering methods and output.

ACORN		K-Means	
Cluster name	# Meters (n_{traces})	Cluster name	# Meters (n_{traces})
ACORN-A	33	K-MEANS-A	43
ACORN-B	7	K-MEANS-B	64
ACORN-C	43	K-MEANS-C	98
ACORN-D	71	K-MEANS-D	85
ACORN-E	402	K-MEANS-E	93
ACORN-F	196	K-MEANS-F	109
ACORN-G	56	K-MEANS-G	122
ACORN-H	134	K-MEANS-H	76
ACORN-I	14	K-MEANS-I	113
ACORN-J	27	K-MEANS-J	83
ACORN-K	47	K-MEANS-K	73
ACORN-L	99	K-MEANS-L	100
ACORN-M	30	K-MEANS-M	85
ACORN-N	45	K-MEANS-N	80
ACORN-O	29	K-MEANS-O	79
ACORN-P	32	K-MEANS-P	103
ACORN-Q	236	K-MEANS-Q	51
ACORN-U	6	K-MEANS-U	50
Total	**1,507**	**Total**	**1,507**

Finally, to further reduce the dataset size while preserving seasonality patterns, we aggregated data samples to obtain a trace with coarser granularity of 1 hour: measurements are then expressed in kWh. An example of processed trace is shown in Fig. 5.

B. CUSTOMERS' CLUSTERING

As shown in the previous subsection, the UK Power Networks dataset inherently provides traces' clustering according to customers' behavioral and socioeconomic aspects (ACORN). As an alternative strategy, we decided to cluster the traces only according to consumption similarities: to do so, we adopted the *K-Means* [65] method. For all the selected 1,507 traces we computed the following features on all the

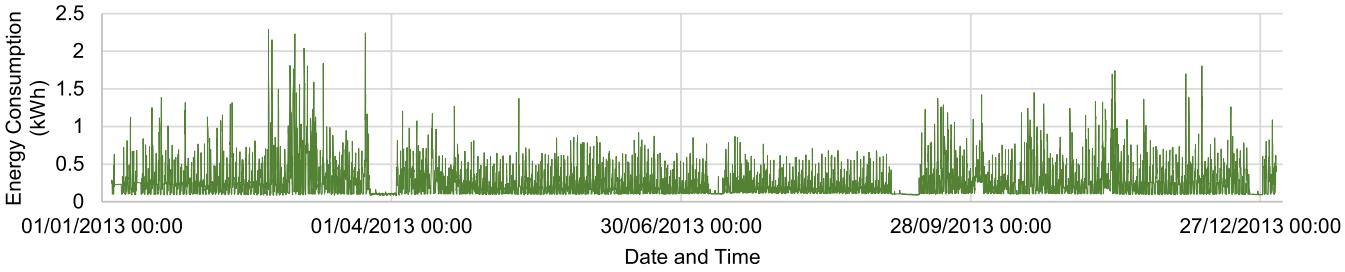


FIGURE 5. Trace example (yearly data with 1 hour resolution).

TABLE 2. Final dataset format for each trace.

Timestamp	Energy consumption (kWh)	Weekday	Hour	AVG4D (kWh)	TempCluster
⋮	⋮	⋮	⋮	⋮	⋮
2013-01-04 0:00:00	0.2600	4	0	0.2006	1
2013-01-04 1:00:00	0.2902	4	1	0.2714	1
2013-01-04 2:00:00	0.2435	4	2	0.2975	1
2013-01-04 3:00:00	0.2490	4	3	0.3036	1
⋮	⋮	⋮	⋮	⋮	⋮

energy consumption samples (8,760 as we focus on hourly granularity for a whole year) of each trace, which were provided as input to K-Means:

- Energy consumption median (kWh);
- Energy consumption average (kWh);
- Energy consumption sum (kWh);
- Weekday with average highest energy consumption, encoded by integers from 0 (Monday) to 6 (Sunday);
- Weekday with average lowest energy consumption, encoded by integers from 0 (Monday) to 6 (Sunday);
- Highest recorded energy consumption (kWh);
- Lowest recorded energy consumption (kWh).

These features disclose less information on customers' behavior than the fine-grained traces, and may be shared with third parties such as the energy company. We chose $k = 18$, so that 18 different clusters are obtained as in the ACORN-based clustering. In this way, it is possible to fairly compare the two clustering methods. How traces are distributed among clusters by K-Means is reported in the right-hand side of Tab. 1: it is easy to see that K-Means distributes traces more evenly among clusters than ACORN.

C. ADDITIONAL ATTRIBUTES

In this subsection we describe how we enriched the dataset with additional attributes than the default ones already available from [62], which are the acquisition time of each sample (*timestamp*) and the hourly *energy consumption* value (see the first two columns of Tab. 2).

Two effective attributes related to calendar that help the LSTM model learn daily and weekly patterns are the *weekday* and the *hour* of the day. As done in the previous subsection, we encoded the weekday as an integer ranging from 0 (Monday) to 6 (Sunday). Concerning the hour, it can be extrapolated from the timestamp and encoded as an integer ranging from 0 to 23. An example for these attributes is shown in the third and fourth columns of Tab. 2.

In the following, we will detail how we obtained other two important attributes, namely *AVG4D* and *TempCluster* (see the last two columns of Tab. 2), which are related to calendar aspects and weather conditions respectively.

1) AVG4D ATTRIBUTE

This attribute is computed and added to the dataset so that the LSTM model can learn energy consumption differences between weekdays and weekends, and existing similarities among different consecutive days at the same hour. To compute AVG4D we took inspiration from [66], which proposes a linear regression technique taking as input the energy consumption at hour t of the previous N days to estimate the energy consumption at hour t of the current day. They have shown that the best prediction performance is obtained when $N = 4$. Even though we adopt a different method for forecasting, adding the average energy consumption of the four previous days, at the same hour, to each sample helps the model learn recent energy consumption trends at different hours of the day, and flattens any possible abnormal consumption at the same hour in the closer days.

The definition of ''four previous days'' is however not univocal in the computation of AVG4D. By analyzing the dataset, we realized that the average daily energy consumption is slightly lower in weekdays (Monday to Friday) than in weekends (Saturday and Sunday), and that the consumption is similar to that of close previous days, at the same hour, that fall within the same category. We thus decided to consider as four previous days the ones that fall in the same category of the current day (i.e., weekdays or weekends). Figure 6 better explains this point: if, for instance, the current day is a Sunday (weekend), the four considered previous days to compute

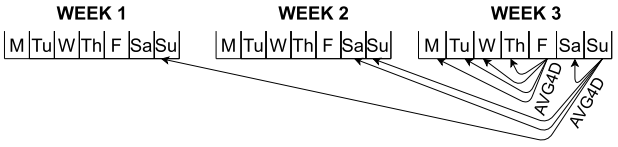


FIGURE 6. Computation of the AVG4D attribute.

AVG4D are the day before (Saturday), Sunday and Saturday of the previous week, and Sunday of two weeks before. Instead, if the current day is a Friday (weekday), the considered days to compute AVG4D are Thursday, Wednesday, Tuesday and Monday of the same week.

AVG4D has thus generally higher values during weekends than weekdays and, besides explicitly providing the information on average consumption in close days at the same hour, implicitly discloses weekends-weekdays patterns that could also be extrapolated by an additional attribute exploiting an one-hot encoding of weekdays and weekend days.

2) TEMPCLUSTER ATTRIBUTE

Seasonality patterns are a key aspect that needs to be considered. In the previous subsections, we introduced some attributes (i.e., Weekday, Hour and AVG4D) that help, in the model training phase, capture any daily or weekly seasonal pattern. The TempCluster attribute is instead introduced to capture yearly seasonal patterns.

By analyzing the available dataset at a first glance, we realized that energy consumption is generally higher in winter than in summer. This is most probably due to greater presence in houses, less available solar light and usage of electrical heating systems. Additionally, being the dataset related to customers from London, where summer is not much warm, it is reasonable to assume that more energy is needed to heat the environment in winter than to cool it in summer. More in general, it looks clear that energy consumption is greatly correlated with weather conditions, as already investigated in some previous works (see Section II).

