BAREKENG: Journal of Mathematics and Its Applications
March 2026 Volume 20 Issue 1 Page 0105–0122
P-ISSN: 1978-7227 E-ISSN: 2615-3017
https://doi.org/10.30598/barekengvol20iss1pp0105-0122

# COMPARATIVE STUDY OF LSTM-BASED MODELS WITH HYPERPARAMETER OPTIMIZATION FOR SHORT-TERM ELECTRICITY LOAD FORECASTING

**Iqbal Kharisudin[1*], Insyiraah Oxaichiko Arissinta[2], Sabrina Aziz Aulia[3], Muhamad Abdul Qodir Dani[4], Galih Kusuma Wijaya[5]**

[1,4,5]*Statistics and Data Science Study Program, Faculty of Mathematics and Natural Sciences, Universitas Negeri Semarang*
[2,3]*Mathematics Study Program, Faculty of Mathematics and Natural Sciences, Universitas Negeri Semarang*
*Kampus Sekaran, Gunungpati, Semarang, 50229, Indonesia*

*Corresponding author's e-mail: * **iqbalkharisudin@mail.unnes.ac.id***

***Article Info***

***Article History:***

*Received: 14th February 2025*
*Revised: 20th March 2025*
*Accepted: 17th June 2025*
*Available online: 24th November 2025*

***Keywords:***

*Bidirectional LSTM;*
*Deep learning modeling;*
*Electricity load;*
*Hyperparameter optimization;*
*Time series forecasting.*

***ABSTRACT***

*This research is focused on the development and comparison of time series models for short-term electrical load forecasting, utilizing several variants of Long Short-Term Memory (LSTM) networks. The specific LSTM variants employed in this study include Vanilla LSTM, Stacked LSTM, Bidirectional LSTM, and Convolutional Neural Network LSTM (CNN-LSTM). We used five years (2016-2020) of daily electricity load data from the Central Java-DIY system, provided by PT PLN (Persero). The primary objective is to ascertain the accuracy and evaluate the performance of these LSTM variants in the context of short-term load forecasting. This is achieved quantitatively through the computation of various error metrics, namely MSE, MAE, RMSE, MAPE, and R-squared. The results of the study reveal that the CNN-LSTM method outperforms the other variants in terms of the calculated metrics. Specifically, the CNN-LSTM method achieved the lowest values for all metrics: an MSE of 0.007 for training and 0.0010 for testing, an MAE of 0.0050 for training and 0.0062 for testing, and an RMSE of 0.083 for training and 0.099 for testing. Among the evaluated models, CNN-LSTM demonstrates the best trade-off between predictive accuracy and training efficiency, making it the most recommended for short-term electricity load forecasting. While BiLSTM achieves higher accuracy, particularly in terms of MAE, it requires a longer training time. In contrast, Stacked LSTM converges faster with slightly lower accuracy, making it a strong alternative when computational efficiency is prioritized..*



***How to cite this article:***

I. Kharisudin, I. O. Arissinta, S. A. Aulia, M. A. K. Dani, and G. K. Wijaya, "COMPARATIVE STUDY OF LSTM-BASED MODELS WITH HYPERPARAMETER OPTIMIZATION FOR SHORT-TERM ELECTRICITY LOAD FORECASTING," *BAREKENG: J. Math. & App.*, vol. 20, iss. 1, pp. 0105-0122, Mar, 2026.


Journal homepage: *https://ojs3.unpatti.ac.id/index.php/barekeng/*
Journal e-mail: *barekeng.math@yahoo.com; barekeng.journal@mail.unpatti.ac.id*
**Research Article · Open Access**

## 1. INTRODUCTION

Forecasting energy load is crucial for a nation's economic growth as it enables effective energy management and reduction in power consumption [1]. The demand for electricity generation, transmission, and distribution must be met by energy providers, which involves considerations of capital investment, efficient power procurement, capacity and network planning, fuel ordering planning, renewable planning, and optimal supply scheduling [2], [3]. The primary objective of energy load forecasting is to produce the most accurate results with minimal errors through proper planning. This can help reduce operational costs, enhance grid reliability, and increase financial profits. Accurate energy load forecasting can enable energy providers to save millions in energy procurement. The losses incurred due to energy conservation can be costly due to the expense of technologies used in energy storage. Accurate peak demand forecasting is essential; failure to do so could lead to underproduction of electricity, resulting in power outages. Conversely, overproduction could lead to energy wastage and increased costs. To optimize power plant operations, reduce operational costs, and improve grid operation reliability, a precise forecast of future grid load is necessary, along with proper scheduling and decision-making [4], [5]. Energy load forecasting can be categorized based on their time horizon into very short-term, short-term, medium-term, and long-term forecasting [6].

Electric load forecasts are classified into four temporal categories: very short-term, short-term, medium-term, and long-term [1], [7]. Very Short-Term Load Forecasting (VSTLF) encompasses a period from several minutes up to an hour ahead [8], while Short-Term Load Forecasting (STLF) extends from one hour up to one week ahead. In contrast, Medium-Term Load Forecasting (MTLF) and Long-Term Load Forecasting (LTLF) cover periods from several weeks to months and from one year to several years into the future, respectively [9]. Forecasts of electricity loads that significantly deviate from the actual loads impose additional costs on electricity suppliers in the market. Consequently, accurate 24-hour load forecasting is a primary concern for these providers. This study, therefore, focuses on forecasting the STLF [1] load for the subsequent 24 hours. Several studies have explored STLF utilizing various deep learning optimization methods, see e.g. [10], [11]. Additionally, the development and integration of fuzzy-based methods have also been conducted, such as [12] employing fuzzy support vector regressions and [13] combined fuzzy optimization model and load feature recognition [14].

Generally, accurate forecasting, particularly short-term forecasting, can significantly contribute to reducing operational costs, stabilizing power supply scheduling, coordinating load management efficiently, and enhancing the safety and security of power supply system's [15], [16], [17], [18]. With technological advancements and the incorporation of smart devices in a smart network environment, accuracy, rapid response, and intelligence have emerged as critical aspects of STLF [19]. Deep learning, which is a specific area within machine learning, has been receiving a lot of attention recently. This is largely due to its ability to manage and analyze large quantities of data and identify complex patterns within this data. When it comes to forecasting power load, deep learning provides innovative ways to model and predict electricity usage. This addresses many of the challenges that are often encountered with traditional forecasting methods [20], [21], [22].

Deep learning (DL), a branch of machine learning (ML), leverages deep, multi-layered artificial neural networks (ANNs) to enhance accuracy in a variety of tasks, including object detection, speech recognition, and language translation, among others. Due to the notable success of DL models in addressing both classification and regression challenges, there has been a growing trend toward exploring various DL architectures across numerous fields. One architecture has gained significant traction due to its effectiveness in capturing both long- and short-term dependencies in time series data, while requiring minimal feature engineering or preprocessing. This architecture is known as recurrent neural networks (RNNs).

RNNs are dynamic systems that efficiently utilize the temporal structure of input sequences, making them a powerful tool for time series forecasting. They are an evolution of feed-forward neural networks (FFNNs), with the key difference being that RNNs propagate information forward through time steps in a sample. The defining feature of RNNs is their memory gate, which allows the model to process sequential data by retaining prior inputs to predict future outputs. The model continuously updates its memory, also called the recurrent hidden state $h_t$, as it processes each time step.

However, standard RNNs are hindered by short-term memory limitations, making it difficult for them to effectively process long input sequences. The longer the input sequence, the less the model can learn from earlier data points. Additionally, RNNs face challenges when dealing with long sequential data due to the issues of exploding and vanishing gradients. Exploding gradients occur when excessively large importance

is assigned to weight matrices without justification, while vanishing gradients result in values that are too small, preventing further learning by the model [25], [26].

LSTM networks solve the short-term memory limitations of RNNs, where backpropagation errors tend to vanish or explode as they pass through many time steps [27]. To address this issue, gated units were introduced to regulate the flow of information. LSTM units have demonstrated successful performance across various domains for time series forecasting. Unlike RNNs, which are typically constrained to learning patterns over about 10-time steps, LSTM networks can model long-term dependencies, effectively learning from sequences with over 1000-time steps.

This study is designed to significantly contribute to the field of Short-Term Load Forecasting (STLF) by employing the Long Short-Term Memory (LSTM) method. LSTM, a multi-layered approach, can map an input sequence to a fixed-dimension vector and subsequently translate the target sequence from this vector. This method is integral to the iterative neural network model, barring the input sequence. LSTM is adept at addressing problems associated with long-term dependencies, which may arise due to the introduction of numerous short-term dependencies into the dataset. Furthermore, both LSTM and its bidirectional variant (BiLSTM) are theoretically suitable for short-term forecasting because they are capable of modeling complex temporal patterns, including non-linear dependencies and local fluctuations that are often present in short-term data. LSTM learns from past temporal dependencies effectively, while BiLSTM enhances predictive power by considering both past and future contexts through its dual-directional processing. This architectural strength allows these models to capture fine-grained dynamics over short horizons, making them particularly well-matched to the needs of STLF tasks [23], [24].

The primary aim of this research is to provide insights into the application of several LSTM variants, namely Vanilla LSTM, Stacked LSTM, Bidirectional LSTM, and CNN-LSTM, for the short-term forecasting of electrical loads. The findings of this study are expected to serve as a valuable resource for managerial decision-makers in STLF, aiding them in leveraging various LSTM variants to address STLF challenges.

