Communication-Efficient Design of Machine Learning System for Energy Demand Forecasting of Electrical Vehicles

Jiacong Xu*, Riley Kilfoyle†, Zixiang Xiong† and Ligang Lu‡
*Dept of CS, Johns Hopkins University, Baltimore, MD 21218
†Dept of ECE, Texas A&M University, College Station, TX 77843
‡Shell International Exploration and Production Inc., Houston, TX 77082
jxu155@jhu.edu; rskilfoyle@tamu.edu; zx@ece.tamu.edu; ligang.lu@shell.com

***Abstract*—Machine learning has shown incredibly potential in many fields of application ranging from ChatGPT and Bard to Tesla's autonomous vehicles. These ML models require vast amounts of data and communications overhead in order to be effective. In this paper we propose a communication-efficient time series forecasting model combining the most recent advancements in MetaFormer architecture implemented across a federated series of learning nodes. The time series prediction performance and communication overhead cost of the distributed model is compared against a similar centralized model and shown to have parity in performance while consuming much lower data rates during training.**

I. Introduction

The Alternative Fuels Data Centre lists more than 54,000 EV charging stations (CSs) currently in operation in the United States. It is projected that the number of EVs on the road will rise by more than 4000% by 2030 [1]. As a result, there is a strong need to effectively and intelligently predict energy demands for EV CSs to mitigate their impact on the power grids without upgrading/expansion capability. Generally, the power grid supplies the energy for CSs once requests from EVs are received. Predicting the amount of energy needed at each CS over different periods of time will allow the power suppliers of CSs to purchase the desired amount of electricity at lower rates in order to save money and make charging at public stations more efficient – a key step towards smart charging [2]. In addition, the power grid can coordinate energy consumption via schedule management to reduce energy costs.

Energy prediction algorithms have been thoroughly studied in the past. Majidpour et al. [3] compared fast machine learning-based time-series prediction algorithms and found that the nearest neighbor algorithm showed improved accuracy. Ryu et al. [4] proposed deep learning (DL) load forecasting models and showed that DL methods exhibited better performance compared to other forecasting models. Paterakis et al. [5] compared a DL method with eight most commonly used machine learning methods such as nearest neighbors, support vector machines, Gaussian processes, regression trees, in energy demand prediction and showed that the DL method outperformed all eight other methods. However, conventional DL algorithms trained on individual CSs may not achieve high prediction accuracy due to insufficient data samples and failure to consider the influence from other CSs, particularly the nearby ones in the neighborhood [6].

We introduce machine learning based approaches which can not only significantly improve the accuracy of energy demand prediction, but also reduce the communication overhead for EV networks. In particular, we first introduce a communication model using the power provider as a centralized node to gather all information from the individual CS nodes in a considered metropolitan area (e.g., Houston, or London, or Amsterdam, etc.). We then develop a DL method to help the power provider accurately predict energy demands for the CSs in this area.

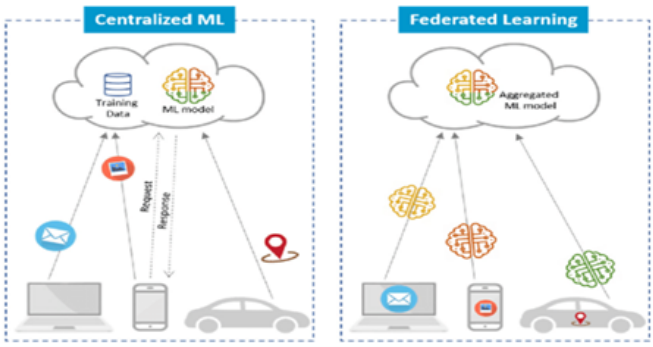

Fig. 1. Centralized vs FL architectures: In centralized learning (left), data is sent to the cloud, where the ML model is built. The model is accessed by a user through an API by sending a request to access one of the available services. In FL [7,8] (right), each device trains a model and sends its parameters to the server for aggregation. Data is kept on-devices and knowledge is shared through an aggregated model with peers.