We more thoroughly analyzed the dataset that, as already mentioned, is accompanied with meteorological variables recorded in London using the DarkSky API. The included variables are: maximum external temperature, relative humidity, visibility, wind speed, dew point and ultraviolet (UV) index. For each of the considered traces as selected in Section V-A, we computed the Pearson correlation coefficient (i) between energy consumption and all the meteorological variables and (ii) among meteorological variables. The resulting correlation matrices (one per trace) were then averaged and the average correlation matrix is reported in Fig. 7. Not surprisingly, it can be seen that energy consumption is highly negatively correlated with the maximum external temperature. It also has a high negative correlation with dew point and UV index but, being them highly positively correlated with the maximum external temperature, we neglected it to avoid multicollinearity. The relative humidity has a moderate correlation with energy consumption, while the wind speed

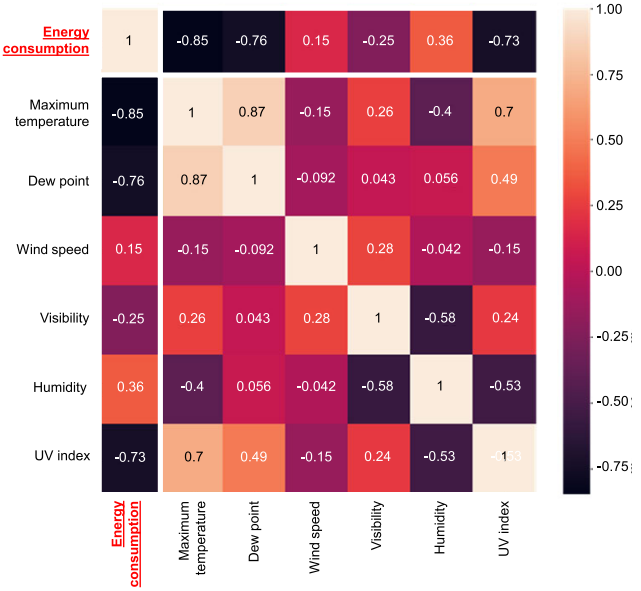


FIGURE 7. Correlation matrix for various meteorological variables and energy consumption.

has a low correlation. Moreover, visibility has a high negative correlation with relative humidity and we neglected it to avoid multicollinearity.

Given the above analysis, the meteorological variables that to some extent are correlated to energy consumption and worth being considered are *maximum external temperature*, *relative humidity* and *wind speed*. Including these three variables as stand-alone attributes to the dataset would add unnecessary complexity to the input data for two reasons.

First, all the three variables contribute to the *apparent temperature* (AT) in shade, which can be compute as [67]:

$$AT = T_{ext} + 0.33 \cdot e - 0.70 \cdot ws - 4 \tag{1}$$

where T_{ext} is the maximum external temperature (°C), ws is the wind speed (m/s) and e is the water vapour pressure (hPa), computed as:

$$e = \frac{rh}{100} \cdot 6.105 \cdot e^{\frac{17.27 \cdot T_{ext}}{237.7 + T_{ext}}} \tag{2}$$

where rh is the relative humidity. To simplify without losing much information, we then further consider only the apparent temperature as weather-related attribute.

Second, the apparent temperature (as its component variables) is very volatile but, clearly, minimal changes among hours are expected not to have any noticeable effect on energy consumption. Conversely, including the hourly apparent temperature would add avoidable noise during the training phase. This pushed us to reduce the granularity of the attribute by clustering it through the K-Means algorithm. We gave as input feature the sole apparent temperature for the whole set of samples in the reference period (i.e., 8,760 samples). To decide the optimal number of clusters k we used the *elbow method* [68], which indicated that the best number of clusters

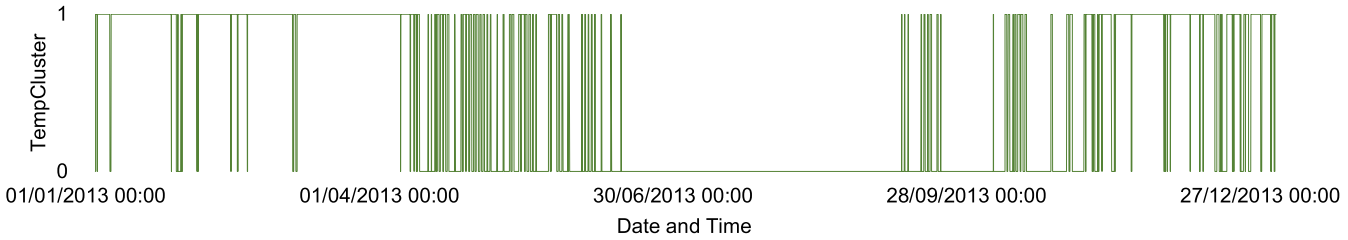


FIGURE 8. Value of TempCluster attribute for all the 8,760 hours of year 2013 (from 00:00 of January 1st to 23:00 of December 31st).

is $k = 2$. This means that an attribute disclosing whether the apparent temperature, in a specific hour of the day, is ''cold'' or ''warm'' is a good-enough weather-related attribute to be given as input to the LSTM model in the training phase.

The output of the aforementioned clustering operation is the attribute that we called TempCluster. For each hour of each day of year 2013, we encoded all the apparent temperatures belonging to the ''cold'' (resp. ''warm'') cluster with 1 (resp. 0): the result is shown in Fig. 8. The figure shows that TempCluster has value 1 for most of the hours in winter, has value 0 for all the summer hours, while it frequently swaps between 0 and 1 in mid seasons (i.e., spring and autumn).

VI. LSTM MODEL CONFIGURATION

This section aims at describing how input data are modelled and how LSTM hyperparameters are chosen in the federated architecture, while also highlighting the differences with the model adopted in the centralized architecture for comparison.

A. INPUT DATA AND DESIGN

At time instant t, a number $X = |\chi|$ of historical samples is given as input to the LSTM model (including the value recorded at t) to forecast the energy consumption at $t + 1$, where $\chi = \{0, \ldots, X - 1\}$. In our experiments, we set $X = 24$: this means that at time t samples at times t, $t - 1, \ldots, t - 23$ are used for forecasting. In other words, our system implements a sliding window with look-backs of size 24 samples (i.e., one day) and look-ahead of size 1 sample.

The model adopted in this work consists of four layers (two hidden) and is similar to the one adopted in [43]:

1) The *first layer* (input layer) takes as input the $X = 24$ historical samples.
2) The *second layer* (hidden layer) includes 32 LSTM cells. The hyperbolic tangent (*tanh*) is chosen as an activation function.
3) The *third layer* (hidden layer) takes as input the output of the 32 LSTM cells of the second layer and includes 16 LSTM cells adopting *tanh* as an activation function.
4) The *fourth layer* (output layer) takes as input the output of the 16 LSTM cells of the second layer and consists of a dense layer, with only one output unit: the output unit value represents the energy consumption forecast for the next hour.

B. LOCAL MODEL TRAINING

We consider a different number of energy consumption samples S, ranging from $S = 2,880$ (four month) to $S = 7,920$ (11 months) for each involved Edge Computing Node. As additional features, we include a subset of the attributes described in the previous Section and reported in Tab. 2. Specifically, we include *weekday*, *hour*, *AVG4D* and *TempCluster*, for a total of $F = 5$ features. We instead do not include *timestamp* since it is a simple text string, and meaningful information has been already extrapolated from it (i.e., the *hour* attribute).

We chose the hyperparameters for model training after careful manual tuning using partial data from the constructed dataset. As seen in Section III, the three most important parameters that have to be tuned to ensure good FedAVG performance are the number of local epochs N_{epoch}, the batch size B_{size} and the number of rounds N_{round}. Considering that Edge Computing Nodes are expected to have relatively limited processing capabilities, we set $N_{epoch} = 5$, $B_{size} = 100$ and $N_{round} = 50$. This design choice ensures a limited burden on the devices when they are selected for local training but, at the same time, good overall forecasting performance, as we will evaluate later. We chose the Mean Absolute Error (MAE) as evaluation metric for the loss function, and we adopt the Adaptive Moment Estimation (Adam) variant of Stochastic Gradient Descent as optimization algorithm, with a learning rate $LR_{edge} = 0.0001$.

We employ two well-known regularization methods to avoid or limit overfitting:

- *Dropout:* it consists on randomly neglecting some LSTM cells during each training step. We adopt this technique during local training at the Edge Computing Nodes for any of the two LSTM layers, and we set as dropout rate $DR_{rate} = 10\%$, meaning that 10% of LSTM are neglected at any training step. This helps the process escape from local minima.
- *Early Stop:* it consists on stopping the training operation after the execution of a certain number of rounds, i.e., when the loss function does not improve anymore on validation data: we adopt this technique at the Aggregator on the global model. Early Stop relies on a counter that is incremented when loss function worsens with respect to the previous round. The training procedure is stopped when the counter reaches a value $ES_{thresh} = 3$ since it is likely that, when this condition

TABLE 3. Recap of adopted parameters (unless otherwise specified).

