Multi-task learning for electrical load forecasting using an enhanced autoencoder-lstm model

Ali Forootani, Reza Ahamid

[Ali.forootani1@ucalgary.ca](mailto:Ali.forootani1@ucalgary.ca), [Reza.ahmadi3@ucalgary.ca](mailto:Reza.ahmadi3@ucalgary.ca)   
Schulich school of engineering

# Abstract

Electrical load forecasting has always been a matter of concern for utility companies, since an accurate load forecasting framework would notably decrease the operational costs and risks in the power grid. Deep learning-based methods are the state-of-the-art approach for this task, showing promising performance. However, further accuracy improvements remain essential. In this project, we aim to study a novel approach to enhance the performance of a long short-term memory (LSTM) model. To be more precise, we propose to develop a multi-task learning model that forecasts the future load and reconstructs the historical load at the same time. The architecture of the model is inspired by Autoencoder. Furthermore, instead of zero or random initialization of hidden state and cell state of LSTM model, we transfer these values from the encoder part of the framework. The obtained results tested on real-world dataset indicate decreasing the error metrics by ten percent.

# 1. Introduction

Accurate electrical load forecasting is playing a significant role in power systems. In other words, a precise load forecasting model addresses the difference between supply and demand which leads to tangible reduction in operational costs [1]. Additionally, having an accurate load forecast enables the utility companies to dynamically participate in electricity market. The knowledge about the future load for one up to 24 hours ahead, provides a competitive advantage in offering better bids [2]. Accordingly, electrical load forecasting has received a huge amount of attention consistently.

Recently, with advances in artificial intelligence, especially in deep learning, there has been a significant increase in the number of publications dedicated to load forecasting using deep models. This was not feasible before due to a lack of adequate data. However, with the development of advanced meter infrastructures (AMIs) providing high-quality data with hourly or minutely resolution, researchers have been able to leverage deep models [3]. Prior to this, the methodologies used in both academia and industry were limited to statistical methods or, at best, shallow machine learning models. Undoubtedly, even shallow machine learning struggles to capture latent behaviors in nonlinear data like electrical load, let alone statistical methods. Furthermore, statistical approaches fail to account for unprecedented uncertainties in the data, leading to poor forecasting metrics [4]. Consequently, deep learning models have become the latest trend in electrical load forecasting, demonstrating promising results.

# 2. related works

As there is a significant amount of research that leverages statistical approaches and artificial intelligence for load forecasting. In the past, statistical methods were predominantly used. The autoregressive integrated moving average (ARIMA) model has been widely applied in the literature [5]-[8]. For example, a study employed an ARIMA model alongside a similar-day approach to forecast intraday load [5]. In this approach, the target day is matched with past days exhibiting similar meteorological conditions, and the load is predicted by averaging the demand from these matched days.

Furthermore, various shallow machine learning methods have been employed for load forecasting. For instance, in [9], support vector regression (SVR) combined with an adaptive genetic algorithm is used to predict power system load, incorporating meteorological data into the forecasting process. To enhance accuracy, a hybrid method using SVR is proposed, integrating an improved adaptive genetic algorithm to optimize the inputs of the SVR. Additionally, the k-nearest neighbor (KNN) algorithm has been successfully applied to load forecasting [10]-[11]. Specifically, the authors in [10] introduce a multivariate k-NN regression method for forecasting electricity demand in the UK market, utilizing binary dummy variables to distinguish between working and non-working days. The performance of this method is evaluated through an extensive empirical analysis of UK electricity load data.

One of the most influential studies in electrical load forecasting, [12], introduces an LSTM recurrent neural network framework tailored to address the complexities of load prediction. This deep learning approach demonstrates superior performance, consistently outperforming other algorithms in forecasting short-term electricity demand for individual residential users. The paper highlights the effectiveness of LSTM models in capturing intricate temporal dependencies, making it a cornerstone in the field.

