Electric Load Forecasting Enhanced by Weather Factors: A Comparative Study of GRU, LSTM, and XGBoost Model in Jamali Power System Network

Dimas Bangun Fiddiansyah
Division of Digital Management
PLN Head Office
Jakarta, Indonesia
dimas.bangun@pln.co.id

Istifa Shania Putri
Graduate Student in Informatics
STEI ITB
Bandung, Indonesia
23523007@std.stei.itb.ac.id

Didik Notosudjono
Professor, Cancellor
Pakuan University
Bogor, Indonesia
didiknotosudjono@unpak.ac.id

Agus Trisusanto
Division of Digital Management
PLN Head Office
Jakarta, Indonesia
agus.ts@pln.co.id

***Abstract*— Electric load forecasting plays a critical role in efficient energy management, helping utility companies balance supply and demand, reduce operational costs, and maintain grid stability. This paper provides a comparative analysis of three predictive models—GRU, LSTM, and XGBoost—to identify the most accurate model for electric load forecasting using time-series data, including historical load data and weather factors such as temperature, humidity, and wind speed. The models were evaluated on metrics including R^2, MAPE, RMSE, and MAE. Results show that GRU outperforms both LSTM and XGBoost, achieving the highest R^2 value of 0.990 and a MAPE of 0.79 %, indicating its superior ability to capture complex temporal dependencies in the load data. The LSTM model closely follows with an R^2 of 0.989 and MAPE of 0.85 %, while XGBoost lags with an R^2 of 0.589 and a MAPE of 4.76 %, demonstrating its limitations in time-series forecasting. Hyperparameter tuning played a crucial role in optimizing model performance, with GRU's simpler architecture providing a slight edge over LSTM. These findings establish the GRU model as the most reliable and accurate model for electric load forecasting for this case, particularly when accounting for both historical load patterns and weather-related variables, highlighting the importance of recurrent neural networks for this application.**

Keywords—Electric load forecasting, GRU, LSTM, XGBoost, time-series data, energy management, machine learning.

I. INTRODUCTION

Electric load forecasting is a critical aspect of energy management, essential for ensuring efficient power generation, distribution, and maintaining grid stability. Accurate load forecasts enable utility companies to balance supply and demand, minimize operational costs, and reduce the risk of blackouts. Traditionally, forecasting methods such as Autoregressive Integrated Moving Average (ARIMA) and regression analysis have been widely employed. However, these methods often face limitations in capturing the complex, nonlinear patterns characteristic of modern energy systems [1].

In recent years, machine learning models have emerged as powerful alternatives to traditional methods, offering good accuracy and adaptability in handling the intricacies of time-series data. Among these, deep learning techniques such as Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) networks, along with ensemble methods like eXtreme Gradient Boosting (XGBoost), have shown significant promise in the field of load forecasting [2]. Despite these advancements, a critical gap remains in the literature: there has been insufficient evidence directly comparing the performance of GRU, LSTM, and XGBoost specifically for electric load forecasting. Previous studies have often evaluated these models independently or under varying conditions, leaving a gap in understanding which model offers the most reliable predictions under identical conditions, particularly when evaluated using metrics such as Mean Absolute Percentage Error (MAPE).

This paper aims to address this gap by conducting a detailed comparative analysis of GRU, LSTM, and XGBoost models using the same test set size and conditions. The primary objective is to determine which model achieves the lowest MAPE, thereby providing empirical evidence to guide the selection of the most suitable model for electric load forecasting. Additionally, this study explores the practical implications of these findings, with a focus on improving the accuracy and reliability of forecasting in real-world energy management scenarios.

The significance of this research lies in its dual contributions. Practically, the findings will provide utility companies and grid operators with evidence-based guidance for model selection, potentially leading to enhanced forecasting accuracy and operational efficiency. Theoretically, this study contributes to the existing body of knowledge by empirically establishing the superiority of GRU over LSTM and XGBoost in the context of electric load forecasting, thereby filling a crucial gap in the literature.

This paper is organized to provide a thorough investigation into electric load forecasting through the application of three machine learning models: GRU, LSTM, and XGBoost. In the Introduction, the importance of accurate load forecasting for effective energy management will be discussed, alongside an identification of existing gaps in the literature regarding a comparative evaluation of these models. The Basic Theory section will then elucidate the theoretical underpinnings of load forecasting, particularly focusing on the methodologies and architectures of GRU, LSTM, and XGBoost models. The Methodology will describe the experimental framework, including data preprocessing techniques, model development, and hyperparameter optimization. In the Results and Analysis section, the performance metrics will be presented and critically analyzed to assess the forecasting efficacy of each

model. Finally, the Conclusions will summarize the key findings, demonstrating the superiority of the GRU model and outlining the implications of this research for future applications in energy management.

