Electrical Load Forecasting Using Edge Computing and Federated Learning

Venkata Rama Sai Badam
Department of Computer Science
University of Central Florida
Email: ve253217@ucf.edu

***Abstract*— In this replication study, we attempt to reproduce the findings of the original paper, "Electrical Load Forecasting Using Edge Computing and Federated Learning" by training deep learning models for load forecasting using federated learning on a dataset of household electricity consumption data from 25 houses in Texas, USA. We employ Tensorflow as the machine learning framework and utilize the slightly different model architecture as compared to the original paper. The results demonstrate that federated learning can be effectively applied for household load forecasting, achieving comparable accuracy to the centralized training approach while preserving data privacy while reducing the amount of data transfer.**

Index Terms : **Federated Learning, Privacy , Load Forecasting**

I. Introduction

In the modern world, where energy systems are becoming more advanced and interconnected, the integration of technology is crucial for the smooth operation of smart grids. One of the major challenges faced by utility providers is predicting short-term electrical loads accurately. This is important because it helps ensure grid stability, proper resource allocation, and overall system reliability. Let's delve into the world of Electrical Load Forecasting and explore how Edge Computing and Federated Learning can be game-changers in overcoming the limitations of traditional methods.

Load forecasting is incredibly important in the world of smart grids. With the rise of interconnected grids, utilities can now optimize resource usage, boost energy efficiency, and respond quickly to changes in demand, all thanks to accurate short-term load predictions. These predictions empower grid operators to make informed decisions and manage peak loads effectively, ensuring a stable and reliable energy infrastructure.

However, traditional machine learning approaches have their limitations when it comes to short-term load forecasting in smart grids. This is because these grids are complex and dynamic, requiring continuous model updates to keep up with changing patterns. Centralized processing can lead to delays, and privacy concerns can arise with the storage of sensitive data. That's where innovative approaches like Federated Learning come into play.

Federated Learning is a revolutionary approach that combines the knowledge of multiple decentralized devices while ensuring data privacy. In the context of short-term load forecasting, it enables the training of machine learning models across a network of devices without the need for a central data hub. This approach not only protects privacy but also makes the model more adaptable to local variations in load patterns. The collaborative nature of Federated Learning leads to more accurate and reliable short-term load forecasts (STLF), benefiting both grid operators and consumers alike.

Federated learning, a decentralized machine learning approach, is broadly categorized into two main types: vertical federated learning and horizontal federated learning.

Vertical federated learning is characterized by different parties possessing distinct types of data related to the same set of entities or individuals. Each party typically holds information on different features or attributes of the shared dataset. Rather than directly exchanging raw data, these parties collaborate by sharing aggregated statistics or gradients specific to their data attributes. This collaborative approach allows for the training of a model across distributed datasets without compromising data privacy.

On the other hand, horizontal federated learning involves multiple parties with similar types of data but for different subsets of entities or individuals. Each party retains data instances for a specific subset of the overall dataset. Collaboration in horizontal federated learning entails joint model training on local datasets, followed by the exchange and aggregation of model parameters to enhance the global model. This approach facilitates collaborative model building while preserving the privacy of individual data instances.

The problem at hand which is the electrical load forecasting falls under vertical federated learning as no two houses have the same appliances, so the data they use to train the model might be different (have different features).

The remainder of this paper is structured as follows: Section II discusses related works focusing on load prediction and privacy. In Section III, we define the proposed approach and used methods. Section IV introduces the simulations. Section V discusses the results. Then in Section VI concludes the paper.

II. Related Work

In terms of privacy, the paper [1] highlights the risks associated with fine-grained consumption data being sent over networks, as it can reveal sensitive information about households. Various methods have been proposed to protect user privacy, such as clustering-based approaches and data aggregation techniques. However, none of the existing papers [2], [3], [4] address

both privacy and prediction accuracy simultaneously.

The proposed approach in the paper combines edge computing and federated learning to address the challenges of privacy and data diversity in load forecasting. The authors suggest using the Edge Equipment in the Home Area Network (HAN) to carry out operations related to client selection and training neural networks at the edge. Federated learning is used to train a global LSTM-based model for STLF, allowing the use of data to train the model without compromising residents' privacy. The paper presents a network architecture and explains the FederatedAveraging algorithm for aggregating client updates and updating the global model.