To recapitulate, recurrent neural networks are currently the state-of-the-art approach for the discussed task. However, there is still room for improvement. In this project, we aim to study the capability of multi-task learning using an LSTM model for load forecasting. The primary task, which is the main focus, is forecasting the load itself, while the secondary task involves reconstructing historical loads (lagged loads). We believe that such a design may improve accuracy compared to a single-task LSTM model. The general architecture of our proposed framework follows an Autoencoder structure. The encoder serves as the shared component, while there are two decoders: one for reconstruction and the other for forecasting. Additionally, inspired by [13], we explore the impact of initializing the hidden state and cell state of the forecasting LSTM from the encoder LSTM, rather than using random or zero initialization. The report is organized as follows: Section 3 details the proposed methodology, including the model’s architecture and design considerations. Section 4 presents the results and provides a discussion. Finally, section 5 concludes the paper by summarizing the achievements of this project.

# 3. material and methods

***3.1 Data preprocessing***

For this project, we utilized the hourly electrical load demand dataset from PJM, a publicly available resource. PJM is a regional transmission organization (RTO) that coordinates the movement of wholesale electricity across parts of Delaware, Illinois, Indiana, Kentucky, Maryland, Michigan, New Jersey, North Carolina, Ohio, Pennsylvania, Tennessee, Virginia, West Virginia, and the District of Columbia.

The initial data preprocessing steps included parsing and extracting time-based features. The time column was converted to a datetime format, enabling the extraction of features such as the hour of the day, day of the month, and month. These features provide essential temporal granularity for forecasting task. To capture weekly patterns in load, each day of the week was mapped to a numerical value using a predefined dictionary (day\_mapping). This encoded feature allows the model to distinguish between weekdays and weekends, which significantly influence electricity demand patterns.

The dataset also includes temperature data from various regions (e.g., Baltimore, Washington, Philadelphia), which serve as critical predictors of electricity demand due to their impact on heating and cooling needs. Several temperature features were engineered, such as maximum and minimum temperatures. Specifically, we calculated the maximum and minimum temperatures across all regions for each time step to capture the temperature extremes within the network. Additionally, we introduced lagged temperature features by shifting each region's temperature data by one time step (i.e., one hour ahead) to incorporate information on temperature trends. For each shifted regional temperature, we also calculated the maximum temperature for the next hour (next\_max\_temperature), enabling the model to leverage temperature forecasts alongside current conditions.

After generating all features, rows with any missing values were removed to ensure dataset consistency and to avoid introducing biases through imputation.

To normalize the feature range and improve model performance, we applied MinMax scaling, transforming all feature values to a range between 0 and 1. This step ensures that no single feature disproportionately influences the model due to scale differences. Also, to enable the time series forecasting model to learn from historical data, we transformed the dataset into a supervised learning format using a custom function (series\_to\_supervised). This function creates lagged versions of each variable for the specified historical time steps (historical\_time\_step), and future forecasting horizon (horizon). For each time step, it constructs a feature set containing both the previous 24-hour data points and the 24-hour forecast horizon. Additionally, to evaluate model performance on recent data, the entire year 2023 was held out for testing.

***3.2 Framework***

***3.2.1 Model Design and Architecture***

The proposed model employs a multi-task learning framework to achieve two objectives simultaneously: forecasting future load values (prediction) and reconstructing historical load patterns (reconstruction). Figure 1 illustrates the general structure of a multi-task learning model.

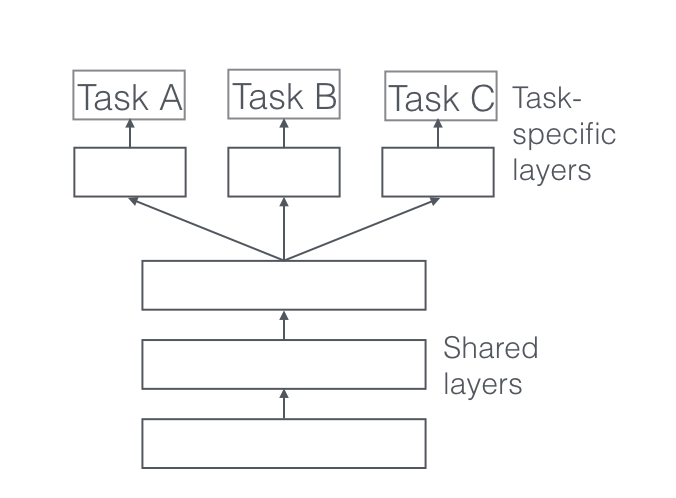


Fig. 1. Multi-task learning high level demonstration.