Specifically, we adopt a federated learning (FL) approach [7], [8] to energy demand prediction. As seen in Fig. 1, FL is a form of distributed learning wherein updates to model parameters are calculated locally on individual devices rather than transmitting massive volumes of data to centralized servers for global model development. These local model updates are

Work supported in part by NSF grants ECCS-1923803, CCF-2007527, and CCF-2324397.

then transmitted to a centralized server for aggregation into a global model. Once the global model has been updated according to various algorithms, the updated global model is transmitted out to the CSs for the next round of training. In this fashion the individual data never leaves the local CSs, some of the computation is offloaded to local CSs, and a globally useful DL model is created. This reduces the increasing strain placed on communication networks while accomplishing the goal of training a useful global model. Thus, the advantage with FL is that the CSs only need to share their trained models obtained from their datasets instead of sharing their real datasets themselves, which generally are of large volume and often are prohibited by data privacy laws and security regulations.

Using real data obtained from charging stations in Fig. 2 of Dundee city, the United Kingdom between 2017 and 2018, which has 65,601 transactions that include IDs from 58 CSs, transaction ID for each CS, EV charging date, EV charging time, and consumed energy (in kWh) for each transaction [9], Saputra et al. [10] showed that simple location-based node clustering improves the accuracy of energy demand prediction up to 24.63% and decreases communication overhead by 83.4% compared with other baseline machine learning algorithms.

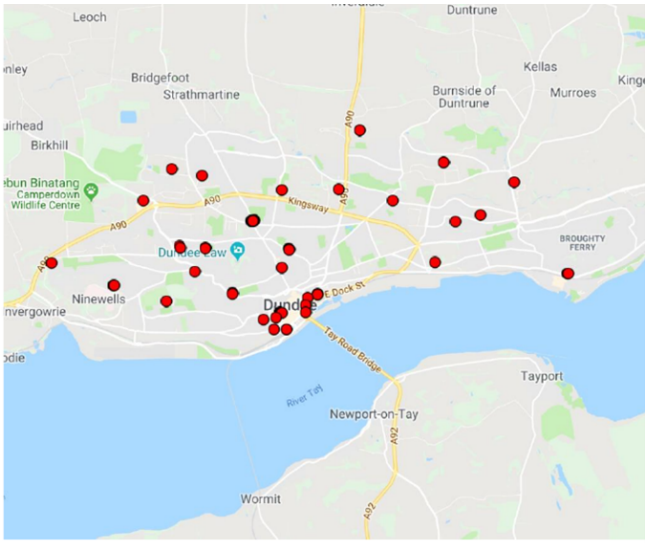

Fig. 2. Centralized vs FL architectures: In centralized learning (left), data is sent to the cloud, where ML model is built. The model is used by a user through an API by sending a request to access one of the available services. In FL [7], [8] (right), each device trains a model and sends its parameters to the server for aggregation. Data is kept on-devices and knowledge is shared through an aggregated model with peers.

The EV charging prediction task for an individual client is a uni-variate time-series forecasting problem, which has been previously investigated. Different from previous works that treat the hourly energy consumption as the prediction output, we aggregate 24 hours' energy consumption as one data point to do daily prediction. The reason is that the hourly data-sets often have no pattern or contain large number of zeros, which presents itself as much randomness, whereas the daily data exhibit clear trends of energy consumption variation.

Inspired by the great success of transformer for language and vision tasks, many advanced transformer architectures have been proposed for time series forecasting in recent two years. These models give better prediction performance compared with traditional RNNs and the most recent work PatchTST [11] out performs its MLP counterpart and shows state-of-the-art performance on many time-series forecasting benchmarks. In this paper, we borrow the the ideas from recent advances on efficient 2-D vision transformers and propose a lighter and better model called Local and Global Time Series Transformer (LoGTST).

