Knowledge Engineering and Data Science (KEDS)
Vol 6, No 2, October 2023, pp. 170–187

pISSN 2597-4602
eISSN 2597-4637

Deep Learning Approaches with Optimum Alpha for Energy Usage Forecasting

Aji Prasetya Wibawa [a,1,*], Agung Bella Putra Utama [a,2], Ade Kurnia Ganesh Akbari [a,3], Akhmad Fanny Fadhilla [a,4], Alfiansyah Putra Pertama Triono [a,5], Andien Khansa'a Iffat Paramarta [a,6], Faradini Usha Setyaputri [a,7], Leonel Hernandez [b,8]

[a] *Department of Electrical Engineering and Informatics, Faculty of Engineering, Universitas Negeri Malang*
Jl. Semarang no. 5, Malang 65145, Indonesia
[b] *Institución Universitaria de Barranquilla IUB*
Cra. 45 #48-31, Nte. Centro Historico, Barranquilla 080020, Colombia
[1]*aji.prasetya.ft@um.ac.id*;* [2]*agungbpu02@gmail.com;* [3]*ade.kurniaganesh.1905356@students.um.ac.id;*
[4]*akhmadfadhil512@gmail.com;* [5]*alfiansyah.putrapt.1905356@student.um.ac.id;* [6]*khansaandien@gmail.com;*
[7]*faradini.usha@gmail.com;* [8]*lhernandezc@unibarranquilla.edu.co*
** corresponding author*

ARTICLE INFO

Article history:
Received 17 October 2023
Revised 17 October 2023
Accepted 17 October 2023
Published online 20 October 2023

Keywords:
Energy Efficiency
Forecasting
Deep Learning
Exponential Smoothing
Optimum Alpha

ABSTRACT

Energy use is an essential aspect of many human activities, from individual to industrial scale. However, increasing global energy demand and the challenges posed by environmental change make understanding energy use patterns crucial. Accurate predictions of future energy consumption can greatly influence decision-making, supply-demand stability and energy efficiency. Energy use data often exhibits time-series patterns, which creates complexity in forecasting. To address this complexity, this research utilizes Deep Learning (DL), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-term Memory (LSTM), Bidirectional LSTM (Bi-LSTM), and Gated Recurrent Unit (GRU) models. The main objective is to improve the accuracy of energy usage forecasting by optimizing the alpha value in exponential smoothing, thereby improving forecasting accuracy. The results showed that all DL methods experienced improved accuracy when using optimum alpha. LSTM has the most optimal MAPE, RMSE, and R2 values compared to other methods. This research promotes energy management, decision-making, and efficiency by providing an innovative framework for accurate forecasting of energy use, thus contributing to a sustainable and efficient energy system.

I. Introduction

Energy usage is a critical factor in various human activities, ranging from individual to industrial scales. It plays a vital role in supporting economic growth, social welfare, and technological development [1]. However, with the increasing global demand for energy and the challenges posed by environmental changes, understanding energy usage patterns has become increasingly important. Accurate predictions about future energy use can provide significant benefits in decision-making [2], demand and supply stability [3], and energy efficiency [4]. Energy usage data often exhibits a time series nature, where information is recorded over a specific time span [5]. For example, hourly energy consumption data may be challenging to interpret directly due to its temporal nature [6]. Additionally, energy usage data can involve various attributes that contribute to the patterns and fluctuations of energy usage. Therefore, accurately forecasting future energy use poses a complex task.

To overcome the complexity of analyzing energy usage data, Deep Learning (DL) has emerged as a practical approach [7]. DL is a branch of machine learning that utilizes neural networks with multiple layers and parameters to learn complex data representations [8]. Various DL models have been developed for time series analysis, including Convolutional Neural Networks (CNN) [9], Recurrent Neural Networks (RNN) [10], Long Short-term Memory (LSTM) [11], Bidirectional LSTM (Bi-LSTM) [12], and Gated Recurrent Unit (GRU) [13]. CNNs have been widely used in image

https://doi.org/10.17977/um018v6i22023p170-187

recognition tasks, but they can also be applied to time series data analysis. They can automatically extract essential features from time series data, such as seasonal patterns, trends, cycles, and irregularities. Unlike 2D-CNNs, which require converting time series data into image format, 1D-CNNs [14] can directly process time series data without the need for image conversion.

RNNs, particularly LSTM, are well-suited for modeling temporal dependencies in time series data [15]. RNNs maintain a hidden state that captures information about previous time steps, allowing them to capture long-term dependencies. LSTM, in particular, addresses the vanishing gradient problem commonly encountered in traditional RNNs. The vanishing gradient problem occurs when the gradient approaches zero, preventing updates to the network weights and causing the loss of time series data characteristics. LSTM overcomes this issue by using memory cells and gates to store and control the temporary state of the network [16]. Bi-LSTM is an extension of the LSTM model that incorporates information from both past and future time steps. It consists of two LSTM layers, one processing the input sequence in the forward direction and the other in the backward direction. By considering information from both directions, Bi-LSTM can capture more comprehensive temporal dependencies in the data [17]. This bidirectional nature makes Bi-LSTM particularly effective in tasks where future information is crucial for accurate predictions, such as energy usage forecasting. GRU, on the other hand, is a simplified version of LSTM that aims to address the computational.