II. Basic Theory

A. Demand Forecast

Power system planning involves a series of processes aimed at making investment decisions within the electric power system. This planning starts with forecasting electricity demand to ensure adequate supply of electrical energy in the future. While there is no universally accepted standard for the time horizon in load forecasting, it is typically categorized into short-term, medium-term, and long-term forecasts [3]. The different types of forecasts, as identified in previous research, are explained below:

- Long-term load forecasting (LTLF): This type of forecasting spans a time range of 1 to 20 years and is used to inform strategic decisions such as planning new power plants and other major electrical system developments.
- Medium-term load forecasting (MTLF): Forecasting in this category covers a time horizon from one week to one year, typically used for operational planning.
- Short-term load forecasting (STLF): This forecast ranges from one hour to one week and is crucial for daily operations and detailed scheduling of plant activities.

B. Gate Recurrent Unit (GRU)

GRU is a variation of the LSTM that also utilizes a gated recurrent neural network architecture. Unlike LSTM, which has three gates (forget gate, input gate, and output gate), GRU only has two gates: the update gate and reset gate. Additionally, GRU involves fewer training parameters compared to LSTM, allowing it to converge faster during the training process [4].

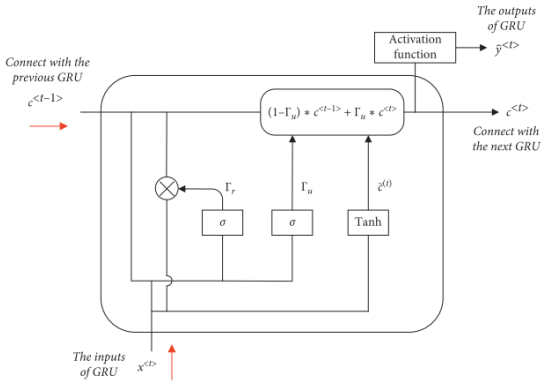

Fig 1. GRU architecture [4]

The GRU structure is depicted in Figure 1 , where σ (sigmoid) and tanh are the activation funtions. The input of the current unit, $c^{(t-1)}$, is also the output of the previous unit, while $c^{(t)}$ represents the output of the current unit, which is connected to input of the next unit. Input data is represented by $x^{(t)}$, and the output $\hat{y}^{(t)}$ is generated by activation functions. The reset gate and update gate are denoted as G_r and G_u, respectively. GRU operates with two gates: the update gate retain past information, while the reset gate controls how much of the previous data is ignored. The value of G_u, ranging from 0 to 1, determines how much past information is kept, and G_r , with values from -1 to 1, dictates how much is forgotten. As illustrated in Figure 2, the GRU formulas can be represented as Equation 1-4 follows [5] :

$$G_u = s\,(\omega_u[c^{<t-1>}, x^{<t>}] + b_u\,, \quad [1]$$
$$G_r = s\,(\omega_r[c^{<t-1>}, x^{<t>}] + b_r\,, \quad [2]$$
$$\tilde{c}^{<t>} = \tanh(\omega_c[G_r * c^{<t-1>}, x^{<t>}] + b_c)\,, \quad [3]$$
$$c^{<t>} = (1 - G_u) * c^{<t-1>} + G_u * \tilde{c}^{<t>}, \quad [4]$$

In this context, $\omega_c, \omega_r, \omega_u$ represent the weight matrices for the update gate, reset gate, and candidate activation $\tilde{c}^{<t>}$ respectively, while b_c, b_r, b_u correspond to the bias vectors.

C. Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) network is a type of Recurrent Neural Network (RNN) designed to handle long-term dependencies in data. Unlike traditional RNNs, which struggle to remember information over time, LSTM solves this by addressing the vanishing gradient problem, allowing it to retain important information for longer periods without altering the basic training process.

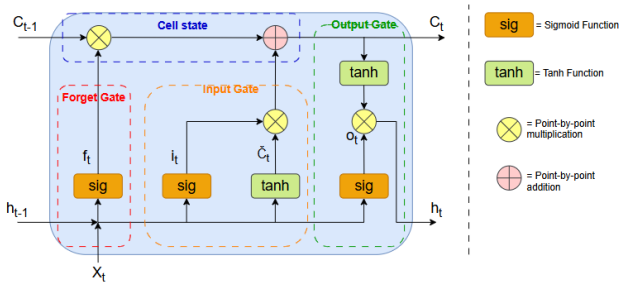

Fig 2. LSTM Architecture [6]

Based figure 2, The LSTM network includes a hidden state (new), h_{t-1} as the previous hidden state, C_t as the new cell state, C_{t-1}as the previous cell state, and X_t as the input data. This model is useful in handling continuous values and noise. LSTMs come equipped with numerous adjustable parameters, such as learning rates, input biases, and output biases. These parameters provide flexibility during training, allowing the network to adapt and enhance its performance. As a result, LSTMs present a more robust and adaptable framework for sequential data modeling, surpassing the limitations of basic deep learning such as hidden markov model.