For generic FL, sharing model parameters among all clients and the server will involve heavy communication overhead. To reduce the communication cost, Online-Fed randomly picks a specific number of clients for model updating. Partial sharing based online FL (PSO-Fed) [12] further alleviates this problem by only randomly sharing partial model parameters to those selected clients. However, in each global iteration, the clients in both of these two models can only access their own information or stay idled, which will hinder the convergence speed and limit its generalization ability. Thus, we propose Partial Sharing Global Forwarding FL (PSGF-Fed) that is built upon PSO-Fed but enables the server to randomly share small amount of partial parameters to all the clients. In this way, PSGF-Fed is able to reduce the total communication overhead by accelerating the convergence speed of each local model.

We strongly believe that our proposed novel LoGTST in conjunction with PSGF-Fed will greatly improve results reported by Saputra et al. [10] and Perry et al. in [6], in which algorithm validation was done by using the root mean squared error (RMSE) for energy demand prediction (with smaller RMSE indicating better prediction accuracy).

II. Method

The learning system in this application consists of two major parts: the DL model and the FL policy. For better prediction accuracy, the model should be able to capture the local and global trend of the input time series and the FL policy should aggregate the information from all the clients and lead the model to a good convergence. With the constraints of communication overhead, the space complexity of the learning model should be as small as possible and the time of information sharing between server and clients should be short as well.

A. MetaFormer

Current research on the model architecture design for time series forecasting can be divided into two camps: Transformer and Multi-Layer Perceptron (MLP). Different from previous transformer architectures, PatchTST [11] splits a time-series into patches and tokenizes these patches into vectors by patch embedding following Vision Transformer (ViT) [13]. This new transformer out-performs MLPs and other transformers such as DLinear [14] and FEDFormer [15] on many popular benchmarks.

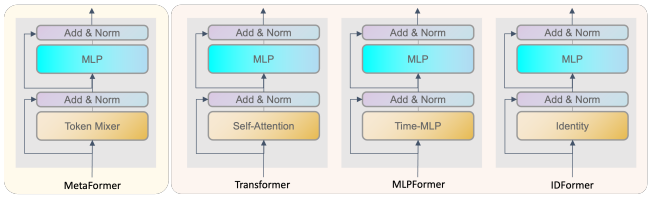

Fig. 3. Architectures of MetaFormer and its variants. Time-MLP refers to the MLP operation along the series of tokens; Identity means there is no operation.

The self-attention operation [16] in traditional transformers possesses the advantages of global dependency parsing and dynamic weight generation. However, self-attention brings a large amount of space and time complexity to the overall model. The authors of MetaFormer [17] summarize recent advances on vision transformer and argues that it is the transformer architecture itself which contributes to performance of ViT on vision tasks. They also replace the self-attention operation by a simple pooling operation and demonstrate that PoolFormer shows better efficiency than most recent variants of ViT.

Similar to PatchTST, we introduce the MetaFormer architecture into the task of 1-D time series forecast. As shown in Figure 3, we replace self-attention by Time-MLP and Identity operations as the token-mixer and propose two variants of MetaFormer: MLPFormer and IDFormer. In our experiments we observed the surprising finding that the simplest identity operation performs as well as PatchTST. Since our goal is to reduce the model complexity while minimizing loss of accuracy, we favor most of the transformer blocks to be the simplest IDFormer in our final model.

B. LoGTST

Figure 4 shows the basic architecture of our proposed LoGTST model.

RevIN. The Reversible Instance Normalization (RevIN) [20] module normalizes the input signal for each sample, which consists of one look-back window and one prediction horizon, and records the normalization factors for future denormalization of the output signal. In this way, RevIN is able to symmetrically remove and restore the statistical information of a time-series instance and improve the model's stability.

