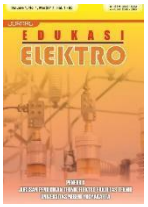
Vol. 9, No. 2, November 2025, pages 114 - 129

JEE

Jurnal Edukasi Elektro

https://doi.org/10.21831/jee.v9i2.86643



Recurrent Neural Network Model for Short-term Electric Load Forecasting

Muhammad Amri Yahya[1], Tegar Prasetyo[1], Muhammad Fikri[2]

[1]Politeknik Negeri Sriwijaya, Palembang, Indonesia

[2]Universitas Lampung, Bandar Lampung, Indonesia

Abstract—Accurate short-term electricity load forecasting is critical for planning power generation and maintaining cost efficiency. Since the amount of electricity generated significantly affects the cost-efficiency of power generation, a forecasting method with high accuracy is required. In response, this study developed a Recurrent Neural Network (RNN) model architecture trained using three different algorithms: Levenberg-Marquardt, Bayesian Regularization, and Scaled Conjugate Gradient. The model's performance was evaluated using the Mean Absolute Percentage Error (MAPE) metric. Historical load data were categorized by day type and divided into training and testing sets. The best-performing RNN model was used to carry out load forecasting, which was then compared to the forecasting results from PT PLN. Among the models tested, the RNN trained with Bayesian Regularization, configured with an 8-16-1 network architecture using a learning rate of 0.01 achieved the highest accuracy. In a two-week forecasting simulation, this model reached a MAPE of 1.4084%, significantly outperforming the 3.3160% error from PLN's conventional forecasting method. These results underpin the effectiveness of RNNs, particularly when trained with Bayesian Regularization, for enhancing short-term electricity load forecasting within the scope of this dataset.

Keywords: recurrent neural network, scaled conjugate gradient, levenberg-marquardt, bayesian regularization

Article submitted 2025 June 11.
Resubmitted 2025 July 7.
Final acceptance 2025 July 9.
Final version published as submitted by the authors.

Corresponding Author:
Muhammad Amri Yahya,
Politeknik Negeri Sriwijaya,
Palembang, Indonesia.
Email: muhammad.amri.yahya@polsri.ac.id

Citation Document:
Yahya, M. A., Prasetyo, T., & Fikri, M. (2025). Recurrent Neural Network Model for Short-term Electric Load Forecasting. *Jurnal Edukasi Elektro*, 9(2), 114-129. https://doi.org/10.21831/jee.v9i2.86643

1 Introduction

The amount of electric power generated has a significant impact on the cost efficiency of electricity production, making accurate load forecasting essential [1]. Proper forecasting ensures that the supply of electricity aligns with demand, minimizing both excess generation and shortages. As the state-owned utility responsible for electricity supply and distribution in Indonesia, the State Electricity Company (PLN) must carry out comprehensive load planning. This includes forecasting

to ensure that electricity generation closely matches real-time demand, optimizing resource use and system stability.

A key component of this planning process is Short-Term Load Forecasting (STLF), which is crucial but also presents complex challenges [2]. With ongoing technological advancements, STLF methods have evolved and are typically grouped into two main categories: conventional and non-conventional approaches. Conventional methods often rely on statistical techniques such as regression analysis and time series models, with popular examples including Autoregressive Integrated Moving Average (ARIMA) and the Holt-Winters exponential smoothing.

In contrast, recent studies have increasingly centered on Artificial Intelligence (AI) and Machine Learning (ML) approaches for load forecasting. Models such as the Convolutional Long Short-Term Memory (ConvLSTM) network [3], Long Short-Term Memory (LSTM) neural networks [4][5], LSTM combined with residual networks [6], LSTM-RNN hybrids [7], Deep Neural Networks (DNN) [8][9], Quantile LSTM networks [10], TCN-LSTM architectures [11], and Deep Recurrent Neural Networks (DRNN) [12] have shown promising results. These ML-based models often outperform traditional methods [13][14], largely due to their ability to capture complex, nonlinear patterns that conventional models struggle to address [15][16].

Several prior studies have examined the performance of different neural network architectures for STLF, including the Backpropagation Neural Network (BPNN), Elman Neural Network, Fuzzy Neural Network, and Wavelet Neural Network. Among these, BPNN was found to produce the most accurate predictions for short-term load forecasting [17]. Similarly, applied Recurrent Neural Networks (RNNs) demonstrated improved forecasting accuracy, with Mean Absolute Percentage Errors (MAPE) of 7.22% using 10 hidden neurons, 7.13% with 20 neurons, and 6.98% with 30 neurons, respectively. Compared to the feedforward neural network model, which had MAPE values of 9.05% with 10 hidden layers, 8.71% with 20 hidden layers, and 8.92% with 30 hidden layers. The training algorithm used was Levenberg-Marquardt [18].

Short-Term Load Forecasting (STLF) provides critical input to ensure informed decision-making in load management, generation scheduling, unit commitment, and precise load dispatch. This enables grid operators to optimize steady-state operation, enhance system reliability, and reduce operational costs. Under forecasting may lead to system overloading or potential blackouts. Conversely, over forecasting can result in increased spinning reserve requirements, consequently incurring higher operating expenses. Based on the discussion above, machine learning offers a promising alternative for improving the accuracy of electric load forecasting. Among the various algorithms available, the Recurrent Neural Network (RNN) is one widely used approach [19][20][21][22]. This research aims to develop and evaluate RNN models using three training algorithms: Levenberg-Marquardt, Scaled Conjugate Gradient, and Bayesian Regularization along with various network parameter settings, to identify the most effective configuration for short-term electric load forecasting.

2 Method

This study utilized MATLAB as the primary tool to support the research process. The materials comprised historical electrical load data and forecast data provided by PLN. These data were sourced from the Java Madura Bali electricity system, specifically from the distribution areas of Central Java and Yogyakarta, obtained through PT. PLN P3B Distribution Office in Central Java and Yogyakarta. The dataset consists of 30-minute interval load measurements collected daily.

The data was organized into seven groups according to the day of the week. Each group was further divided into training and testing subsets. To enhance forecasting accuracy, data from national holidays were excluded, as load patterns on holidays differ significantly from regular days. This research follows several stages to achieve its objectives, as illustrated in Figure 1.

