# Decentralized Time Series Forecasting with Federated Learning and Convolutional Neural Networks

Ankith Boggaram
ankith.boggaram@utah.edu

Sharath Satish
sharath.satish@utah.edu

## Abstract

*This study looks into the application of federated learning to predict household energy consumption using a time-series dataset that includes energy usage and weather data. It aims to assess the effectiveness of a convolutional neural network (CNN) architecture in a federated learning framework compared to a standalone model, where the standalone model is trained using a set of hyperparameters optimized through Bayesian optimization. Results show that the federated learning setup achieved relatively better performance as compared to the standalone CNN on the test dataset with no signs of overfitting. The findings suggest that federated learning can effectively predict energy consumption without compromising model accuracy despite having a distributed dataset, offering a viable solution for distributed machine learning applications in variety of domains, including smart homes.*

## 1. Introduction

Understanding energy consumption in residential households is a vital issue to address, as it provides substantial benefits in identifying consumption patterns and decreasing usage to lessen our environmental footprint. However, the challenge arises from the difficulty of obtaining energy consumption patterns from individual households and aggregating these data for analysis. This complexity is due to the variability in consumption behaviors and concerns about privacy when consolidating user-specific data at a central location, which may be vulnerable during transmission or storage at a centralized server.

One approach to address this issue is Federated Learning (FL) [1], which employs a decentralized method for training models directly at the data source. In this process, individual nodes send only the local model weights from a machine learning or deep learning model, trained over a specified number of epochs, to a central server. The server then synchronously aggregates the received weights from all nodes before returning the updated weights to them, continuing this cycle until the desired accuracy is achieved, or the number of epochs are exhausted. This approach guarantees that the data collected at the source remains private and is not shared with other entities in the system, while still allowing access to the collective data from all other entities. Aggregating the weights of remote client nodes in a federated learning system is relatively straightforward when there are few clients. However, as the number of client nodes grows, this process may become less efficient as some nodes may contribute more valuable information through their parameters than others, and challenges arise when the data is non-IID. These factors require the use of alternative methods for aggregating weights, which might not be suitable for all scenarios.

Efficiently analyzing the data collected at each source node is essential to understand energy consumption patterns which may be specific to that household. Utilizing deep learning models that require fewer parameters to train and can be updated when a sufficient amount of data is collected in future timestamps would help minimize the computational load of decentralized forecasting on remote node devices. Convolutional neural networks (CNN), although mainly employed for image processing, can also be used to analyze multivariate time-series data, as demonstrated by Chen et al. [2], as time steps can be used by the convolution layers of the model to forecast data at future time steps. Although CNN models may struggle to maintain long-term contextual information from earlier time steps compared to Long Short-Term Memory (LSTM) models as the context length grows, they have the advantage of requiring far fewer parameters to train. This makes CNNs a promising choice for analyzing and forecasting time-series energy consumption data across households in a neighborhood, over LSTMs as demonstrated by Taik et al. [3].

The aim of this study is to explore whether convolutional neural networks, trained on energy consumption data in the form of time-series forecasts collected at the source, can benefit from a federated learning architecture involving a central server node and four client nodes. The focus is on forecasting the total energy consumption for a future timestamp based on data from previous timestamps. The project code is available at this GitHub repository.

## 2. Related Work

Recent studies highlight the potential of federated learning in prediction tasks, but there is a notable lack of research specifically addressing the combination of FL and household energy forecasting. Yang et al. [4] introduced multiple FL frameworks, showing they can achieve comparable accuracy to centralized methods while preserving data privacy. Liu et al. [5] applied FL to household energy demand prediction, demonstrating its ability to handle time-series data across distributed devices. However, these studies overlook critical practical challenges.

As mentioned earlier in the introduction, existing research does not address the challenge of partitioning time-series data effectively among clients in an FL setup. Additionally, while studies focus on general FL frameworks, they lack tailored aggregation strategies for non-IID household energy data, which we address in this study.

Incorporating weather data alongside historical energy readings to perform convolutions in 2-dimensions within a federated learning framework for energy forecasting is a relatively unexplored area in the literature. While some studies such as Savi et al. [6] have examined federated learning for energy applications.

Taik et al. [3] employed LSTM deep learning models at client nodes to predict and visualize energy consumption based on data from a private energy contractor. While the model excelled at personalizing energy expenditure for various households, it faced significant limitations, including reliance on private data repositories, the complexity of federated learning architectures, and the computational demands of the LSTM model, which can be challenging for edge devices with limited processing power. This highlights another novelty in our approach, which aims to enhance energy prediction accuracy by leveraging the interplay between weather conditions and prior energy consumption data in a federated setting by using a CNN model.