As shown in Figure 1, a multi-task learning model consists of shared layers that enhance the learning process across tasks, as well as task-specific layers designed to fulfill different objectives. This structure facilitates knowledge sharing, improving model performance compared to single-task models [14].

Given that one of the tasks is to reconstruct historical data, the proposed framework for this project adopts an encoder-decoder LSTM architecture. Figure 1 depicts the framework designed for this project.

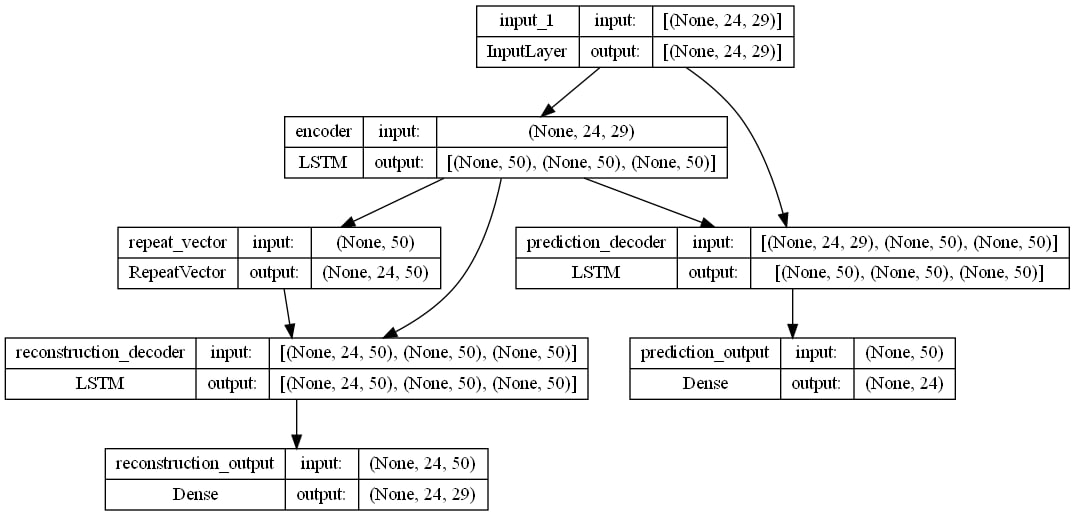


Fig. 1. proposed deep model.

This approach is structured with an encoder-decoder framework to handle both reconstruction and forecasting tasks. The encoder, implemented as an LSTM, processes sequences over a specified historical time window, capturing essential temporal dependencies in the data. It generates two outputs: a hidden state (state\_h) and a cell state (state\_c), which encapsulate the learned information from the input sequence. These states are then transferred to the decoders as initial conditions for their respective tasks. Figure 2 illustrates the internal structure of an LSTM model.

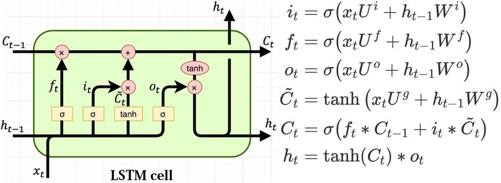


Fig. 2. LSTM associated with its parameters.

The reconstruction decoder reuses the encoder’s final hidden and cell states to initialize its own LSTM layers. This setup enhances the decoder’s ability to accurately reproduce the input sequence, reinforcing the model’s understanding of temporal dependencies. By reconstructing historical load patterns, the reconstruction task contributes to the overall effectiveness of the model.