The paper [5] proposes a decentralized approach to training deep neural networks using data from mobile devices. The authors propose a practical algorithm called FederatedAveraging, which combines local stochastic gradient descent (SGD) on each client with a server that performs model averaging. They conduct extensive experiments to evaluate the approach, showing its robustness to unbalanced and non-IID data distributions, as well as its ability to significantly reduce communication rounds compared to synchronized SGD. The paper also discusses the privacy advantages of federated learning compared to centralized training on persisted data.

They also mention previous studies on distributed training and optimization, but highlight the unique challenges and considerations of federated learning. They discuss the work of McDonald et al. [6] and Povey et al. [7] on distributed training of perceptrons and speech recognition DNNs, respectively. However, these works focus on the cluster/data center setting and do not consider the unbalanced and non-IID data distributions that are characteristic of federated learning. The authors adapt the algorithmic approach of these works to the federated setting and conduct an empirical evaluation that addresses the specific questions and challenges of federated learning.

[5] also mentions the work of Neverova et al. [8], which discusses the advantages of keeping sensitive user data on the device. Additionally, the work of Shokri and Shmatikov [9] is related as they also emphasize privacy and address communication costs by sharing only a subset of parameters during each round of communication. However, these works do not consider unbalanced and non-IID data distributions, and their empirical evaluations are limited. The authors of this paper aim to fill these gaps by focusing on the non-IID and unbalanced properties of federated optimization and addressing the critical communication constraints.

III. SYSTEM MODEL

In this section, we present the model architecture and how the model aggregation works and brief working of the federated averaging algorithm.

A. Model Architecture

The proposed model is a recurrent neural network (RNN) with two Long Short-Term Memory (LSTM) layers and two fully connected (dense) layers. The first LSTM layer has 64 units and is configured to return sequences, meaning that it will output a sequence of vectors for each input sequence. The second LSTM layer has 32 units and is configured to return a single vector for each input sequence. The two dense layers each have 10 units and are activated with the rectified linear unit (ReLU) activation function. The L2 regularization penalty is applied to the weights and biases of the dense layers to help prevent overfitting. The final dense layer has a number of units equal to the desired horizon of the model, which is the number of future time steps that the model should predict.

Here's a more detailed breakdown of the model architecture:

The first LSTM layer takes as input a sequence of vectors, where each vector represents a single time step. The layer processes the input sequence and outputs a sequence of vectors, where each vector represents the hidden state of the LSTM cell at the corresponding time step. The 'return_sequences=True' parameter ensures that the output of the LSTM layer is a sequence of vectors, rather than a single vector.

The second LSTM layer takes as input the sequence of vectors output by the first LSTM layer. It processes the input sequence and outputs a single vector, which represents the final hidden state of the LSTM cell.

The first dense layer takes as input the single vector output by the second LSTM layer. It transforms the vector into a 10-dimensional vector, which represents the output of the first dense layer. The 'activation='relu'' parameter specifies that the ReLU activation function should be applied to each element of the output vector.

The second dense layer takes as input the 10-dimensional vector output by the first dense layer. It transforms the vector into a vector of length 'horizon', which represents the predicted values for the 'horizon' future time steps. The 'activation='relu'' parameter specifies that the ReLU activation function should be applied to each element of the output vector.

The L2 regularization penalty is applied to the weights and biases of the dense layers to help prevent overfitting. Overfitting is a common problem in machine learning, and it occurs when a model learns the training data too well and is unable to generalize to new, unseen data. The L2 regularization penalty helps to reduce overfitting by adding a penalty to the loss function that is proportional to the sum of the squares of the weights and biases. This penalty encourages the model to learn smaller weights and biases, which can help to improve its generalization performance. One thing to note is that the final model architecture chosen for this work was selected after testing out multiple configurations. The hyperparameters of this model were not tuned using any grid search or random search algorithms.