## 2. RESEARCH METHODS

This research is an experimental study aimed at finding the best model for forecasting daily electricity usage data. The data used is daily electricity usage data over a period of 5 years (2016-2022), obtained from the State Electricity Company (PT PLN Persero) for the Central Java-DIY region. From the data obtained, selection, ranking, transformation, and encoding are performed as needed. The method used in this research experiment focuses on the use of deep learning, specifically LSTM and several of its variations, such as Stacked LSTM, Bidirectional LSTM, and CNN-LSTM. Each model undergoes experiments to find the optimal parameter combination to obtain the best model. Model evaluation is carried out using several evaluation metrics, such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Average Percentage Error (MAPE). The best-performing model(s) based on the evaluation metrics are then implemented for short-term electricity load forecasting. The final stage of the research involves drawing conclusions from the modeling analysis results, discussing the implications of these findings, and suggesting areas for future research in STLF using LSTM methods.

### 2.1 Vanilla LSTM

The vanilla LSTM network consists of three primary gates: the forget gate, the input gate, and the output gate. The forget gate determines which information should be retained, assigning a value between 0 and 1, with values near 0 being discarded and those near 1 being kept. The input gate controls what new information is allowed to enter the memory cell and be stored.

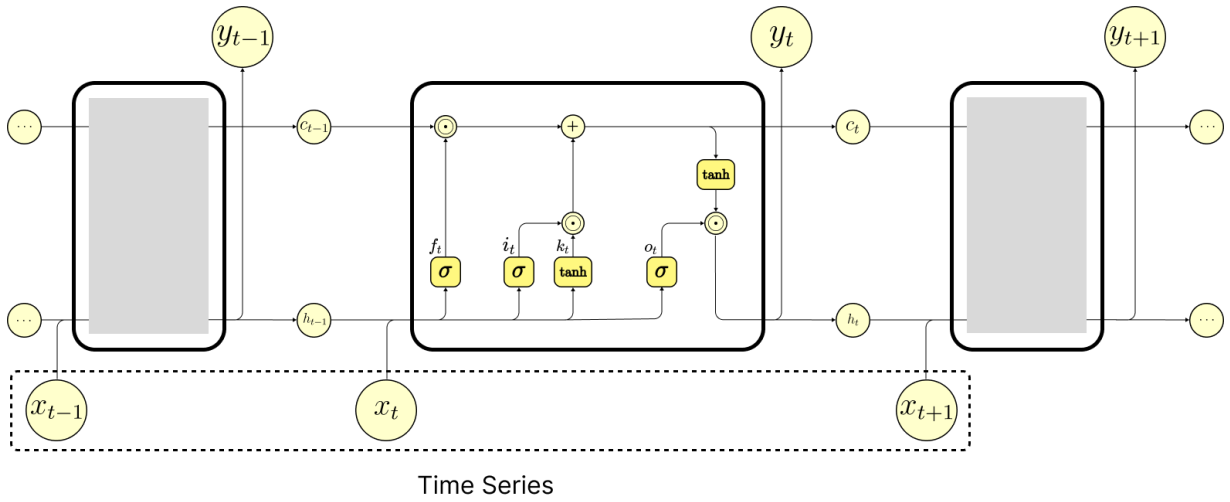



**Figure 1**. LSTM Architecture

This process is represented by Eqs. (2) and (3), where $k_t$ generates a vector of new values to be added to the memory cell, guided by the update signal $i_t$. The memory cell is then updated as described in Eq. (4), where $c_t$ (the long-term state) decides whether to retain or discard past information and incorporate new data. Finally, the output, $h_t$ (the short-term state), is produced by two layers. First, $o_t$ is the output determined by the $\sigma$ activation function, which selects which values from the memory cell are passed as output. This is followed by a tanh layer, which constrains the output within the range [−1, 1]. The process of updating the network is shown in Eqs. (1) to (6) and illustrated in Fig. 1.

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big), \tag{1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2}$$

$$k_t = \tanh(W_k \cdot [h_{t-1}, x_t] + b_k), \tag{3}$$

$$c_t = f_t \times c_{t-1} + i_t \times k_t, \tag{4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{5}$$

$$h_t = o_t \times \tanh(c_t), \tag{6}$$

where $f_t$, $i_t$, $k_t$, and $o_t$ represent the output values of the forget gate, input gate, update signal, and output gate, respectively. These gates receive input values $x_t$ at the current time step $t$ and the output value $h_{t-1}$ from the previous time step $(t-1)$. The corresponding weight matrices are denoted as $W_f$, $W_i$, $W_k$, and $W_o$, while the bias vectors are represented by $b_f$, $b_i$, $b_k$, and $b_o$ for each gate. The activation function $\sigma$ is nonlinear, and $c_t$ refers to the memory unit within the LSTM [25], [26], [27].

## 2.2 Stacked LSTM

A stacked LSTM is a neural network architecture that consists of multiple layers of LSTM units, arranged sequentially to increase the depth of the model and enhance its capacity to learn complex patterns from data [28]. Under a standard LSTM model, the layer would receive an input sequence and output a value or a specified number of values to be predicted. However, under a stacked model, as it is shown in Fig. 5, the outputs that can either correspond to each timestep or represent an aggregate prediction for the sequence, depending on the task's configuration and the network's structure [29]. In this way, the hidden states are a function of all previously hidden states. Consequently, the $i$-th layer can be updated by Eqs. (7) to (13).

$$f_t^l = \sigma\big(W_f^l \cdot \big[h_{t-1}^l, h_t^{l-1}\big] + b_f^l\big), \tag{7}$$

$$i_t^l = \sigma\big(W_i^l \cdot \big[h_{t-1}^l, h_t^{l-1}\big] + b_i^l\big), \tag{8}$$

$$k_t^l = \tanh\big(W_k^l \cdot \big[h_{t-1}^l, h_t^{l-1}\big] + b_k^l\big), \tag{9}$$

$$c_t^l = f_t^l \times c_{t-1}^l + i_t^l \times k_t^l, \tag{10}$$

$$o_t^l = \sigma\big(W_o^l \times \big[h_{t-1}^l, h_t^{l-1}\big] + b_o^l\big), \tag{11}$$

$$h_t^l = o_t^l \times \tanh\left(c_t^l\right), \tag{12}$$

$$h_t^0 = x_t \tag{13}$$

where the resulting output acts as an abstracted input representation, which is then passed as a hierarchical feature to the next LSTM layer. The final layer contains a set of neurons equal to the number of time steps the model aims to forecast. Studies highlight several benefits of this architecture, noting that stacking multiple layers enables the network to extract and refine features from the raw time series at different levels and moments. The model's parameters are distributed throughout the entire architecture, which helps speed up convergence and improves the tuning of the nonlinear transformations applied to the data.

### 2.3 Bidirectional LSTM

The Bidirectional Long Short-Term Memory (BiLSTM) network is an advanced variant of the LSTM architecture that captures long-term dependencies in both forward and backward temporal directions [30]. In short-term electricity load forecasting, where future load values depend on both past and recent load patterns, by leveraging the BiLSTM's dual-layer approach, the model can better capture complex temporal relationships and improve the accuracy of predictions, particularly for non-linear and fluctuating load trends [31].

While the traditional LSTM processes data sequentially from past to future, BiLSTM comprises two LSTM layers: one processes the input sequence from start to end (forward pass), and the other processes it from end to start (backward pass). By merging these two sequences, BiLSTM has access to a more complete context at every time step, allowing it to incorporate information from both previous and subsequent points. This is particularly advantageous for load forecasting, where fluctuations may correlate with both past and forthcoming load values, influenced by external factors like temperature or day of the week.

Each LSTM layer in the BiLSTM model follows the standard LSTM gating mechanisms, namely the forget, input, and output gates. Forget gate ($f_t$) regulates which portions of the previous cell state $c_{t-1}$ are retained or discarded. Input gate ($i_t$) selects the new information to be stored in the current cell state. Output gate ($o_t$) decides what part of the cell state contributes to the output at the current time step [31].

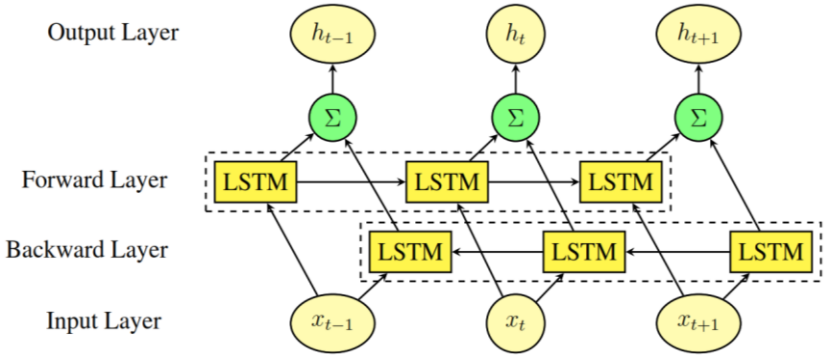



**Figure 2**. BiLSTM Architecture

For BiLSTM, these gates function independently in the forward and backward layers, which are later concatenated or averaged to produce a final output that reflects information from both directions. The combined effect of these gates enables the BiLSTM to learn more intricate patterns in the time series data, even when there are subtle shifts in electricity load trends. The mathematical framework for each direction in the BiLSTM is the same as the vanilla LSTM model Eqs. (1) to (6), consists of forget gate ($f_t$), input gate and update vector ($i_t$) and ($k_t$), memory cell update ($c_t$), output gate ($o_t$), and short-term state (final output, $h_t$). The outputs of the forward and backward layers are then concatenated or combined (e.g., averaged or summed) to produce a final output that encapsulates information from both directions. The mathematical framework for each direction in the BiLSTM is consistent with the vanilla LSTM equations. Let $\overrightarrow{h_t}$ represent the hidden state in the forward layer at time step $t$, and $\overleftarrow{h_t}$ represent the hidden state in the backward layer. The final output of the BiLSTM, denoted as $h_t^{\mathrm{Bi}}$, is computed as:

$$h_t^{\mathrm{Bi}} = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]. \tag{14}$$