The forecasting decoder is responsible for predicting future load values over a specified horizon of 24 time-steps (hours). Like the reconstruction decoder, it initializes its LSTM layers using the encoder’s final hidden and cell states. This transfer of state information provides the forecasting decoder with a rich representation of past context, allowing it to generate accurate predictions. By leveraging the encoder states, the forecasting decoder maintains a continuity of information, enabling it to model both short- and long-term dependencies critical for effective load forecasting.

***3.2.2. Custom Loss Function***

To ensure both tasks achieve high accuracy, we define a custom loss function (custom\_loss\_forecast), which scales the mean squared error by a factor of 100. This scaling emphasizes the error impact, making even small deviations in predictions significant. By using this custom loss, the model is encouraged to minimize errors more aggressively, aiming for precision in both reconstruction and forecasting tasks.

***3.2.3 Loss Weights***

To balance the dual objectives of the model, specific weights were assigned to each output. A weight of 1.0 was allocated to the reconstruction task to ensure the model effectively captured and reconstructed historical patterns. Meanwhile, a higher weight of 2.0 was assigned to the prediction task, reflecting the greater emphasis on accurate load forecasting as the primary goal. This weighting scheme enabled the model to prioritize forecasting while maintaining sufficient focus on reconstruction to enhance overall performance.

***3.2.4 Model Compilation and Optimizer***

The model is compiled with the Adam optimizer, chosen for its adaptive learning rate properties, which facilitate efficient convergence. The custom loss function is applied to both outputs, with respective weights assigned to each loss component.

***3.2.5 Training Process with Callback Functions***

To optimize model performance and prevent overfitting, several callback functions were incorporated. Early stopping was implemented to monitor the (prediction\_output\_loss) and halt training if no improvement was observed over 20 consecutive epochs. This approach is particularly critical for time series models as it helps to avoid overfitting while reducing unnecessary training time. Additionally, a model checkpoint callback was employed to save the model weights whenever an improvement in (prediction\_output\_loss) was detected, ensuring that the best-performing model was retained during training.

To further enhance convergence, a "reduce learning rate on plateau" callback was utilized. If no improvement in (prediction\_output\_loss) occurred for 10 epochs, this callback reduced the learning rate by a factor of 0.5. This adaptive adjustment allowed the model to make finer updates in the later stages of training, facilitating convergence to a more accurate solution. Together, these callbacks provided an effective mechanism for maintaining model stability and achieving optimal performance.

***3.2.6 Model Training***

During training, validation is performed using a test set to monitor generalization performance. Importantly, the training data is not shuffled, preserving the temporal sequence within each batch, which is crucial for time series forecasting.

# 4. results and discussion

***4.1 Settings and case definitions***

To conduct a comprehensive analysis, we evaluated the performance of the proposed model using commonly cited load forecasting metrics in the literature: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and the 𝑅2 score. These metrics were calculated both for the overall forecast and for each step in the prediction horizon.

Due to the size of the dataset and the computational cost of LSTM models, we were unable to use libraries such as Keras Tuner for hyperparameter tuning. Instead, we first relied on existing literature to narrow down potential hyperparameter values, such as the number of neurons (units) in the LSTM, the number of epochs, and the optimizer. We then employed a trial-and-error approach to test possible values, using a validation set comprising 20% of the training set. Through this process, we determined that the optimal number of neurons for the LSTM was 50. It is worth noting that a Dense layer follows the LSTM model, but tuning its number of neurons was unnecessary, as it must equal 24 (the 24-hour forecasting horizon). The loss function used in the model was Mean Squared Error (MSE). However, we observed that multiplying the loss by 100 improved convergences in terms of smoothness and speed.

To ensure a fair comparison, we compared the results of the proposed method with two alternative models: (1) an Autoencoder LSTM-based model without the reconstruction component but with the same design as the proposed method, and (2) a regular LSTM model with the same hyperparameters as the proposed model.

***4.2 Results and discussion***

Primarily, the overall error metrics for all models over the 24-hour horizon are listed in Table 1.