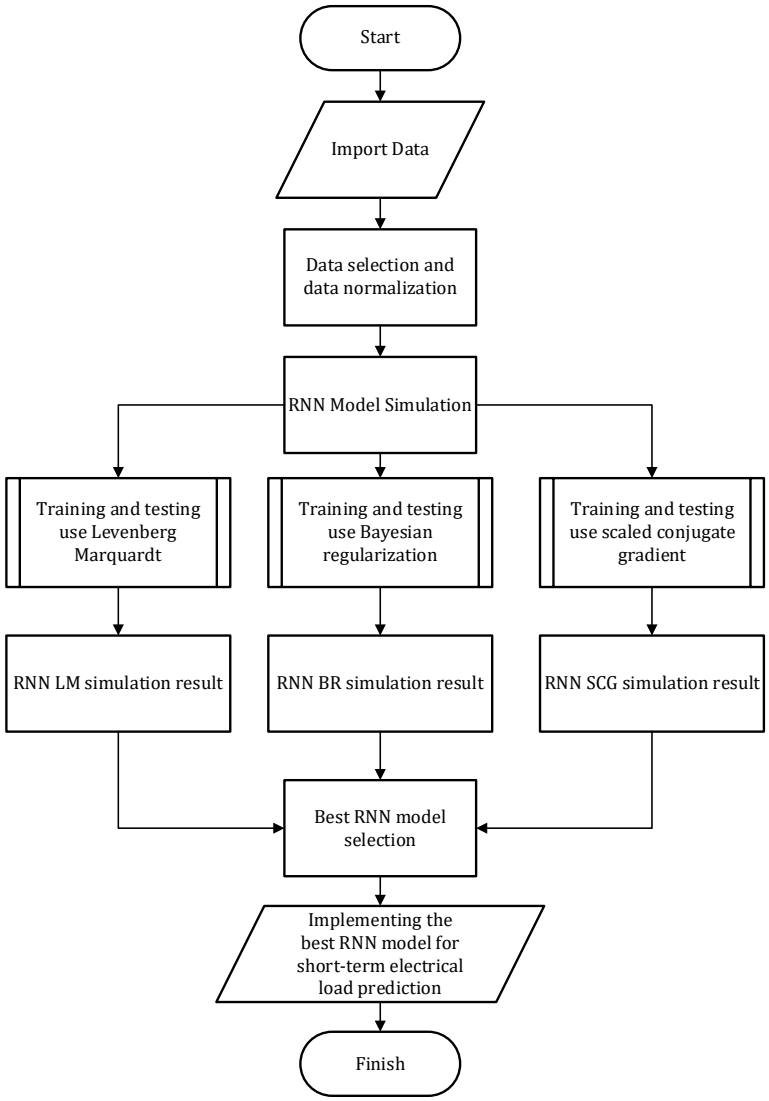


Figure 1. Flowchart illustrating the simulation stages for load forecasting with an RNN

The research materials needed in this study are historical data on electrical loads and load forecast data carried out by PLN. The data comes from the Java Madura Bali electricity system in the distribution areas of Central Java and Yogyakarta which were obtained from the office of PLN P3B distribution in Central Java and DIY. The data needed is the data load per unit 30 minutes every day. Historical data on load usage from January to March, shown in Figure 2. Data is classified into seven groups of data; each data grouped according to the name of the same day. Each group of data is divided into training and testing data. Data on national holidays are omitted to improve forecasting accuracy because the characteristics of the use of electrical loads on holidays are different from ordinary days. The input data is categorized based on the corresponding day of the week. The division is 80% for training data and 20% for testing data.

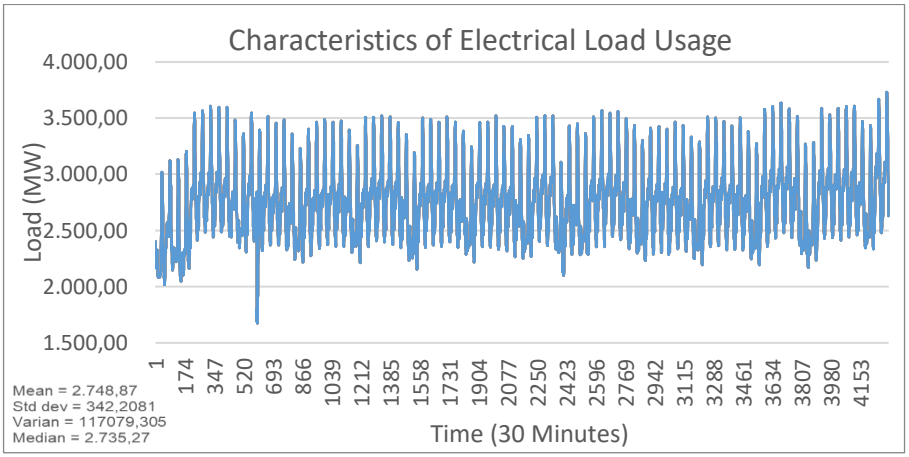


Figure 2. Characteristics of electricity load usage January 2015 – March 2015 per half hour

After selecting the data, normalization is applied to scale values into the interval [-1, 1], which helps improve the efficiency of the learning process.

2.1 RNN Model Training

Training on electrical load data is conducted using three algorithms: Levenberg-Marquardt for RNN model 1, Bayesian Regularization for RNN model 2, and Scaled Conjugate Gradient for RNN model 3.

a. Levenberg-Marquardt

The Levenberg-Marquardt algorithm aims to match the speed of second-order methods without directly computing the Hessian matrix. For the sum-of-squares performance function, the Hessian matrix and gradient are determined by Equations (1) and (2) respectively[23][24],

$$H = J^T J \tag{1}$$

$$g = J^T e \tag{2}$$

The Jacobian matrix (J) consists of the first-order partial derivatives of the neural network error function with respect to the network's weights and biases. The vector e represents the residual error between the network output and the target values. Computing the Jacobian typically involves standard backpropagation, which is more computationally intensive than direct Hessian approximation. The Levenberg–Marquardt algorithm employs a quasi-linearization strategy by approximating the inverse of the Hessian matrix using the Jacobian, as reflected in the iterative update formulation akin to Newton's method (see Equation 3), where $x$denotes the vector of connection weight parameters.

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \tag{3}$$

When the regularization parameter μ approaches zero, the algorithm approximates Newton's Method using an estimated Hessian, enabling rapid convergence. In contrast, for large values of μ, the algorithm behaves more like Gradient Descent with very small step sizes. Newton's Method is known for its quadratic convergence rate and high accuracy near optimal solutions. To leverage this, the Levenberg–Marquardt algorithm employs an adaptive strategy that reduces μ after each successful iteration and increases it only when the objective function (i.e., the network error) rises. This mechanism ensures a generally monotonic decrease in the objective function. By combining the fast convergence of Newton's Method with the stability of Gradient Descent, the Levenberg–

Marquardt algorithm achieves greater robustness and efficiency than conventional gradient-based approaches.

b. Bayesian Regularization

Regularization mitigates neural network overfitting. The Bayesian approach is ideal for this task because it automatically tunes regularization parameters while retaining the fast convergence of traditional backpropagation. The main objective of neural network training is to develop a model that maintains low error and performs effectively on unseen data. A model that performs consistently on both training and new data is considered to have good generalization. Regularization contributes to this by limiting the magnitude of the network's parameters [25].