In this study, we aim to explore the application of DL models with optimum alpha for energy usage forecasting. We will compare the performance of different DL models and evaluate their effectiveness in capturing the complex patterns and fluctuations in energy usage data. Additionally, we will investigate the impact of data normalization techniques on the performance of DL models. The findings of this research will contribute to the development of accurate and efficient energy usage forecasting models, which can aid in decision-making and promote energy efficiency in various sectors. Overall, this study aims to address the challenges in analyzing energy usage data by leveraging the power of DL models. By utilizing DL models, we can extract meaningful features and capture temporal dependencies in the data, leading to improved energy usage forecasting. The results of this research will provide valuable insights for energy management and planning, contributing to a more sustainable and efficient energy future.

II. Methods

To facilitate a more systematic research approach, experiments were devised, as illustrated in Figure 1. In essence, a comparison was made between the Smoothed Deep Learning (S-DL) method using optimum alpha and the primary DL method. Various evaluation metrics were also employed to assess the performance of the optimum alpha-enhanced results. Further details regarding Figure 1 will be expounded upon in the following subsections.

A. Dataset

The dataset used in this study uses the Hourly Energy Demand Time Series Forecast dataset from kaggle [18]. This dataset encompasses a span of four years (January 2015 to December 2018) and encompasses information regarding electricity usage, production, pricing, and meteorological conditions in Spain. Specifically, data on electricity consumption and generation was sourced from ENTSOE, a publicly accessible platform for Transmission Service Operator (TSO) data. Settlement prices, on the other hand, were acquired from the Spanish TSO, Red Electric España. Additionally, weather data for the five largest cities in Spain was procured as part of a personal project, and it was subsequently made available to the public through the Open Weather API. What sets this dataset apart is its inclusion of detailed hourly records for electricity consumption, alongside forecasts provided by the TSO for both consumption and pricing.

This dataset consists of 29 attributes that have 35064 instances with float data type. The target attribute used in this study is the actual total load attribute whose data visualization can be seen in Figure 2. As for the total load forecast attribute, it is not used in the research because the presence of this attribute in the data serves as a benchmark or comparison attribute with the target attribute. In addition, there are 2 attributes that are deleted because they have NaN values. The total load forecast attribute was also not used in the study because the presence of this attribute in the data serves as a

benchmark or comparison attribute with the target attribute. Therefore, the total attributes to be used are 26.