Tokenization and DeTokenization. The tokenization process is the same as Patch Embedding in PatchTST [11]. Here we call it Tokenization to follow the convention in computer vision and natural language processing. Given the input uni-variate time series as $\mathbf{x}$ and its length as L, the tokenization process can be simply accomplished by applying 1-D convolution on $\mathbf{x}$ with a predefined kernel of size P and stride S. The kernel size can also be seen as the patch length and the number of tokens is $N = [L/S]$. Different from previous works that directly flatten feature vectors $\mathbf{x}_{hidden}^{(i)}$, where $i = 1, 2, ..., N$ and apply MLP for prediction, which introduces a large amount of trainable parameters, the DeTokenization process in LoGTST firstly changes the number of tokens to satisfy the prediction horizon and then compresses the number of channel of each $\mathbf{x}_{hidden}^{(i)}$ to S by 3×1 convolution before flattening the vector to recover the output.

IDFormer and Transformer: In the transformer branch, we replace the early two transformer blocks by IDFormer blocks to reduce total model complexity and prevent early attention on naive features and keep the last transformer block to parse dependencies between hidden vectors. The patch embeddings $\mathbf{x}_p \in \mathbb{R}^{D \times N}$ should be merged with additive learnable positional encoding $\mathbf{W}_{pos}$ by $\mathbf{x}_d = \mathbf{x}_p + \mathbf{W}_{pos}$ in case of the attention mechanism treating all the feature vectors equally. For each head $h = 1, 2, ..., H$, we define three learnable matrices: $W_h^Q, W_h^K \in \mathbb{R}^{D \times d_k}$ and $W_h^V \in \mathbb{R}^{D \times D}$. Then, the calculation of multi-head self-attention can be written as:

$$O_h^T = Attention(Q_h, K_h, V_h) = softmax\{\frac{Q_h K_h^T}{\sqrt{d_k}}\}V_h \quad (1)$$

Loss: Following previous work for time series forecasting [11], [15], we also use MSE to measure the discrepancy between the prediction and the reality. Defining the number of input variables as M and the prediction horizon as T, then the total loss can be calculated by $\mathcal{L} = 1/M \sum_{i=1}^{M} \|\hat{\mathbf{x}}_{L+1:L+T}^{(i)} - \mathbf{x}_{L+1:L+T}^{(i)}\|^2$. Note that there is only one channel ($M = 1$) for the task EV charging forecasting.

C. PSGF-Fed

The difference between our proposed PSGF-Fed and previous works is illustrated in Figure 5.

Online-Fed. Instead of exchanging the model parameters with all the clients, the server of Online FL (Online-Fed) will randomly select a subset of clients in ever iteration for communication efficiency. Denote the selected subset of client indices as S_n, where $C = |S_n|$ refers to the number of selected clients and n represents current iteration. There is no need to perform local updates for unselected clients because the local model will be directly replaced by the server model when these clients are selected in future iterations. Define the local model parameters for client i in iteration n after local update as $\mathbf{w}_{n+1}^i$, then the global model can be updated by:

$$\mathbf{w}_{n+1} = \frac{1}{C} \sum_{i \in S_n} \mathbf{w}_{n+1}^i \quad (2)$$

PSO-Fed. Partial sharing based online FL (PSO-Fed) [12] reduces the granularity of communication to the parameter level for better efficiency and randomness. PSO-Fed randomly selects a subset of clients S_n and also randomly selects a subset of parameters $\mathbf{S}_n^i$ for client i for parameter exchange between server and clients. Different from Online-Fed, the unselected clients can still update their local models because when they are selected, not all of the local parameters will be replaced. The $\mathbf{S}_n^i$ can be a $D \times D$ diagonal matrix with M ones for selected diagonal elements and $D - M$ zeros. Then the selected local model can be updated by:

$$\mathbf{w}_{n+1}^i = LocalUpdate(\mathbf{S}_n^i \mathbf{w}_n + (\mathbf{I}_D - \mathbf{S}_n^i)\mathbf{w}_n^i) \quad (3)$$