Regularization improves a model's ability to generalize by constraining the size of its weights. When weights are smaller, the model's output becomes more stable in response to variations in the input data. By applying regularization, even a complex neural network can be simplified to better approximate the underlying true function. The standard backpropagation method seeks to minimize the function $F = E_d$, where E_d is determined using Equation (4).

$$E_d = \sum_{i=1}^{n}(t_i - a_i)^2 \quad (4)$$

In this context, n is the number of input data in the training set, t_i is the target value for the *i-th* data point, and a_i is the output from the neural network with respect to the data. Regularization alters the performance function by incorporating the standard deviation of the weights and biases, as illustrated in Equation (5) [26][27].

$$F = \beta E_d + \alpha E_w \quad (5)$$

α, β are regularization parameters, and E_w is defined in Equation (6)

$$E_w = \frac{1}{n}\sum_{i=1}^{n}(W_i)^2 \quad (6)$$

W_i represents the network's weights or thresholds. While applying Equation (6) encourages the minimization of these weights and thresholds, it does not inherently guarantee effective performance. Traditional methods often face challenges in selecting appropriate parameter values. To address this, MacKay proposed a Bayesian framework for neural networks that adaptively tunes these parameters, enabling improved generalization and model performance. Equation (7) presents the formulation for computing the regularization parameters.

$$\begin{cases} \alpha = \frac{\gamma}{2E_w} \\ \beta = \frac{n-\gamma}{2E_D} \end{cases} \quad (7)$$

$$\gamma = n - 2\alpha\, tr(H)^{-1}$$

where H is the Hessian matrix of the performance function F.

The following outlines the core steps of this approach.

1) Initialize α, β, weights, and biases.
   Minimize the error function based on the Levenberg-Marquardt algorithm using Equation (5).
2) Compute the Hessian matrix approximation using Equation (8)

   $$H = \beta J^T J + \alpha I \quad (8)$$

   The value of the effective number of parameters, γ, is then determined using.

   $$\gamma = n - 2\alpha\, tr(H)^{-1}$$

3) Update α, β using Equation (8)
   Repeat until the network converges.

c. Scaled Conjugate Gradient

While the standard backpropagation algorithm updates weights via steepest descent (the direction of quickest error decrease), this method doesn't guarantee the fastest overall convergence. The Conjugate Gradient (CG) algorithm improves upon this by searching along specially chosen "conjugate directions." These directions typically accelerate convergence while ensuring that error reductions achieved in earlier steps are not undone.

CG algorithms typically modify the step size during each iteration. To find the optimal step size within the conjugate direction and minimize the error function, they perform a line search (Equation 10). This line search approach is efficient as it avoids the computational burden of calculating the Hessian matrix.

All CG implementations begin the first iteration by using the steepest descent direction (Equation 9). For subsequent iterations, the new search direction is generated to be conjugate to the previous one (Equation 11). This is achieved by combining the current steepest descent gradient with the direction used in the prior search steps [28].

$$p_o = -g_o \tag{9}$$

$$x_{k+1} = x_k + \alpha_k g_k \tag{10}$$

$$p_k = -g_k + \beta_k p_{k-1} \tag{11}$$

Different versions of the Conjugate Gradient (CG) algorithm vary in how the β_k is calculated.

An alternative step-size estimation method is the Scaled Conjugate Gradient (SCG), which amalgamates the trust region approach from the Levenberg-Marquardt algorithm with the conjugate gradient approach. Introduced by Møller (1993), this method uses Equation (12), where s is the Hessian approximation, E is the aggregate error function, E' is the gradient E , λ_k and σ_k are scaling factors initialized by the user at the algorithm's start so that $0 < \lambda_k < 10^{-6}$ and $0 < \sigma_k < 10^{-4}$. For *SCG*, β_k computation of the search direction and scaling factor are given in Equations (13) and (14) as follows [29].

$$s_k = \frac{E'(w_k\, \sigma_k p_k) - E'(w_k)}{\sigma_k} + \lambda_k p_k \tag{12}$$

$$\beta_k = \frac{(|g_{k+1}|^2 - g_{k+1}{}^T g_k}{g_k{}^T g_k} \tag{13}$$

$$p_{k+1} = -g_{k+1} \beta_k p_k \tag{14}$$

SCG updates its design parameters independently at each iteration, which is crucial to its success. This gives SCG a main advantage over line-search-based algorithms.

d. Proposed Architecture Design

A neural network architecture generally comprises input layers, one or several intermediate layers called hidden layers, and an output layer. In this study, the hidden layers were configured with 4, 8, 12, and 16 neurons. Based on several references, the size of the hidden layer is typically chosen to be between the sizes of the input and output layers. In this study, the number of neurons in the hidden layer was kept to less than twice that of the input layer. Figure 3 show the recurrent neural network architecture.