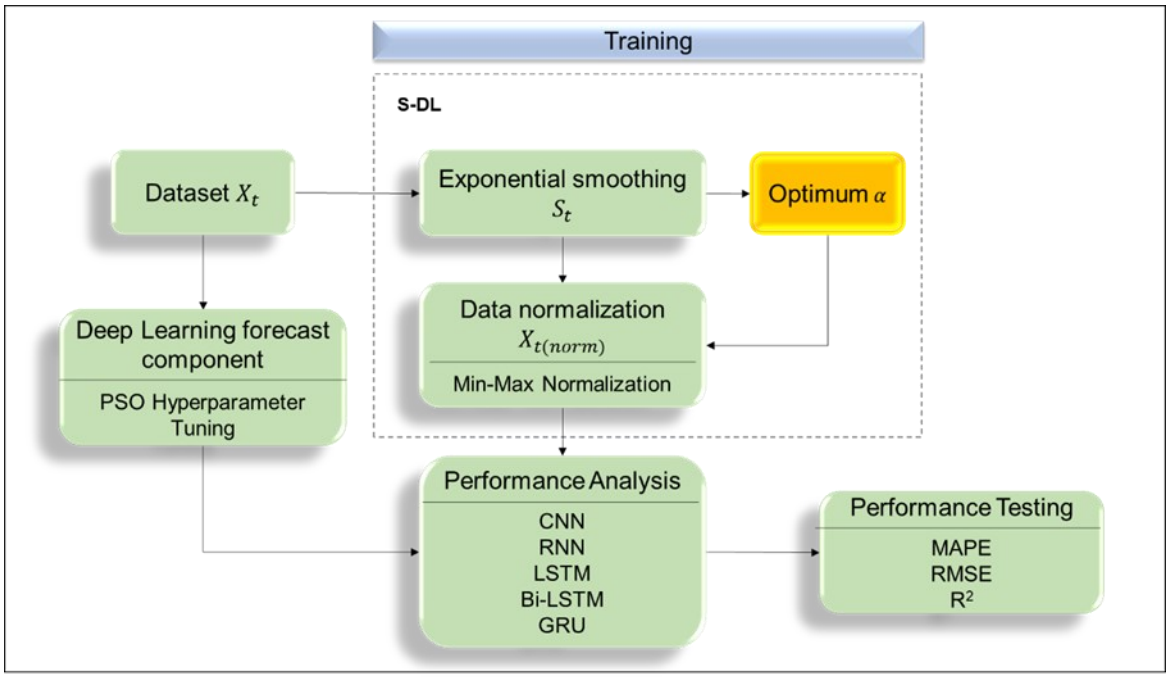

Fig. 1. Experimental schema

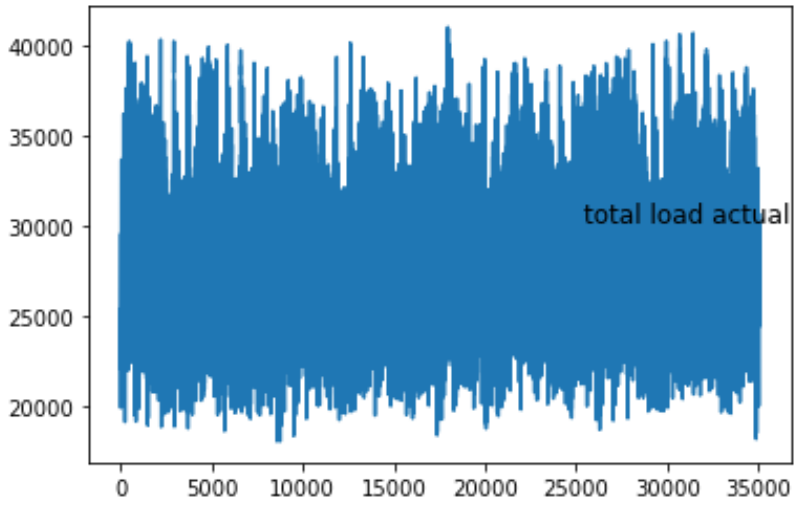

Fig. 2. Total load actual

B. Exponential Smoothing with Optimum α

Exponential smoothing is a widely used technique in time series forecasting that aims to eliminate noise and capture underlying patterns in data [19]. It achieves this by assigning weights to previous observations, with higher weights given to more recent data points. The smoothing factor, denoted as α (alpha), determines the weight assigned to the most recent observation [20]. The concept of optimum α arises from the need to find the best value for the smoothing factor that maximizes the accuracy of the forecasting model [21]. The choice of α depends on the specific characteristics of the time series data and the desired forecasting task. The goal is to select the value of α that minimizes the forecasting error or maximizes the accuracy of the predictions.

To determine the optimum α, various approaches can be employed. One standard method is to perform a grid search or optimization algorithm to evaluate different values of α and select the one that yields the lowest forecasting error. The process of finding the optimum α involves balancing the

trade-off between responsiveness to recent changes in the data and the level of smoothing applied. A higher α value gives more weight to recent observations, making the model more responsive to short-term fluctuations but potentially less stable. Conversely, a lower α value places more emphasis on historical data, resulting in a smoother forecast but potentially slower to adapt to changes. This process considers the characteristics of the time series data and the specific forecasting objectives, striking a balance between responsiveness and stability in the model's predictions.

Equation (1) and (2) offer the single exponential smoothing [22] when $t = 0$. The smoothed data S_t is the result of smoothing the raw data $\{X_t\}$. The smoothing factor, α is a value that determines the level of smoothing. The range of α is between 0 and 1 ($0 \leq \alpha \leq 1$). When α close to 1, the learning process is fast because it has a less smoothing effect. In contrast, values of α closer to 0 have a more significant smoothing effect and are less responsive to recent changes (slow learning).

$$S_t = \alpha X_t + (1-\alpha) S_{t-1}, \quad t > 0 \tag{1}$$

$$S_t = S_{t-1} + \alpha(X_t - S_{t-1}) \tag{2}$$

$$Optimum\ \alpha = \frac{(X_{max} - X_{min}) - \frac{1}{n}\sum_{i=1}^{n} X_t}{X_{max} - X_{min}} \tag{3}$$

The substitution of Equation (3) to (2) results in the following Equation (4). We use the optimum smoothed result (S_t) to improve the DL methodperformance [21]. Pseudocode 1 show how to find the optimum alpha for exponential smoothing

$$S_t = S_{t-1} + \frac{(X_{max} - X_{min}) - \frac{1}{n}\sum_{i=1}^{n} X_t}{X_{max} - X_{min}}(X_t - S_{t-1}) \tag{4}$$

PSEUDOCODE 1. Find the optimum alpha for exponential smoothing

```
Input:
- Data time series

Output:
- Optimum value of alpha

Procedure FindOptimumAlpha(data):
    Set alpha_min = 0.1  // minimum value of alpha
    Set alpha_max = 0.9  // maximum value of alpha
    Set alpha_step = 0.1  // increment step for alpha
    Set error_min = infinity  // minimum error value
    Set alpha_optimum = 0  // optimum value of alpha

    For alpha = alpha_min to alpha_max step alpha_step:
        Apply exponential smoothing with alpha to the data
        Calculate the error by comparing the predicted values with the actual
   data

        If error < error_min:
            Set error_min = error
            Set alpha_optimum = alpha

    Return alpha_optimum as the optimum value of alpha

End Procedure
```