Name	Description	Federated (Value)	Centralized (Value)
	Input data and LSTM model structure		
X	Historical samples for forecasting	24 (one day)	
Layers	Number and type of layers	4 (1 input, 2 hidden, 1 output)	
(LSTM) cells	Number of cells	24 (input), 32 (1st hidden), 16 (2nd hidden), 1 (output)	
	Model training		
Features	Input features	Energy, Weekday, Hour, AVG4D, TempCluster	
S	Input samples per trace	$\{2{,}880; \ldots; 7{,}920\}$	(from 4 to 11 months)
N_{epoch}	Number of (local) epochs	5	50
B_{size}	Batch size	100	
N_{round}	Max number of rounds	50	-
C	Selected Edge Computing Nodes per round	100%	-
LR_{edge}	Learning rate (Edge Computing Nodes)	0.0001	-
$n_{pretrain}$	Traces in pre-train	0	-
LR	Learning rate (global)	-	0.0002
DR_{rate}	Dropout rate	10%	
ES_{thresh}	Early Stop threshold	3	
	Simulation parameters		
Traces	Overall number of traces	1,507	
Clusters (ACORN)	Number of ACORN clusters (traces)	18 (see Tab. 1)	
Clusters (K-Means)	Number of K-Means clusters (traces)	18 (see Tab. 1)	
n_{traces} (Random)	Traces of Random	80	
Traces training set	% traces in training	70%	
Traces test set	% traces in test	30%	
N^{ES}_{round}	Average rounds before Early Stop	30	
S_{model}	Size of the LSTM model	0.032 Mb	
S_{data}	Size of the dataset	0.36 Mb	

occurs, no further loss function improvement can be obtained.

C. CENTRALIZED MODEL FOR COMPARISON

For comparison purposes, we designed and implemented an LSTM model for the state-of-the-art centralized architecture depicted in Fig. 2. The model structure is analogous to the one described in Section VI-A and includes four layers, as done in the state-of-the-art work [43], which embraces a centralized architecture and is thus considered as benchmark: an input layer, two LSTM hidden layers including 32 and 16 LSTM cells respectively, and an output dense layer with one output unit. All the hyperparameters were chosen as specified in Section VI-B with only two differences:

- Given the centralized nature of the learning approach, time is not slotted in rounds, so the N_{round} parameter is not considered. Instead, the maximum number of training epochs N_{epoch} at the centralized server is set to $N_{epoch} = 50$, and Early Stop is used among epochs.
- The learning rate LR_{edge} is not considered. Instead, a global learning rate is specified, i.e., $LR = 0.0002$. This value has been chosen after careful manual tuning.

We designed the LSTM model adopted by the centralized architecture as similar as possible to the one adopted by the federated architecture, and we set the hyperparameters for both approaches with the goal to ensure similar forecasting performance, so that the solutions can be fairly compared on other aspects, as we will show in the next Section.

VII. PERFORMANCE EVALUATION

In this Section we thoroughly evaluate the proposed federated architecture. Unless otherwise specified, we set the parameters as recalled in Tab. 3. We adopted the TensorFlow Federated framework[1] and the Python-based Keras API[2] to implement the LSTM model and test the solution. We used the Google Colab Cloud platform[3] to gather results.

A. TRAINING AND TEST SET CONSTRUCTION

We group the dataset traces according to three criteria:

- *ACORN Clustering:* they are grouped as specified in Section V-A and as reported in the left-hand side of Tab. 1; training and testing are performed per-cluster.
- *K-Means Clustering:* they are grouped as specified in Section V-B and as reported in the right-hand side of Tab. 1; training and testing are performed per-cluster.
- *Random:* a random set of traces, among the 1,507 traces selected as described in Section V-A, is chosen. Unless otherwise specified, such a set will include $n_{traces} = n_{nodes} = 80$ traces, which is the average cluster size. The definition of such a set allows us to test the proposed solution when customers' clustering is not performed.

For any cluster/random set defined as specified above and including n_{traces} traces, we follow the methodology depicted in Fig. 9(a) to define the training and test sets. We randomly include the samples of 70% of the traces in the training set and the samples of 30% of the traces in the test set. Training is performed on data collected from January to November, while testing is done on data collected in December, which is thus considered as testing month unless otherwise specified. As introduced in Section VI, not all the samples of the traces selected for training are included in the training set, as shown

[1] https://www.tensorflow.org/federated
[2] https://keras.io
[3] https://colab.research.google.com

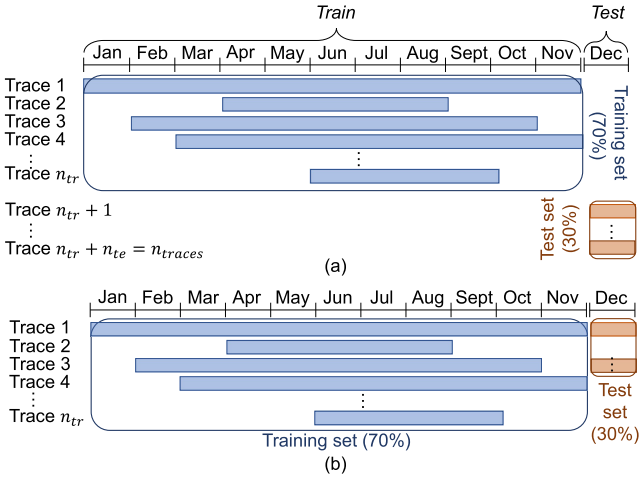


FIGURE 9. Adopted methodology for the creation of training and test sets. In (a) the test traces are not included in the training set, while in (b) they are.

in the figure: for each trace, only samples related to a random number of months between 4 and 11 is selected. We decided to follow this approach to limit computational needs, instead of further reducing the number of traces in the training set, because we expect that, in this way, more heterogeneous patterns can be captured in the training phase. In the figure, n_{tr} (resp. n_{te}) indicates the number of traces used for training (resp. testing), and none of the traces used for testing is included in the training set. It is clear that $n_{tr} = 0.7 \cdot n_{traces}$ and $n_{te} = 0.3 \cdot n_{traces}$.

We also consider a slightly different methodology, shown in Fig. 9(b). In this case, all the traces used for testing in December have their samples from January to November included in the training set. This simulates a scenario where an Edge Computing Node contributes to model training and also benefits from the constructed global model to forecast energy consumption. 70% of traces are randomly selected for training and, among them, 45% are used for testing purposes. In this way, roughly 30% of the traces of any cluster/random set is used for testing (since $0.7 \cdot 0.45 \simeq 0.3$), making the methodology comparable to the one shown in Fig. 9(a).

For each one of the evaluations reported in this Section we randomly created 10 training/test sets following the methodologies described above. Any evaluated performance metric is the average value from each of the 10 resulting instances.

B. EVALUATED METRICS

The following metrics are evaluated:

- *Root Mean Square Error (RMSE):* it is used to measure the *forecasting performance* and is computed as:

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{N} \left(Y_t - \hat{Y}_t\right)^2}{N}} \tag{3}$$

where Y_t is the recorded energy consumption measurement at time t, $\hat{Y}_t$ the forecast value and N the overall number of testing samples. It is expressed in kWh. We choose RMSE instead of MAE because large errors, which are the most undesirable in our context, are better highlighted by RMSE.
- *Training Time:* it is used to evaluate the time (in minutes) that is needed to train the LSTM network, before that Early Stop intervenes.
- *Transmitted Data:* it captures the amount of data (in Mb) that are exchanged between the Edge Computing Nodes and the Aggregator in any direction (i.e., from Edge Computing Nodes to Aggregator or vice-versa). It measures the *communication overhead*.

C. FEDERATED VS. CENTRALIZED ARCHITECTURE

This subsection compares the performance of the state-of-the art centralized architecture and our proposed federated architecture, as depicted in Figs. 2 and 4. As specified in Section VI-C, the state-of-the art centralized architecture considered for comparison is the one proposed in [43].

1) TRAINING TIME AND FORECASTING PERFORMANCE

Table 4 reports a performance comparison between federated and centralized approaches in terms of RMSE and training time, considering a Random set including $n_{traces} = 80$ traces. As already specified in the previous Section, we were able to set the hyperparameters for the federated architecture so that no performance loss is experienced with respect to the centralized one. Additionally, a 10%-15% performance gain is experienced when test traces are included in the training set (Fig. 9(b)) with respect to when they are not included (Fig. 9(a)), meaning that an Edge Computing Node should participate in the training process to maximize its forecasting ability, but that a global model trained by other Edge Computing Nodes can still be used if this is not possible (e.g. when the node joins the system and not enough samples have been collected yet). To be as general as possible and pose ourselves in the worst-case scenario, in all the following evaluations we will consider the case where testing traces are not included in the training set.

TABLE 4. Federated vs. centralized architecture performance (Random set).

Approach	Forecasting performance (RMSE)		Training Time (min)
	Test traces in training set	Test traces not in training set	
Federated	0.1124	0.1333	2.87
Centralized	0.1210	0.1330	20.80

Table 4 also shows a comparison in terms of training time. For the centralized approach, such time has been directly measured on the Google Colab Cloud platform. In fact, it is reasonable to assume that the centralized architecture relies on public (or private) Cloud servers for model training. Conversely, the training time for the federated approach can only

be estimated, since we are using a centralized Cloud platform to simulate a system where computation should be instead distributed. Specifically, the training time can be estimated in the following way:

$$T_{training}(n_{tr}) = \bar{T}_{round}(n_{tr} = 1) \cdot N^{ES}_{round}(n_{tr}) \quad (4)$$

where $\bar{T}_{round}(n_{tr} = 1)$ is the average training time per round when only one trace is included in the training set, as it happens in Edge Computing Nodes when the federated architecture is adopted, and $N^{ES}_{round}(n_{tr})$ is the number of rounds that are executed before that Early Stop intervenes. This training time estimation implicitly assumes that the Round Trip Time (RTT) between Edge Computing Nodes and Aggregator is negligible with respect to the training time per round. In our settings, the latter is around 10 *s*, so the assumption is reasonable since RTT is expected to be in the order of hundreds of *ms* at most [69]. Also the FedAVG execution time at the Aggregator can be considered negligible, being an inexpensive sequence of weighted sums.