The forward layer computes its states using the equations:

$$\overrightarrow{f_t} = \sigma\left(\overrightarrow{W_f} \cdot \left[\overrightarrow{h_{t-1}}, x_t\right] + \overrightarrow{b_f}\right), \tag{15}$$

$$\overrightarrow{i_t} = \sigma\left(\overrightarrow{W_i} \cdot \left[\overrightarrow{h_{t-1}}, x_t\right] + \overrightarrow{b_i}\right), \tag{16}$$

$$\overrightarrow{k_t} = \tanh\left(\overrightarrow{W_k} \cdot \left[\overrightarrow{h_{t-1}}, x_t\right] + \overrightarrow{b_k}\right), \tag{17}$$

$$\overrightarrow{c_t} = \overrightarrow{f_t} \times \overrightarrow{c_{t-1}} + \overrightarrow{i_t} \times \overrightarrow{k_t}, \tag{18}$$

$$\overrightarrow{o_t} = \sigma\left(\overrightarrow{W_o} \cdot \left[\overrightarrow{h_{t-1}}, x_t\right] + \overrightarrow{b_o}\right), \tag{19}$$

$$\overrightarrow{h_t} = \overrightarrow{o_t} \times \tanh(\overrightarrow{c_t}). \tag{20}$$

Similarly, the backward layer processes the sequence in reverse, with analogous equations for $\overleftarrow{f_t}$, $\overleftarrow{i_t}$, $\overleftarrow{k_t}$, $\overleftarrow{c_t}$, $\overleftarrow{o_t}$, and $\overleftarrow{h_t}$. By considering both past and future contexts simultaneously, BiLSTM are highly effective in capturing intricate dependencies in sequential data, such as subtle variations in electricity load trends across time. This bidirectional architecture is particularly beneficial for time series forecasting, where understanding both historical and potential future influences is critical.

### 2.4 CNN-LSTM

Convolutional Neural Network - Long Short-Term Memory (CNN-LSTM) is an extension of the traditional LSTM architecture designed to handle spatiotemporal data by integrating convolution operations into its structure. While conventional LSTMs employ fully connected layers to process input and hidden states, CNN-LSTM replaces these layers with convolutional operations, enabling it to capture spatial dependencies alongside temporal patterns [32]. Convolutional Neural Networks (CNNs) are primarily used for extracting features from data with a grid-like structure, such as images or spatially organized data. CNN architectures were originally designed for two-dimensional data, enabling the model to learn hierarchical patterns that are recognized across the entire network. This ability to capture spatial hierarchies made CNNs particularly effective in image processing. However, with the increasing application of sequence data such as text and time series, one-dimensional CNNs have gained popularity. These 1D CNNs adapt the original CNN framework to work with sequential data by learning local patterns along a single dimension, making them well-suited for tasks like time-series forecasting and natural language processing [33]. However, their performance tends to be less effective when compared to other models, such as Long Short-Term Memory (LSTM) networks. CNNs excel in feature extraction, especially for spatial data like images, they often struggle with sequential data where the order and context of inputs matter significantly [32]. CNNs typically consist of three main layers: the convolutional layer, which extracts features from the input data, the pooling layer, which reduces the spatial dimensions of the data while retaining essential features, and the fully connected layer, which makes predictions based on the learned features [34]. First, the input layer receives the vector of input values, which is then processed by the convolutional layer where filters (or kernels) extract relevant features before passing them to a fully connected layer. The basic architecture of CNNs is illustrated in Figure 6. An LSTM network can be attached to the dense or fully connected layer in this setup, allowing the resulting CNN–LSTM architecture to be updated similarly to a standard LSTM, with matrix additions and dot products replaced by convolutional operations. In this configuration, the CNN–LSTM integrates convolutional mechanisms into both the input-to-state and state-to-state transitions. The network is reformulated so that the operators originally defined in Eqs. (1) to (6) are substituted with the convolution operator ($*$) and the Hadamard product ($\odot$), as shown in Eqs. (15) to (19), where the latter preserves the constant-bias property of the cell. Compared to the traditional LSTM, these equations incorporate information from the cell state into all gate computations, enabling the model to maintain or discard information appropriately across the CNN–LSTM structure. Additionally, the weight matrices of each gate now employ convolutional operators to embed the filter behavior. The network is updated according to the following equations:

$$f_t = \sigma\left(W_f * x_t + U_f * h_{t-1} + b_f\right), \tag{21}$$

$$i_t = \sigma\left(W_f * x_t + U_i * h_{t-1} + b_i\right), \tag{22}$$

$$\tilde{c}_t = \tanh(W_c * x_t + U_c * h_{t-1} + b_c), \tag{23}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{24}$$

$$o_t = \sigma(W_o * x_t + U_o * h_{t-1} + b_o), \tag{25}$$

$$h_t = o_t \odot \tanh(C_t)\,. \tag{26}$$

where $x_t$ stands for input tensor at time step $t$; typically a multidimensional array (e.g., images or sequences); $h_t$ hidden state tensor at time $t$; $c_t$ cell state tensor at time $t$, $f_t, i_t$, $o_t$ are forget, input, and output gates, respectively; $\tilde{c}_t$ candidate cell state; $W$ and $U$ convolutional kernel weights; $b$ bias terms; $*$ convolution operator; $\odot$ element-wise multiplication; $\sigma$ sigmoid activation function; and $\tanh$ hyperbolic tangent function [25], [32].

### 2.5 Hyperparameter Optimization

Hyperparameter optimization is a critical step in training deep learning models, as it involves finding the optimal set of hyperparameters to maximize model performance. Optuna, a Python-based library for automatic hyperparameter optimization, has gained prominence due to its flexibility and efficiency. Optuna employs a sequential model-based optimization (SMBO) framework, where a surrogate model predicts the performance of hyperparameter combinations. Specifically, Optuna uses Tree-structured Parzen Estimator (TPE) as its default optimization algorithm. TPE builds two probability density functions: $l(x)$, which models hyperparameters leading to poor performance, and $g(x)$, which models hyperparameters resulting in good performance. The optimization problem is then formulated as:

$$x^* = \arg\max_x \frac{g(x)}{l(x)}. \tag{28}$$

This approach enables Optuna to prioritize hyperparameter regions with a higher likelihood of improved performance while avoiding exhaustive search. Optuna operates in three key stages: (1) objective function definition: the user defines an objective function that returns a performance metric (e.g., validation loss) for a given set of hyperparameters; (2) search space specification: optuna provides an intuitive API to define hyperparameter ranges (e.g., learning rate, number of units, dropout rates). These ranges can include discrete, continuous, or categorical variables; and (3) optimization: optuna iteratively samples hyperparameters, evaluates the objective function, and updates its surrogate model to improve future sampling.

In practice, Optuna works by trying out many combinations of hyperparameters sampled from predefined ranges (e.g., learning rate, number of hidden units, dropout rate, batch size), then training the model with each set on the training data. The model's performance, typically measured using a loss metric such as Mean Squared Error (MSE) on the validation set, is then used to inform the surrogate model in selecting the next, potentially better, combination of hyperparameters. This iterative process continues, with each trial refining the understanding of which regions in the hyperparameter space yield better model performance. In time series forecasting using LSTM and BiLSTM, such tuning is especially crucial due to the models' sensitivity to temporal patterns and parameter configurations. Since all optimization trials are based on training data and validation MSE, it is important that this process be clearly described in the Methods section—especially how the time series was split for validation—to ensure clarity, reproducibility, and that temporal dependencies are preserved during model selection.

## 3. RESULTS AND DISCUSSION

The plot of dataset consists of daily electricity usage records spanning five years (2016–2022) is illustrated in Figure 1. To ensure data suitability for analysis, preprocessing steps such as selection, ranking, transformation, and encoding are applied. The dataset is subsequently split into training and testing portions, with 80% used for training and the remaining 20% reserved for testing. This proportion ensures that the models are trained on a sufficiently large dataset while preserving a representative portion for performance evaluation.

The identification of multi-seasonal characteristics is evident from two boxplots presented in Fig. 3 and Fig. 4, which show monthly and daily electricity consumption patterns, respectively. The monthly boxplot reveals a recurring pattern similar to the daily plot, with a notable dip in electricity usage around late May to early June. This decline corresponds to the Idul Fitri holiday period, during which industries and schools in Indonesia temporarily close, leading to a significant reduction in electricity demand. On the daily

scale, a consistent weekly pattern emerges where electricity consumption decreases on Saturdays and Sundays, reflecting lower industrial and commercial activity during weekends.