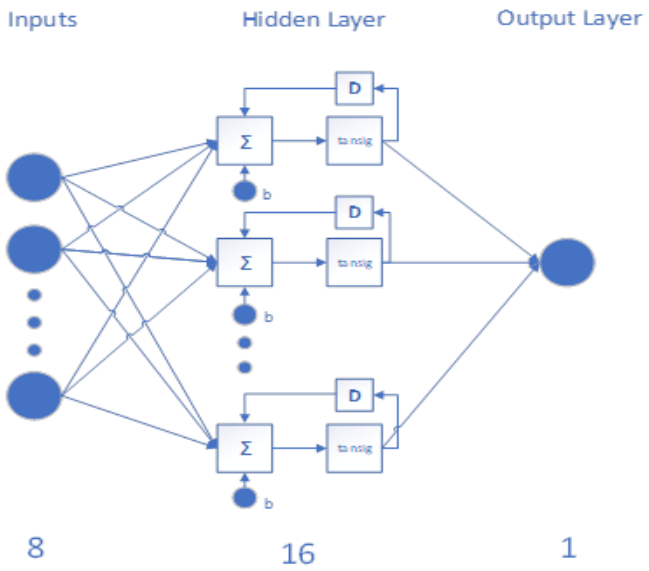


Figure 3. Recurrent neural network architecture

e. Setting the Parameters

In this study, the maximum number of epochs was set to 500, with a target error of 0.0001. The learning rates tested were 0.01, 0.05, and 0.1. Table 1 summarizes the parameter settings for each training algorithm. The Levenberg-Marquardt algorithm uses a damping parameter, initialized at 0.001, to balance between Gradient Descent and Gauss-Newton methods. The μ is adaptively adjusted (decreased by a factor of 0.1 or increased by a factor of 10) to optimize convergence and avoid saddle points. Bayesian regularization incorporates Bayesian inference to automatically tune the regularization parameters α and β (as shown in Equation 7), penalizing large weights (E_w) to reduce overfitting by maximizing the model evidence (marginal likelihood). The Scaled Conjugate Gradient method avoids explicit Hessian computation by using line searches (Equation 10) and conjugate directions. It relies on parameters controlling step size and scaling factors, which are crucial for accurately estimating curvature on non-quadratic error surfaces. All algorithms used the same maximum epoch of 500 and target error of 0.0001 to ensure a fair comparison. Notably, Bayesian regularization starts with a higher initial μ value (0.005), reflecting its focus on regularization rather than rapid convergence.

Table 1. Parameter settings for each training algorithm

Parameter Configuration	Levenberg-Marquardt	Bayesian regularization	Scaled Conjugate Gradient
Maximum epoch	500	500	500
Target error/Performance goal	0.001	0.0001	0.0001
Initial μ	0.001	0.005	N/A
μ decay factor	0.1	0.1	N/A
μ growth factor	10	10	N/A
Maximum μ	1.00×10^{10}	1.00×10^{10}	N/A
σ_0	N/A	N/A	5.0×10^{-5}
λ_0	N/A	N/A	5.0×10^{-7}

2.2 Network Testing

This study uses data from weeks 2 to 9, totaling 384 records. Prediction accuracy is evaluated using several metrics, with the most used being [30]. Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{n}\frac{\sum_{i=1}^{n}|X_t - F_t|}{X_t} \quad (15)$$

Equation (15) defines the Mean Absolute Percentage Error (MAPE):
Explanation:
X_t = actual value at time t
F_t= forecasted (simulated) value at time t
e = error or difference between (X_t and F_t)
n = number of observations
After training and testing, the best-performing network model is selected based on the lowest MAPE value. This model is then used to forecast the electricity load one day ahead, and the results are compared with those generated by PT. PLN.

3 Result

3.1 Simulation Results of the RNN-Based Models

a. Levenberg-Marquardt Algorithm

The simulation results show that variations in the number of hidden-layer neurons and learning rates led to only minor differences in MAPE values. The optimal configuration by using 8 hidden neurons and a learning rate of 0.1 achieved the lowest testing MAPE of 1.0276%. The short training time (1 second) and rapid convergence highlight the efficiency of the Levenberg–Marquardt algorithm, particularly for convex optimization tasks. Table 2 show the simulation results using the levenberg-marquardt algorithm.

Table 2. Simulation results using the levenberg-marquardt algorithm

No.	Neuron Hidden Layer	Target	Learning Rate	MAPE		Time
				Training	*Testing*	
1	4	0.0001	0.01	0.1870	1.1584	1 s
2			0.05	0.1559	1.0559	1 s
3			0.1	0.1654	1.1720	1 s
4	8	0.0001	0.1	0.1816	1.1885	1 s
5			0.5	0.1967	1.1462	1 s
6			0.1	0.1995	1.0276	1 s
7	12	0.0001	0.01	0.1812	1.0688	1 s
8			0.05	0.1789	1.0339	1 s
9			0.1	0.1642	1.2184	1 s
10	16	0.0001	0.01	0.1704	1.0333	1 s
11			0.05	0.1678	1.0406	1 s
12			0.1	0.1435	1.0571	1 s

b. Scaled Conjugate Gradient Algorithm

The simulation results indicate that changes in the number of artificial neurons in the hidden layer and the value of the learning rate resulted in no significant variation in MAPE values. The best testing MAPE, 1.0096%, was achieved using 16 hidden neurons with a learning rate of 0.01. Table 3 show the simulation results using the scaled conjugate gradient algorithm.

Table 3. Simulation results using the scaled conjugate gradient algorithm

No.	Neuron Hidden Layer	Target	Learning Rate	MAPE		Time
				Training	*Testing*	
1	4	0.0001	0.01	0.2398	1.0718	3 s
2			0.05	0.1982	1.1006	3 s
3			0.1	0.2034	1.2673	3 s
4	8	0.0001	0.01	0.2034	1.2673	3 s
5			0.05	0.2049	1.1002	2 s
6			0.1	0.1848	1.1217	1 s
7	12	0.0001	0.01	0.1910	1.0815	1 s
8			0.05	0.1959	1.0423	1 s
9			0.1	0.1901	1.3025	2 s
10	16	0.0001	0.01	0.1931	1.0096	1 s
11			0.05	0.1822	1.1588	3 s
12			0.1	0.1918	1.0840	1 s

c. Bayesian Regularization Algorithm

The simulation results suggest that variations in the number of hidden-layer neurons and learning rates led to minimal differences in MAPE values. The optimal configuration consisted of 16 neurons with a learning rate of 0.01 and yielded the lowest testing MAPE of 0.9909%. The longer training duration of 14 seconds is attributed to Bayesian hyperparameter updates. The narrow train-test MAPE gap (approximately 0.8%) demonstrates the effectiveness of Bayesian regularization in preventing overfitting. Moreover, Bayesian regularization consistently achieved lower MAPE values (≤ 1.0787%) across all architectures, whereas Levenberg–Marquardt and Scaled Conjugate Gradient showed greater variability, with train-test differences reaching up to 1.2%.

Table 4. Simulation results using the bayesian regularization algorithm

No.	Neuron Hidden Layer	Target	Learning Rate	MAPE		Time
				Training	*Testing*	
1	4	0.0001	0.01	0.2033	0.9991	3 s
2			0.05	0.2033	0.9991	2 s
3			0.1	0.2033	0.9991	2 s
4	8	0.0001	0.01	0.1995	0.9952	2 s
5			0.05	0.1944	1.0027	2 s
6			0.1	0.1993	0.9940	3 s
7	12	0.0001	0.01	0.1929	1.0787	4 s
8			0.05	0.1967	1.0587	4 s
9			0.1	0.1960	1.0028	4 s
10	16	0.0001	0.01	0.1972	0.9909	14 s
11			0.05	0.1921	1.0642	22 s
12			0.1	0.1927	1.1379	22 s

3.2 Selection of the Best Architecture and Network Parameters

This section is a description of the simulation results in section 3.1. where the Bayesian regularization training algorithm produces the highest accuracy. The training process yields different error values due to variations in the calculation methods of each training algorithm. The rate at which these error values change also differs, with some converging quickly and others gradually getting more gradually toward a steady state. Daily estimations are performed using test data, and the average error over one week is calculated using the MAPE formula. The training algorithm that produces the lowest MAPE is selected for this study.