C. Data Normalization

In this research, preprocessing is done by changing the original data so that it can be processed for further testing [23]. The inherent characteristics of the majority of time-series data exhibit dynamic and non-linear behavior [24]. The preprocessing carried out in this study is by normalize the data. Data normalization is an essential preprocessing step in energy usage forecasting to ensure that the input data is standardized and comparable across different scales. Normalization techniques transform the data into a standard range, typically between 0 and 1, without distorting the original distribution. This process helps to eliminate the influence of outliers and extreme values, making the data more suitable for training DL models.

The choice of normalization technique depends on the characteristics of the energy usage data and the specific requirements of the forecasting task. It is essential to experiment with different normalization techniques and evaluate their impact on the performance of DL models. Proper data normalization can improve the convergence speed of the models, prevent numerical instability, and enhance the overall accuracy of energy usage forecasting. One commonly used normalization technique in this research is using Min-Max scaling, also known as feature scaling. This method rescales the data by subtracting the minimum value and dividing by the range (maximum value minus minimum value). The resulting values are then within the range of 0 to 1 [25]. Min-Max scaling preserves the relative relationships between the data points and is particularly useful when the distribution of the data is known to be bounded as in (5). Pseudocode 2 present the process for the normalization.

$$X_{t(norm)} = \frac{X_t - X_{min}}{X_{max} - X_{min}} \tag{5}$$

$X_{t(norm)}$ is the result of normalization, X_t is the data to be normalized, while X_{min} and X_{max} stand for the minimum and maximum value of the entire data.

PSEUDOCODE 2. Normalization using Min-Max

```
Input:
-Data to be normalized (X), minimum value of the data (X_min), maximum value
   of the data (X_max)

Output:
-Normalized data (X_norm)

Procedure Min-Max Normalization
  Calculate the range of the data:
     a. Set X_range = X_max - X_min

  Normalize the data:
     a. For each data point X_t in X:
        i. Calculate the normalized value X_norm_t using the formula:
           X_norm_t = (X_t - X_min) / X_range
        ii. Append X_norm_t to the normalized data X_norm

  Return the normalized data X_norm

End Procedure
```

D. PSO Hyperparameter Tuning

Particle Swarm Optimization (PSO) is a metaheuristic optimization algorithm inspired by the social behavior of bird flocking or fish schooling [26]. It is commonly used to tune the hyperparameters of machine learning models, including Deep Learning (DL) models [27]. In this section, we will discuss the application of PSO for hyperparameter tuning in DL models for energy usage forecasting. Hyperparameters are parameters that are not learned directly from the data but are set by the user before training the model. They control the behavior and performance of the DL model, such as the learning rate, number of hidden layers, and number of neurons in each layer. Finding the optimal values for these hyperparameters is crucial for achieving the best performance of the DL model.

PSO works by simulating the movement of particles in a multidimensional search space. Each particle represents a potential solution, and its position in the search space corresponds to a set of hyperparameters. The particles move towards the best solution found so far, called the global best, and are influenced by their own best solution, called the personal best. Through iterations, the particles explore the search space and converge towards the optimal solution. A general outline of the PSO hyperparameter tuning process can be seen in the Pseudocode 3. In the context of DL models for energy usage forecasting, PSO can be used to tune hyperparameters such as the number of DL layers, the number of neurons in each layer, the batch size, and the dropout rate like in Table 1. By searching the hyperparameter space using PSO, we can find the combination of hyperparameters that leads to the best performance of the DL model in terms of accuracy and prediction error.

PSEUDOCODE 3. PSO hyperparameter tuning

```
Input:
- Data for training and validation
- Hyperparameter search space

Output:
- Best hyperparameter settings

Procedure PSO_Hyperparameter_Tuning(data):
    Set population_size = 50  // Number of particles in the swarm
    Set max_iterations = 100  // Maximum number of iterations
    Set c1 = 2.0  // Cognitive parameter
    Set c2 = 2.0  // Social parameter
    Set w = 0.7  // Inertia weight

    // Initialize the swarm
    Initialize_swarm(population_size)

    // Evaluate initial particle positions
    Evaluate_particles(data)

    // Set the global best position and fitness
    Set_global_best()

    // Main PSO loop
    for iteration = 1 to max_iterations do:
        for each particle in the swarm do:
            // Update particle velocity
            Update_velocity(particle, global_best)

            // Update particle position
            Update_position(particle)

            // Evaluate new particle position
            Evaluate_particle(data, particle)

            // Update personal best position and fitness
            Update_personal_best(particle)

            // Update global best position and fitness
            Update_global_best(particle)

    // Return the best hyperparameter settings
    Return global_best_position

End Procedure
```

To apply PSO for hyperparameter tuning, we need to define the fitness function that evaluates the performance of the DL model with a specific set of hyperparameters. The PSO algorithm then iteratively updates the positions of the particles based on their personal best, global best, and the inertia weight, which controls the balance between exploration and exploitation. By searching the hyperparameter space using PSO, we can find the optimal combination of hyperparameters that leads to improved performance and accurate predictions. This approach can enhance the effectiveness of

DL models in energy usage forecasting and contribute to better decision-making and energy management.