The LSTM model consists of three key gates—input, forget, and output gates with unit storage. For each time step t, the LSTM process can be described as Equation 5 follows [6]:

$$\begin{cases} i_t = sig\,(Wi \cdot [h_{t-1}, x_t] + b_i) \\ f_t = sig\,(Wf \cdot [h_{t-1}, x_t] + b_f) \\ o_t = sig\,(Wo \cdot [h_{t-1}, x_t] + b_o) \\ \overline{C_t} = tanh\,(Wc \cdot [h_{t-1}, x_t] + b_c) \end{cases} \quad [5]$$

Where i_t represents the input gate, f_t is the forget gate, o_t is the output gate, C_t and C_{t-1} denote the cell states, and h_t is the hidden state. The weight matrices are W_i, W_f, W_o, W_c, while the bias terms are b_i, b_f, b_o, b_c. The sigmoid activation function, referred to as σ (sig), and the hyperbolic tangent activation function (tanh) are also employed.

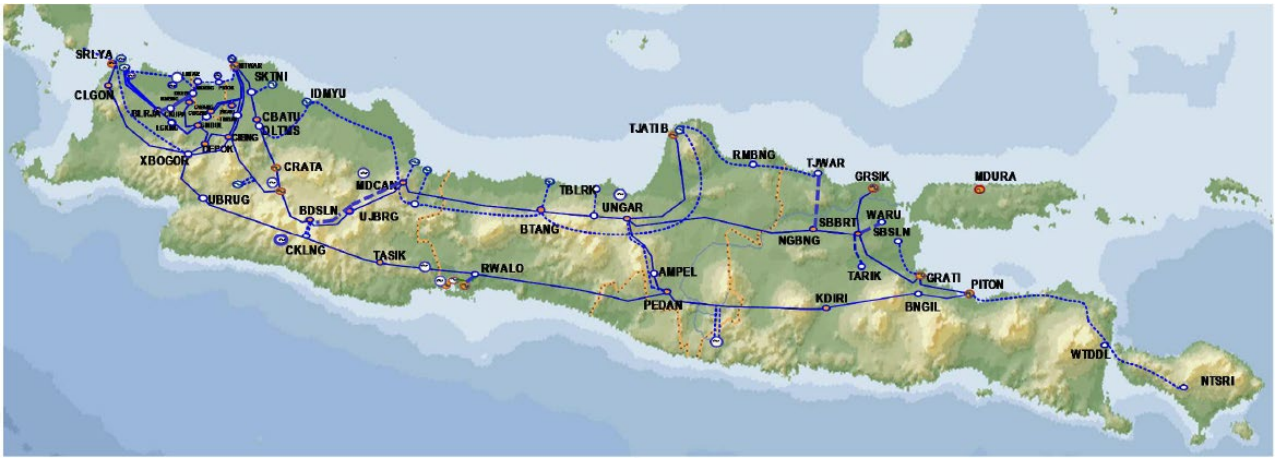

Fig 3. Jawa- Madura- Bali 500 kV single line diagram [10]

D. *Extreme Gradient Boosting (XGBoost)*

XGBoost belongs to the family of gradient-boosting algorithms and incorporates several improvements that build upon the core principles of gradient boosting. XGBoost is an ensemble learning algorithm that combines the predictions of multiple decision trees. It uses boosting technique to sequentially build decision trees that correct errors made by previous trees. XGBoost employs gradient boosting to minimize errors by adjusting tree parameters in the direction that reduces loss the most. The algorithm includes regularization techniques to prevent overfitting and handle large datasets efficiently [7].

XGBoost uses a computational procedure that starts by assigning uniform weights to all samples in the training set. At each iteration, the classification error rate is calculated to identify areas for improvement. The model updates sample weights based on the error rate at every iteration to enhance prediction accuracy.

XGBoost constructs an objective function with optimization in mind, using a second-order Taylor expansion method to handle the loss function and incorporate regularization terms. This approach balances minimizing the objective function and managing model complexity, thus reducing the risk of overfitting. The regularization term $\Omega(f_k)$ controls the intricacy of the tree structure by penalizing overly complex models, using an L2 norm of the weights to maintain a simpler tree and avoid overfitting [8].