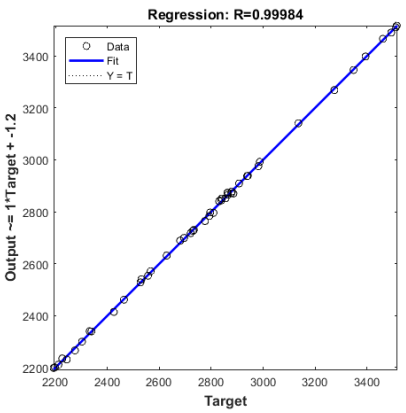


Figure 4. Linear regression of model output versus target values

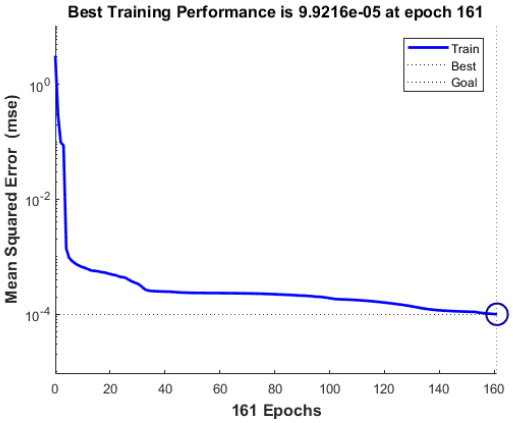


Figure 5. Training means squared error (MSE) over epochs

Figure 4 depicts the actual versus predicted load along with R^2/RMSE metrics. The high R^2 value (>0.98) indicates a strong linear correlation between the predictions and actual data. However, residual patterns suggest a slight model bias, with underprediction occurring during peak loads.

Figure 5 illustrates the error reduction during training with Bayesian regularization over 161 epochs. The plot shows a sharp initial decline corresponding to the nonlinear optimization phase, followed by asymptotic convergence as the gradient approaches zero. The final MSE of approximately 0.0001, consistent with the values reported in Table 4, along with the absence of spikes, confirms the stability and effectiveness of Bayesian regularization.

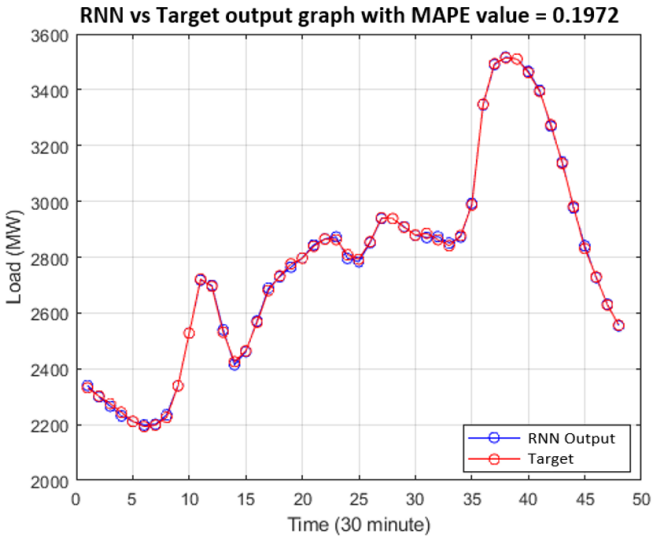


Figure 6. Training results: predicted load using Bayesian regularization compared to actual data

Figure 6 shows the training results of the Bayesian regularization model compared to the actual load. The close overlap indicates low bias, exemplified by a MAPE of 0.1972% for Monday, demonstrating accurate fitting of the training data. Minor deviations at peak and valley points suggest slight underfitting. Figure 7 presents the forecasting results, where a high agreement with actual data (MAPE of 0.9909%) validates the model's effectiveness in capturing temporal features such as daily load cycles.

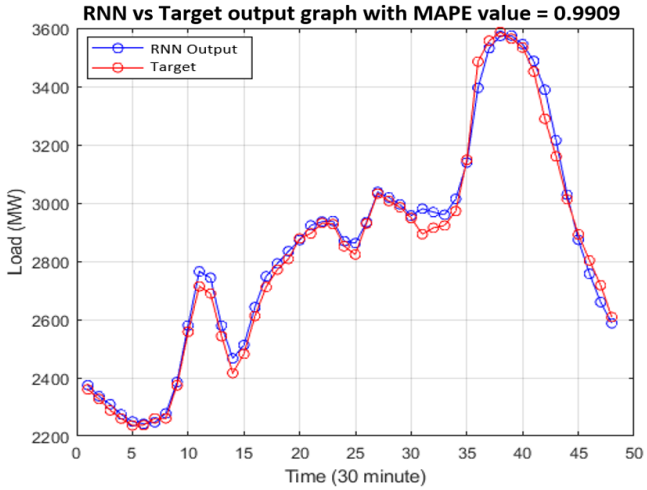


Figure 7. Testing results: predicted load using Bayesian regularization compared to actual data

Bayesian Regularization (BR) demonstrated clear superiority, achieving the lowest test MAPE of 0.9909% with an 8-16-1 architecture. This outperformed the Levenberg–Marquardt algorithm, which recorded a MAPE of 1.0276%, and the Scaled Conjugate Gradient method, with 1.0096%.

Regarding architectural impact, both Bayesian Regularization and Scaled Conjugate Gradient required 16 neurons in the hidden layer to effectively model temporal dependencies, whereas Levenberg–Marquardt achieved optimal results with only 8 neurons, indicating suitability for simpler dynamics. In terms of learning rate sensitivity, Bayesian Regularization and Scaled Conjugate Gradient performed best at a low learning rate of 0.01, while Levenberg–Marquardt achieved optimal performance at a higher learning rate of 0.1.

Training results showed that the Levenberg–Marquardt algorithm achieved the lowest mean training MAPE of 0.1435%, compared to Bayesian Regularization at 0.1921% and Scaled Conjugate Gradient at 0.1848%. However, on the test dataset, Levenberg–Marquardt exhibited the highest MAPE of 1.0276%. In contrast, Bayesian Regularization attained the lowest test MAPE of 0.9909%, followed by Scaled Conjugate Gradient with 1.0096%. Based on these findings, Bayesian Regularization was selected as the preferred training algorithm for this study.