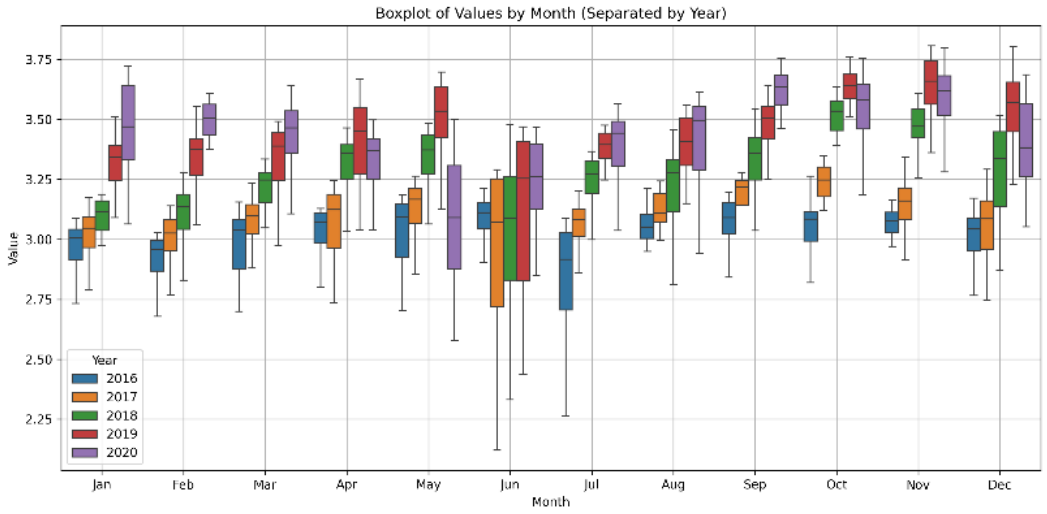



**Figure 3**. Monthly Electricity Load Distribution from 2016 to 2020

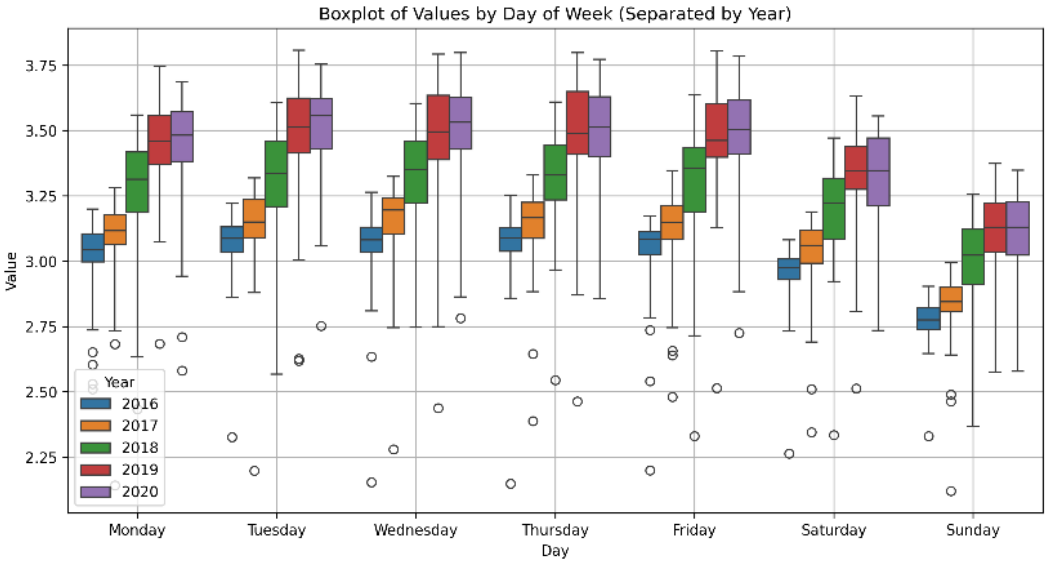



**Figure 4**. Daily Electricity Load Distribution by Day of The Week from 2016 to 2020

These observations highlight the presence of multi-seasonality in the electricity load data, characterized by both weekly and annual seasonal fluctuations. Such complex, overlapping seasonal patterns pose challenges for traditional forecasting models, which often struggle to capture multiple seasonality effectively. Consequently, deep learning approaches, particularly LSTM-based models, are well-suited for this task, as they can learn intricate temporal dependencies and multi-scale seasonal patterns inherent in electricity consumption data.

The study focuses on deep learning methodologies, specifically variations of Long Short-Term Memory (LSTM) models, including Vanilla LSTM, Stacked LSTM, Bidirectional LSTM (BiLSTM), and CNN-LSTM. These models are evaluated to determine the most accurate and efficient approach for short-term electricity load forecasting.

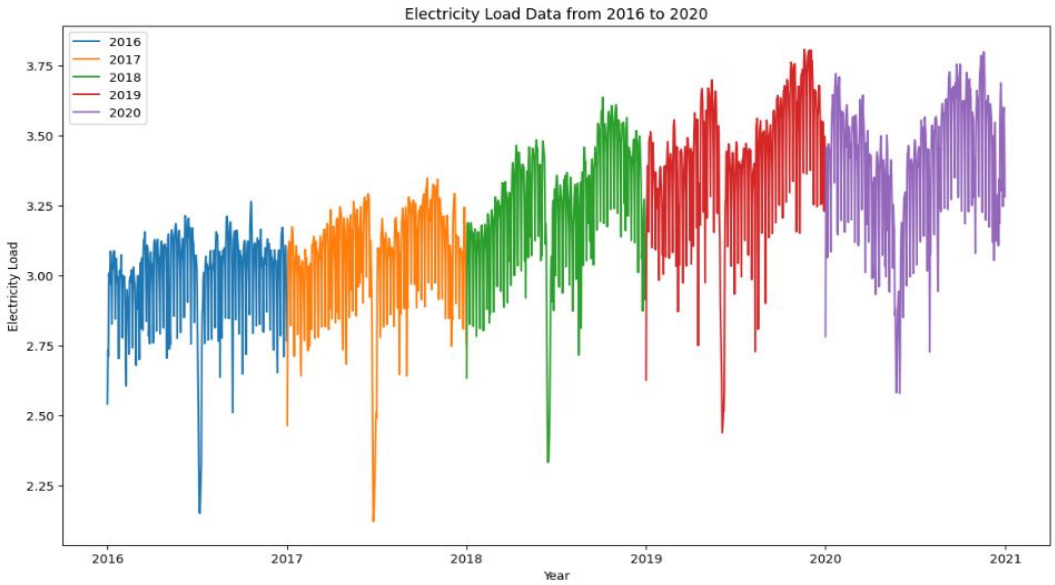



**Figure 5**. Electricity Load Data from 2016 to 2020

Each model undergoes rigorous experimentation to identify the optimal combination of hyperparameters for enhanced forecasting accuracy. The evaluation process employs multiple performance metrics, including MSE, RMSE, MAE, MAPE, and the coefficient of determination ($R^2$). These metrics provide a comprehensive assessment of model performance, ensuring the reliability and robustness of the forecasting results. Hyperparameter optimization is conducted iteratively using Optuna, where each trial involves training the model with a specific set of hyperparameters and evaluating its performance on validation data. The iteration process is governed by two termination criteria: (1) a maximum of 50 trials per study to manage computational efficiency, and (2) early stopping during model training if no improvement in validation performance is observed for 15 consecutive epochs. By systematically comparing different LSTM-based architectures, the study aims to determine the model best suited for capturing the complex temporal patterns in electricity consumption data.

Table 1 summarizes the parameter value ranges explored during the optimization of LSTM-based models using Optuna. Four model types are considered: Vanilla LSTM, Stacked LSTM, Bidirectional LSTM, and CNN-LSTM. To analyze different temporal dependencies, two-time step configurations, 7 and 30, are examined, corresponding to weekly and monthly patterns. The models differ in layer configurations: Vanilla LSTM employs a single layer, Stacked LSTM varies between 1 and 5 layers, Bidirectional LSTM integrates bidirectional layers with 1 to 5 additional LSTM layers, and CNN-LSTM combines 1 to 5 convolutional layers followed by 1 to 5 LSTM layers. The number of neurons in dense layers ranges from 35 to 256, while convolutional layers in CNN-LSTM utilize between 32 and 256 neurons. Activation functions tested include ReLU, tanh, and sigmoid, while dropout rates range from 0.1 to 0.5 to mitigate overfitting. Additionally, batch sizes between 16 and 256 are explored to optimize the training process. This extensive hyperparameter search ensures a thorough exploration of model configurations to achieve optimal forecasting performance.

**Table 1**. Parameter Value Range used in Optimization Optuna

| Model | Time step | Layers | Dense | Activation function | Dropout rate | Batch size |
|---|---|---|---|---|---|---|
| Vanilla LSTM | 7 & 30 | LSTM:1 layer | LSTM layer: 35 – 256 neurons | [relu, tanh, sigmoid] | 0.1 – 0.5 | 16 – 256 |
| Stacked LSTM | 7 & 30 | LSTM: 1-5 layers | LSTM layer: 35 – 256 neurons | [relu, tanh, sigmoid] | 0.1 – 0.5 | 16 – 256 |
| Bidirectional LSTM | 7 & 30 | Bidirectional LSTM: 1 – 5 layers | BiLSTM layer: 35 – 256 neurons | [relu, tanh, sigmoid] | 0.1 – 0.5 | 16 – 256 |
| CNN-LSTM | 7 & 30 | Convolutional 1d: 1-5 layers<br>LSTM: 1 – 5 layers | Convolutional layer: 32 – 256 neurons<br>LSTM layer: 32 – 256 neurons | [relu, tanh, sigmoid] | 0.1 – 0.5 | 16 – 256 |

Table 2 presents the optimal parameter values determined for each model after the optimization process, showcasing the best configurations for effective short-term electricity load forecasting. The Vanilla

LSTM model achieves its optimal performance with a single LSTM layer consisting of 236 neurons, utilizing the ReLU activation function, a dropout rate of 0.12, and a batch size of 25, all configured with a time step of 30. Similarly, the Stacked LSTM model performs best with two LSTM layers, containing 216 and 209 neurons, respectively, while employing the ReLU activation function, a higher dropout rate of 0.21, and a batch size of 16, also with a time step of 30. The Bidirectional LSTM model is optimized using two bidirectional LSTM layers, with 252 and 242 neurons, respectively, leveraging the tanh activation function, a dropout rate of 0.3, and a batch size of 17, maintaining a time step of 30.

In contrast, the CNN-LSTM model demonstrates an optimal structure with three convolutional layers containing 179, 91, and 220 neurons, respectively, followed by two LSTM layers with 106 and 104 neurons. This model utilizes the tanh activation function, a dropout rate of 0.1, and a batch size of 55, but with a notably shorter time step of 7. This distinction underscores the varying structural requirements across different architectures, where deeper recurrent models benefit from longer time steps, while hybrid models like CNN-LSTM leverage shorter sequences to capture spatial and temporal dependencies efficiently. These optimized configurations highlight the importance of tailoring hyperparameters to specific model architectures to achieve the best forecasting performance.