## 3. Approach

Our approach can be broadly divided into three parts, data preparation, designing the CNN architecture and setting up the federated learning framework.

### 3.1. Data Preparation

To estimate household energy consumption, we utilize a time-series dataset that records energy usage at 1-second intervals, along with corresponding weather data at those times. The dataset we are using for this purpose is the Smart Home dataset from Kaggle.

Federated learning requires unique data for each client in order to simulate real-world scenarios where data is distributed and localized to individual nodes. To achieve this, the dataset is split into five non-overlapping parts using a round-robin algorithm. Each of the four client nodes receives one segment for training, reflecting the diverse data ownership in federated settings. The fifth segment is reserved for training a standalone CNN model, serving as a baseline comparison against the federated approach.

The dataset is preprocessed to ensure least resistance while training the CNN. The preprocessing steps we followed are:

- **Feature Selection**: Only relevant numeric columns, such as total energy usage, energy usage per zone of the house such as kitchen, garage, living room, home office etc. and weather-related features (e.g., temperature, humidity), are retained for modeling.
- **Target Variable Calculation**: A new target column, `Target`, is added to represent the percentage change in energy usage (`use [kW]`). Missing values in this column are filled with zeros.
- **Scaling**: Numeric columns are standardized using `StandardScaler` from scikit-learn. The scaler is fitted on a subset of training data to prevent data leakage.

Figure 1 helps provide a better view of the data split.

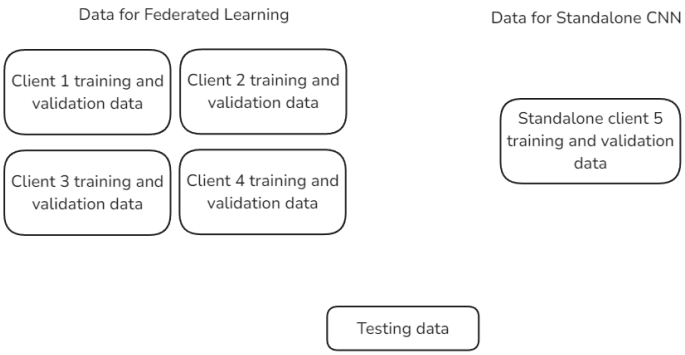


Figure 1. Separation of data

A central test dataset, held out from all training data, is used across all nodes for evaluation. This ensures that the model's performance is assessed uniformly and provides a consistent benchmark for comparing the federated learning approach with the centralized CNN model.

### 3.2. Convolutional Neural Network Architecture

For this project, we designed a custom Convolutional Neural Network (CNN) architecture tailored for time-series energy consumption forecasting. This architecture was used both in the federated learning framework and in the standalone setup to ensure a fair comparison of their performance. The CNN was implemented using PyTorch and consisted of the following components:

1. **Input Layer**: The network accepts inputs with a shape of $(\text{batch\_size}, 1, \text{sequence\_length}, \text{n\_features})$, where `sequence_length` represents the number of time steps in a sequence, and `n_features` denotes the number of features per time step.

2. **Convolutional Layers**:

   - The first convolutional layer applies `filters_layer1` filters with a kernel size of $(1, \text{n\_features})$ to capture feature interactions across the input channels.
   - The second convolutional layer applies `filters_layer2` filters with a kernel size of $(3, 1)$ to learn temporal patterns in the data.
   - The third convolutional layer applies `filters_layer3` filters with the same kernel size to extract deeper representations.

3. **Activation Function**: Each convolutional layer is followed by the Leaky ReLU activation function to introduce non-linearity and prevent vanishing gradients.

4. **Pooling and Regularization**: Max-pooling layers with a kernel size of $(3, 1)$ are applied after the second and third convolutional layers to reduce the spatial dimensions and prevent overfitting. A dropout layer with a drop rate of 0.1 is used before the fully connected layer for further regularization.

5. **Flattening and Fully Connected Layer**: After the final pooling layer, the output is flattened and passed through a fully connected layer to generate a single scalar output, representing the predicted energy consumption. The size of the linear layer's input is dynamically computed based on a dummy input.

6. **Output Layer**: The final output is a single value representing the forecasted energy consumption, which aligns with the regression objective of the model.