As shown in Table 5, the Levenberg–Marquardt algorithm exhibits the highest average error values compared to Bayesian Regularization and Scaled Conjugate Gradient, likely due to its excessively rapid convergence. This rapid convergence leads to suboptimal performance, particularly with limited datasets like the one used in this study. In contrast, Bayesian Regularization demonstrates superior generalization, producing the lowest error values when tested with new input data. After evaluating various network architectures and training algorithms, the optimal recurrent neural network model was identified with an 8-16-1 architecture and a learning rate of 0.01. This model, trained using Bayesian Regularization, achieved the lowest MAPE of 0.9909%, as reported in Table 5.

Table 5. Comparison of best results from each training algorithm

No.	Algorithm	Architecture	Target Error	Learning Rate	MAPE %	
					Training	*Testing*
1	Levenberg-Marquardt	8-8-1	0.0001	0.1	0.1995	1.0276
2	Bayesian regularization	8-16-1	0.0001	0.01	0.1972	0.9909
3	scaled conjugate gradient	8-16-1	0.0001	0.01	0.1931	1.0096

3.3 Load Forecasting Results Using the Selected Models

The simulation results for overall load estimation are presented in Table 6. During the first week, the lowest forecast error (MAPE) of 0.9537% was observed on Friday, while the highest MAPE of 1.9401% occurred on Saturday. The notable difference between the training MAPE for Saturday (0.8367%) and its testing MAPE (1.9401%) suggests that the model did not fully capture the complexity of weekend load patterns.

Table 6. Load forecast results for the first week

Day	*Training*	*Testing*	Time
Monday	0.1972	0.9909	14 s
Tuesday	0.5120	1.6952	10 s
Wednesday	0.3751	1.7629	37 s
Thursday	0.6455	1.1239	25 s
Friday	0.5146	0.9537	20 s
Saturday	0.8367	1.9401	19 s
Sunday	0.4565	1.5742	36 s

For the second week, the lowest estimated error (MAPE) was 1.1328%, observed on Monday, as shown in Table 7.

Table 7. Load forecast results for the second week

Day	*Training*	*Testing*	Time
Monday	0.3563	1.1328	25 s
Tuesday	0.6187	1.4379	14 s
Wednesday	0.8115	1.7012	18 s
Thursday	0.7399	1.4109	15 s
Friday	0.4553	1.3666	22 s
Saturday	0.1376	1.3594	12 s
Sunday	0.1588	1.2689	5 s

The superiority of the RNN model is demonstrated by its average MAPE of 1.4084%, compared to PLN's 3.3160%, representing a 57.5% reduction in forecasting error. This improvement indicates potential for significant cost savings in power generation. PLN's limitations are highlighted by its highest errors on Monday (5.5448%) and Saturday (4.1951%), likely due to inadequate modeling of public holiday effects. In contrast, the RNN model consistently achieves daily MAPE values below 2%, with a peak of 1.7629% on Wednesday, well within the 5% industry tolerance threshold. Table 8 presents the overall simulation results for the one-week electricity load forecast alongside PLN's forecast accuracy, comparing MAPE values.

Table 8. Comparison of MAPE values: RNN vs. PLN

Day	MAPE RNN (%)		MAPE PLN (%)	
	Week 1	Week 2	Week 1	Week 2
Monday	0.9909	1.1328	2.8273	5.5448
Tuesday	1.6952	1.4379	3.0174	4.0197
Wednesday	1.7629	1.7012	3.0755	2.8847
Thursday	1.1239	1.4109	2.8061	2.6185
Friday	0.9537	1.3666	3.1788	3.7201
Saturday	1.9401	1.3594	3.3138	4.1951
Sunday	1.5742	1.2689	2.3546	2.8676
Average	1.4344	1.3825	2.9391	3.6929
	1.4084		3.3160	

PLN's load forecasting exhibited variable MAPE performance. During the first week, forecasts ranged from a minimum MAPE of 2.3138% to a maximum of 3.3138%, with a weekly average of 2.9391%, which remains below the 5% industry tolerance threshold. In the second week, MAPE values varied between 2.6185% and 5.5448%, resulting in a weekly average of 3.6929%. The overall two-week forecast MAPE for PLN was 3.3160%.

In contrast, the RNN-based forecasting model consistently produced lower daily MAPE values. The most accurate prediction during the first week was on Friday with a MAPE of 0.9537%, while the highest error was on Saturday at 1.9401%, leading to a weekly average MAPE of 1.4344%. In the second week, the RNN model's MAPE ranged from 1.1328% on Monday to 1.7012% on Wednesday, with a weekly average of 1.3825%. The consolidated two-week MAPE for the RNN model was 1.4084%. These performance differences are illustrated in Figure 8 and Figure 9.

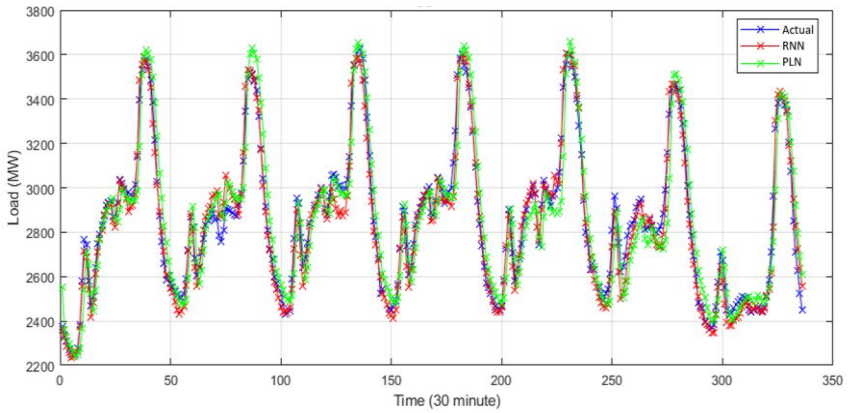


Figure 8. Comparison of load forecast results for the first week between RNN model and PLN