**Table 2**. Best Parameter Value from Each Model

| Model | Time step | Layers | Dense | Activation function | Dropout rate | Batch size |
|---|---|---|---|---|---|---|
| Vanilla LSTM | 30 | LSTM: 1 layer | LSTM layer 1:236 | relu | 0.12 | 25 |
| Stacked LSTM | 30 | LSTM: 2 layers | LSTM layer 1:216<br>LSTM layer 2:209 | relu | 0.21 | 16 |
| Bidirectional LSTM | 30 | Bidirectional LSTM: 2 layers | BiLSTM layer 1:252<br>BiLSTM layer 2:242 | tanh | 0.3 | 17 |
| CNN-LSTM | 7 | Convolutional 1d: 3 layers<br>LSTM: 2 layers | Convolutional Layer 1:179<br>Convolutional Layer 2:91<br>Convolutional Layer 3:220<br>LSTM layer 1:106<br>LSTM layer 2:104 | tanh | 0.10 | 55 |

Expanding on the previously discussed in Table 2, which outlined the best parameter values for each model at a broader scale, Table 3 specifically examines the optimal configurations when using a shorter time step of 7. This comparison provides insights into how model architectures adapt to a more immediate forecasting window. The Vanilla LSTM model achieves its best performance with a single LSTM layer comprising 167 neurons, employing the ReLU activation function, a dropout rate of 0.19, and a batch size of 16. The Stacked LSTM, designed to leverage deeper feature representations, performs optimally with two LSTM layers containing 86 and 34 neurons, respectively. This model utilizes the tanh activation function, a slightly higher dropout rate of 0.22, and a batch size of 19, indicating an adjustment in complexity to accommodate the shorter time horizon.

The Bidirectional LSTM, structured to capture temporal dependencies in both forward and backward directions, achieves its best results with a single bidirectional LSTM layer consisting of 167 neurons. This model uses the ReLU activation function, a lower dropout rate of 0.10, and a batch size of 27, suggesting a balance between depth and regularization. Finally, the CNN-LSTM model, which integrates convolutional layers for spatial feature extraction before sequential modeling, consists of three convolutional layers with 179, 91, and 220 neurons, followed by two LSTM layers with 106 and 104 neurons. It employs the tanh activation function, a dropout rate of 0.10, and a batch size of 55. These configurations highlight how each model adapts its structure and hyperparameters to effectively process shorter sequences, optimizing performance for short-term electricity load forecasting.

**Table 3**. Best Parameter Value from Each Model at Time Step 7

| Model | Layers | Dense | Activation function | Dropout rate | Batch size |
|---|---|---|---|---|---|
| Vanilla LSTM | LSTM: 1 layer | LSTM layer 1: 167 | relu | 0.19 | 16 |
| Stacked LSTM | LSTM: 2 layers | LSTM layer 1:86<br>LSTM layer 2:34 | tanh | 0.22 | 19 |
| Bidirectional LSTM | Bidirectional LSTM: 1 layer | BiLSTM layer 1: 167 | relu | 0.10 | 27 |

| Model | Layers | Dense | Activation function | Dropout rate | Batch size |
|---|---|---|---|---|---|
| CNN-LSTM | Convolutional 1d: 3 layers<br>LSTM: 2 layers | Convolutional Layer 1: 179<br>Convolutional Layer 2: 91<br>Convolutional Layer 3: 220<br>LSTM layer 1: 106<br>LSTM layer 2: 104 | tanh | 0.10 | 55 |

Table 4 presents the optimal model parameters for a time step of 30, showcasing the best configurations for each deep learning architecture. The Vanilla LSTM model achieves its best performance with a single LSTM layer comprising 236 neurons, utilizing the ReLU activation function, a dropout rate of 0.12, and a batch size of 25. In contrast, the Stacked LSTM, which leverages deeper network structures, performs optimally with two LSTM layers containing 216 and 209 neurons, respectively. This model also adopt the ReLU activation function but with a higher dropout rate of 0.21 and a batch size of 16. The Bidirectional LSTM, designed to capture dependencies in both forward and backward directions, consists of two bidirectional LSTM layers with 252 and 242 neurons, respectively, utilizing the tanh activation function, a dropout rate of 0.3, and a batch size of 17.

**Table 4**. Best Parameter Value from Each Model at Time Step 30

| Model | Layers | Dense | Activation function | Dropout rate | Batch size |
|---|---|---|---|---|---|
| Vanilla LSTM | LSTM: 1 layer | LSTM layer 1:236 | relu | 0.12 | 25 |
| Stacked LSTM | LSTM: 2 layers | LSTM layer 1:216<br>LSTM layer 2:209 | relu | 0.21 | 16 |
| Bidirectional LSTM | Bidirectional LSTM: 2 layers | BiLSTM layer 1:252<br>BiLSTM layer 2:242 | tanh | 0.3 | 17 |
| CNN-LSTM | Convolutional 1d: 2 layers<br>LSTM: 3 layers | Convolutional Layer 1:60<br>Convolutional Layer 2:185<br>LSTM layer 1:57<br>LSTM layer 2:192<br>LSTM layer 3:172 | tanh | 0.20 | 85 |

Meanwhile, the CNN-LSTM model integrates convolutional layers to extract spatial-temporal features before feeding the data into the LSTM layers. Its optimal configuration consists of two convolutional layers with 60 and 185 neurons, followed by three LSTM layers with 57, 192, and 172 neurons. This model employs the tanh activation function, a dropout rate of 0.20, and a batch size of 85. These diverse architectures and parameter settings demonstrate the distinct strategies used to optimize each model for short-term electricity load forecasting. By fine-tuning the number of layers, neuron counts, activation functions, dropout rates, and batch sizes, the study aims to determine the most effective deep learning approach for capturing complex temporal dependencies in electricity demand.

**Table 5**. Number of Epochs for Training the Best Model

| Model | Epochs | |
|---|---|---|
| | Time steps 7 | Time steps 30 |
| Vanilla LSTM | 121 | 105 |
| Stacked LSTM | 41 | 87 |
| Bidirectional LSTM | 123 | 104 |
| CNN-LSTM | 54 | 88 |

Following the discussion on model parameters, the next section focuses on the training and evaluation of the models. The training process is analyzed in terms of the number of epochs required to achieve optimal performance, as summarized in Table 5. This table presents the training durations for each model at both time steps 7 and 30, offering insights into the computational demands of different architectures. The number of epochs required varies depending on the complexity of the model and the selected time step. For example, the Vanilla LSTM model reaches optimal performance after 121 epochs for time step 7 and 105 epochs for time step 30, whereas the Stacked LSTM model requires 41 epochs at time step 7 and 87 epochs at time step 30.

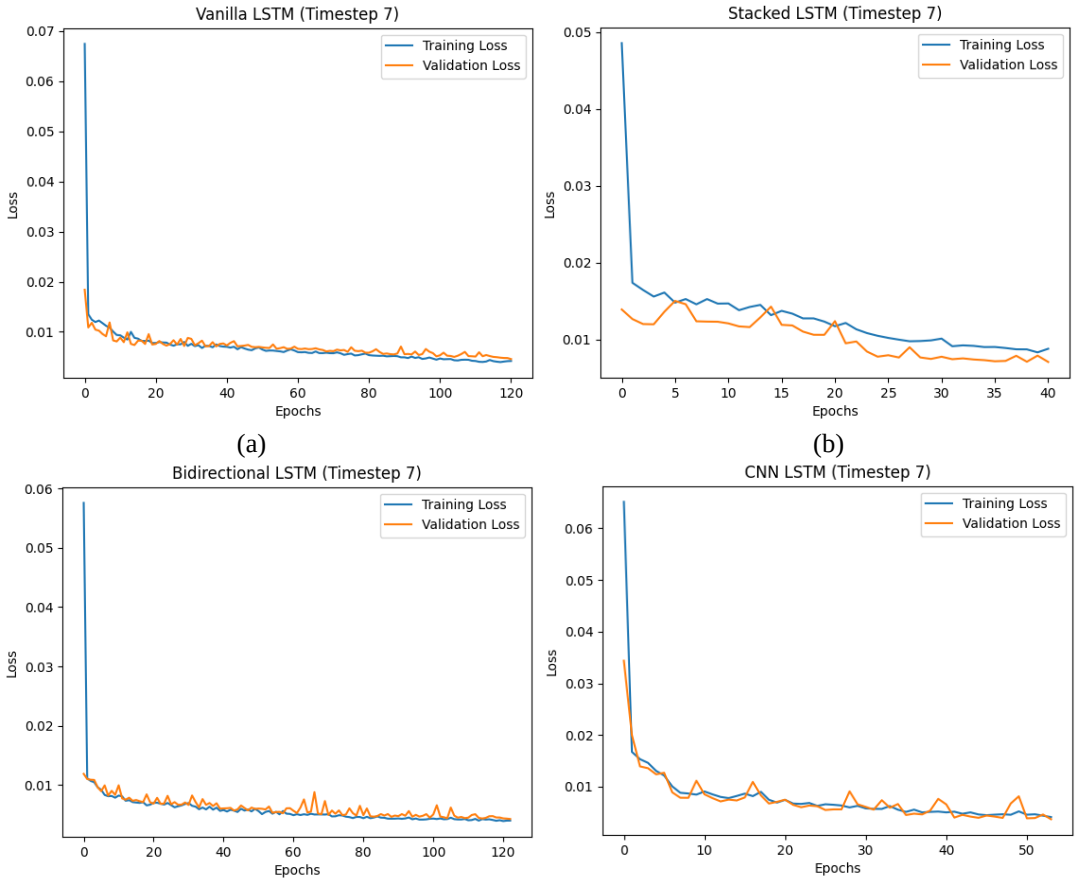
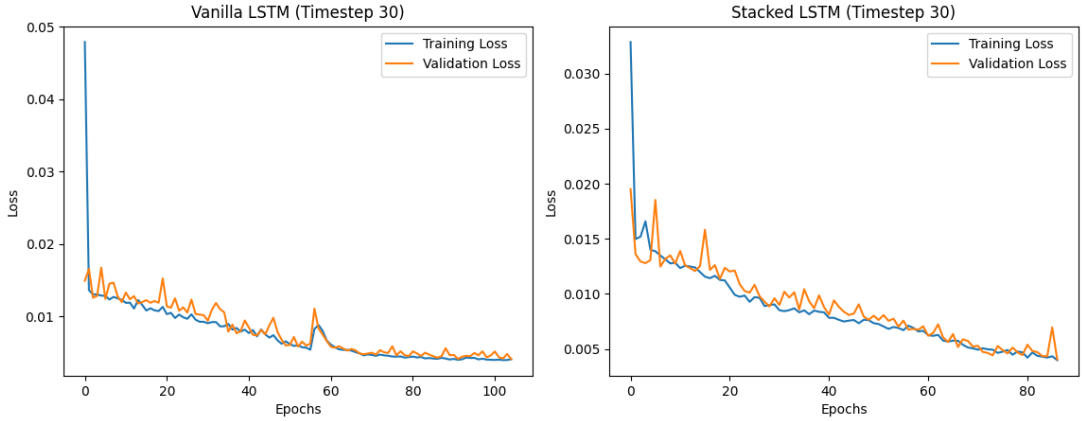
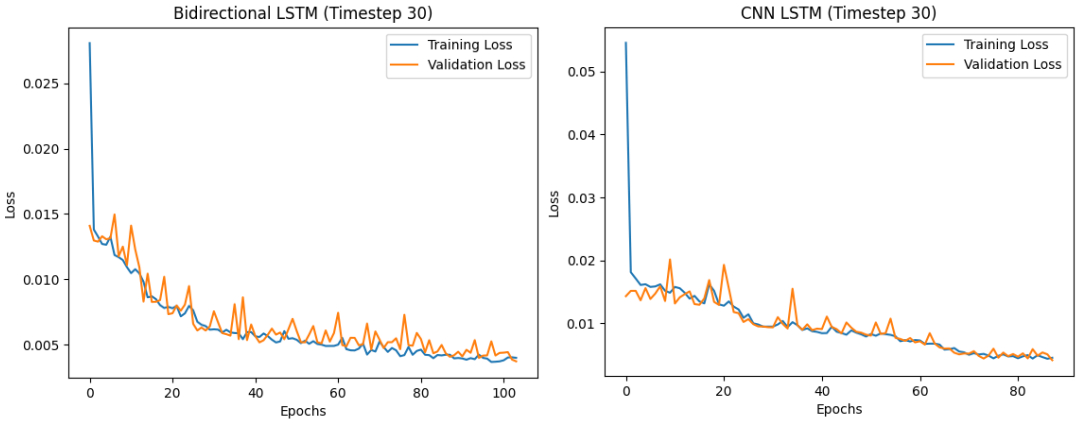