This model architecture is well-suited for time series forecasting tasks, particularly those involving long-range dependencies and non-linear relationships. The LSTM layers effectively capture temporal patterns, while the Dense layers introduce non-linearity and provide additional feature extraction capabilities. The L2 regularization helps prevent overfitting and improve generalization performance.

```
lstm_model = tf.keras.models.Sequential([
  tf.keras.layers.LSTM(64, return_sequences=True, input_shape=x_train.shape[-2:]),
  tf.keras.layers.Dense(10, activation='relu',
                        kernel_regularizer=tf.keras.regularizers.L2(l2=0.01)),
  tf.keras.layers.LSTM(32),
  tf.keras.layers.Dense(10, activation='relu',
                        kernel_regularizer=tf.keras.regularizers.L2(l2=0.01),
                        bias_regularizer=tf.keras.regularizers.L2(l2=0.01)),
  tf.keras.layers.Dense(units=horizon),
])
```

Fig. 1. Model Structure

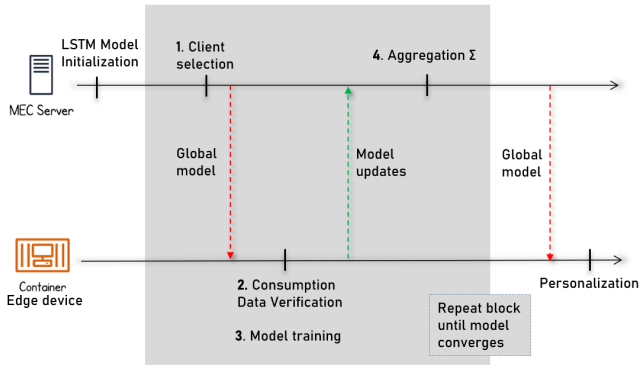

Fig. 2. Algorithm

B. Algorithm

The authors in the original paper discuss the use of the FederatedAveraging algorithm in the context of federated learning for electrical load forecasting. The FederatedAveraging algorithm is used to aggregate the updates from individual clients and update the global model in a privacy-preserving manner.

The algorithm works as follows:

- Initialization: The initial global model is randomly initialized or pre-trained using publicly available data.
- Training Rounds: The training process consists of multiple rounds. In each round, a subset of clients is selected to participate in the training. The server sends the current global model to these clients.
- Local Training: Each client performs local training using its own data. The clients compute Stochastic Gradient Descent (SGD) updates on their locally-stored data. These updates represent the changes to the model's weights based on their local data.
- Model Aggregation: After the local training, the clients send their updated models (weights) back to the server. The server aggregates these client updates to build a new global model. The aggregation is typically done by averaging the weights of the models received from the clients.
- Model Distribution: The new global model is then sent to another subset of clients for the next round of training. This process is repeated until the desired prediction accuracy is reached or a maximum number of rounds is reached.

IV. Simulation Study

A. Data Pre-processing

In the initial stages of the data preprocessing pipeline, a fundamental step involves addressing missing values within the dataset obtained from pecanstreet.org. The original research encompassed a dataset comprising 200 houses for training and testing. However, the available free sample contains data from 25 houses. Given the inherent variability in the types of appliances across different households, it becomes necessary to standardize the input shape across all houses. To achieve this, null values in the columns have been imputed with zeroes. The resultant dataset, post the removal of non-contributory columns such as dataid, encompasses a total of 73 variables.

The datasets available from the source have diverse frequency intervals, ranging from 1 second to 15 minutes. Although the authors advocate for the utilization of data with a 1-hour frequency, the pecanstreet.org source only provides data at a maximum frequency of 15 minutes. To align with the authors preference for 1-hour frequency data, a conversion process from 15 minutes to 1 hour is implemented.

Noteworthy is the organization of the dataset, wherein data for all houses is initially combined into a single dataset. To be inline with the federated learning paradigm adopted in this work, a subsequent step involves the segregation of data pertaining to each house. This separation is facilitated by leveraging the dataid parameter, which uniquely identifies individual houses. This setup makes sure we have a dataset for each house that's specific to that house. This is important for the way we're using federated learning in the later parts of the project.