III. METHODOLOGY

A. *Dataset*

The dataset used to train the load forecasting model is comprised of historical electrical load data from the Pembangkit Jawa Madura Bali (PJB) system. As shown in Fig. 3, illustrates the transmission network PJB system, highlighting key substations, power plants, and transmission lines across the Java and Madura islands. This transmission map is critical in understanding the spatial distribution of electricity demand, which forms the basis of the historical load dataset used for training the load forecasting model. The interconnected nature of this system provides insight into how load is distributed and managed across various regions

The historical data that has been used spanning the period from January 2024 to July 2024. This data represents the actual load consumption patterns within the region and serves as the primary foundation for the model. To improve the predictive accuracy, weather data was also integrated into the dataset. The weather information, obtained from a reliable open-source platform, includes critical environmental features such as temperature, dew point, humidity, and rainfall with same period from Januari 2024 to July 2024. These variables are essential in load forecasting, as weather conditions can significantly influence electricity demand. By combining both the load and weather data, the model is designed to better capture the complex interactions that impact electricity usage in the PJB system.

B. *Flowchart*

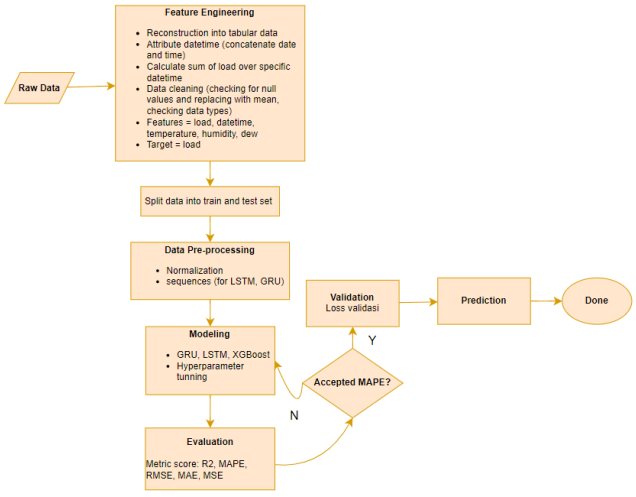

Fig 4. Workflow

Figure 4 illustrates the experimental process for building and evaluating a load forecasting model using time-series data. The pipeline begins with feature engineering, where raw data is transformed into a tabular format suitable for machine learning models. During this stage, the datetime attribute is created by combining date and time, and the sum of the load

is calculated over specific datetime intervals. Data cleaning is also performed by checking for missing values, which are replaced with the mean, and ensuring data types are correct. The features used for training include load, datetime, temperature, humidity, and dew, while the target variable is the load. After feature engineering, the data is split into training and test sets, followed by the pre-processing stage, where normalization is applied, and sequences are created for models like LSTM and GRU.

The modeling step involves training GRU, LSTM, and XGBoost models with hyperparameter tuning to enhance performance. After training, the models are evaluated using metrics such as R^2, MAPE, RMSE, MAE and the validation process checks the performance by calculating the validation loss. If the results are satisfactory, the model moves to the prediction phase, but if the performance is not acceptable (e.g., MAPE is too high), the model parameters are re-tuned. Once an acceptable MAPE score is reached, the process concludes with predictions.

To effectively evaluate the performance of the models in this study, several metrics were employed, each providing a unique perspective on prediction accuracy. R^2, also known as the coefficient of determination, measures how well the predicted values align with the actual values, offering insights into the overall goodness of fit. Alongside this, MAPE (Mean Absolute Percentage Error) was used to assess the prediction error as a percentage, making it easier to interpret relative performance. Additionally, RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) were applied to measure the magnitude of prediction errors. RMSE is particularly useful for penalizing larger errors, while MAE provides a straightforward average of absolute differences between predicted and actual values. Together, these metrics offer a comprehensive evaluation of the models' effectiveness in forecasting load demand [9]. R^2, MAPE, RMSE, MAE can be expressed by the following Equations 6-9:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - p_1)^2}{\sum_{i=1}^{n}(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i)^2} \quad [6]$$

$$MAPE = \frac{1}{n}\sum \left|\frac{y_i - \widehat{y_i}}{y_i}\right| x100 \quad [7]$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - p_i)^2}{n}} \quad [8]$$

$$MAE = \frac{\sum_{i=1}^{n}|y_i - p_i|}{n} \quad [9]$$

C. *Exploratory Data Analysis*

An exploratory data analysis was conducted to examine the relationships between key environmental factors and electricity load. Understanding these correlations is crucial for selecting relevant features and building accurate predictive models. The analysis focuses on the interaction between variables such as temperature, humidity, dew point, and rainfall, and their potential impact on the electricity load (MW). This initial exploration helps identify significant patterns and trends that can inform the subsequent modeling process.