This architecture was used consistently across all client nodes in the federated learning framework. Each client trained the model independently on its local data, and periodic weight aggregation was performed at the server. The PyTorch model architecture used for forecasting can be seen in Figure 2

## 3.3. Federated Learning Framework

To implement federated learning for neighborhood energy consumption prediction, we adopted a server-client architecture. The framework was designed to enable multiple client nodes to collaboratively train a global model while

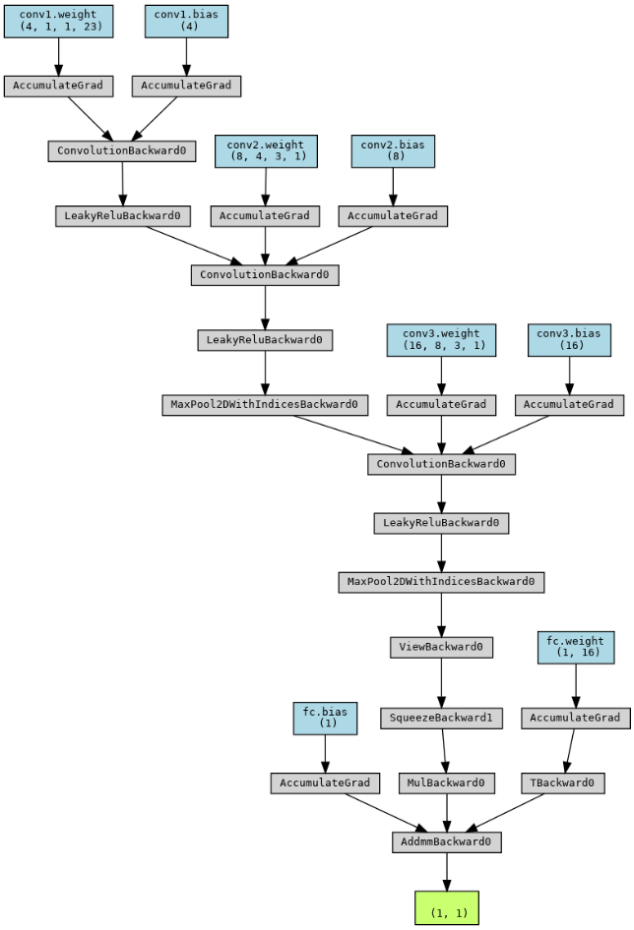


Figure 2. CNN Architecture

maintaining data privacy by keeping data localized on individual clients. Below, we describe the framework components and workflow in detail.

### 3.3.1 Framework Overview

**Server Node**

The central server facilitates communication between clients and coordinates the federated learning process. It performs the following tasks:

- Initializes and distributes the global model to all client nodes at the start of each training round.
- Aggregates model updates received from clients to update the global model using the Federated Averaging (FedAvg) algorithm proposed by Li et al. [7]
- Distributes the updated global model back to the clients for the next round of training.

**Client Nodes**

Each client represents a distinct household in the dataset,

simulating a real-world decentralized data scenario. Each client in a federated learning setup have the following tasks:

- Train their local models using their respective datasets.
- Compute and send model weight updates to the server at the end of each training round.
- Do not share raw data with the server or other clients, ensuring data privacy.

#### 3.3.2 Training Workflow

**Data Partitioning**
As mentioned in Section 3.1, the preprocessed dataset was divided into five parts using a round-robin approach. Four parts were allocated to the client nodes for training, while the fifth part was retained for training a standalone CNN model for baseline comparison. A shared test set was used for evaluation to ensure consistent comparison.

**Model Training**
Each client node trained its local CNN model using its data partition for a fixed number of local epochs. The same CNN architecture is required across all clients to ensure consistency and to enable weight aggregation.

**Weight Aggregation**
At the end of each training round, clients sent their updated model weights to the server. The server aggregated these weights using the Federated Averaging (FedAvg) algorithm:

$$\theta_{\text{global}} = \frac{1}{N} \sum_{i=1}^{N} \theta_i$$

where $\theta_i$ are the weights from the $i$-th client and $N$ is the total number of clients.
Since the data among clients is evenly distributed, each contribute equally to the model being trained. This in turn means each client's contribution is equally weighted during the aggregation. This can be adjusted based on the relevance of real world clients and how much data they contribute during training.

**Global Model Distribution**
After aggregation, the updated global model was distributed by the server back to all clients for the next round of training. This iterative process was repeated for several rounds until convergence.