Table 1. Overall error metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | MAPE | MAE | RMSE | R2 |
| Proposed model | 5.785 | 5127.373 | 6803.425 | 0.776 |
| Autoencoder LSTM | 6.396 | 5639.766 | 7648.169 | 0.717 |
| Regular LSTM | 8.697 | 7707.660 | 9806.485 | 0.534 |

The proposed model demonstrates superior performance compared to the Autoencoder LSTM and Regular LSTM, achieving a MAPE of 5.785, an MAE of 5127.373, an RMSE of 6803.425, and an 𝑅2 score of 0.776. These metrics indicate that the proposed model is both accurate and reliable in capturing patterns for load forecasting across the 24-step horizon. The relatively low MAPE suggests that the model consistently provides predictions with minimal percentage-based errors, while the high 𝑅2 score indicates that the proposed approach explains a significant proportion of the variance in the data.

Figures 3 and 4 illustrate the MAPE trends across the forecasting horizon for the proposed model and the Autoencoder LSTM, respectively. These figures provide a step-by-step evaluation of prediction accuracy over 24 time steps, highlighting how errors evolve as the forecast horizon extends. The comparison focuses on these two models because, as shown in Table 1, the Autoencoder LSTM is the strongest counterpart model, achieving better overall metrics compared to the Regular LSTM. Therefore, the Regular LSTM is excluded from this detailed analysis due to its significantly higher error rates and lower 𝑅2 score.

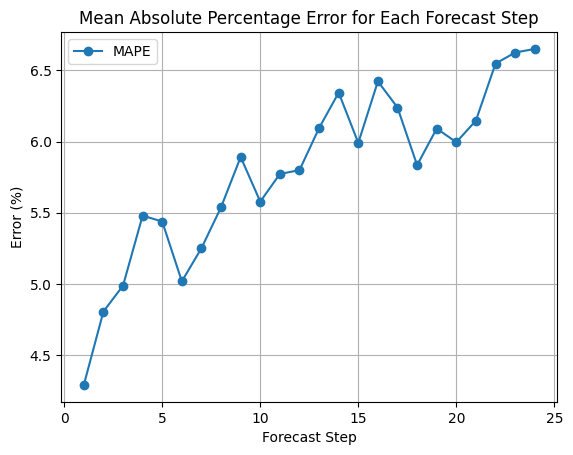


Fig. 3. MAPE of proposed model over 24 steps.

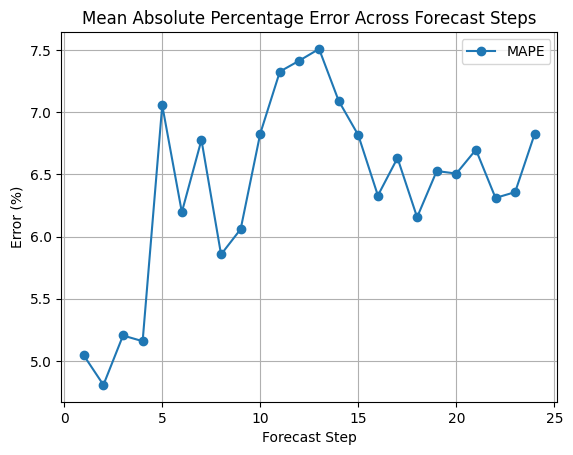


Fig. 4. MAPE of Autoencoder LSTM model over 24 steps.

Regarding the above figures, the proposed model exhibits a smooth and gradual increase in MAPE as the forecast horizon progresses. Starting at approximately 4.5% in the initial steps, the MAPE rises steadily to around 6.5% by the 24th step. This consistency indicates that the proposed model effectively handles the accumulation of uncertainty over extended horizons, maintaining robust predictions throughout. The stability of the proposed model’s error trend can be attributed to its targeted architecture, which simultaneously focuses on forecasting and reconstruction. By transferring the encoder’s hidden and cell states directly to the forecasting decoder, the model efficiently captures temporal dependencies while also learning reconstruction tasks.