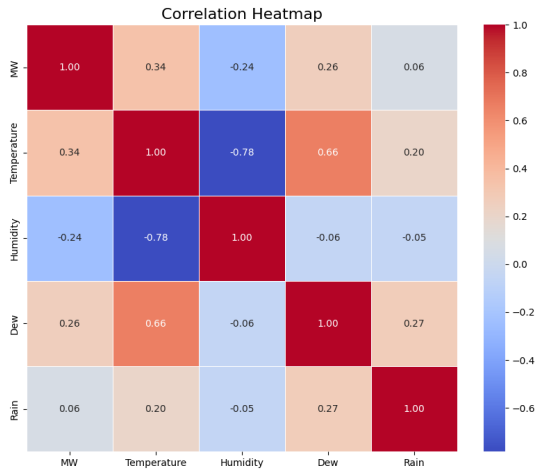

Fig 5. Correlation heatmap

As seen in Figure 5, the correlation analysis shows that temperature has the strongest positive correlation with the target variable, MW (load), with a correlation value of 0.34. This indicates that higher temperatures are associated with increased electricity load, likely due to higher cooling demands during warmer periods. Given that this correlation is the most significant among the variables examined, temperature stands out as a critical factor for inclusion in the forecasting model. Other variables like humidity, dew, and rain display weaker correlations with MW, ranging from -0.24 to 0.06, suggesting that they have a lesser impact on load fluctuations. For this study, the decision to focus on temperature is justified, as it provides the most substantial link to load variations. Additionally, it's important to note that the weather data used for this analysis was gathered from open-source databases, ensuring accessibility and transparency in the data collection process.

D. *Model Architecture*

The XGBoost model in this study is implemented using XGBRegressor with the 'reg:squarederror' objective function, which is optimized for minimizing squared error in regression tasks such as predicting electricity load. The model employs gradient boosting, an ensemble learning method that builds multiple decision trees, where each tree attempts to correct the errors of the previous ones. XGBoost enhances traditional gradient boosting by incorporating regularization, tree pruning, and learning rate adjustments to prevent overfitting and improve generalization.

The LSTM model is built using the Keras Sequential API. The architecture consists of multiple LSTM layers. The first LSTM layer has 50 units, configured with return_sequences=True, which ensures that the output of this layer is passed to the next layer in the sequence. This helps in learning temporal dependencies over longer sequences. A Dropout layer with a dropout rate of 0.3 is added to prevent overfitting by randomly disabling 30 % of neurons during training. The second LSTM layer also has 50 units but with return_sequences=False, meaning it only returns the final output, allowing the model to focus on the most important features of the sequence. Another Dropout layer is added after this LSTM layer. The model then has a Dense layer with 25 units and a ReLU activation function to introduce non-linearity, followed by a final Dense layer with 1 unit and a linear activation function to predict the output load value.

This LSTM model is tailored to handle the sequential nature of the data effectively, capturing temporal dependencies crucial for load estimation.

The GRU model follows a similar architecture but uses Gated Recurrent Units instead of LSTMs. The model is constructed through a custom function, create_gru_model, which allows flexibility in configuring the number of units and dropout rate. The first GRU layer is configured with the specified number of units (e.g., 50, 100, or 150) and return_sequences=True, meaning it returns the full sequence for further processing. A Dropout layer follows to prevent overfitting, using the specified dropout rate. A second GRU layer is added without return_sequences, which processes the entire sequence and returns only the final output. Another Dropout layer follows to prevent overfitting. The final Dense layer with 1 unit is responsible for outputting the predicted value. The model is compiled with the Adam optimizer and uses mean squared error (MSE) as the loss function and mean absolute error (MAE) as a metric for evaluating model performance.

Both models aim to capture the underlying temporal dependencies in the load prediction problem while leveraging the strengths of LSTM and GRU in handling sequential data.

E. Hyperparameter Testing

In the process of optimizing model performance, hyperparameter tuning played a critical role in improving the accuracy and efficiency of the models. For each model—GRU, LSTM, and XGBoost—several hyperparameters were tested and adjusted to find the optimal configuration.

The hyperparameter grid for XGBoost tests multiple key aspects of the model's configuration. The n_estimators parameter controls the number of boosting rounds, with higher values often leading to more accurate models at the cost of increased computational time. By exploring different max_depth values, we can control the complexity of the trees used in the boosting process. A deeper tree (higher max_depth) captures more patterns but risks overfitting. The learning_rate is another important parameter, balancing the step size taken during updates. A lower value allows the model to converge more slowly and carefully, while a higher rate speeds up training but may miss optimal solutions. Testing different values for subsample helps to generalize the model better by training on a portion of the dataset. Finally, reg_alpha and reg_lambda (L1 and L2 regularization) prevent overfitting by penalizing large weights. Tuning these parameters can help achieve a balance between model complexity and performance. This parameter testing generated 486 models.