**Communication**
To enable efficient communication between the server and clients, we utilized **gRPC**, a high-performance, open-source RPC framework. gRPC facilitated asynchronous communication between the nodes and the server, ensuring seamless interaction in the federated learning setup.

**Evaluation**
At the end of training, the global model and the standalone CNN model were evaluated using the shared test dataset. Metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were used to compare the performance of the federated model with the baseline standalone model.

A pictorial representation of our distributed ferderated learning setup can be seen in Figure 3

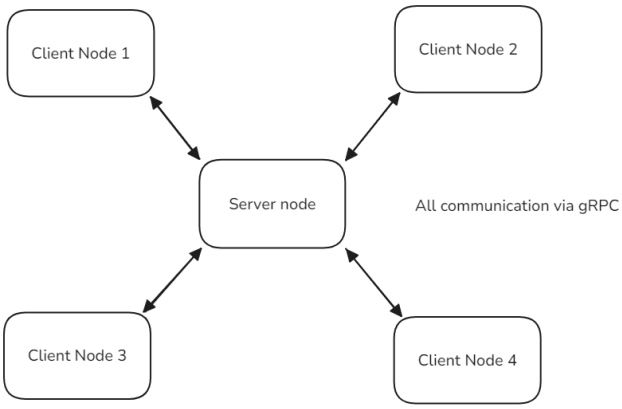


Figure 3. Federated Learning Setup

For the purpose of this study, we have used the Advanced Privacy Preserving Federated Learning Framework (APPFL) [8] for communication between the server and clients.

## 4. Experiments

To evaluate the performance of the federated learning framework for neighborhood energy consumption prediction, we conducted two separate experiments as mentioned earlier in Section 3.

1. Federated Learning with 4 clients
2. Standalone CNN with Bayesian Optimization

### 4.1. Federated Learning Setup

In the federated setup, we trained the CNN model of each client with the following hyperparameters:

- Sequence Length: 18
- Learning Rate: 0.01
- Epochs: 20

These hyperparameters were chosen to balance training efficiency and model performance.

### 4.2. Standalone CNN setup

In the standalone CNN setup, we first adopted the same model architecture and hyperparameters as those used in the

federated learning setup to ensure a fair comparison of the results.

Next, given the results of the standalone CNN model on the test dataset, we optimized 7 hyperparameters of the model using Bayesian Optimization through the ax-platform package by Facebook [9]. The optimization procedure utilized a combination of 8 SOBOL and 11 Bayesian optimization steps to tune the hyperparameters of the model across 20 iterations.

The hyperparameter values suggested by the Bayesian Optimization model are as follows,

- Sequence Length: 161
- Learning Rate: 0.0001
- Epochs: 50
- Drop Rate: 0.2
- Number of Filters: 32, 1, 1

This approach allowed us to optimize the standalone model's performance while maintaining a fair evaluation against the federated model.

## 4.3. Results

### 4.3.1 Federated learning results

During the training process, we monitored the training and validation errors across all client nodes in each epoch. The individual client training and validation losses can be seen in the figures 4, 5, 6 and 7.
Please note that due to the initial disparity in values, the validation loss starts significantly higher than the training loss. We opted not to overlap the training and validation curves as this approach would obscure the representation of the training loss curve.

### 4.3.2 Standalone CNN results

Similar to the federated learning architecture, during the training process of the standalone CNN model, we monitored the training and validation MAE at each epoch, initially through the baseline hyperparameters defined in section 4.1, leading to an RMSE score of 37308.6512 on the test dataset. However, by optimizing the hyperparameters of the CNN model using Bayesian Optimization, the algorithm was able to find a set of hyperparameters which produced the least RMSE on the test dataset of 331.5852.

The training and validation error loss over the epochs after Bayesian Optimization have been shown in Figure 8.

### 4.3.3 Final results

The results of both approaches on a common test dataset can be seen in Table 1.