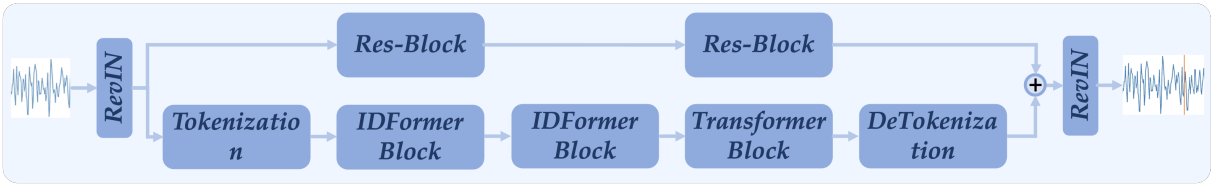

Fig. 4. Overview of the architecture of our proposed **LoGTST**. The input and output signals are processed by Reversible Instance Normalization (RevIN) module before and after the learning model, respectively. Res-Block refers to simple residual block consists of linear, ReLU, dropout, residual connection [18], and layernorm [19] operations. The signal is forwarded into two branches for processing and then merge together to construct the final output.

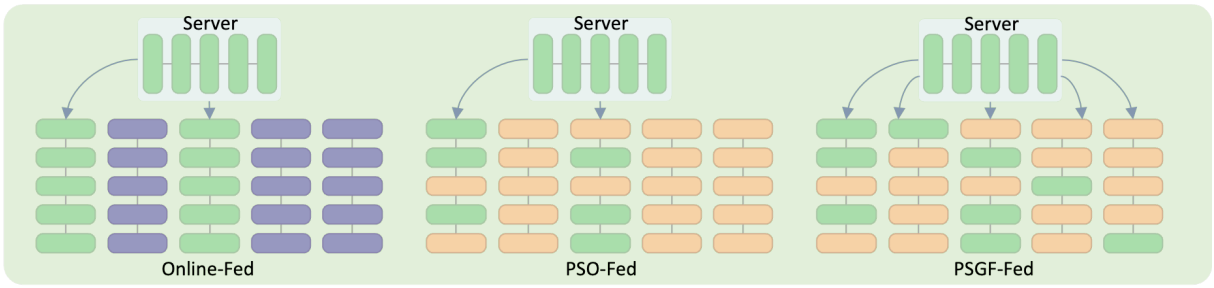

Fig. 5. Illustration of the model parameters' sharing from server to clients for Online-Fed, PSO-Fed [12], PSGF-Fed (ours). The green rectangles refer to shared parameters. The purple rectangles will remain idle during following local update while the orange and green rectangles will be updated.

The server generates the global model by aggregating the local parameters and this can be accomplished by:

$$\mathbf{w}_{n+1} = \frac{1}{C} \sum_{i \in S_n} \mathbf{S}^i_{n+1} \mathbf{w}^i_{n+1} + (\mathbf{I}_D - \mathbf{S}^i_{n+1}) \mathbf{w}_n \qquad (4)$$

The communication overhead is then significantly reduced compared with Online-Fed while maintaining convergence speed.

PSGF-Fed. PSO-Fed enables the self-learning for each unselected client, which guarantees its fast convergence speed even though only partial parameters are shared. However, training a local model only on limited local data for many local iterations (or even several global iterations) without any global information has the risk of over-fitting or weakening the generalization ability. To alleviate this issue, we propose a new online FL policy, namely PSGF-Fed, where the server will randomly selects a small subset of model parameters and share them with each client. In this way, all the clients will receive some global information from the server, which will regularize the local training process. The number of selected parameters for each client can be adjusted to reach the best trade-off between convergence speed and communication overhead. Then all the local models will be updated by (3), but the M are different for selected and unselected clients.

III. Experiments

A. Results from centralized learning using LoGTST

B. Results from FL using PSGF-Fed

As seen in the above table, when trained and tested against the NN5 time series dataset, the initial online federated learning, data intensive training performs well with an MSE of 6.02 but at the cost of transferring approximately 1.5 billion parameters during training. The previous generation PSO federated learning architecture achieves slightly worse performance of 6.10 while only transferring 0.75 billion parameters. Our PSGF federated learning model achieves 6.08 MSE while only transferring 0.38 billion parameters under the 30/30 hyperparameter implementation. This is a significant improvement in communication cost overhead with a marginal performance improvement in addition when compared to the PSO federated learning model.