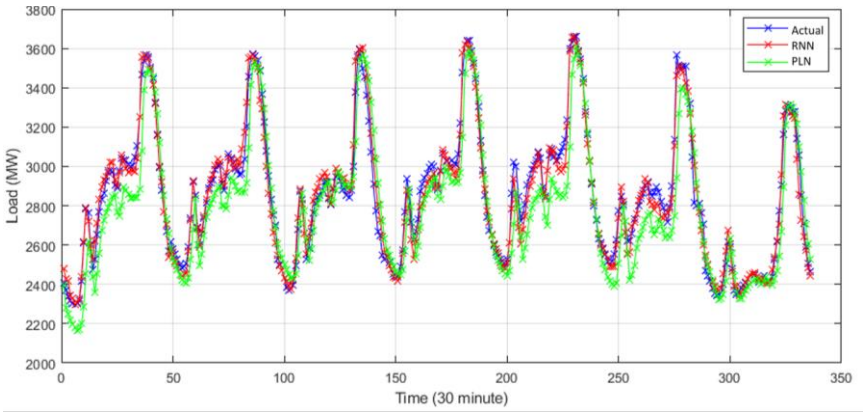


Figure 9. Comparison of the load forecast results for the second week between RNN model and PLN

As shown in Figure 8 and Figure 9, the RNN-based forecasts consistently produce lower MAPE values than PLN's forecasts. Moreover, the RNN's MAPE remains well below the 5% tolerance threshold for load forecast deviation, indicating superior accuracy. These results suggest that implementing RNN for load forecasting can enhance both the reliability and economic efficiency of the electrical system.

4 Discussion

The findings of this study highlight overfitting as a common challenge in neural network training. Although the training error can be minimized to very small values, the error tends to increase significantly when the model is tested on new data. This occurs because the network memorizes the training samples without learning to generalize to unseen situations. Techniques such as early stopping and regularization are effective in promoting generalization. In this study, early stopping was implemented with a target error of 0.0001. Using Bayesian regularization with an 8-16-1 architecture, the model achieved the lowest MAPE of 0.9909%, demonstrating strong generalization. This improvement is attributed to the adjustment of regularization parameters α and β (see Eq. 7), which help mitigate overfitting [26][27].

The model's accuracy decreases during weekends and holidays due to irregular consumption patterns. Nevertheless, the RNN achieved a 57% lower error compared to PLN's forecast, demonstrating the effectiveness of AI-based approaches for short-term load forecasting. Based on these findings, it is recommended that PLN adopt the Bayesian regularization RNN model to optimize electricity generation, thereby reducing overproduction and operational costs. Additionally, preprocessing the data by grouping national holiday data separately could enhance prediction accuracy for these irregular days.

This study has some limitations. The data is limited to Central Java and Yogyakarta, so validation in other regions with different load patterns (such as those in island areas) is necessary. Excluding holiday data to reduce noise may impair forecast accuracy during anomalous periods; integrating a holiday calendar as an input feature is suggested as a solution. Parameter settings were constrained during testing: the maximum epoch was set to 500, target error at 0.0001, learning rates were varied (0.01, 0.05, and 0.1), and hidden neurons were tested at 4, 8, 12, and 16.

For future research, deeper architectures such as Deep Recurrent Neural Networks could be explored by increasing hidden layers. Hybrid models that combine RNN with LSTM or GRU may better capture long-term dependencies. Incorporating k-fold cross-validation can improve model stability assessment. Further, integrating statistical methods like ARIMA could capture both linear and nonlinear patterns. To further improve the model performance, supplementary data such as holidays events, weather reports, and economic patterns could be incorporated. Ultimately, to

improve forecasting accuracy, one can implement autotuning that is based on dynamic optimization via reinforcement learning to auto-tune the hyperparameter.

5 Conclusion

This study confirms that Recurrent Neural Networks (RNNs) are highly effective for short-term electricity load forecasting. In this study, we tested three training algorithms. Bayesian regularization came out on top, using an 8-16-1 network and a learning rate of 0.01, proved optimal for short-term load forecasting, achieving a two-week MAPE of 1.4084%, which represents a 57% improvement over PLN's conventional methods. These findings underscore the transformative potential of AI-driven forecasting systems for PLN, while also highlighting the need for further research on dynamic data integration and regional scalability.

6 References

[1] H. Zhu, Q. Lin, X. Li, H. Xiao, and T. Shao, "Short-term electrical load forecasting based on pattern label vector generation," *Energy Build*, vol. 331, p. 115383, Mar. 2025, doi: 10.1016/J.ENBUILD.2025.115383.

[2] Z. Peng and X. Yang, "Short- and medium-term power load forecasting model based on a hybrid attention mechanism in the time and frequency domains," *Expert Syst Appl*, vol. 278, p. 127329, Jun. 2025, doi: 10.1016/J.ESWA.2025.127329.

[3] S. H. Rafi, N. Al Masood, and S. R. Deeba, "An effective short-term load forecasting methodology using convolutional long short term memory network," *Proceedings of 2020 11th International Conference on Electrical and Computer Engineering, ICECE 2020*, pp. 278–281, Dec. 2020, doi: 10.1109/ICECE51571.2020.9393067.

[4] H. Xu, Y. Zhang, and Y. Zhao, "Short-Term Electricity Load Forecasting Based on Ensemble Empirical Mode Decomposition and Long Short-Term Memory Neural Network," *Proceedings - 2023 IEEE International Conference on Energy Internet, ICEI 2023*, pp. 271–275, 2023, doi: 10.1109/ICEI60179.2023.00058.

[5] W. T. Zheng, Q. Z. Wan, L. Zhu, N. Lv, and M. Li, "Short Term Power Load Forecasting Based on Clustering and Long Short Term Memory Network," *Proceedings of 2021 IEEE 4th International Electrical and Energy Conference, CIEEC 2021*, May 2021, doi: 10.1109/CIEEC50170.2021.9510632.

[6] M. Xu, J. Dong, and X. Sun, "Optimization Research on Short-Term Power Load Forecasting Model Combining LSTM and Residual Networks," *2025 2nd International Conference on Smart Grid and Artificial Intelligence (SGAI)*, pp. 1140–1144, Mar. 2025, doi: 10.1109/SGAI64825.2025.11009588.

[7] F. Li, X. Yu, X. Tian, and Z. Zhao, "Short-Term Load Forecasting for an Industrial Park Using LSTM-RNN Considering Energy Storage," *2021 3rd Asia Energy and Electrical Engineering Symposium, AEEES 2021*, pp. 684–689, Mar. 2021, doi: 10.1109/AEEES51875.2021.9403118.

[8] G. M. U. Din and A. K. Marnerides, "Short term power load forecasting using Deep Neural Networks," *2017 International Conference on Computing, Networking and Communications, ICNC 2017*, pp. 594–598, Mar. 2017, doi: 10.1109/ICCNC.2017.7876196.