In contrast, the Autoencoder LSTM shows a more fluctuating error trend. While the MAPE starts at approximately 5.0%, similar to the proposed model, it experiences notable peaks and troughs, with errors exceeding 7.5% at certain steps. These fluctuations suggest that the Autoencoder LSTM struggles to maintain consistent accuracy, particularly at intermediate forecast steps. The fluctuating performance of the Autoencoder LSTM may stem from its limited focus on forecasting without leveraging shared learning from additional tasks, as in the proposed model.

The side-by-side comparison of MAPE trends highlights the strengths of the proposed model over the Autoencoder LSTM. Across the entire forecast horizon, the proposed model consistently achieves lower MAPE values, demonstrating superior prediction accuracy. This is evident in the overall metrics presented in Table 1, where the proposed model achieves a MAPE of 5.785 compared to 6.396 for the Autoencoder LSTM.

In addition to its accuracy, the proposed model exhibits greater stability in MAPE trends. Its MAPE increases gradually and predictably across the forecast horizon, while the Autoencoder LSTM shows significant variability, with notable peaks and troughs in its error rates. This stability indicates that the proposed model is more robust to changes in data patterns and better equipped to handle long-term dependencies.

A key advantage of the proposed model is its multi-task learning design, which integrates forecasting and reconstruction tasks. By leveraging shared learning between the two decoders, the model enhances its ability to capture temporal dependencies and improve performance across both tasks. In contrast, the single-task design of the Autoencoder LSTM limits its capacity to achieve comparable levels of accuracy and robustness.

The proposed model consistently outperforms the other models across all metrics, achieving lower MAPE (e.g., 4.29% at hour 1 vs. 5.04% for the Autoencoder LSTM and 8.22% for the Regular LSTM) and MAE (e.g., 3746.65 at hour 1 vs. 4373.52 for the Autoencoder LSTM). Its RMSE values are also the lowest, demonstrating superior robustness, while its R2-scores remain higher across all time points (e.g., 0.89 at hour 1). While the Autoencoder LSTM performs competitively, it falls short due to its single-task design, which sacrifices the benefits of shared learning seen in the proposed model. The Regular LSTM lags significantly, with higher errors and lower variance explanation. These results highlight the advantages of the proposed model’s multi-task design, which integrates forecasting and reconstruction to prioritize accurate predictions.

Table 1. Error metrics for different horizons.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Proposed model | Autoencoder LSTM | Regular LSTM |
| MAPE hour 1 | 4.29 | 5.04 | 8.22 |
| MAPE hour 6 | 5.44 | 6.20 | 8.68 |
| MAPE hour 18 | 5.83 | 6.16 | 7.99 |
| MAPE hour 24 | 6.65 | 6.83 | 7.87 |
| MAE hour 1 | 3746.65 | 4373.52 | 7093.26 |
| MAE hour 6 | 4414.70 | 5436.50 | 7553.97 |
| MAE hour 18 | 5215.96 | 5460.26 | 7123.61 |
| MAE hour 24 | 5930.97 | 6078.82 | 6882.03 |
| RMSE hour 1 | 4844.72 | 6258.89 | 9039.30 |
| RMSE hour 6 | 5789.10 | 7120.19 | 9514.69 |
| RMSE hour 18 | 6844.72 | 7008.09 | 8934.96 |
| RMSE hour 24 | 7624.41 | 7804.83 | 8551.33 |
| R2 hour 1 | 0.89 | 0.81 | 0.60 |
| R2 hour 6 | 0.84 | 0.75 | 0.56 |
| R2 hour 18 | 0.77 | 0.70 | 0.61 |
| R2 hour 24 | 0.72 | 0.71 | 0.65 |