For the LSTM model, we explore two different optimizers: Adam and RMSprop. Adam is an adaptive learning rate optimizer that generally works well across a range of problems, while RMSprop adjusts learning rates based on recent gradient changes, which can make it useful for time-series problems. Epochs (number of training iterations) are tested at two levels (50 and 100), allowing us to assess how much training is necessary for the model to converge. Shorter epochs may underfit the model, while longer epochs may risk overfitting. Finally, sequence length (24 and 40) tests how far back in the time-series the model looks when making predictions. A longer sequence captures more historical patterns, but may introduce unnecessary complexity and noise, depending on the dataset. This parameter testing generated 8 models.

For the GRU model, the number of units (50, 100, 150) controls the number of neurons in each hidden layer. Testing different values for dropout_rate helps prevent overfitting by randomly "dropping" some neurons during training. Batch_size is tested at two levels, 16 and 32, to determine the optimal number of samples to process before updating the model's weights. Smaller batch sizes tend to generalize better but take longer to train, while larger batch sizes speed up training but may lead to a less generalized model. Finally, the number of epochs is tested at two levels (40 and 60) to ensure the model trains for the right amount of time without overfitting. This parameter testing generated 36 models.

IV. Result and Analysis

TABLE 1 METRIC SCORE EVALUATION

Model	R^2	MAPE(%)	RMSE (MW)	MAE (MW)
XGBoost	0.589	4.76	1810.02	1152.77
LSTM	0.989	0.85	290.84	218.58
GRU	0.990	0.79	273.14	205.05

We assess the top-performing model among GRU, LSTM, and XGBoost using key metrics like R^2, MAPE, RMSE, and MAE to evaluate their effectiveness in forecasting electricity load. The performance metrics, see Table 1, clearly indicate that the GRU model outperformed both the LSTM and XGBoost models across all metrics. The GRU model achieved the highest R^2 value of 0.990, indicating that it explains 99 % of the variance in the actual load data. Additionally, the MAPE score of 0.79 % for GRU shows its remarkable accuracy, as the prediction error is less than 1 % on average. The RMSE and MAE values, 273.14 and 205.05 respectively, further demonstrate GRU's ability to minimize both larger and smaller errors, making it the most reliable model for this task.

The LSTM model performed comparably to GRU, achieving an R^2 of 0.989 and a slightly higher MAPE of 0.85 %. The LSTM's RMSE and MAE values (290.84 and 218.58) suggest that while it is highly accurate, it lags slightly behind GRU in terms of error reduction. This difference, although minimal, may be due to GRU's simpler architecture, which can learn dependencies in time-series data more efficiently than LSTM in some cases.

On the other hand, XGBoost underperformed significantly compared to both GRU and LSTM. With an R^2 of 0.589, XGBoost explains only 58.9 % of the variance in the data, which is far lower than the neural network models. The MAPE score of 4.76 % indicates that the average prediction error is considerably higher. XGBoost's RMSE and MAE values, 1810.02 and 1152.77 respectively, show a much larger spread of errors, reflecting its struggle to capture the temporal dependencies in the load data. This result highlights the limitations of tree-based models like XGBoost in time-series forecasting tasks, where recurrent neural networks such as LSTM and GRU excel due to their ability to learn from sequential data.