**Figure 6**. Training Loss Curves for Each Model at Time Step 7, (a) Vanilla LSTM, (b) Stacked LSTM, (c) Bidirectional LSTM, (d) CNN-LSTM

Similarly, the Bidirectional LSTM model demands 123 epochs at time step 7 and 104 epochs at time step 30, reflecting its bidirectional processing complexity. In contrast, the CNN-LSTM model exhibits a smaller variance in training duration, requiring 54 epochs for time step 7 and 88 epochs for time step 30. These variations in the number of epochs highlight differences in model convergence rates and training dynamics. Models with deeper architectures or more complex dependencies, such as the Bidirectional LSTM, generally require more epochs to learn long-term dependencies effectively. Fig. 6 and Fig. 7 illustrate the training progress for the time steps 7 and 30, respectively, providing a visual representation of model convergence trends.






(c) (d)

**Figure 7**. Training Loss Curves for Each Model at Time Step 30, (a) Vanilla LSTM, (b) Stacked LSTM, (c) Bidirectional LSTM, (d) CNN-LSTM

Table 6 provides a comprehensive evaluation of the four LSTM-based models by analyzing multiple performance metrics, including MSE, MAE, RMSE, MAPE, and the coefficient of determination ($R^2$).

**Table 6**. Best Evaluation Value from Each Model

| Model | MSE | | MAE | | RMSE | | MAPE | | R2 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test | Train | Test | Train | Test |
| Vanilla LSTM | 0.008 | 0.011 | 0.052 | 0.065 | 0.089 | 0.104 | 1.71% | 1.99% | 88.49% | 80.38% |
| Stacked LSTM | 0.007 | 0.011 | 0.049 | 0.062 | 0.086 | 0.103 | 1.62% | 1.90% | 89.31% | 80.68% |
| Bidirectional LSTM | 0.008 | 0.010 | 0.051 | 0.058 | 0.089 | 0.100 | 1.68% | 1.79% | 88.50% | 81.68% |
| CNN-LSTM | 0.007 | 0.010 | 0.050 | 0.062 | 0.083 | 0.099 | 1.62% | 1.88% | 89.98% | 81.90% |

These metrics serve as critical indicators of the models' accuracy, reliability, and ability to generalize unseen data. Lower values of MSE, MAE, RMSE, and MAPE suggest higher predictive accuracy, while an $R^2$ value approaching 1 signifies a stronger alignment between predicted and observed outcomes. The comparison of these metrics across both training and testing datasets helps assess the effectiveness of each model under different time step configurations. Fig. 8 and Fig. 9 provide a visual representation of these performance comparisons for time steps 7 and 30, respectively.

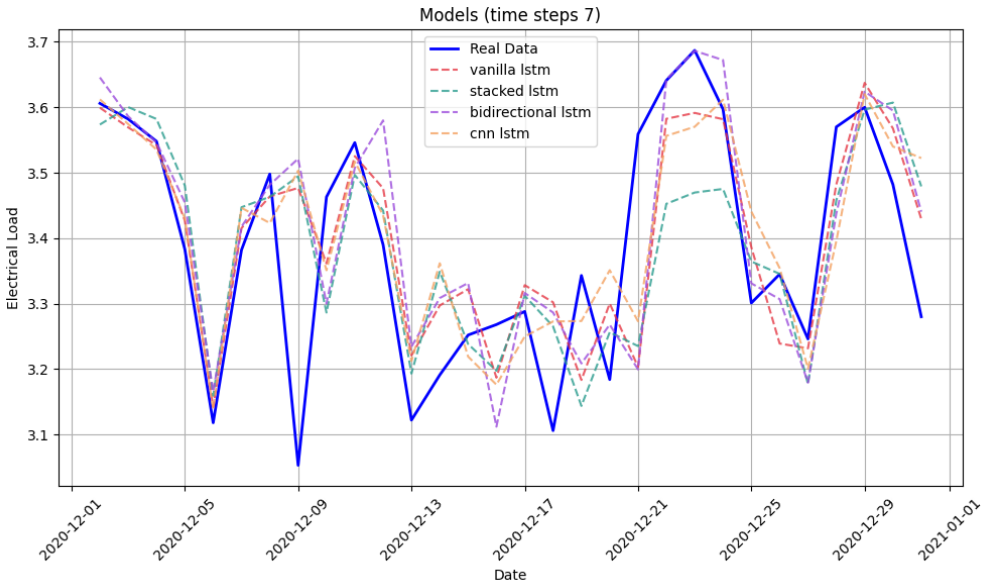



**Figure 8**. Forecast comparison of the four models at time step 7, predicting the last 30 days (1 month)

Among the evaluated models, the CNN-LSTM and Bidirectional LSTM architectures demonstrate superior performance in terms of minimizing prediction errors, as reflected in their lowest test MSE values

of 0.010. This suggests that these hybrid architectures effectively capture temporal dependencies and patterns in the electricity load data. Meanwhile, the Vanilla LSTM and Stacked LSTM models produce slightly higher test MSE values of 0.01, indicating that while they remain competitive, their predictive accuracy is marginally lower. Although these variations are minimal, they highlight the nuanced differences in each model's capacity to learn and generalize from the dataset. The results suggest that leveraging convolutional layers in combination with LSTM, as seen in CNN-LSTM, or employing bidirectional architectures, as in Bidirectional LSTM, may contribute to better forecasting performance by capturing both past and future contextual information, thereby enhancing the model's ability to learn from temporal dependencies in both directions.

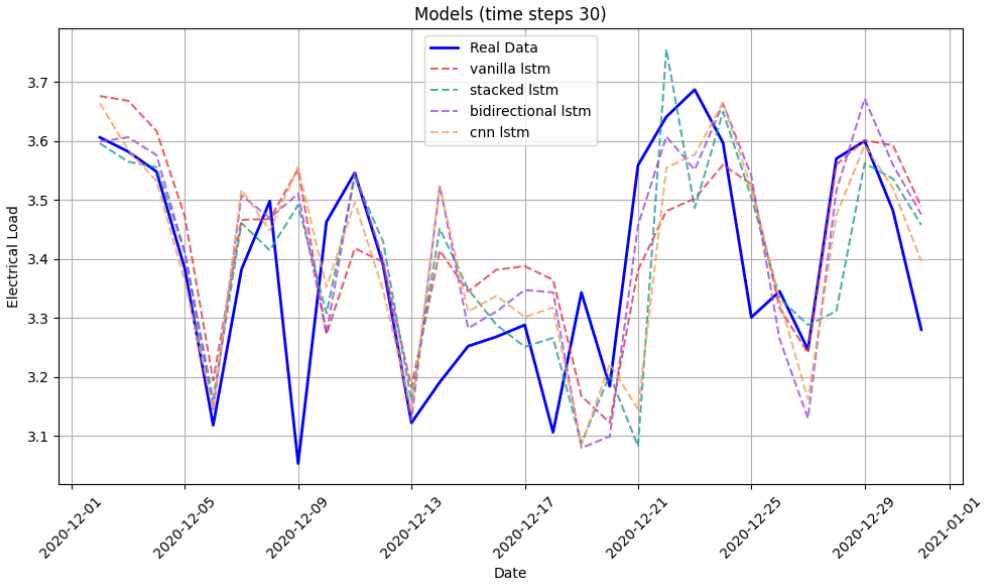



**Figure 9**. Forecast Comparison of the Four Models at Time Step 30, Predicting the Last 30 Days (1 Month)

When evaluated using the Mean Absolute Error (MAE), the CNN-LSTM, Stacked LSTM, and Bidirectional LSTM models achieve the lowest test MAE values of 0.062, 0.062, and 0.058, respectively. These low MAE values indicate that the models yield the smallest average absolute deviations from the actual values, suggesting a high level of accuracy in capturing the underlying patterns in electricity consumption data. In contrast, the Vanilla LSTM model yield slightly higher test MAE values of 0.065. Although the differences in MAE among the models are relatively minor, they highlight subtle variations in performance. Interestingly, the Bidirectional LSTM achieves a slightly lower MAE than the CNN-LSTM and Stacked LSTM models, suggesting its potential advantage in leveraging bidirectional dependencies within the data. However, despite this marginal improvement in absolute error, the CNN-LSTM and Stacked LSTM models offer a more balanced trade-off between model complexity and predictive accuracy, making them particularly suitable for scenarios where reducing absolute error is crucial.