Table 1. PSO hyperparameter tuning search space

Parameter	Search Space
Batch Size	'100', '1000'
Epoch	'50', '100'
Hidden Layer	'2', '5', '10'
Loss Function	'MSE', 'MAE', 'huberloss'
Neuron	'32', '64'
Optimizer	'adam', 'rmsprop'

E. Performance Analysis

To measure the performance analysis in the study, we used methods in DL. DL is a subset of machine learning algorithms. DL itself is often called a deep neural network [28]. Neural networks are computational models that work by mimicking the behavior of the human brain [29]. Basically, DL is a neural network that has many layers and parameters [30]. The number of layers in DL allows the model to be able to analyze large amounts of data and have complex relationships. Early layers are used to learn simple features, while deeper layers learn more complex features [31].

- Convolutional Neural Network (CNN)

CNN, especially 2D-CNN, have revolutionized picture classification. However, one dimension (1D-CNN), excel at time-series data classification [14]. 1D-CNN can automatically learn the internal representation of time-series data and detect essential characteristics without operator intervention [21]. The internal representation of time-series data includes seasonality, trends, cycles, and abnormalities. These properties are essential for time-series data analysis and prediction. These internal representations can be recorded and used for classification by 1D-CNN. 1D-CNN operate directly on time-series data, unlike 2D-CNN, which convert input data into numbers. This simplifies workflow by eliminating preprocessing processes. 1D-CNNs may capture temporal connections and identify significant patterns by directly examining sequential data [27] Overall, 1D-CNN for time-series data categorization have many benefits. This allows automatic feature extraction for more efficient and accurate time-series data analysis. Direct processing of time-series data eliminates complex data transformations, simplifying modeling. Thus, 1D-CNN are useful for time-series data analysis and classification. The 1D-CNN architecture present in Figure 3 and for the pseudocde of CNN forecasting process can be seen in Pseudocode 4.

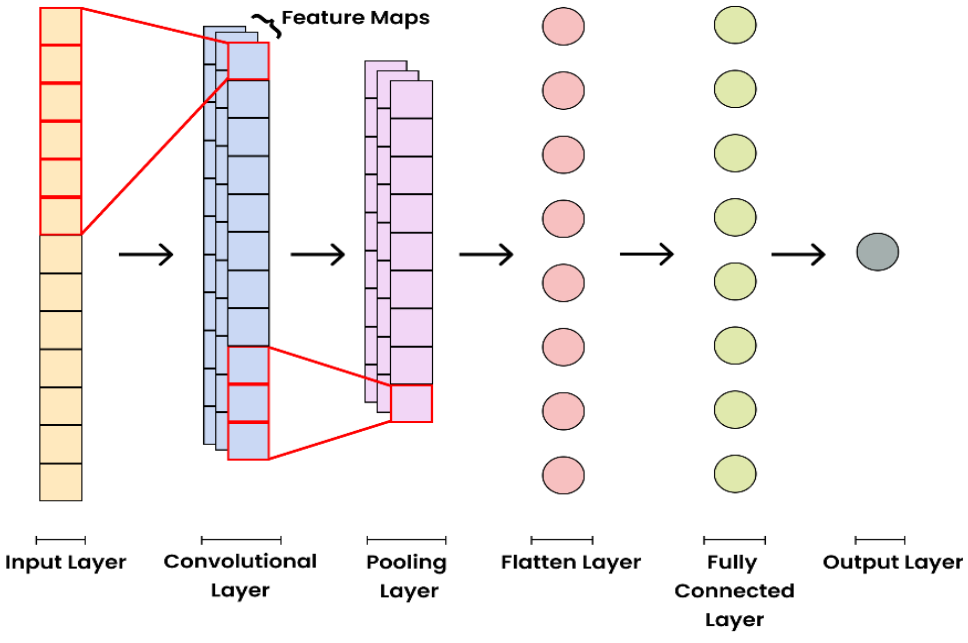

Fig. 3. 1D-CNN architecture

PSEUDOCODE 4. CNN forecasting process

```
Input:
- Energy Dataset
- Setting parameters according to the results of PSO hyperparameter tuning

Output:
- Trained CNN model

Procedure Train_CNN(training_data, validation_data, num_conv_layers,
    num_filters, filter_size, num_fc_layers, num_neurons, learning_rate,
    num_epochs):
     Initialize CNN model

     // Add convolutional layers
     for i = 1 to num_conv_layers do:
         Add convolutional layer with num_filters[i] filters and filter_size[i]
    filter size

     // Flatten the output from convolutional layers
     Flatten()

     // Add fully connected layers
     for i = 1 to num_fc_layers do:
         Add fully connected layer with num_neurons[i] neurons

     // Compile the model
     Compile model with appropriate loss function and optimizer

     // Train the model
     Train model on training_data with validation_data, using learning_rate and
    num_epochs

     // Return the trained model
     Return trained CNN model

End Procedure
```