Both Tab. 4 and Fig. 10 show that the LSTM network training time is one order of magnitude lower when federated learning is adopted. The reason is that the federated architecture distributes training computation among Edge Computing Nodes and, as shown in Fig. 10, this makes training time almost invariant to the training set size, since each Edge Computing Node is in charge of model training based only on locally-collected samples. Instead, in the centralized architecture, the training time consistently increases as the training set size increases, confirming that the centralized architecture provides a much less scalable solution. It is however important to point out that the shown results assume the same processing capability for both architectures (i.e., that offered by Google Cloud servers), while it is possible that, in a real deployment, Edge Computing Nodes would be more resource-constrained devices than Cloud servers, thus reducing the training time gain of our proposed solution. This evaluation is left for future work.

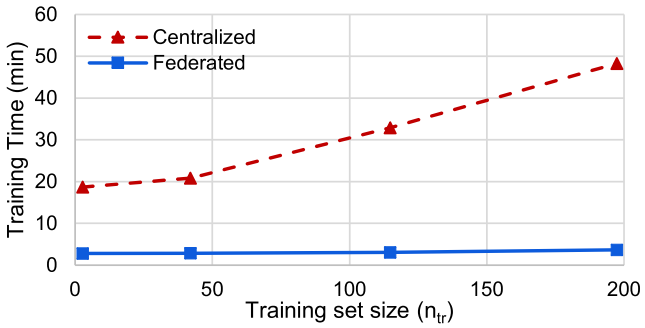


FIGURE 10. Training time ($T_{training}$) comparison between federated and centralized architectures while increasing the training set size n_{tr}.

Figure 11 reports the forecasting performance comparison when the training set size varies. It is shown that the RMSE decreases as the training set size increases, since the trained model is able to recognize more and more occurring patterns.

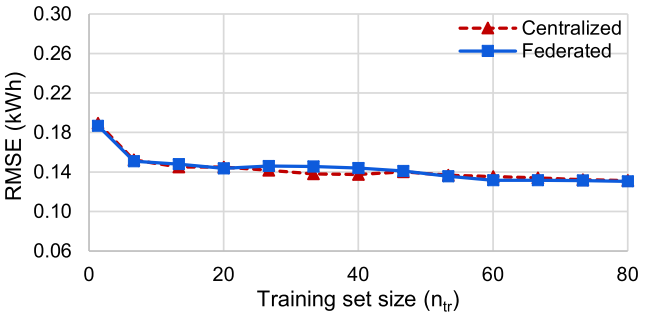


FIGURE 11. Forecasting performance comparison between federated and centralized architectures while increasing the training set size n_{tr}.

The figure also shows that the federated and centralized approaches always lead to similar performance.

2) IMPACT OF CLUSTERING

We evaluate the impact of ACORN and K-Means clustering. We compare the proposed clustering strategies with Random. Figures 12 and 13 show the forecasting performance for ACORN and K-Means; they also include the average RMSE value among all clusters and the results obtained for the Random training set. The federated and centralized architectures lead to very similar forecasting performance for all the clusters. Additionally, it can be seen that the cluster ACORN-U leads to much worse performance than

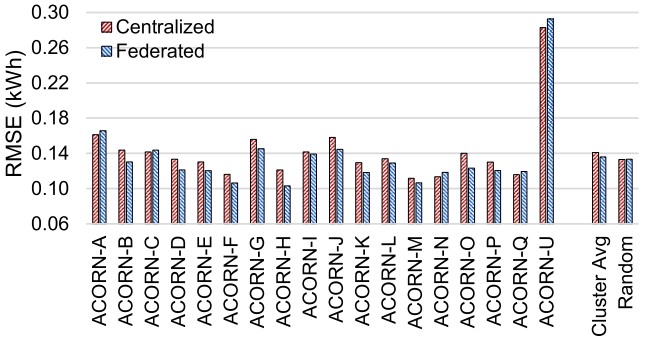


FIGURE 12. Forecasting performance comparison between federated and centralized architectures with *ACORN* clustering.

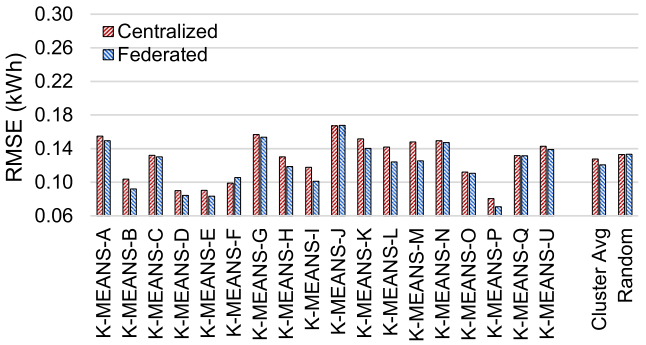


FIGURE 13. Forecasting performance comparison between federated and centralized architectures with *K-Means* clustering.

the clusters' average and than Random. By looking at the ACORN macro-groups [61], it can be seen that ACORN-U, recently also called ACORN-R, refers to *not private households*. It is thus a quite different group than the others, since it includes communal accommodations like military bases, hostels, care homes etc., which generate much more heterogeneous energy consumption patterns than in the other ACORN macro-groups, causing bad performance.

Table 5 summarizes the average performance in the case of clustering and compare it with Random, both for federated and centralized approaches; it also reports the standard deviation among different clusters. We also include in the table the performance of ACORN clustering when ACORN-U is excluded. K-Means clustering always outperforms all the other strategies (with a performance gain of around 10%-15%), while ACORN clustering has slightly worse performance than Random if ACORN-U is included in the evaluation, and better performance if excluded (more realistic case). We can conclude that a clustering strategy helps improve the performance and should be considered by-design in the federated architecture, and that clustering based on traces similarities (K-Means) leads to slightly better performance (around 5%) than clustering based on demographics (ACORN). However, this latter case leads to more than acceptable performance if homogeneous clusters are considered (in this evaluation, if ACORN-U is excluded). The case of ACORN-U also indicates that appropriate clustering is needed to ensure some performance gain.

TABLE 5. Forecasting performance of different clustering strategies.

Clustering	Federated (RMSE)		Centralized (RMSE)	
	Average	Std. Dev.	Average	Std. Dev.
ACORN	0.1360	0.0410	0.1422	0.0370
ACORN (w/o -U)	0.1268	0.0160	0.1310	0.0150
K-Means	0.1208	0.0263	0.1277	0.0257
No clustering (Rand.)	0.1333	-	0.1330	-

3) COMMUNICATION OVERHEAD

We now compare the federated and centralized architectures in terms of communication overhead, by analytically computing the minimum amount of transmitted data between the edge (i.e., Edge Computing Nodes or Smart Meters) and the energy company (i.e., Aggregator or centralized server) to train the LSTM model. In the federated architecture, the adopted formula is:

$$D_{trans}^{fed} = N_{round}^{ES} \cdot 2(C \cdot n_{nodes} \cdot S_{model}) \tag{5}$$

where D_{trans}^{fed} is the overall transmitted data, N_{round}^{ES} the estimated number of rounds before that Early Stop intervenes, C the fraction of Edge Computing Nodes randomly selected in each round, n_{nodes} the number of Edge Computing Nodes in the system and S_{model} the size of the model (i.e., the overall size of the transmitted LSTM cell weights). In the federated architecture the local models need to be sent, for all the N_{round}^{ES} rounds, by all the selected Edge Computing Nodes ($C \cdot n_{nodes}$) to the Aggregator, and then the Aggregator needs to re-distribute the computed global model to the selected Edge Computing Nodes (reason why a factor 2 is included in the formula). In our calculations we set $C = 10\%$, $N_{round}^{ES} = 30$ (experimentally chosen according to Fig. 17) and $S_{model} = 0.032$ Mb (real value observed for the adopted LSTM model). Note that we do not include the overhead due to model initialization in Eq. 5 (see Step 1 in Section III-B1) since it is paid only once for each Edge Computing Node, i.e., when they join the federated architecture for the first time, and can thus be considered negligible.

For the centralized architecture, the adopted formula is instead the following:

$$D_{trans}^{centr} = n_{nodes} \cdot S_{data} \tag{6}$$

where n_{nodes} is the overall number of Smart Meters in the system and S_{data} is the size of the local dataset that needs to be transmitted to the centralized server. Unless otherwise specified, we set $S_{data} = 0.36$ Mb, which is the size of a local dataset including historical data and features for 12 months with hourly resolution (i.e., 8760 energy consumption samples and related attributes as shown in Tab. 2).