For RMSE, a metric that penalizes larger errors more heavily, the CNN-LSTM and Bidirectional LSTM models again demonstrate superior performance with lower RMSE values of 0.099 and 0.100, respectively. These results indicate that these architectures are more effective at mitigating large deviations in predictions, which is essential for practical forecasting applications where minimizing extreme errors is critical. Meanwhile, the Vanilla LSTM and Bidirectional LSTM models record RMSE values of 0.104 and 0.103, showing slightly higher error magnitudes. Similarly, in terms of Mean Absolute Percentage Error (MAPE), which measures prediction accuracy in percentage terms, the CNN-LSTM and Bidirectional LSTM models achieve the lowest test MAPE values of 1.88% and 179, respectively, signifying their robustness in providing precise percentage-based predictions. In comparison, the Vanilla LSTM and Stacked LSTM models have slightly higher MAPE values of 1.99% and 1.90%, respectively, indicating a relatively lower degree of precision. These results suggest that CNN-LSTM and Stacked LSTM models not only maintain lower absolute and squared errors but also demonstrate higher accuracy in relative terms, reinforcing their suitability for time series forecasting tasks.

The coefficient of determination ($R^2$) values further highlight the effectiveness of the CNN-LSTM and Bidirectional LSTM models in capturing variance within the time series data. The CNN-LSTM achieves the highest $R^2$ value of 81.90% on the test set, followed closely by the Bidirectional LSTM at 81.68%. These

results suggest that both models can explain a substantial portion of the variability in the data, making them valuable tools for understanding complex temporal patterns. Meanwhile, the Stacked LSTM and Vanilla LSTM exhibit slightly lower $R^2$ values of 80.68% and 80.38%, respectively, indicating a marginally reduced ability to capture data variance. Although the differences are small, they reinforce the notion that CNN-LSTM and Bidirectional LSTM have a slight edge in predictive capability.

In terms of training performance, CNN-LSTM and Stacked LSTM demonstrate superior generalization, as evidenced by their lower MSE, MAE, and RMSE values on both training and test sets. efficiency of training is another critical factor, with CNN-LSTM and Stacked LSTM requiring significantly fewer epochs to converge. For time step 7, CNN-LSTM and Stacked LSTM converges in just 54 and 41 epochs, respectively, whereas Vanilla LSTM and Bidirectional LSTM take 121 and 123 epochs, respectively. A similar trend is observed at time step 30, where CNN-LSTM and Stacked LSTM require 88 and 87 epochs, compared to Vanilla LSTM and Bidirectional LSTM that require 105 and 104 epochs, respectively. This efficiency makes CNN-LSTM and Stacked LSTM more suitable for large-scale applications, reducing computational costs without sacrificing accuracy.

In terms of predictive accuracy, the Bidirectional LSTM and CNN-LSTM models consistently outperform the other architectures across multiple evaluation metrics, including MSE, MAE, RMSE, MAPE, and $R^2$. Bidirectional LSTM achieves the lowest MAE (0.058) and highly competitive scores in MSE (0.010) and RMSE (0.100), demonstrating strong capability in modeling temporal dependencies in both forward and backward directions. CNN-LSTM, on the other hand, achieves the highest $R^2$ value (81.90%) and the lowest RMSE (0.099), indicating its strength in minimizing large prediction errors and capturing overall data variance. While Stacked LSTM and Vanilla LSTM also perform reasonably well, their slightly higher error values and lower $R^2$ scores suggest comparatively reduced effectiveness in learning complex temporal dynamics. These results confirm that Bidirectional LSTM and CNN-LSTM offer superior forecasting accuracy, making them more reliable for short-term electricity load prediction tasks.

## 4. CONCLUSION

This study comprehensively evaluates the performance of four LSTM-based models—Vanilla LSTM, Stacked LSTM, Bidirectional LSTM (BiLSTM), and CNN-LSTM—for daily electricity load forecasting. By employing extensive hyperparameter optimization, the study ensures that each model operates under its best possible configuration, providing a fair comparison of their predictive capabilities. The findings reveal that, of all the models evaluated, CNN-LSTM demonstrates the highest predictive accuracy, consistently achieving the lowest error metrics (MSE, MAE, RMSE, MAPE) and the highest $R^2$ score. This indicates its strong capability in capturing complex temporal patterns in electricity load data. Although slightly less accurate, Stacked LSTM emerges as the most efficient model, requiring significantly fewer training epochs to converge, making it well-suited for large-scale applications where computational efficiency is critical. Bidirectional LSTM offers higher accuracy than Stacked LSTM, particularly in MAE, but demands longer training time due to its bidirectional structure. In contrast, Vanilla LSTM performs the weakest in both accuracy and training efficiency, making it the least suitable for time series forecasting in this context. Overall, CNN-LSTM is the most recommended model, striking a strong balance between predictive performance and training efficiency, while Stacked LSTM remains a competitive alternative when computational resources are limited.

Despite these findings, the study has several limitations. First, the time step configurations used in model training are fixed and may not fully capture all temporal dependencies present in the data. Second, the hyperparameter search space, although carefully selected, is bounded and may exclude potentially better configurations outside the defined range. Third, the evaluation is limited to four specific LSTM-based architectures; other advanced deep learning models such as Transformers or hybrid models incorporating exogenous variables are not considered. These limitations suggest directions for future research, including broader model comparisons, adaptive time step strategies, and expanded hyperparameter tuning.

## Author Contributions

Iqbal Kharisudin: Conceptualization, Funding Acquisition, Methodology, Writing - Review and Editing. Insyiraah Oxaichiko Arissinta: Data Curation, Writing - Original Draft. Sabrina Aziz Aulia: Formal Analysis, Investigation,

Validation. Muhamad Abdul Qodir Dani: Visualization. Galih Kusuma Wijaya: Software. All authors discussed the results and contributed to the final manuscript.

## Funding Statement

This research was supported by research funding from the Faculty of Mathematics and Natural Sciences, Universitas Negeri Semarang.

## Acknowledgment

The authors would like to express their sincere gratitude to all participants of the Konferensi Nasional Matematika (KNM) XXII, held at Universitas Bengkulu on July 15-16, 2024. The insightful discussions, constructive feedback, and active engagement have significantly enriched this study. We deeply appreciate the opportunity to present and exchange ideas in such a vibrant academic forum. We also extend our deepest appreciation to PLN Jawa Tengah and D.I. Yogyakarta for their invaluable contribution and support in providing the data essential for this research.

## Declarations

The authors declare no conflict of interest.

## REFERENCES

[1] W. Holderbaum, F. Alasali, and A. Sinha, *ENERGY FORECASTING AND CONTROL METHODS FOR ENERGY STORAGE SYSTEMS IN DISTRIBUTION NETWORKS*, vol. 85. Cham, Switzerland: Springer, 2023. doi: https://doi.org/10.1007/978-3-030-82848-6.

[2] I. K. Nti, M. Teimeh, O. Nyarko-Boateng, and A. F. Adekoya, "ELECTRICITY LOAD FORECASTING: A SYSTEMATIC REVIEW," *Journal of Electrical Systems and Information Technology*, vol. 7, no. 1, pp. 1–19, Dec. 2020, doi: https://doi.org/10.1186/s43067-020-00021-8.

[3] R. Chandrasekaran and S. K. Paramasivan, "ADVANCES IN DEEP LEARNING TECHNIQUES FOR SHORT-TERM ENERGY LOAD FORECASTING APPLICATIONS: A REVIEW," *Archives of Computational Methods in Engineering 2024*, pp. 1–30, Jun. 2024, doi: https://doi.org/10.1007/s11831-024-10155-x.

[4] Z. Guo, K. Zhou, X. Zhang, and S. Yang, "A DEEP LEARNING MODEL FOR SHORT-TERM POWER LOAD AND PROBABILITY DENSITY FORECASTING," *Energy*, vol. 160, pp. 1186–1200, Oct. 2018, doi: https://doi.org/10.1016/j.energy.2018.07.090.

[5] C. Tian, J. Ma, C. Zhang, and P. Zhan, "A DEEP NEURAL NETWORK MODEL FOR SHORT-TERM LOAD FORECAST BASED ON LONG SHORT-TERM MEMORY NETWORK AND CONVOLUTIONAL NEURAL NETWORK," *Energies (Basel)*, vol. 11, no. 12, p. 3493, Dec. 2018, doi: https://doi.org/10.3390/en11123493.