Converting a time series dataset to a windowed format is imperative when employing Long Short-Term Memory (LSTM) models for time series forecasting. LSTMs excel at capturing sequential dependencies and temporal patterns in data, making them well-suited for time-sensitive predictions. By creating input-output pairs in the form of windows, where each window represents a sequence of consecutive time steps, the model gains the ability to discern and learn from temporal relationships within the data. This conversion ensures that the LSTM is provided with a structured input that encapsulates the historical context necessary for accurate predictions. The windowed dataset enables the LSTM to effectively learn short-term and long-term dependencies, facilitating the extraction of meaningful patterns critical for forecasting future values in the time series. For example, examining Fig 3 gives a good picture of how the raw timeseries dataset is converted to windowed dataset. The function keeps on iterating the dataset until it reaches the last data point in the dataset. In essence, the transformation from a raw time series to a windowed dataset enhances the LSTM's capacity to comprehend the sequential nature of the data, ultimately leading to more robust and precise time series forecasts.

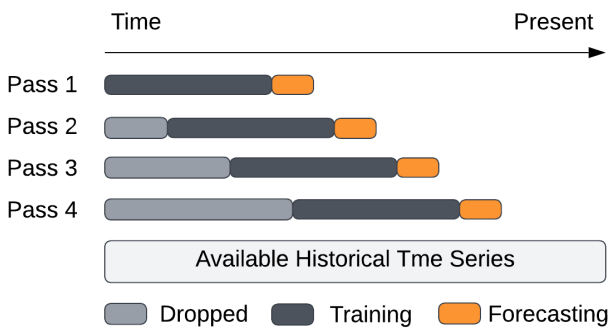

Fig. 3. Sliding Window

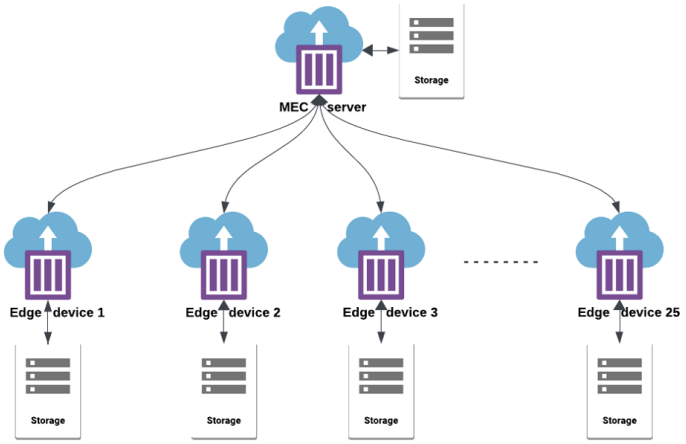

Fig. 4. Docker setup

B. Evaluation metrics

The original work employed RMSE and MAPE as performance evaluation metrics for the constructed models. In this study, MAPE has been replaced with WMAPE to alleviate the potential for division-by-zero errors during runtime, an issue that can arise with MAPE.

1) RMSE: The root mean square error (RMSE) measures the average difference between a model's predicted values and the actual values.

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N}(y_i - \hat{y}_i)^2}{N}}$$

where :
y_i is the actual value
$\hat{y}_i$ is the predicted value
N is the number of observations

2) WMAPE: WMAPE stands for Weighted Mean Absolute Percentage Error. It is a metric used to assess the effectiveness of regression or forecasting models. It is a variation of MAPE, where the mean absolute percentage errors are treated as a weighted arithmetic mean. Typically, the absolute percentage errors are weighted by the actual values. For instance, in sales forecasting, errors are weighted by sales volume. This effectively addresses the "infinite error" issue.

$$WMAPE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\sum_{i=1}^{n} y_i} \times 100$$

C. Simulation Setup

In our simulation, Docker containers were selected as the preferred virtualization environment. The MEC server, responsible for model aggregation, operates within its designated container, coupled with a Docker volume to facilitate efficient data handling. Each individual edge device functions within its dedicated container, also associated with a Docker volume housing localized data. Given the participation of 20 houses in the training phase and 5 houses in the testing phase, three distinct images were generated to encapsulate the MEC server, clients involved in training, and clients participating in testing, respectively. The host machine, an AMD Ryzen 9 6900HS with 16 GB of RAM, serves as the underlying infrastructure for this simulation environment.