Figure 14 compares the communication overhead between federated and centralized architectures as the number of nodes n_{nodes} in the system varies. It is seen that the transmitted data linearly increase as the number of nodes increases and that it is very similar for the two architectures. Considering the values for S_{model} and S_{data} as extrapolated from our considered dataset, no gain in communication overhead is experienced when the federated architecture is adopted.

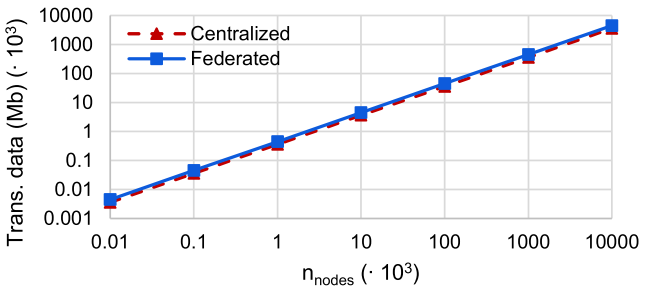


FIGURE 14. Communication overhead for federated and centralized architectures as the number of edge computing nodes n_{nodes} increases ($C = 10\%$, $N_{round}^{ES} = 30$, $S_{model} = 0.032$ Mb, $S_{data} = 0.36$ Mb).

However, any possible gain strictly depends on the size of the local dataset S_{data}. Figure 15 compares the communication overhead as the size of local dataset S_{data} varies, considering 1-million users in the system. Being the transmitted data invariant to the dataset local size S_{data} in the federated architecture (as shown in Eq. 5), while it is not in the centralized, the federated architecture leads to much less transmitted data as the resolution (or granularity) of the dataset increases. With a resolution of 1 minute (i.e., 525,600 samples per year), around 50 times less data need to be transmitted if a federated approach is chosen, since

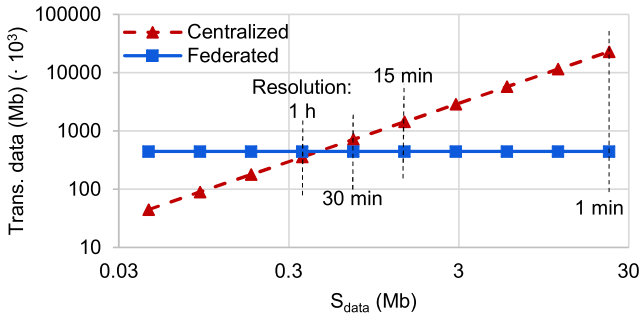


FIGURE 15. Communication overhead for federated and centralized architectures as the size of local dataset S_{data} increases ($C = 10\%$, $N^{ES}_{round} = 30$, $S_{model} = 0.032$ Mb, n_{nodes} =1,000,000).

only the model, whose size is always fixed to S_{model}, needs to be sent to/from the Aggregator.

D. SENSITIVITY TO INPUT FEATURES

We now focus our evaluation on the federated architecture. As first step, we evaluate the impact on forecasting performance of the features described in Section V. The results are shown in Tab. 6. The table shows the performance when the features are or are not provided as input to the training phase. We always include *energy consumption*, and we consider it jointly with any of the other features, that is, *weekday*, *hour*, *AVG4D* and *TempCluster*. The last row of the table shows the forecasting performance when all those features are considered, as for all the other evaluations in this paper.

TABLE 6. Forecasting performance comparison for different input features.

Considered features					RMSE
Energy consumption	Weekday	Hour	AVG4D	Temp Cluster	
✓					0.1467
✓	✓				0.1424
✓		✓			0.1620
✓			✓		0.1551
✓				✓	0.1392
✓	✓	✓	✓	✓	0.1333

The results show that considering only *energy consumption* as feature leads to better forecasting performance than considering it jointly with *AVG4D* or *hour*, meaning that these latter features, in our evaluation scenario, do not add any insightful information during the training process if taken alone. Instead, adding as feature *weekday* and especially *TempCluster* improves forecasting performance. This means that, in general, the traces experience strong weekly patterns and that exploiting clustered apparent temperature data, discriminating between hot and cold hours, is an effective way to improve the forecasting performance. Moreover, when all the features are jointly considered (last row), some performance gain is experienced with respect to only considering *weekday* or *TempCluster*, meaning that *AVG4D* and *hour* carry some exploitable information when put aside the other features.

E. IMPACT OF TUNING PARAMETERS

We evaluate the impact of some tuning parameters on forecasting performance in the federated architecture. As described in Section IV the Aggregator can, in the initialization phase, centrally train the LSTM model using a number $n_{pretrain}$ of *pre-training traces* that are obtained from customers that are willing to share them. Figure 16 shows the forecasting performance and the number of rounds before that Early Stop intervenes when the percentage of pre-training traces increases. The figure shows that, as such a percentage increases, (i) a smaller number of rounds is needed to ensure model convergence and (ii) the forecasting performance is not affected. This means that using pre-training data helps speed up the FL-based training procedure without any impact on forecasting performance. In our settings, if only 15% of the traces are used in the pre-training phase, the number of rounds N^{ES}_{round} can be reduced by 25%, and federated training time is thus significantly reduced.

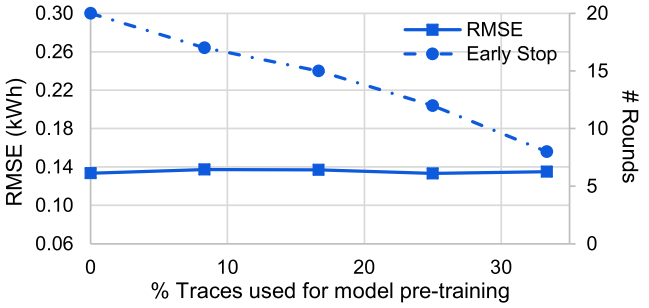


FIGURE 16. Impact of pre-training on forecasting performance and on number of early stop rounds (# Rounds or N^{ES}_{round}).

The parameter C also plays an important role in the federated architecture. A higher value for C implies that more Edge Computing Nodes are involved in the training process in each round, with bad implications on communication overhead (see Eq. 5) and on processing effort for the nodes, since they are likely to be selected more frequently. However, as shown in Fig. 17, a higher C is beneficial to speed up the training procedure. In other words, the figure shows that a variation of C does not have an impact on training performance, but considering more nodes per round helps reduce N^{ES}_{round}. In our small-scale settings, the number of rounds N^{ES}_{round} can be reduced by around 30% if $C = 40\%$.

Figure 18 shows instead the impact of the amount of local training data on forecasting performance and on training time. The same procedure as the one described in Fig. 9 is adopted for the definition of training and test sets, with only one difference. In this case, the Random traces selected for training are all between 1 and 11 months long, depending on the experiment. The x-axis (*# Months*) indicates the number of months that are considered in each experiment. For example, *# Months* equal to 6 means that, for all the Random traces, six months are randomly selected for training. Figure 18 reports the obtained results: the more data are used as input by any Edge Computing Node, the better forecasting performance is

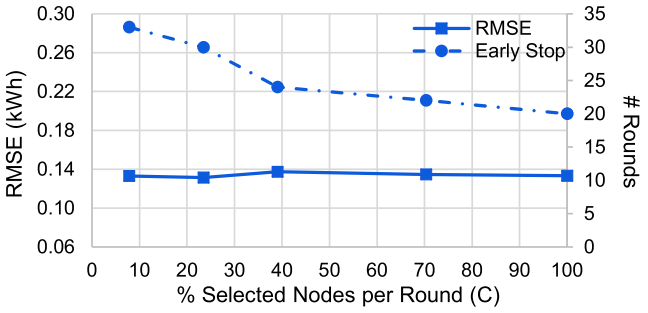


FIGURE 17. Impact of C on forecasting performance and on number of early stop rounds (# Rounds or N^{ES}_{round}).

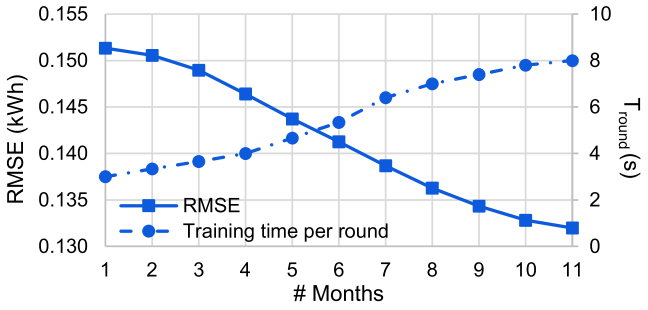


FIGURE 18. Impact of amount of training data on forecasting performance and on training time per round T_{round}.

ensured. The best performance is obtained when the historical data of the previous 11 months (*# Months* equal to 11) are used for local model training. However, not surprisingly, the higher amount of local data are used, the higher training time per round T_{round} is experienced, indicating that the amount of historical samples must be carefully chosen to avoid excessive training time overheads.