# 5. conclusion

In this project, we developed a novel approach for load forecasting using a multi-task learning framework inspired by Autoencoders. The architecture forecasts future loads while reconstructing historical patterns, utilizing the transfer of hidden and cell states from the encoder to the decoders. This state transfer ensures that the forecasting decoder is enriched with temporal dependencies, enhancing predictive accuracy. Although the Autoencoder framework balances reconstruction and forecasting, it introduces trade-offs that may compromise forecasting performance. The results of this project highlight the effectiveness of simplifying the design by focusing exclusively on forecasting while retaining the state transfer mechanism. The proposed model outperforms the Autoencoder LSTM, achieving a 9.6% reduction in MAPE (5.785 vs. 6.396) and a 10.4% improvement in 𝑅2-score (0.776 vs. 0.717). These findings demonstrate that this project’s targeted architectural modifications significantly improve forecasting accuracy. This project showcases the potential of efficient encoder-decoder architectures and provides a robust solution for load forecasting, setting the foundation for further advancements in predictive modeling.

# 6. References

[1] M. Gilanifar, H. Wang, L. M. K. Sriram, E. E. Ozguven and R. Arghandeh, "Multitask Bayesian Spatiotemporal Gaussian Processes for Short-Term Load Forecasting," in *IEEE Transactions on Industrial Electronics*, vol. 67, no. 6, pp. 5132-5143, June 2020.

[2] S. Schreck, I. Prieur de La Comble, S. Thiem and S. Niessen, "A Methodological Framework to support Load Forecast Error Assessment in Local Energy Markets," in *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3212-3220, July 2020.

[3] U.S. Energy Information Administration, “Frequently Asked Questions”, Accessed: July. 2021, [online]. Available: https://www.eia.gov/tools/faqs/faq.php? id=108&t=3.

[4] M. Afrasiabi, M. Mohammadi, M. Rastegar, L. Stankovic, S. Afrasiabi and M. Khazaei, "Deep-Based Conditional Probability Density Function Forecasting of Residential Loads," in *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3646-3657, July 2020.

[5] X. Cao, S. Dong, Z. Wu, and Y. Jing, “A data-driven hybrid optimization model for short-term residential load forecasting,” in Proc. IEEE Int. Conf. Comput. Inf. Technol. Ubiquitous Comput. Commun. Dependable Auton. Secure Comput. Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM), Liverpool, U.K., 2015, pp. 283–287.

[6] K. S. L.Madhavi et al., “Advanced electricity load forecasting combining electricity and transportation network,” in Proc. North Amer. Power Symp., Sep. 2017, pp. 1–6.

[7] W. Jian-jun, N. Dong-Xiao, and L. Li, “AnARMAcooperate with artificial neural network approach in short-term load forecasting,” in Proc. 5th Int. Conf. Natural Comput., Aug 2009, vol. 1, pp. 60–64.

[8] K. G. Boroojeni, M. H. Amini, S. Bahrami, S. Iyengar, A. I. Sarwat, and O. Karabasoglu, “A novel multi-time-scale modeling for electric power demand forecasting: From short-term to medium-term horizon,” Elect. Power Syst. Res., vol. 142, pp. 58–73, Jan. 2017.

[9] G. Zhang and J. Guo, "A Novel Method for Hourly Electricity Demand Forecasting," in IEEE Transactions on Power Systems, vol. 35, no. 2, pp. 1351-1363, March 2020.

[10] R. Zhang, Y. Xu, Z. Y. Dong, W. Kong, and K. P. Wong, “A composite k-nearest neighbor model for day-ahead load forecasting with limited temperature forecasts,” presented at the IEEE Gen. Meeting, Boston, MA, USA, 2016, pp. 1–5.

[11] F. H. Al-Qahtani and S. F. Crone, “Multivariate k-nearest neighbour regression for time series data—A novel algorithm for forecasting UK electricity demand,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Dallas, TX, USA, 2013, pp. 1–8.

[12] 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," in IEEE Transactions on Smart Grid, vol. 10, no. 1, pp. 841-851, Jan. 2019.

[13] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[14] Y. Guo *et al*., "BiLSTM Multitask Learning-Based Combined Load Forecasting Considering the Loads Coupling Relationship for Multienergy System," in *IEEE Transactions on Smart Grid*, vol. 13, no. 5, pp. 3481-3492, Sept. 2022.