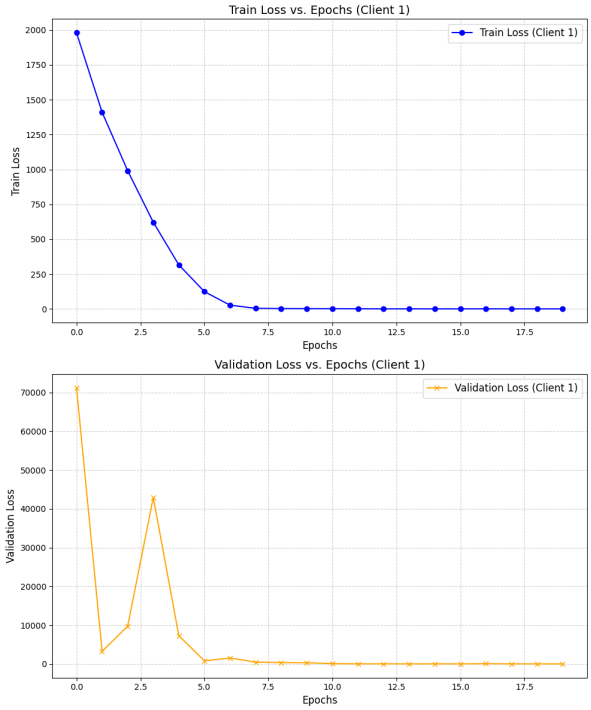


Figure 4. Client 1 Training and Validation errors

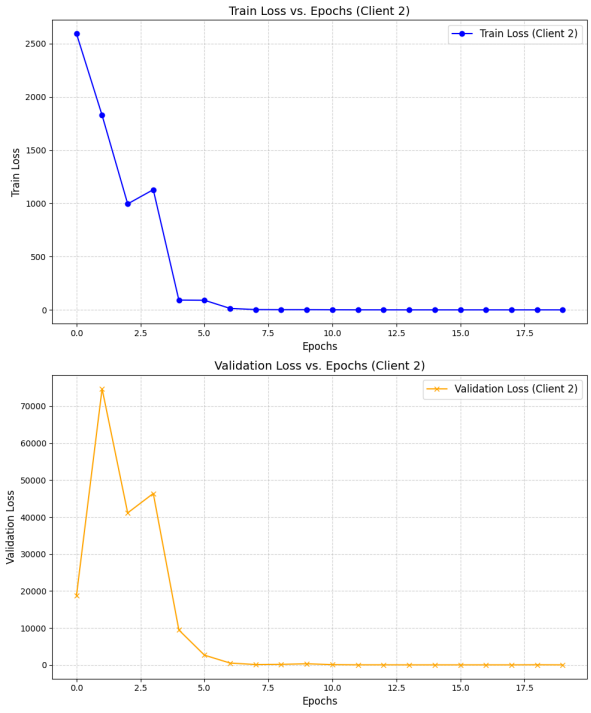


Figure 5. Client 2 Training and Validation errors

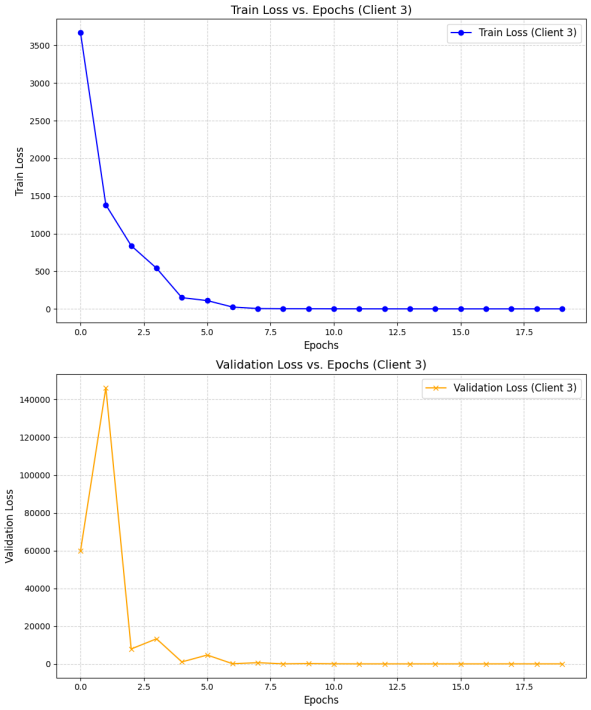


Figure 6. Client 3 Training and Validation errors

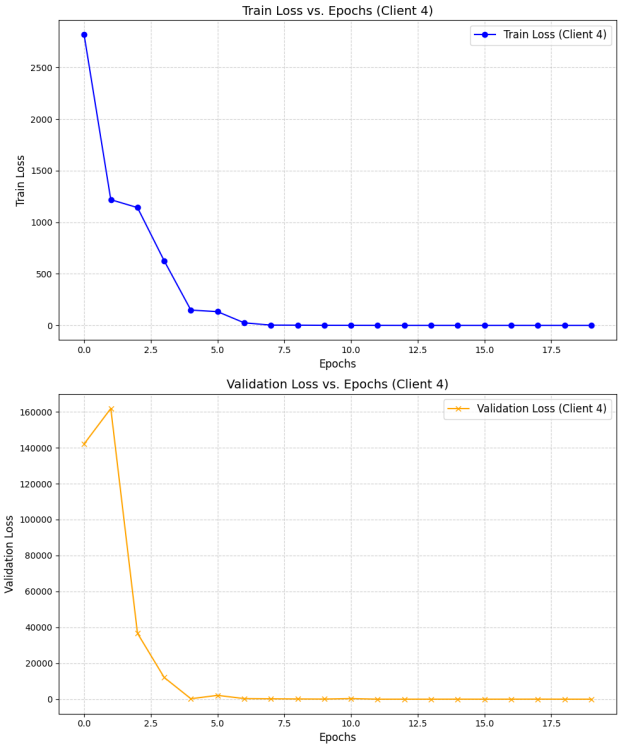


Figure 7. Client 4 Training and Validation errors

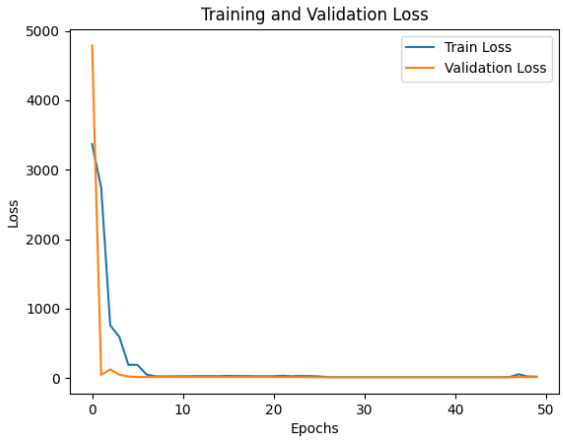


Figure 8. Standalone CNN Training and Validation errors

| Model | Final RMSE |
|---|---|
| Federated CNN | 36.043475 |
| Standalone CNN | 331.5852 |

Table 1. Final RMSE Evaluation on Test Set

## 5. Conclusions

Energy consumption forecasting is a crucial area of study due to its significant environmental implications. However, safeguarding the privacy of household energy usage data, while enabling residents to gain insights into their consumption patterns through lightweight models, is essential for scaling these solutions to a broader population. Our experiment demonstrates that convolutional neural networks (CNNs) within a federated learning framework outperform standalone CNN models trained solely on individual energy usage profiles, by a factor of 10, a significant increase in performance, while safeguarding the privacy of individual households.

This has broad implications not only for energy consumption forecasting, but also for other domains such as healthcare, communications, and business, where maintaining the privacy of data collected and managed at the source is crucial to ensure anonymity among different agents in the system. A limitation of our experiment was the simulation