F. SENSITIVITY TO TESTING MONTH

We evaluate the forecasting performance of a model trained over the whole 2013 year on different testing months. With respect to the other subsections, here we consider a model trained as specified in Section VII-A, with the difference that also the December month is part of the training set. We focus on a Random set and we vary the testing month from January to December. The goal is to understand if the trained model is able to capture patterns occurring in different seasons of the year. The results are reported in Tab. 7. It can be seen that an acceptable forecasting performance is achieved for all the testing months, confirming that the trained model is able to capture changing patterns. However, better results are obtained in summertime, since in this season more predictable patterns occur, being people not at home more frequently.

G. COMPARISON WITH A FL-BASED STRATEGY

In this subsection we compare our proposal with an existing FL-based solution [54]. We focus our comparison on a Random set (with $n_{nodes} = 80$ and $n_{tr} = 0.7 \cdot n_{nodes} = 56$), which

TABLE 7. Forecasting performance comparison for different testing months.

Testing month	RMSE
January	0.1463
February	0.1530
March	0.1535
April	0.1345
May	0.1304
June	0.1259
July	0.1057
August	0.1106
September	0.1302
October	0.1335
November	0.1471
December	0.1328

is the most general case also evaluated in [54]. We do not consider any further forecasting performance enhancement due to clustering (as done in this work) or to personalization (as done in [54]). We consider the four scenarios defined and evaluated in the reference paper:

- *Scenario 1:* $C = 9\%$ (i.e., 5 selected nodes per round) and $N_{epoch} = 1$;
- *Scenario 2:* $C = 36\%$ (i.e., 20 selected nodes per round) and $N_{epoch} = 1$;
- *Scenario 3:* $C = 9\%$ and $N_{epoch} = 5$;
- *Scenario 4:* $C = 36\%$ and $N_{epoch} = 5$.

For the state-of-the-art approach, we consider $N_{round} = 20$ as specified in the paper, while in our strategy the training phase is stopped when Early Stop intervenes. Results are shown in Tab. 8. It is shown that our proposal outperforms the state of the art for all the evaluated scenarios. This happens because the approach defined in [54] only considers *energy consumption* as input feature in the training phase, while our strategy includes four additional features (i.e., *weekday*, *hour*, *AVG4D* and *TempCluster*), here proven to enhance the forecasting performance with respect to the state of the art both when C and N_{epoch} vary.

TABLE 8. Comparison with the state of the art (federated architecture).

Scenario	RMSE	
	Existing strategy [54]	Our strategy
1) $C = 9\%$, $N_{epoch} = 1$	0.1624	0.1413
2) $C = 36\%$, $N_{epoch} = 1$	0.1485	0.1351
3) $C = 9\%$, $N_{epoch} = 5$	0.1425	0.1339
4) $C = 36\%$, $N_{epoch} = 5$	0.1500	0.1412

H. COMPARISON WITH OTHER PRIVACY-PRESERVING TRAINING STRATEGIES

The adoption of a federated architecture makes it possible to globally train a model while meeting customers' privacy requirements. However, from a customer perspective, two other trivial privacy-preserving strategies could be chosen:

- Training a model using own collected data, without any interaction between the Edge Computing Node and the federated architecture. We call this training strategy *single trace (own)*.

- Training a model using a trace gathered somewhere else (e.g. from a public dataset), without any interaction between the Edge Computing Node and the federated architecture. We assume that only one trace is chosen for training to keep the processing burden on the Edge Computing Node low and manageable (as it occurs in the *federated* and in the *single trace (own)* cases). We call this training strategy *single trace (other)*.

For both the *single trace* strategies described above, we consider 30% of the traces of a Random set (that is, 24 traces) to train 24 models using the historical data, from January to November, of 24 Smart Meters. Then each model is tested using the same trace that trained it in the case of *own*, while it is tested with a randomly-chosen different trace from the Random set in the case of *other*. December is always selected as testing month. Table 9 (*Standard* column) reports the comparison between *single trace* (*own* and *other*) and *federated* strategies, where results obtained with the 24 models are averaged. For *federated* we consider the case where the test traces are included in the training set, simulating a scenario where customers, to maximize their forecasting performance, join the federated architecture and participate to the training process (see also Tab. 4). Table 9 shows that the *own* strategy has the best performance, since the model is perfectly tailored on the customers' patterns, while *other* leads to the worst results, showing that using a different trace to train the model is not an effective strategy. *Federated* leads instead to only slightly worse performance than *own*.

TABLE 9. Comparison between privacy-preserving training strategies.

Training strategy	Trace type (RMSE)	
	Standard	Composed
Single trace (own)	0.1108	0.1950
Single trace (other)	0.1763	
Federated	0.1124	0.1530

However, *federated* is always a much more effective strategy than *single trace* if, for some reason, energy consumption patterns of a customer suddenly change (e.g. because a new appliance is connected to the electrical system, or in the case of short-term housing). To evaluate this scenario we modified the December testing month: we created 24 fake composed testing traces, where the consumption pattern changes four times (i.e., every week) as shown in Fig. 19. In each week, the energy consumption of another randomly-chosen trace from the Random set is taken. These artificial composed traces pose us in the worst-case scenario where consumption patterns vary consistently and very frequently. For *single trace* we evaluate the forecasting performance considering the composed testing traces while using the 24 models as trained above (there is no difference between *own* and *other* in this case), while for *federated* we use the model trained by the federated architecture. Table 9 (*Composed* column) shows the obtained results. As it can be seen, *federated* leads to much better forecasting performance than *single trace* (around 25% better), since the model trained by the federated architecture is more generalized and able to better forecast different patterns, also in the case of frequent changes.

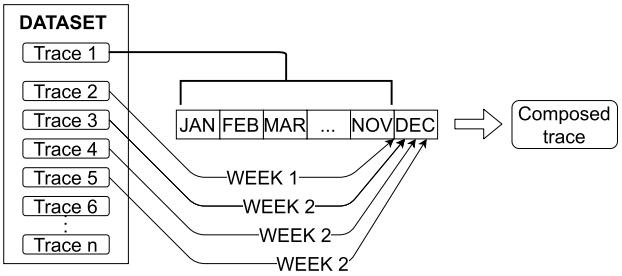


FIGURE 19. Methodology for the creation of the composed testing traces. Each trace is randomly chosen from the dataset.

I. ASSESSMENT ON RECENT DATA

Finally, we used an already-deployed IoT system to collect energy consumption measurements occurring in a house located in Lombardy (Italy) during September 2020. We adopted a Shelly Energy Meter[4] connected to the magnetothermic switch of the house to gather energy consumption measurements at hourly granularity. We then considered ten models trained using ten different Random sets to forecast the house's load in any specific hour, using as input the collected data in the previous 24 hours. Table 10 shows the experienced forecasting performance (average among the ten instances) and its comparison with results obtained so far, while Fig. 20 compares predicted and real consumption values for one of the instances. Results (although preliminary) are very promising. Despite the model has been trained on a seven-year-old dataset by using samples collected in a city from a different country, the performance is not too badly affected.

TABLE 10. Short-term forecasting performance on recently-collected data.

Approach	Testing data (RMSE)	
	Recent data	Dataset [60]
Federated	0.1337	0.1333

J. BENEFITS OF OUR SOLUTION: A RECAP

To summarize, in contrast to the centralized architecture recalled in Fig. 2, our proposed federated architecture is based on a decentralized approach that brings many benefits, thoroughly investigated through the paper and especially in this Section.

First, our solution embraces the *privacy-by-design* principle, since it guarantees that fine-grained energy consumption measurements are kept at the edge (i.e., where they are generated) to mitigate any customers' privacy breach that could occur if such data would be transferred to a centralized location. However, in some specific cases of unbalanced data, privacy could still be breached by simply analyzing the shared model weights [70]. Techniques such as differential privacy

[4]https://shelly.cloud/products/shelly-em-smart-home-automation-device/

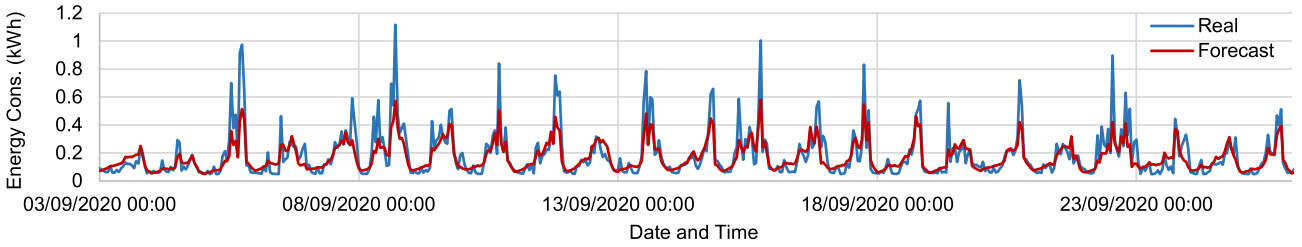


FIGURE 20. Comparison between real and forecast trace (data collected in September 2020).

can be straightforwardly adopted [71]–[73] to mitigate this issue, but this is left for future work.