Within the Docker environment, the Multi-Edge Computing (MEC) server orchestrates the federated learning process, initiating the cycle by initializing a random model. Subsequently, a predetermined number of clients, typically either 5 or 10, are selected based on the specific scenario. Following client selection, the random model is distributed to these chosen clients. Each client then embarks on local training of the model, utilizing its own unique dataset. Upon completion of local training, the updated models are transmitted back to the MEC server for aggregation, carrying the collective insights gained from each client's local training. The resulting aggregated model, having trained on patterns of the participating clients, is then deployed to designated testing containers for metric evaluation. Each testing container, equipped with the aggregated model, utilizes it to generate predictions on unseen testing data.

The averaged metrics obtained from the testing containers, such as Root Mean Square Error (RMSE), serve as a benchmark for assessing the performance of the aggregated model. The lower the RMSE, the better the model's ability to accurately predict future load patterns. Should the model's performance fall below satisfactory thresholds, or if it fails to meet predefined criteria, a subsequent round of training is initiated. In this subsequent training round, the aggregated model from the previous iteration serves as the foundational base model for further refinement, incorporating the newly acquired knowledge from the clients. This iterative process continues until the aggregated model achieves the desired level of performance, consistently delivering accurate load forecasts that empower grid operators to optimize resource utilization, enhance energy efficiency, and maintain a stable and reliable smart grid infrastructure.

V. Results

A. Evaluated Scenarios

As outlined in Table I, in alignment with the methodology employed in the initial paper, four distinct scenarios were examined during the course of this investigation. Each scenario randomly selects either five or ten clients from the training set. In contrast to the original paper, the federated learning algorithm in this study was executed for a duration of 5 rounds for each scenario.

TABLE I
SCENARIOS CONSIDERED

scenarios	No. of clients	Local Epochs
1	5	1
2	5	5
3	10	1
4	10	5

B. Global models

The scenarios we explored led to global models created through federated learning. These models performance was evaluated using Root Mean Square Error (RMSE) and Weighted Mean Absolute Percentage Error (WMAPE), which you can find in Tables II and III. Instead of using Mean Absolute Percentage Error (MAPE), we used WMAPE to handle cases where there were no consumption values, a change from the original paper's method.

Table II summarizes how the clients performed in different scenarios. We focused our load forecast on individual households over a short one-hour period. One thing we noticed is that the global model fits some clients better than others, likely because not all clients have similar profiles. We also found that it's better to choose a larger number of clients in each round.

An examination of Table II reveals that each metric encompasses sub-columns for "min," "mean," and "max" values, representing the minimum, average, and maximum recorded for the models during training and testing phases. Notably, the WMAPE exhibited excessively low minimum values when evaluating models on training data. This suggests potential overfitting, implying that the models memorized the training data too closely, leading to biased performance on specific household datasets. Conversely, the exceedingly high maximum WMAPE values observed for models evaluated on testing data (e.g., 94.23%, 96.20%) indicate a lack of generalizability. This signifies that the models struggled to adapt effectively to unseen data.

Furthermore, it is self-evident that models evaluated on testing data cannot outperform their training counterparts. As the models haven't been exposed to the testing data, minimum WMAPE values such as 19.69%, 18.97% are reasonable, while maximum values like 89.91%, 87.98% reflect the inherent challenge of predicting unseen data with complete accuracy.

However, the mean values for both training and testing data offer a more nuanced perspective. Ranging between 23.77% and 41.87% for training data and between 40.39% and 56.49% for testing data, these values suggest that the model's performance is neither exceptional nor abysmal. This indicates merit in further investigating the model's capabilities and potential for improvement, particularly in addressing the observed overfitting and generalizability limitations.