Similar to the NN5 results, the PSO federate learning is able to reduce the number of parameters passed during training by approximately 50 percent while maintaining performance within 2 percent of the original online federate learning model. Our PSGF architecture is demonstrated to reduce passed parameter numbers by 77 percent while maintaining performance similar to that of the PSO model. Additionally, when given a matched communication budget as the PSO model, our PSGF architecture is able to outperfom the PSO by 0.25 MSE. This shows the flexibility and robustness of our improved PSGF model in data overhead as well as performance.

IV. Conclusions

We will include simulation results in our final submission.

V. Acknowledgements

Z. Xiong would like to acknowledge fruitful discussions with Michail Gkagkos.

Models		LoGTST		PatchTST/64		PatchTST/42		FEDformer		Autoformer		Informer		Pyraformer	
#Parameters		**5.39E+05**		1.19E+06		9.21E+05		1.63E+07		1.05E+07		1.13E+07		1.01E+07	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.151	0.199	**0.149**	**0.198**	0.152	0.199	0.238	0.314	0.249	0.329	0.354	0.405	0.896	0.556
	192	0.195	**0.240**	**0.194**	0.241	0.197	0.243	0.275	0.329	0.325	0.370	0.419	0.434	0.622	0.624
	336	0.246	**0.280**	**0.245**	0.282	0.249	0.283	0.339	0.377	0.351	0.391	0.583	0.543	0.739	0.753
	720	0.318	**0.333**	**0.314**	0.334	0.320	0.335	0.389	0.409	0.415	0.426	0.916	0.705	1.004	0.934
ETTh1	96	0.379	0.404	**0.370**	0.400	0.375	**0.399**	0.376	0.415	0.435	0.446	0.941	0.769	0.664	0.612
	192	**0.413**	**0.421**	**0.413**	0.429	0.414	**0.421**	0.423	0.446	0.456	0.457	1.007	0.786	0.790	0.681
	336	0.426	**0.431**	**0.422**	0.440	0.431	0.436	0.444	0.462	0.486	0.487	1.038	0.784	0.891	0.738
	720	**0.447**	**0.463**	**0.447**	0.468	0.449	0.466	0.469	0.492	0.515	0.517	1.144	0.857	0.963	0.782
ETTh2	96	0.275	**0.336**	**0.274**	0.337	**0.274**	**0.336**	0.332	0.374	0.332	0.368	1.549	0.952	0.645	0.597
	192	**0.338**	**0.379**	0.341	0.382	0.339	0.379	0.426	0.446	0.426	0.434	3.792	1.542	0.788	0.683
	336	**0.327**	0.381	0.329	0.384	0.331	**0.380**	0.477	0.447	0.477	0.479	4.215	1.642	0.907	0.747
	720	**0.378**	**0.421**	0.379	0.422	0.379	0.422	0.453	0.469	0.453	0.490	3.656	1.619	0.963	0.783
ETTm1	96	**0.288**	**0.342**	0.293	0.346	0.290	**0.342**	0.326	0.390	0.510	0.492	0.626	0.560	0.543	0.510
	192	**0.331**	0.370	0.333	0.370	0.332	**0.369**	0.365	0.415	0.514	0.495	0.725	0.619	0.557	0.537
	336	**0.360**	**0.390**	0.369	0.392	0.366	0.392	0.392	0.425	0.510	0.492	1.005	0.741	0.754	0.655
	720	**0.416**	0.425	**0.416**	**0.420**	0.420	0.424	0.446	0.458	0.527	0.493	1.133	0.845	0.908	0.724
ETTm2	96	**0.163**	**0.253**	0.166	0.256	0.165	0.255	0.180	0.271	0.205	0.293	0.355	0.462	0.435	0.507
	192	0.221	0.293	0.223	0.296	**0.220**	**0.292**	0.252	0.318	0.278	0.336	0.595	0.586	0.730	0.673
	336	0.278	0.330	**0.274**	**0.329**	0.278	**0.329**	0.324	0.364	0.343	0.379	1.270	0.871	1.201	0.845
	720	0.366	**0.383**	**0.362**	0.385	0.367	0.385	0.410	0.420	0.414	0.419	3.001	1.267	3.625	1.451

TABLE I
EXPERIMENTAL RESULTS FOR NN5 DATA.