Another important benefit consists on the higher training dynamicity at a lower computational cost. In fact, in the centralized architecture, a model that has to keep track of novel patterns (possibly occurring in the time-series data) needs to be periodically re-trained with massive amount of new samples, (i) requiring a significant processing effort at the centralized server and (ii) leading to high training times. Moreover, if the model is not frequently re-trained, it may not be able to forecast novel patterns for a long time (low adaptability). Our proposal is instead able to continuously re-train the model and dynamically adapt it to new patterns, while distributing the training effort among many Edge Computing Nodes that only require moderate processing capabilities. Moreover, since local training involves a much lower amount of data, per-round training times can also be kept low.

Finally, the federated architecture generally leads to a communication overhead reduction with respect to the centralized one, especially when samples are collected at a high granularity. In fact, only the LSTM cell weights need to be exchanged between the Aggregator and the Edge Computing Nodes, while fine-grained energy consumption measurements, which have generally a higher size, are kept local.

VIII. CONCLUSION

In this paper we proposed an architecture that adopts Federated Learning to unlock effective short-term load forecasting at the edge of the network. The architecture allows multiple participants to collaboratively train an LSTM neural network while keeping sensitive fine-grained energy consumption measurement local, under the coordination of a centralized aggregation node. It also includes the possibility to use weather- and calendar-related information as input in the model training phase, and envisions the possibility to opportunely cluster participants with similar consumption trends or with similar socioeconomic conditions, with the proven benefit of enhancing the forecasting performance.

We thoroughly evaluated the proposed solution and compared it with a state-of-the-art architecture, where the LSTM network is trained in a centralized location using massive amounts of data as collected from the borders of the network. The results show that our proposal can lead to forecasting performance as good as the state of the art, but outperforms it in terms of model training time (of up to one order of magnitude) and privacy awareness. With respect to communication overhead, our method outperforms the existing approach when the consumption measurements are collected with resolution higher than one hour. We also demonstrated that our architecture comes with a set of parameters (that is, amount of pre-training data, number of selected participants per FL round, amount of training data) that need to be properly tuned to strike the best balance between forecasting performance, training time and communication overhead, and that our approach outperform other privacy-preserving training strategies, especially when energy consumption patterns rapidly change over time.

As future work we plan to extend the architecture to embed differential privacy principles, so that any model inversion attack can be mitigated. In this way, the participants' privacy can be preserved also when their locally-trained model discloses some information on the data used to train it. We also plan to implement a small testbed where the Federated Learning logic is executed on resource-constrained devices, so that we will be able to thoroughly evaluate our architecture in a real (and not simulated) deployment, and we plan to evaluate the forecasting performance when other clustering algorithms are adopted (e.g. DBSCAN) and/or other features are considered, possibly extracted through appropriate machine learning methods.

LIST OF ACRONYMS

ACORN	Classification of Residential Neighbourhood
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
AMI	Advanced Metering Infrastructure
ANN	Artificial Neural Network
API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
AT	Apparent Temperature
AVG4D	Average of Previous Four Days
CNN	Convolutional Neural Network
CRBM	Conditional Restricted Boltzmann Machine
ES	Early Stop
FedAVG	Federated Averaging
FedSGD	Federated SGD
FL	Federated Learning
GPU	Graphical Processing Unit

IoT	Internet of Things
LS-SVM	Least Squares SVM
LR	Learning Rate
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
RNN	Recurring Neural Network
RTT	Round Trip Time
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
UV	Ultraviolet

REFERENCES

[1] A. R. Khan, A. Mahmood, A. Safdar, Z. A. Khan, and N. A. Khan, "Load forecasting, dynamic pricing and DSM in smart grid: A review," *Renew. Sustain. Energy Rev.*, vol. 54, pp. 1311–1322, Feb. 2016.

[2] S. Fallah, R. Deo, M. Shojafar, M. Conti, and S. Shamshirband, "Computational intelligence approaches for energy load forecasting in smart energy management grids: State of the art, future challenges, and research directions," *Energies*, vol. 11, no. 3, p. 596, Mar. 2018.

[3] Y. Peng, Y. Wang, X. Lu, H. Li, D. Shi, Z. Wang, and J. Li, "Short-term load forecasting at different aggregation levels with predictability analysis," in *Proc. IEEE Innov. Smart Grid Technol. Asia (ISGT Asia)*, May 2019, pp. 3385–3390.

[4] B. Yildiz, J. I. Bilbao, J. Dore, and A. B. Sproul, "Recent advances in the analysis of residential electricity consumption and applications of smart meter data," *Appl. Energy*, vol. 208, pp. 402–427, Dec. 2017.

[5] P. Zhang, X. Wu, X. Wang, and S. Bi, "Short-term load forecasting based on big data technologies," *CSEE J. Power Energy Syst.*, vol. 1, no. 3, pp. 59–67, Sep. 2015.

[6] Research and Markets. *Global Smart Meters Market—Growth, Trends, Forecasts (2020—2025)*. Accessed: Jan. 11, 2021. [Online]. Available: https://tinyurl.com/y4ujl8f3

[7] Z. Fan, G. Kalogridis, C. Efthymiou, M. Sooriyabandara, M. Serizawa, and J. McGeehan, "The new frontier of communications research: Smart grid and smart metering," in *Proc. Int. Conf. Energy-Efficient Comput. Netw.*, 2010, pp. 115–118.

[8] M. R. Asghar, G. Dán, D. Miorandi, and I. Chlamtac, "Smart meter data privacy: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2820–2835, Jun. 2017.

[9] F. Mármol, C. Sorge, O. Ugus, and G. Pérez, "Do not snoop my habits: Preserving privacy in the smart grid," *IEEE Commun. Mag.*, vol. 50, no. 5, pp. 166–172, May 2012.

[10] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Gener. Comput. Syst.*, vol. 97, pp. 219–235, Aug. 2019.

[11] G. Premsankar, M. D. Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[13] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–9, 2019.

[14] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving Google keyboard query suggestions," 2018, *arXiv:1812.02903*. [Online]. Available: http://arxiv.org/abs/1812.02903

[15] X. Wu, Z. Liang, and J. Wang, "FedMed: A federated learning framework for language modeling," *Sensors*, vol. 20, no. 14, p. 4048, Jul. 2020.

[16] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *J. Healthcare Informat. Res.*, vol. 5, no. 1, pp. 1–19, Mar. 2021.

[17] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, "Differential privacy-enabled federated learning for sensitive health data," 2019, *arXiv:1910.02578*. [Online]. Available: http://arxiv.org/abs/1910.02578

[18] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Med. Informat.*, vol. 112, pp. 59–67, Apr. 2018.

[19] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4734–4746, Aug. 2020.

[20] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23920–23935, 2020.

[21] R. Cioffi, M. Travaglioni, G. Piscitelli, A. Petrillo, and F. De Felice, "Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions," *Sustainability*, vol. 12, no. 2, p. 492, Jan. 2020.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] A. Tascikaraoglu and B. M. Sanandaji, "Short-term residential electric load forecasting: A compressive spatio-temporal approach," *Energy Buildings*, vol. 111, pp. 380–392, Jan. 2016.

[24] F. Kaytez, M. C. Taplamacioglu, E. Cam, and F. Hardalac, "Forecasting electricity consumption: A comparison of regression analysis, neural networks and least squares support vector machines," *Int. J. Electr. Power Energy Syst.*, vol. 67, pp. 431–438, May 2015.

[25] K. Amasyali and N. M. El-Gohary, "A review of data-driven building energy consumption prediction studies," *Renew. Sustain. Energy Rev.*, vol. 81, pp. 1192–1205, Jan. 2018.

[26] K. Gajowniczek and T. Zabkowski, "Electricity forecasting on the individual household level enhanced based on activity patterns," *PLoS ONE*, vol. 12, no. 4, pp. 1–26, 2017.

[27] H. Chitsaz, H. Shaker, H. Zareipour, D. Wood, and N. Amjady, "Short-term electricity load forecasting of buildings in microgrids," *Energy Buildings*, vol. 99, pp. 50–60, Jul. 2015.

[28] M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renew. Sustain. Energy Rev.*, vol. 50, pp. 1352–1372, Oct. 2015.

[29] K. Li, C. Hu, G. Liu, and W. Xue, "Building's electricity consumption prediction using optimized artificial neural networks and principal component analysis," *Energy Buildings*, vol. 108, pp. 106–113, Dec. 2015.

[30] F. Abbas, D. Feng, S. Habib, U. Rahman, A. Rasool, and Z. Yan, "Short term residential load forecasting: An improved optimal nonlinear auto regressive (NARX) method with exponential weight decay function," *Electronics*, vol. 7, no. 12, p. 432, Dec. 2018.

[31] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustain. Energy, Grids Netw.*, vol. 6, pp. 91–99, Jun. 2016.

[32] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—A novel pooling deep RNN," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5271–5280, Sep. 2018.