TABLE II
RMSE AND WMAPE FOR GLOBAL MODELS IN THE CONSIDERED SCENARIOS FOR THE 20 PARTICIPATING CLIENTS

	RMSE			WMAPE		
Scenario	Min	Mean	Max	Min	Mean	Max
1	0.0867	0.1319	0.2549	0.25%	23.73%	94.23%
2	0.0744	0.1306	0.2829	5.73%	27.66%	72.29%
3	0.1079	0.153	0.2723	1.45%	30.45%	85.56%
4	0.1511	0.2046	0.265	6.02%	41.87%	96.20%

TABLE III
RMSE AND WMAPE FOR GLOBAL MODELS IN THE CONSIDERED SCENARIOS FOR 5 NON-PARTICIPATING CLIENTS

	RMSE			WMAPE		
Scenario	Min	Mean	Max	Min	Mean	Max
1	0.1019	0.1313	0.1585	19.69%	56.49%	72.43%
2	0.0823	0.1284	0.1694	16.06%	40.39%	66.82%
3	0.1154	0.1532	0.1921	18.97%	48.68%	87.98%
4	0.0951	0.1908	0.2159	17.95%	49.74%	89.91%

C. Personalized models

In this section, we explore the potential enhancements in metrics achievable through the utilization of personalized models. Initially, we investigate whether retraining the model locally for participant clients yields improved results. Subsequently, we extend the same approach to clients who were not part of the training process. The models are retrained for 1 and 5 epochs for each client, and the outcomes for the participating clients are presented in Table IV, while those for the non-participating clients are outlined in Table V.

Examining the impact of varying the number of epochs on

TABLE IV
RMSE AND MAPE AFTER PERSONALIZATION FOR 20 PARTICIPATING CLIENTS

Scenario	Epochs	RMSE			WMAPE		
		Min	Mean	Max	Min	Mean	Max
1	1	0.0806	0.1137	0.1704	0.01%	28.20%	59.40%
	5	0.0533	0.0988	0.1351	0.33%	11.97%	35.31%
2	1	0.0868	0.1186	0.1656	1.33%	36.46%	76.24%
	5	0.0604	0.0971	0.1348	0.79%	11.56%	45.40%
3	1	0.0685	0.1149	0.1685	2.22%	31.32%	62.42%
	5	0.0535	0.0965	0.1395	1.88%	12.60%	41.19%
4	1	0.0727	0.1199	0.1658	6.81%	37.27%	65.76%
	5	0.0652	0.1016	0.1379	1.25%	16.13%	45.04%

personalized models for participating clients reveals intriguing trends. When transitioning from 1 epoch to 5 epochs, substantial shifts in metrics become apparent. Taking Model 1 as an example, the WMAPE for participating clients decreases from 59.40% to 35.31%, representing a significant improvement of approximately 40.69%. Similar patterns are observed across all models, demonstrating a consistent enhancement in forecasting accuracy as the number of epochs increases.

Comparing the metrics between 1 epoch and 5 epochs for both participating and non-participating clients reinforces the positive influence of additional training epochs. The observed reductions in WMAPE percentages indicate that extended training contributes to refining personalized models and improving their predictive capabilities.

Moving on to the comparison between training and testing sets, the personalized models consistently outperform in the training phase. This is expected, as the models are tailored to the specific characteristics of the training data. However, it's noteworthy that the gap between training and testing metrics diminishes as the number of epochs increases, indicating a more robust generalization of the models to unseen data.

To visually demonstrate the enhancements in predictions through personalization, we randomly chose a client from the participant set (client 4031) and a client from the non-participant set (client 8565). We employed global model 3 alongside the corresponding personalized models. The actual load profiles and the predicted profiles are presented in Fig. 3 and Fig. 4. Notably, both models effectively capture the overall behavior of the consumption profiles.

In conclusion, the investigation into personalized models for participating and non-participating clients highlights the efficacy of personalized training, particularly with an increased number of epochs. The substantial improvements in metrics underscore the adaptability of these models to diverse client scenarios. Additionally, the comparison between training and testing sets indicates the models' ability to generalize to new data, with enhanced performance as training epochs are extended. These findings emphasize the potential of personalized models in federated learning scenarios, showcasing their versatility and effectiveness in achieving accurate and reliable predictions.