of different households within a neighborhood by partitioning the original dataset, which is non-IID. This approach may not accurately reflect the true energy consumption patterns of individual households, given the variability in their usage behaviors. This issue could be addressed in future research on this topic, which could use separate datasets to train the clients of a federated learning architecture.

## References


[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 1

[2] Tingting Chen, Xueping Liu, Bizhong Xia, Wei Wang, and Yongzhi Lai. Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder. *IEEE Access*, 8:47072–47081, 2020. 1

[3] Afaf Taïk and Soumaya Cherkaoui. Electrical load forecasting using edge computing and federated learning. In *ICC 2020-2020 IEEE international conference on communications (ICC)*, pages 1–6. IEEE, 2020. 1, 2

[4] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019. 2

[5] Haizhou Liu, Xuan Zhang, Xinwei Shen, and Hongbin Sun. A federated learning framework for smart grids: Securing power traces in collaborative learning. *arXiv preprint arXiv:2103.11870*, 2021. 2

[6] Marco Savi and Fabrizio Olivadese. Short-term energy consumption forecasting at the edge: A federated learning approach. *IEEE Access*, 9:95949–95969, 2021. 2

[7] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020. 3

[8] Zilinghan Li, Shilan He, Ze Yang, Minseok Ryu, Kibaek Kim, and Ravi Madduri. Advances in appfl: A comprehensive and extensible federated learning framework. *arXiv preprint arXiv:2409.11585*, 2024. 4

[9] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. 5