[6] P. H. Kuo and C. J. Huang, "A HIGH PRECISION ARTIFICIAL NEURAL NETWORKS MODEL FOR SHORT-TERM ENERGY LOAD FORECASTING," *Energies 2018*, vol. 11, no. 1, p. 213, Jan. 2018, doi: https://doi.org/10.3390/en11010213.

[7] J. Zheng, X. Chen, K. Yu, L. Gan, Y. Wang, and K. Wang, "SHORT-TERM POWER LOAD FORECASTING OF RESIDENTIAL COMMUNITY BASED ON GRU NEURAL NETWORK," *2018 International Conference on Power System Technology, POWERCON 2018 - Proceedings*, pp. 4862–4868, Jul. 2018, doi: https://doi.org/10.1109/POWERCON.2018.8601718.

[8] H. Choi, S. Ryu, and H. Kim, "SHORT-TERM LOAD FORECASTING BASED ON RESNET AND LSTM," *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, SmartGridComm 2018*, Dec. 2018, doi: https://doi.org/10.1109/SmartGridComm.2018.8587554.

[9] 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: https://doi.org/10.1109/ICCNC.2017.7876196

[10] N. A. Nguyen, T. D. Dang, E. Verdú, and V. Kumar Solanki, "SHORT-TERM FORECASTING ELECTRICITY LOAD BY LONG SHORT-TERM MEMORY AND REINFORCEMENT LEARNING FOR OPTIMIZATION OF HYPER-PARAMETERS," *Evol Intell*, vol. 16, no. 5, pp. 1729–1746, Oct. 2023, doi: https://doi.org/10.1007/s12065-023-00869-5.

[11] E. Uwimana, Y. Zhou, and N. M. Sall, "A SHORT-TERM LOAD DEMAND FORECASTING: LEVENBERG–MARQUARDT (LM), BAYESIAN REGULARIZATION (BR), AND SCALED CONJUGATE GRADIENT (SCG) OPTIMIZATION ALGORITHM ANALYSIS," *Journal of Supercomputing*, vol. 81, no. 1, pp. 1–30, Jan. 2025, doi: https://doi.org/10.1007/s11227-024-06513-y

[12] J. Luo, Y. Zheng, T. Hong, A. Luo, and X. Yang, "FUZZY SUPPORT VECTOR REGRESSIONS FOR SHORT-TERM LOAD FORECASTING," *Fuzzy Optimization and Decision Making*, vol. 23, no. 3, pp. 363–385, Sep. 2024, doi: https://doi.org/10.1007/s10700-024-09425-x.

[13] Y. Xie *et al.*, "SHORT-TERM LOAD FORECASTING METHOD BASED ON FUZZY OPTIMIZATION COMBINED MODEL OF LOAD FEATURE RECOGNITION," *Electrical Engineering*, pp. 1–14, Jun. 2024, doi: https://doi.org/10.21203/rs.3.rs-3836587/v1.

[14] Nirwanto, S. Bahri, and L. Harsyiah, "COMPARATIVE ANALYSIS OF FUZZY TIME SERIES CHEN AND MARKOV CHAIN METHODS FOR FORECASTING ELECTRICITY CONSUMPTION IN MATARAM CITY," *BAREKENG: Jurnal Ilmu Matematika dan Terapan*, vol. 18, no. 4, pp. 2375–2386, Oct. 2024, doi: https://doi.org/10.30598/barekengvol18iss4pp2375-2386.

[15] S. Bahrami, R. A. Hooshmand, and M. Parastegari, "SHORT TERM ELECTRIC LOAD FORECASTING BY WAVELET TRANSFORM AND GREY MODEL IMPROVED BY PSO (PARTICLE SWARM OPTIMIZATION) ALGORITHM," *Energy*, vol. 72, pp. 434–442, Aug. 2014, doi: https://doi.org/10.1016/j.energy.2014.05.065.

[16] M. Djukanovic, S. Ruzic, B. Babic, D. J. Sobajic, and Y. H. Pao, "A NEURAL-NET BASED SHORT TERM LOAD FORECASTING USING MOVING WINDOW PROCEDURE," *International Journal of Electrical Power & Energy Systems*, vol. 17, no. 6, pp. 391–397, Dec. 1995, doi: https://doi.org/10.1016/0142-0615(94)00009-3

[17] P. J. Santos, A. G. Martins, and A. J. Pires, "DESIGNING THE INPUT VECTOR TO ANN-BASED MODELS FOR SHORT-TERM LOAD FORECAST IN ELECTRICITY DISTRIBUTION SYSTEMS," *International Journal of Electrical Power & Energy Systems*, vol. 29, no. 4, pp. 338–347, May 2007, doi: https://doi.org/10.1016/j.ijepes.2006.09.002.

[18] G. Sudheer and A. Suseelatha, "SHORT TERM LOAD FORECASTING USING WAVELET TRANSFORM COMBINED WITH HOLT–WINTERS AND WEIGHTED NEAREST NEIGHBOR MODELS," *International Journal of Electrical Power & Energy Systems*, vol. 64, pp. 340–346, Jan. 2015, doi: https://doi.org/10.1016/j.ijepes.2014.07.043.

[19] J. Wang, J. Gao, and D. Wei, "ELECTRIC LOAD PREDICTION BASED ON A NOVEL COMBINED INTERVAL FORECASTING SYSTEM," *Appl Energy*, vol. 322, p. 119420, Sep. 2022, doi: https://doi.org/10.1016/j.apenergy.2022.119420.

[20] A. Haque and S. Rahman, "SHORT-TERM ELECTRICAL LOAD FORECASTING THROUGH HEURISTIC CONFIGURATION OF REGULARIZED DEEP NEURAL NETWORK," *Appl Soft Comput*, vol. 122, p. 108877, Jun. 2022, doi: https://doi.org/10.1016/j.asoc.2022.108877.

[21] I. Yazici, O. F. Beyca, and D. Delen, "DEEP-LEARNING-BASED SHORT-TERM ELECTRICITY LOAD FORECASTING: A REAL CASE APPLICATION," *Eng Appl Artif Intell*, vol. 109, p. 104645, Mar. 2022, doi: https://doi.org/10.1016/j.engappai.2021.104645.

[22] X. Wen, J. Liao, Q. Niu, N. Shen, and Y. Bao, "DEEP LEARNING-DRIVEN HYBRID MODEL FOR SHORT-TERM LOAD FORECASTING AND SMART GRID INFORMATION MANAGEMENT," *Sci Rep*, vol. 14, no. 1, pp. 1–16, Dec. 2024, doi: https://doi.org/10.1038/s41598-024-63262-x.

[23] J. Bedi and D. Toshniwal, "DEEP LEARNING FRAMEWORK TO FORECAST ELECTRICITY DEMAND," *Appl Energy*, vol. 238, pp. 1312–1326, Mar. 2019, doi: https://doi.org/10.1016/j.apenergy.2019.01.113.

[24] E. Mora, J. Cifuentes, and G. Marulanda, "SHORT-TERM FORECASTING OF WIND ENERGY: A COMPARISON OF DEEP LEARNING FRAMEWORKS," *Energies (Basel)*, vol. 14, no. 23, p. 7943, Dec. 2021, doi: https://doi.org/10.3390/en14237943.

[25] I. Goodfellow, *DEEP LEARNING*. MIT Press, 2016.

[26] A. Graves, A. Mohamed, and G. Hinton, "SPEECH RECOGNITION WITH DEEP RECURRENT NEURAL NETWORKS," in *2013 IEEE international conference on acoustics, speech and signal processing*, 2013, pp. 6645–6649.doi: https://doi.org/10.1109/ICASSP.2013.6638947

[27] "1997 - SEPP HOCHREITER DAN JÜRGEN SCHMIDHUBER - LSTM".

[28] C. Li *et al.*, "LONG SHORT-TERM MEMORY NETWORKS IN MEMRISTOR CROSSBAR ARRAYS," *Nat Mach Intell*, vol. 1, no. 1, pp. 49–57, 2019.doi: https://doi.org/10.1038/s42256-018-0001-4

[29] S. Hochreiter, "LONG SHORT-TERM MEMORY," *Neural Computation MIT-Press*, 1997, doi: https://doi.org/10.1162/neco.1997.9.8.1735

[30] I. R. Siregar *et al.*, "THE COMPARISON OF LONG SHORT-TERM MEMORY AND BIDIRECTIONAL LONG SHORT-TERM MEMORY FOR FORECASTING COAL PRICE," *BAREKENG: Jurnal Ilmu Matematika dan Terapan*, vol. 19, no. 1, pp. 245–258, Jan. 2025, doi: https://doi.org/10.30598/barekengvol19iss1pp245-258

[31] C. Cai, Y. Tao, T. Zhu, and Z. Deng, "SHORT-TERM LOAD FORECASTING BASED ON DEEP LEARNING BIDIRECTIONAL LSTM NEURAL NETWORK," *Applied Sciences*, vol. 11, no. 17, p. 8129, 2021, doi: https://doi.org/10.3390/app11178129

[32] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W. Woo, "CONVOLUTIONAL LSTM NETWORK: A MACHINE LEARNING APPROACH FOR PRECIPITATION NOWCASTING," *Adv Neural Inf Process Syst*, vol. 28, 2015.

[33] F. M. Butt *et al.*, "OPTIMIZING PARAMETERS OF ARTIFICIAL INTELLIGENCE DEEP CONVOLUTIONAL NEURAL NETWORKS (CNN) TO IMPROVE PREDICTION PERFORMANCE OF LOAD FORECASTING SYSTEM," in *IOP conference series: earth and environmental science*, 2022, p. 12028, doi: https://doi.org/10.1088/1755-1315/1026/1/012028

[34] J. Gu *et al.*, "RECENT ADVANCES IN CONVOLUTIONAL NEURAL NETWORKS," *Pattern Recognit*, vol. 77, pp. 354–377, 2018.doi: https://doi.org/10.1016/j.patcog.2017.10.013