D. Network gain

Instead of sending raw data to a central server for training, the model is sent to the data, the local model is trained on the local data, and then the updated model parameters are aggregated without the raw data leaving the local device.

This section analyzes the data transfer requirements for a single round of federated learning compared to the centralized approach.

Federated Learning

1) Local Training:
   Clients perform local training on the global model using their local data.

TABLE V
RMSE AND MAPE AFTER PERSONALIZATION FOR 5 NON-PARTICIPATING CLIENTS

Scenario	Epochs	RMSE			WMAPE%		
		Min	Mean	Max	Min	Mean	Max
1	1	0.0569	0.1117	0.1704	0.01%	27.20%	59.40%
	5	0.0533	0.0977	0.1351	0.33%	13.00%	35.31%
2	1	0.0546	0.1158	0.1656	1.33%	33.80%	76.24%
	5	0.0590	0.0956	0.1348	0.21%	12.24%	45.40%
3	1	0.0554	0.1134	0.1685	2.22%	31.18%	62.42%
	5	0.0535	0.0953	0.1395	1.88%	13.53%	41.19%
4	1	0.0550	0.1177	0.1658	1.17%	35.71%	65.76%
	5	0.0648	0.1009	0.1379	1.25%	17.99%	51.45%

2) Model Upload:
   Each client uploads the updated model to the MEC server (aggregator). Each model is approximately 495Kb.
   - Total upload: 20 clients * 495Kb/client = 9.9Mb
3) Model Testing:
   - Aggregated model download: 5 non-training clients download the aggregated model (188Kb each).
   - Total download: 5 clients * 188Kb/client = 0.94Mb (3.76Mb when rounded to the nearest 0.01Mb)
   - Performance Evaluation: Clients send prediction results and metrics (175bytes each) to MEC server.
   - Total transfer: 5 clients * 175bytes/client = 875bytes.

- Total Data Transfer Per Round: 9.9Mb + 3.76Mb + 0.875 Mb $\approx$ 13.54Mb (rounded to 0.01Mb)

Centralized Learning

Centralized learning requires transferring all training and testing data to the MEC server:

1) Individual dataset size: 3.1Mb/house
2) Total training data: 20 houses * 3.1Mb/house $\approx$ 62Mb
3) Total testing data: 5 houses * 3.1Mb/house $\approx$ 15.5Mb
4) Total data transfer: 62Mb + 15.5Mb $\approx$ 77.5Mb

Network Gain Analysis

Network gain is calculated as the reduction in data transfer using federated learning compared to centralized learning:

$$\text{Network Gain} = \frac{\text{Centralized Transfer} - \text{Federated Transfer}}{\text{Centralized Transfer}} \times 100\%$$

$$= \frac{77.5\text{Mb} - \ 13.54\text{Mb}}{77.5\text{Mb}} \times 100\%$$

Federated learning significantly reduces data transfer compared to centralized learning, achieving a network gain of 82.26%. This is due to the reduced need to transfer large datasets, especially when dealing with geographically distributed clients.

This analysis provides valuable insights into the network efficiency of federated learning, demonstrating its potential for resource-constrained environments.

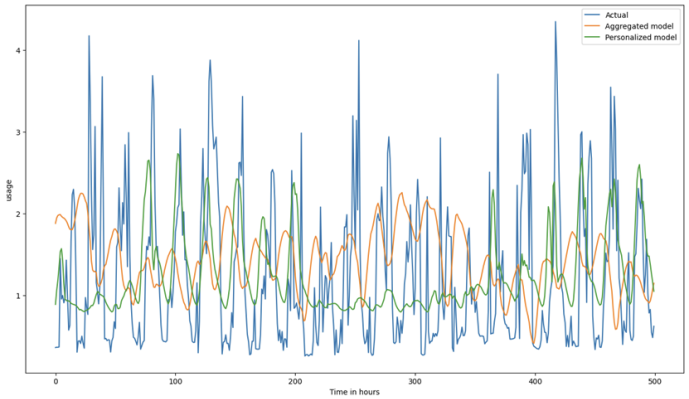

Fig. 5. Predictions for next hour consumption for client 4031 who participated in training the global model 3. Local training for 5 epochs reduced RMSE from 3.9 kW to 2.4 kW.