- Recurrent Neural Network (RNN)

The RNN developed by Paul Werbos and Ronald J. Williams in the 1980s and 1990s is the most commonly used model in deep learning [32]. RNNs are a class of deep learning models designed to process sequential data. The main characteristic of RNNs is the presence of recurrent connections in the network, which allows them to maintain a hidden state that captures information about previous time steps [33]. This hidden state makes RNNs particularly suitable for modeling temporal dependencies in time series data. The architecture includes a series of recurrent cells, each processing input data and updating the hidden state using recurrent connections. This recurrent structure allows the RNN to cope with sequences of varying length. The RNN architecture can be seen in Figure 4. Pseudocde 5 present the RNN forecasting process.

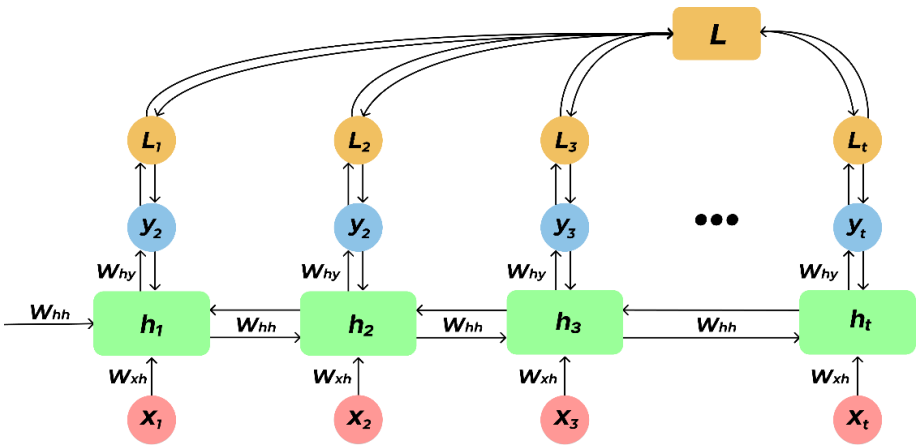

Fig. 4. RNN architecture

PSEUDOCODE 5. RNN forecasting process

```
Input:
- Energy Dataset
- Setting parameters according to the results of PSO hyperparameter tuning

Output:
- Trained RNN model

Procedure RNN_Training(training_data, num_hidden_units, learning_rate,
    num_epochs):
     Initialize weights and biases for input-to-hidden and hidden-to-hidden
    connections
     Initialize the hidden state

     for epoch = 1 to num_epochs do:
         for each training example in training_data do:
             // Forward pass
             for t = 1 to sequence_length do:
                 Update hidden state using the current input and previous hidden
    state

             // Backward pass
             for t = sequence_length to 1 do:
                 Calculate the gradient of the loss with respect to the output
                 Update the weights and biases of the hidden-to-hidden
    connections

                 Calculate the gradient of the loss with respect to the hidden
    state
                 Update the weights and biases of the input-to-hidden
    connections

     // Return the trained RNN model
     Return trained_model

End Procedure
```

- Long Short-term Memory (LSTM)

Vanishing gradient found in RNN is a condition when the gradient approaches 0 so that the gradient cannot provide updates to the weights in the network and make the time series data lose its characteristics [34]. Vanishing gradient is caused by using the same weight at each time-step. LSTM can overcome the vanishing gradient problem in RNN. The concept of Long Short-term Memory (LSTM) was first published in 1997 by Hochreiter and Schmidhuber [35]. LSTM analyzes time series data for the long term by applying a collection of short-term memories. This model develops the information storage capacity of RNNs by using "memory cells" [36]. Memory cells have connections that store the temporary state of the network and are controlled through 3 "gates", namely forget gate, input gate, and output gate [37]. Figure 5 represents the memory cell of LSTM and for the pseudocde of LSTM forecasting process can be seen in Pseudocode 6.