Method		#Params (Comm.)	Loss (RMSE)
Online-Fed		1.53E+09	**6.02**
PSO-Fed [12]	50%	7.35E+08	6.10
	40%	4.03E+08	6.14
	30%	3.52E+08	6.15
	20%	**1.24E+08**	6.28
PSGF-Fed-20%	50%	5.75E+08	6.10
	40%	4.21E+08	6.10
	30%	3.45E+08	6.09
	20%	2.80E+08	6.14
PSGF-Fed-30%	50%	5.21E+08	6.11
	40%	5.25E+08	6.10
	30%	3.77E+08	6.08
	20%	3.18E+08	6.11

TABLE II
EXPERIMENTAL RESULTS FOR UK DATA.

Method		#Params (Comm.)	Loss (RMSE)
Online-Fed		9.07E+06	10.46
PSO-Fed [12]	50%	4.84E+06	10.68
	40%	3.77E+06	10.89
	30%	2.75E+06	10.85
	20%	2.11E+06	11.14
PSGF-Fed-20%	50%	4.82E+06	**10.43**
	40%	4.28E+06	10.54
	30%	2.96E+06	10.67
	20%	2.11E+06	10.64
PSGF-Fed-30%	50%	4.46E+06	10.64
	40%	4.18E+06	10.63
	30%	3.28E+06	10.65
	20%	**2.08E+06**	10.60

REFERENCES

[1] "The growth of electric vehicles," *https://www.cnbc.com/2018/05/30/electric-vehicles-will-grow-from-3-million-to-125-million-by-2030-iea.html*,

[2] M. Muratori, "Role of electric vehicles in the u.s. power sector transition," in *Keynote Session, 4-th E-Mobility Power System Integration Symposium*, November 3rd, 2020.

[3] M. Majidpour, C. Qiu, P. Chu, R. Gadh, and H. R. Pota, "Fast prediction for sparse time series: Demand forecast of ev charging stations for cell phone applications," *IEEE Trans. on Industrial Informatics*, vol. 11, no. 1, pp. 242–250, 2014.

[4] S. Ryu, J. Noh, and H. Kim, "Deep neural network based demand side short term load forecasting," *Energies*, vol. 10, no. 1, p. 3, 2017.

[5] N. G. Paterakis, E. Mocanu, M. Gibescu, B. Stappers, and W. van Alst, "Deep learning versus traditional machine learning methods for aggregated energy demand prediction," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pp. 1–6, 2017.

[6] D. Perry, N. Wang, and S. Ho, "Energy demand prediction with optimized clustering-based federated learning," in *2021 IEEE Globecom Conference*, pp. 1-6, 2021.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[8] S. Abdulrahman, H. Tout, O. Hanine, *et al.*, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet of Things Journal*, vol. 8, pp. 5476-5497, Oct. 2020.

[9] "Electric vehicle charging sessions dundee," *https://data.dundeecity.gov.uk/dataset/ev-charging-data*,

[10] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, and S. Srikanteswara, "Energy demand prediction with federated learning for electric vehicle networks," in *2019 IEEE Globecom Conference*, pp. 1-6, 2019.

[11] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *International Conference on Learning Representations*, 2023.

[12] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning strategies for kernel regression," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4531–4544, 2022.

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy.

[14] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" In *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, 2023, pp. 11 121–11 128.

[15] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proc. 39th International Conference on Machine Learning (ICML 2022)*, Baltimore, Maryland, 2022.

[16] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[17] W. Yu, M. Luo, P. Zhou, *et al.*, "Metaformer is actually what you need for vision," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 819–10 829.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[19] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[20] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *International Conference on Learning Representations*, 2021.