[9] W. Waheed and Q. Xu, "Data-driven short term load forecasting with deep neural networks: Unlocking insights for sustainable energy management," *Electric Power Systems Research*, vol. 232, p. 110376, Jul. 2024, doi: 10.1016/J.EPSR.2024.110376.

[10] S. A. H. Shah *et al.*, "Improved electric load forecasting using quantile long short-term memory network with dual attention mechanism," *Energy Reports*, vol. 13, pp. 2343–2353, Jun. 2025, doi: 10.1016/J.EGYR.2025.01.058.

[11] J. Tian, H. Liu, W. Gan, Y. Zhou, N. Wang, and S. Ma, "Short-term electric vehicle charging load forecasting based on TCN-LSTM network with comprehensive similar day identification," *Appl Energy*, vol. 381, p. 125174, Mar. 2025, doi: 10.1016/J.APENERGY.2024.125174.

[12] P. B. Atosha, E. Özbilge, and Y. Kırsal, "Comparative Analysis of Deep Recurrent Neural Networks for Speech Recognition," *32nd IEEE Conference on Signal Processing and Communications Applications, SIU 2024 - Proceedings*, 2024, doi: 10.1109/SIU61531.2024.10600944.

[13] A. Wan, Q. Chang, K. AL-Bukhaiti, and J. He, "Short-term power load forecasting for combined heat and power using CNN-LSTM enhanced by attention mechanism," *Energy*, vol. 282, Nov. 2023, doi: 10.1016/j.energy.2023.128274.

[14] T. Bashir, C. Haoyong, M. F. Tahir, and Z. Liqiang, "Short term electricity load forecasting using hybrid prophet-LSTM model optimized by BPNN," *Energy Reports*, vol. 8, pp. 1678–1686, Nov. 2022, doi: 10.1016/j.egyr.2021.12.067.

[15] K. Y. Lee, Y. T. Cha, and J. H. Park, "Short-term load forecasting using an artificial neural network," *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 124–132, 1992, doi: 10.1109/59.141695.

[16] S. Ding, C. Su, and J. Yu, "An optimizing BP neural network algorithm based on genetic algorithm," *Artif Intell Rev*, vol. 36, no. 2, pp. 153–162, Aug. 2011, doi: 10.1007/S10462-011-9208-Z.

[17] W. Jie-sheng and Z. Qing-wen, "Short-term Electricity Load Forecast Performance Comparison Based on Four Neural Network Models," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, IEEE, May 2015, pp. 2946–2950. doi: 10.1109/CCDC.2015.7162426.

[18] H. Park, B. Lee, J. Son, and H. Ahn, "A comparison of neural network-based methods for load forecasting with selected input candidates," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, IEEE, Mar. 2017, pp. 1100–1105. doi: 10.1109/ICIT.2017.7915516.

[19] L. Paranhos, "Predicting inflation with recurrent neural networks," *Int J Forecast*, Mar. 2025, doi: 10.1016/J.IJFORECAST.2024.07.010.

[20] M. Szymkowiak, "Using recurrent neural networks to build a data-driven model of an autonomous underwater vehicle," *Transportation Research Procedia*, vol. 83, pp. 417–424, Jan. 2025, doi: 10.1016/J.TRPRO.2025.03.008.

[21] Y. Chu, J. Fei, and S. Hou, "Adaptive Global Sliding-Mode Control for Dynamic Systems Using Double Hidden Layer Recurrent Neural Network Structure," *IEEE Trans Neural Netw Learn Syst*, vol. 31, no. 4, pp. 1297–1309, Apr. 2020, doi: 10.1109/TNNLS.2019.2919676.

[22] P. Amos, S. Narendran, and M. Keerthivasan, "Analysis Of Traffic Sign Recognition Using Artificial Neural Network Algorithm Compared With Accuracy Of Recurrent Neural Networks," *2024 9th International Conference on Applying New Technology in Green Buildings, ATiGB 2024*, pp. 502–506, 2024, doi: 10.1109/ATIGB63471.2024.10717662.

[23] S. Mishra, R. Prusty, and P. K. Hota, "Analysis of Levenberg-Marquardt and Scaled Conjugate gradient training algorithms for artificial neural network based LS and MMSE estimated channel equalizers," *Proceedings - 2015 International Conference on Man and Machine Interfacing, MAMI 2015*, no. Lm, 2016, doi: 10.1109/MAMI.2015.7456617.

[24] L. M. Saini and M. K. Soni, "Artificial neural network based peak load forecasting using Levenberg–Marquardt and quasi-Newton methods," *IEE Proceedings - Generation, Transmission and Distribution*, vol. 149, no. 5, p. 578, 2002, doi: 10.1049/ip-gtd:20020462.

[25] B. G. Heydecker and J. Wu, "Identification of sites for road accident remedial work by Bayesian statistical methods: An example of uncertain inference," *Advances in Engineering Software*, vol. 32, no. 10–11, pp. 859–869, 2001, doi: 10.1016/S0965-9978(01)00037-0.

[26] Z. Sun, Y. Chen, X. Li, X. Qin, and H. Wang, "A Bayesian regularized artificial neural network for adaptive optics forecasting," *Opt Commun*, vol. 382, pp. 519–527, Jan. 2017, doi: 10.1016/J.OPTCOM.2016.08.035.
[27] J. L. Ticknor, "A Bayesian regularized artificial neural network for stock market forecasting," *Expert Syst Appl*, vol. 40, no. 14, pp. 5501–5506, 2013, doi: 10.1016/j.eswa.2013.04.013.
[28] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*. Boston: PWS Publishing, 1996.
[29] M. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," 1993. doi: DOI: 10.1016/S0893-6080(05)80056-5.
[30] Q. Dong *et al.*, "Short-Term Electricity-Load Forecasting by deep learning: A comprehensive survey," *Eng Appl Artif Intell*, vol. 154, p. 110980, Aug. 2025, doi: 10.1016/J.ENGAPPAI.2025.110980.

7 Authors Biography

Muhammad Amri Yahya is a lecturer in the Department of Electronics Engineering at Politeknik Negeri Sriwijaya, with academic interests in power systems, energy forecasting, and applications of artificial intelligence in power system (email: muhammad.amri.yahya@polsri.ac.id).

Tegar Prasetyo is a lecturer in the Department of Electrical Engineering at Politeknik Negeri Sriwijaya, where his teaching and research explore applications of artificial intelligence in power systems (email: tegar.prasetyo@polsri.ac.id).

Muhammad Fikri is a lecturer in the Department of Electrical Engineering at Universitas Lampung. His academic focus spans control theory, with particular interest in dynamical systems, state estimation, and model reduction (email: laniakea@eng.unila.ac.id).