[33] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Trans. Power Syst.*, vol. 33, no. 1, pp. 1087–1088, Jan. 2018.

[34] M. Sajjad, Z. A. Khan, A. Ullah, T. Hussain, W. Ullah, M. Y. Lee, and S. W. Baik, "A novel CNN-GRU-based hybrid approach for short-term residential load forecasting," *IEEE Access*, vol. 8, pp. 143759–143768, 2020.

[35] K. Aurangzeb, S. Aslam, S. I. Haider, S. M. Mohsin, S. U. Islam, H. A. Khattak, and S. Shah, "Energy forecasting using multiheaded convolutional neural networks in efficient renewable energy resources equipped with energy storage system," *Trans. Emerg. Telecommun. Technol.*, p. e3837, Dec. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/ett.3837

[36] H. Nie, G. Liu, X. Liu, and Y. Wang, "Hybrid of ARIMA and SVMs for short-term load forecasting," *Energy Procedia*, vol. 16, pp. 1455–1460, 2012.

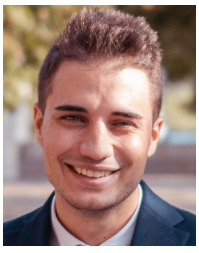
[37] R. Ahmadiahangar, T. Häring, A. Rosin, T. Korotko, and J. Martins, "Residential load forecasting for flexibility prediction using machine learning-based regression model," in *Proc. IEEE Int. Conf. Environ. Electr. Eng. IEEE Ind. Commercial Power Syst. Eur. (EEEIC/I CPS Europe)*, Jun. 2019, pp. 1–4.

[38] Y. Hong, Y. Zhou, Q. Li, W. Xu, and X. Zheng, "A deep learning method for short-term residential load forecasting in smart grid," *IEEE Access*, vol. 8, pp. 55785–55797, 2020.

[39] R. E. Edwards, J. New, and L. E. Parker, "Predicting future hourly residential electrical consumption: A machine learning case study," *Energy Buildings*, vol. 49, pp. 591–603, Jun. 2012.

[40] Y.-H. Hsiao, "Household electricity demand forecast based on context information and user daily schedule analysis from meter data," *IEEE Trans. Ind. Informat.*, vol. 11, no. 1, pp. 33–43, Feb. 2015.
[41] M. N. Fekri, H. Patel, K. Grolinger, and V. Sharma, "Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network," *Appl. Energy*, vol. 282, Jan. 2021, Art. no. 116177.
[42] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.
[43] A. M. Alonso, F. J. Nogales, and C. Ruiz, "A single scalable LSTM model for short-term forecasting of massive electricity time series," *Energies*, vol. 13, no. 20, p. 5328, Oct. 2020.
[44] M. Imani and H. Ghassemian, "Residential load forecasting using wavelet and collaborative representation transforms," *Appl. Energy*, vol. 253, Nov. 2019, Art. no. 113505.
[45] M. Alhussein, K. Aurangzeb, and S. I. Haider, "Hybrid CNN-LSTM model for short-term individual household load forecasting," *IEEE Access*, vol. 8, pp. 180544–180557, 2020.
[46] R. Khalid, N. Javaid, F. A. Al-zahrani, K. Aurangzeb, E.-U.-H. Qazi, and T. Ashfaq, "Electricity load and price forecasting using Jaya-long short term memory (JLSTM) in smart grids," *Entropy*, vol. 22, no. 1, p. 10, Dec. 2019.
[47] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using long short-term memory networks on groups of similar series," 2017, *arXiv:1710.03222v1*. [Online]. Available: https://arxiv.org/abs/1710.03222v1
[48] F. L. Quilumba, W.-J. Lee, H. Huang, D. Y. Wang, and R. L. Szabados, "Using smart meter data to improve the accuracy of intraday load forecasting considering customer behavior similarities," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 911–918, Mar. 2015.
[49] K. Aurangzeb, M. Alhussein, K. Javaid, and S. I. Haider, "A pyramid-CNN based deep learning model for power load forecasting of similar-profile energy customers based on clustering," *IEEE Access*, vol. 9, pp. 14992–15003, 2021.
[50] M. Beccali, M. Cellura, V. L. Brano, and A. Marvuglia, "Short-term prediction of household electricity consumption: Assessing weather sensitivity in a Mediterranean area," *Renew. Sustain. Energy Rev.*, vol. 12, no. 8, pp. 2040–2065, Oct. 2008.
[51] K. M. Powell, A. Sriprasad, W. J. Cole, and T. F. Edgar, "Heating, cooling, and electrical load forecasting for a large-scale district energy system," *Energy*, vol. 74, pp. 877–885, Sep. 2014.
[52] M. G. Fikru and L. Gautier, "The impact of weather variation on energy consumption in residential houses," *Appl. Energy*, vol. 144, pp. 19–30, Apr. 2015.
[53] P. Lusis, K. R. Khalilpour, L. Andrew, and A. Liebman, "Short-term residential load forecasting: Impact of calendar effects and forecast granularity," *Appl. Energy*, vol. 205, pp. 654–669, Nov. 2017.
[54] A. Taïk and S. Cherkaoui, "Electrical load forecasting using edge computing and federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.
[55] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. Brendan McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*. [Online]. Available: http://arxiv.org/abs/1902.01046
[56] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.
[57] G. Dileep, "A survey on smart grid technologies and applications," *Renew. Energy*, vol. 146, pp. 2589–2625, Feb. 2020.
[58] Y. Wang, Q. Chen, T. Hong, and C. Kang, "Review of smart meter data analytics: Applications, methodologies, and challenges," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 3125–3148, May 2019.
[59] M. Gaur, S. Makonin, I. V. Bajic, and A. Majumdar, "Performance evaluation of techniques for identifying abnormal energy consumption in buildings," *IEEE Access*, vol. 7, pp. 62721–62733, 2019.
[60] *Smart Meter Energy Consumption Data in London Households*. Accessed: Jan. 11, 2021. [Online]. Available: https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households
[61] *The ACORN User Guide*. Accessed: Jan. 11, 2021. [Online]. Available: https://acorn.caci.co.uk/downloads/Acorn-User-guide.pdf
[62] *Smart Meters Data in London Including Weather Variables*. Accessed: Jan. 11, 2021. [Online]. Available: https://www.kaggle.com/jeanmidev/smart-meters-in-london
[63] *DarkSky Service*. Accessed: Jan. 11, 2021. [Online]. Available: https://darksky.net
[64] O. Parson, G. Fisher, A. Hersey, N. Batra, J. Kelly, A. Singh, W. Knottenbelt, and A. Rogers, "Dataport and NILMTK: A building data set designed for non-intrusive load monitoring," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2015, pp. 210–214.
[65] J. A. Hartigan, *Clustering Algorithms*. Hoboken, NJ, USA: Wiley, 1975.
[66] H. Rashid, N. Batra, and P. Singh, "Rimor: Towards identifying anomalous appliances in buildings," in *Proc. 5th Conf. Syst. Built Environ.*, Nov. 2018, pp. 33–42.
[67] R. G. Steadman, "A universal scale of apparent temperature," *J. Climate Appl. Meteorol.*, vol. 23, no. 12, pp. 1674–1687, Dec. 1984.
[68] C. Yuan and H. Yang, "Research on K-value selection method of K-means clustering algorithm," *J*, vol. 2, no. 2, pp. 226–235, Jun. 2019.
[69] X. Chen, H. Kim, J. M. Aman, W. Chang, M. Lee, and J. Rexford, "Measuring TCP round-trip time in the data plane," in *Proc. Workshop Secure Program. Netw. Infrastruct.*, Aug. 2020, pp. 35–41.
[70] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. S. Quek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Netw.*, vol. 34, no. 4, pp. 242–248, Jul. 2020.
[71] C. Dwork, "Differential privacy: A survey of results," in *Proc. Int. Conf. Theory Appl. Models Comput.*, 2008, pp. 1–19.
[72] M. Savi, C. Rottondi, and G. Verticale, "Evaluation of the precision-privacy tradeoff of data perturbation for smart metering," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2409–2416, Sep. 2015.
[73] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.

MARCO SAVI received the Ph.D. degree in information technology from Politecnico di Milano, in 2016. In 2020, he worked for four years with Fondazione Bruno Kessler, Trento, Italy. He is currently an Assistant Professor with the University of Milano-Bicocca, Italy. His research interests include the design and optimization of telecommunication networks, cloud computing, and smart grids. He has been involved in some European Research Projects Advancing Access and Core Network Technologies (FP7 COMBO, H2020 ACINO, H2020 DECENTER, and H2020 GN4-3).



FABRIZIO OLIVADESE received the master's degree (Hons.) in computer science from the University of Milano-Bicocca, in 2020, working on a thesis on the application of federated learning in the field of residential energy consumption forecasting. He is an IT Digital Innovation Specialist with the Artsana Group, the leading Italian company on parenting, supplements, cosmetics, and health space.