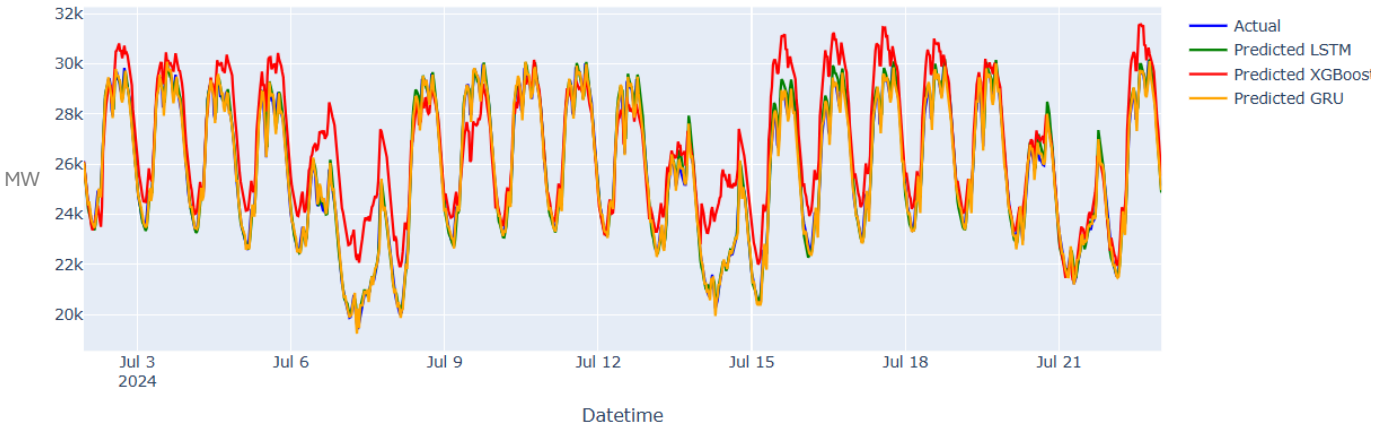

Fig 6. Comparison between actual and predicted for LSTM, XGBoost, and GRU

The time-series plot (see Fig.6) visually corroborates the performance metrics. Both the GRU and LSTM models closely follow the actual load data, with minimal deviation from the true values. In contrast, the XGBoost predictions deviate more frequently, particularly during periods of rapid load changes. The GRU model's predictions (in yellow) align almost perfectly with the actual data (in blue), indicating its ability to adapt to both steady and fluctuating load patterns. LSTM (green) also shows strong alignment but with slightly more deviations during peak load times. XGBoost (red), however, displays more pronounced errors, particularly underestimating load during peaks and overestimating during off-peak periods.

A. Error Analysis

The substantial gap between the neural network models (GRU and LSTM) and XGBoost can be attributed to the fundamental differences in how these models handle sequential data. While XGBoost is powerful in capturing non-linear relationships, it struggles with sequential dependencies that are prevalent in time-series data. On the other hand, both GRU and LSTM are designed specifically to capture these dependencies, making them more suitable for this task. The slight edge that GRU has over LSTM may be due to its simpler architecture, which helps it avoid overfitting and allows for faster convergence during training.

B. Best parameter

- XGBoost best model: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 500, 'reg_alpha': 0.01, 'reg_lambda': 0.1, 'subsample': 0.8}

  The best-performing configuration for XGBoost was found with a learning_rate of 0.1, which strikes a balance between convergence speed and the ability to find optimal solutions. A max_depth of 5 was chosen, indicating that moderately deep trees perform well for this task by capturing the non-linear relationships without overfitting. The high number of n_estimators (500) ensures that the model builds a large ensemble of trees, allowing for fine adjustments to predictions. The reg_alpha (L1 regularization) and reg_lambda (L2 regularization) values of 0.01 and 0.1, respectively, help prevent overfitting by penalizing large weights in the model. Finally, the subsample parameter, set at 0.8, indicates that the model uses 80 % of the data at each boosting iteration, which introduces diversity among the trees and prevents overfitting. Together, these parameters aim to balance model complexity and generalization ability, though the performance still lagged behind recurrent models in handling time-series data.

- GRU best model: {'batch_size': 32, 'dropout_rate': 0.2, 'epochs': 60, 'units': 150}

  The optimal GRU configuration involved a batch_size of 32, which is a moderate size that allows for faster training while maintaining sufficient gradient updates. A dropout_rate of 0.2 was applied, which helps to reduce overfitting by randomly dropping 20 % of the neurons during training. The model trained for 60 epochs, which provided enough iterations for the GRU to learn from the data without overfitting. The model utilized 150 units in the GRU layers, which allowed the network to capture complex patterns in the time-series data and retain long-term dependencies. This parameter configuration highlights that GRU benefits from having a larger number of units and a well-tuned dropout rate, which is essential for maintaining generalization without losing information during training. The performance metrics show that these parameters provided superior results compared to XGBoost, confirming GRU's effectiveness in time-series forecasting.

- LSTM best model: {epoch=100, optimizer=adam, seq_length=40}

  The LSTM model performed best with 100 epochs, which allowed the model to converge properly without overfitting. The Adam optimizer was selected, which is known for its efficiency and ability to handle noisy gradient updates by dynamically adjusting the learning rate during training. The sequence length of 40 means that the model looks back 40 time steps to make predictions, capturing a broader context of the historical data. This longer sequence length enabled the LSTM to effectively capture dependencies in the data, improving its ability to forecast future load values. The combination of these parameters reflects the LSTM's ability to process longer sequences of input data and adjust its learning rate dynamically, resulting in accurate forecasts, though it still slightly underperforms compared to GRU.