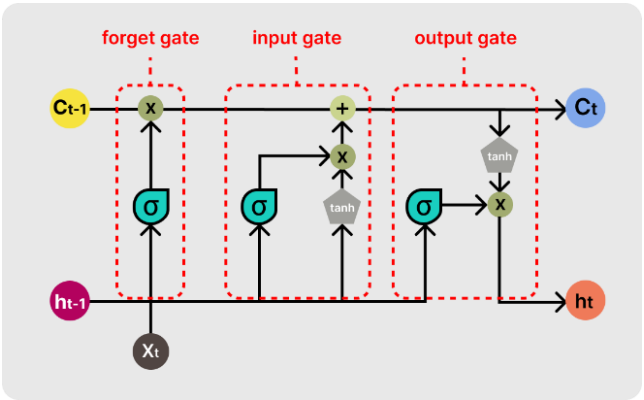

Fig. 5. Memory cell LSTM

PSEUDOCODE 6. LSTM forecasting process

```
Input:
- Energy Dataset
- Setting parameters according to the results of PSO hyperparameter tuning

Output:
- Trained LSTM model

Procedure LSTM_Model(training_data, testing_data, num_layers, num_hidden_units,
    num_output_units, learning_rate, num_epochs):
    // Initialize LSTM model
    Initialize_LSTM(num_layers, num_hidden_units, num_output_units)

    Set_Num_Epochs(num_epochs)

    // Train LSTM model
    for epoch = 1 to num_epochs do:
        // Forward pass
        for each training example in training_data do:
            // Reset LSTM hidden state
            Reset_Hidden_State()

            // Iterate through each time step
            for t = 1 to length(training_example) do:
                // Perform LSTM forward pass
                LSTM_Forward_Pass(training_example[t])

        // Backward pass
        for each training example in training_data do:
            // Reset LSTM gradients
            Reset_Gradients()

            // Iterate through each time step in reverse order
            for t = length(training_example) to 1 do:
                // Perform LSTM backward pass
                LSTM_Backward_Pass(training_example[t])

            // Update LSTM weights
            Update_Weights()

    // Test LSTM model
    for each testing example in testing_data do:
        // Reset LSTM hidden state
        Reset_Hidden_State()

        // Iterate through each time step
        for t = 1 to length(testing_example) do:
            // Perform LSTM forward pass
            LSTM_Forward_Pass(testing_example[t])

    // Return trained LSTM model
    Return LSTM_Model

End Procedure
```

- Bidirectional LSTM (Bi-LSTM)

The Bi-LSTM model is a variant of the LSTM model that incorporates bidirectional processing. It consists of two LSTM layers, one processing the input sequence in the forward direction and the other processing it in the backward direction. This bidirectional processing allows the model to capture both past and future dependencies in the data, making it particularly effective for time series analysis [38]. In the forward LSTM layer, the input sequence is processed from the beginning to the end, capturing the temporal dependencies and patterns in the data. This layer maintains a hidden state that stores information about the past time steps. The backward LSTM layer, on the other hand, processes the input sequence in reverse order, capturing the dependencies and patterns in the opposite direction. This layer maintains a separate hidden state that stores information about the future time

steps. By combining the outputs of both LSTM layers, the Bi-LSTM model can effectively capture the dependencies in both directions [39]. This allows the model to have a more comprehensive understanding of the temporal dynamics in the data. The outputs of the Bi-LSTM layers are then fed into a fully connected layer, which performs a non-linear transformation on the data and produces the final forecasted values. The Bi-LSTM architecture can be seen in Figure 6. Pseudocde 7 show the Bi-LSTM forecasting process

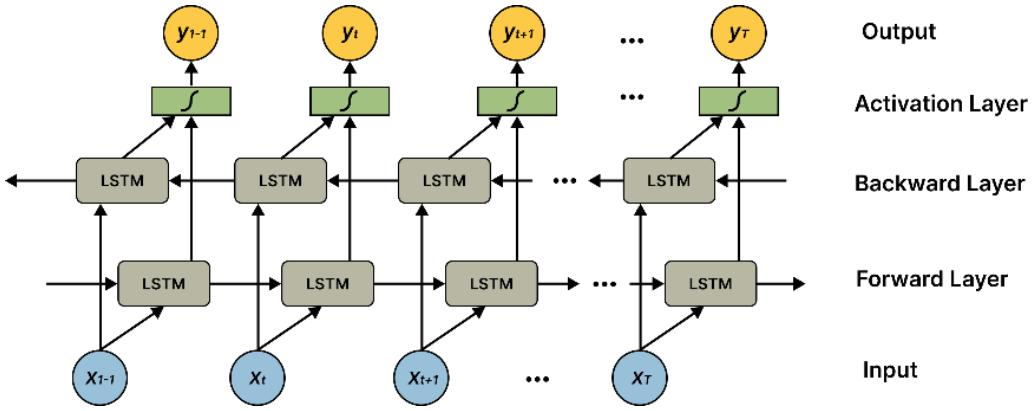

Fig. 6. Bi-LSTM Architecture

PSEUDOCODE 7. Bi-LSTM forecasting process

```
Input:
- Energy Dataset
- Setting parameters according to the results of PSO hyperparameter tuning

Output:
- Trained Bi-LSTM model

Procedure Train_BiLSTM(training_data, validation_data, num_lstm_layers,
    num_lstm_units, learning_rate, num_epochs):
     Initialize Bi-LSTM model

     // Add LSTM layers
     for i = 1 to num_lstm_layers do:
         Add forward LSTM layer with num_lstm_units[i] units
         Add backward LSTM layer with num_lstm_units[i] units

     // Compile the model
     Compile model with appropriate loss function and optimizer

     // Train the model
     Train model on training_data with validation_data, using learning_rate and
    num_epochs

     // Return the trained model
     Return trained Bi-LSTM model

End Procedure
```