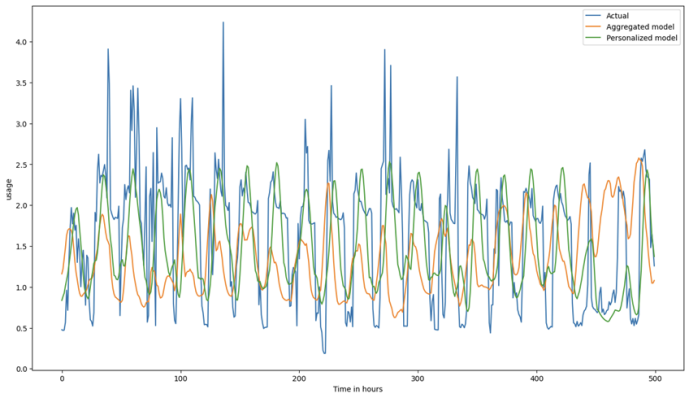

Fig. 6. Predictions for next hour consumption for client 8565 who did not participate in training the global model 3. Local training for 5 epochs reduced RMSE from 3.8 kW to 3.1 kW.

VI. Conclusion

The personalized models demonstrate superior metric values compared to their aggregated counterparts, showcasing the efficacy of individualized approaches in federated learning. Unlike the findings presented in the referenced paper, the personalized models in my study did not exhibit maximum values for WMAPE, potentially attributed to the limitation of available data. This discrepancy suggests that further data collection and diverse client representation could potentially enhance the performance of personalized models.

Moreover, federated learning presents a compelling solution for situations where a client lacks previous data and aims to predict power consumption while preserving privacy. Although models trained through centralized learning perform well in comparison to federated learning, the latter excels in maintaining data privacy. But we have already seen how much data we have saved by using the federated learning approach. you could easily estimate the amount of bandwidth saved if this were to apply for millions of houses.

While there may be a slight sacrifice in accuracy for privacy preservation, federated learning becomes a crucial tool, allowing clients to benefit from collective insights without compromising sensitive individual data. This makes federated learning an ideal choice for scenarios where privacy concerns are paramount, demonstrating its versatility in addressing a spectrum of real-world challenges.

References

[1] A. Taïk and S. Cherkaoui, "Electrical load forecasting using edge computing and federated learning," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[2] M. Badra and S. Zeadally, "Design and performance analysis of a virtual ring architecture for smart grid privacy," *IEEE Transactions on Information Forensics and Security*, vol. 9, pp. 321–329, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:10378045

[3] Y. Gong, Y. Cai, Y. Guo, and Y. Fang, "A privacy-preserving scheme for incentive-based demand response in the smart grid," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1304–1313, 2016.

[4] H. Park, H. Kim, K. Chun, J. Lee, S. Lim, and I. Yie, "Untraceability of group signature schemes based on bilinear mapping and their improvement," *Fourth International Conference on Information Technology (ITNG'07)*, pp. 747–753, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:17502784

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[6] R. McDonald, K. Hall, and G. Mann, "Distributed training strategies for the structured perceptron," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, R. Kaplan, J. Burstein, M. Harper, and G. Penn, Eds. Los Angeles, California: Association for Computational Linguistics, Jun. 2010, pp. 456–464. [Online]. Available: https://aclanthology.org/N10-1069

[7] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of deep neural networks with natural gradient and parameter averaging," *CoRR*, vol. abs/1410.7455, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:5464187

[8] N. Neverova, C. Wolf, G. Lacey, A. Fridman, D. Chandra, B. Barbello, and G. W. Taylor, "Learning human identity from motion patterns," *IEEE Access*, vol. 4, pp. 1810–1820, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:215827018

[9] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1310–1321. [Online]. Available: https://doi.org/10.1145/2810103.2813687