V. CONCLUSIONS

In this study, three models—GRU, LSTM, and XGBoost—were evaluated for their effectiveness in electricity load forecasting, with a focus on capturing temporal dependencies in time-series data. Additionally, weather factors were integrated into the dataset to enhance the model's predictive performance. The best parameter configurations were determined through hyperparameter tuning for each model. The XGBoost model, while optimized with parameters such as a learning rate of 0.1, a max depth of 5, and strong regularization, struggled with time-series forecasting, achieving significantly lower performance metrics compared to the neural network models. This limitation highlights XGBoost's difficulty in handling sequential data, despite its robustness in non-sequential tasks.

On the other hand, the GRU model, tuned with 150 units, a dropout rate of 0.2, and 60 epochs, outperformed the other models, achieving the best metrics across all evaluation measures, including an R^2 of 0.990 and a MAPE of 0.79 %. The LSTM model, with 100 epochs and a sequence length of 40, also performed very well, though it slightly lagged behind GRU, with an R^2 of 0.989 and a MAPE of 0.85 %. Both GRU and LSTM demonstrated their ability to capture the long-term dependencies in the data effectively, with GRU's simpler architecture giving it a slight performance edge.

In summary, the results indicate that GRU is the most suitable model for electricity load forecasting in this study, followed closely by LSTM. Both models significantly outperformed XGBoost, confirming the importance of recurrent neural networks in handling time-series data where temporal dependencies are critical. This study reaffirms that while traditional machine learning models like XGBoost are powerful in structured data tasks, recurrent neural networks (especially GRU) are superior for time-series forecasting, making them ideal for real-world load prediction applications.

ACKNOWLEDGMENT

The authors would like to extend their sincere gratitude to PLN UIP2B Jamali (Jawa-Madura-Bali) and Digital Management Division (DIVMDG) of PLN Head office for their support in providing the historical load data and weather-related information for this study. Their collaboration was crucial in constructing a comprehensive dataset, enabling the development of an accurate and robust electricity load forecasting model.

REFERENCES

[1] N. E. Benti, M. D. Chaka, and A. G. Semie, "Forecasting Renewable Energy Generation with Machine Learning and Deep Learning: Current Advances and Future Prospects," May 01, 2023, *MDPI*. doi: 10.3390/su15097087.

[2] K. Zor and K. Bulus, "A benchmark of GRU and LSTM networks for short-term electric load forecasting," in *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2021*, Institute of Electrical and Electronics Engineers Inc., Sep. 2021, pp. 598–602. doi: 10.1109/3ICT53449.2021.9581373.

[3] M. A. Hammad, B. Jereb, B. Rosi, and D. Dragan, "Methods and Models for Electric Load Forecasting: A Comprehensive Review," *Logistics & Sustainable Transport*, vol. 11, no. 1, pp. 51–76, Feb. 2020, doi: 10.2478/jlst-2020-0004.

[4] X. Gao, X. Li, B. Zhao, W. Ji, X. Jing, and Y. He, "Short-term electricity load forecasting model based on EMD-GRU with feature selection," *Energies (Basel)*, vol. 12, no. 6, 2019, doi: 10.3390/en12061140.

[5] L. Wu, C. Kong, X. Hao, and W. Chen, "A Short-Term Load Forecasting Method Based on GRU-CNN Hybrid Neural Network Model," *Math Probl Eng*, vol. 2020, 2020, doi: 10.1155/2020/1428104.

[6] R. Bareth, A. Yadav, S. Gupta, and M. Pazoki, "Daily average load demand forecasting using LSTM model based on historical load trends," *IET Generation, Transmission and Distribution*, vol. 18, no. 5, pp. 952–962, Mar. 2024, doi: 10.1049/gtd2.13132.

[7] N. Shabbir, R. Ahmadiahangar, A. Rosin, M. Jawad, J. Kilter, and J. Martins, "XgBoost based Short-term Electrical Load Forecasting Considering Trends & Periodicity in Historical Data," in *2023 IEEE International Conference on Energy Technologies for Future Grids, ETFG 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ETFG55873.2023.10407926.

[8] L. Zhang and D. Jánošík, "Enhanced short-term load forecasting with hybrid machine learning models: CatBoost and XGBoost approaches," *Expert Syst Appl*, vol. 241, May 2024, doi: 10.1016/j.eswa.2023.122686.

[9] A. Elanchezhian *et al.*, "Evaluating different models used for predicting the indoor microclimatic parameters of a greenhouse," *Appl Ecol Environ Res*, vol. 18, no. 2, pp. 2141–2161, 2020, doi: 10.15666/aeer/1802_21412161.

[10] PT. PLN (Persero), Rencana Usaha Penyediaan Tenaga Listrik PT PLN (Persero) 2019-2028. Jakarta, Indonesia, 2019