- Gated Recurrent Units (GRU)

The GRU model is a sophisticated RNN variation used for sequential data processing and forecasting. It captures long-term time series dependencies well. Training the GRU model with the training set helps it understand data patterns and relationships. The GRU model uses gating techniques to preserve or discard past time step information, unlike RNNs [40] The reset and update gates control network information flow. The reset gate specifies which bits of the prior concealed state to forget, while the update gate determines how much new information to add. GRU models data temporal dependencies by selectively updating and forgetting information. This adaptive retention or discard capacity allows the model to capture short-term and long-term patterns, making it suitable for time series forecasting, speech recognition, and natural language processing. Its gating

features and capacity to identify long-term dependencies make the GRU model a formidable sequential data analysis and forecasting tool. It is used in many fields where understanding and predicting time series data patterns is crucial because to its flexibility and efficacy [41]. Figure 7 represents the structure of GRU cell and for the pseudocde of GRU forecasting process can be seen in Pseudocode 8.

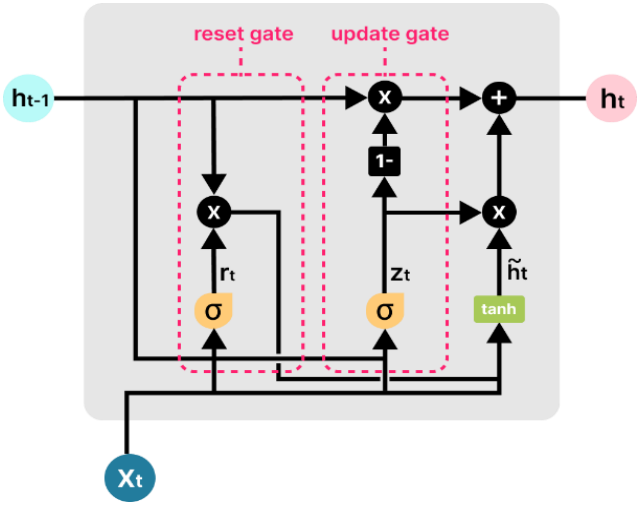

Fig. 7. GRU cell structure

PSEUDOCODE 8. GRU forecasting process

```
Input:
- Energy Dataset
- Setting parameters according to the results of PSO hyperparameter tuning

Output:
- Trained GRU model

Procedure Train_GRU(training_data, validation_data, num_gru_layers,
    num_hidden_units, learning_rate, num_epochs):
     Initialize GRU model

     // Add GRU layers
     for i = 1 to num_gru_layers do:
         Add GRU layer with num_hidden_units[i] hidden units

     // Compile the model
     Compile model with appropriate loss function and optimizer

     // Train the model
     Train model on training_data with validation_data, using learning_rate and
    num_epochs

     // Return the trained model
     Return trained GRU model

End Procedure
```

F. Data Analysis

Performance testing is an essential step in evaluating the effectiveness and efficiency of energy usage forecasting models [42]. It involves assessing the model's ability to accurately predict future energy usage based on historical data. In this section, we will discuss the performance testing process and metrics used to evaluate the DL models' forecasting performance. This research uses Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and the Coefficient of Determination (R^2) as calculations.

MAPE measures the extent to which forecasting or prediction distinguishes between predicted and actual energy values in percentage terms as in (6). A lower MAPE indicates a more accurate model

[43]. RMSE calculates the square root of the average squared difference between the predicted and actual energy usage values [44]. RMSE is used to determine how sensitive the existing DL model can detect outliers in the energy forecasting value compared to the original value, as in (7). Additionally, the R^2 is often used to assess the goodness of fit of the model as in (8). R^2 represents the proportion of the variance in the energy usage data that is predictable by the model. A higher R^2 value indicates a better fit of the model to the data [45].

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|A_i - F_i|}{A_i} \tag{6}$$

$$MSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(F_i - A_i)^2} \tag{7}$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{8}$$

A_i is the actual value, F_i is the predicted value, n is the number of predictions, SS_{res} is the residual sum of squares, and SS_{tot} is the total sum of squares. We also logged the computational time for each method, which serves as an additional performance metric. We designate the best method with the shortest computational time expenditure. By conducting performance testing and evaluating the accuracy and efficiency of DL models, we can gain insights into their effectiveness in energy usage forecasting. This information can guide decision-making processes, improve energy management strategies, and contribute to the development of more sustainable and efficient energy systems.

III. Result and Discussion

Figure 8 through Figure 11 illustrate the comparison between DL and S-DL across all method, with a smoothing factor of α = 0.1 applied to the S-DL. The setting paramter of all method is used from the PSO hyperparameter tuning search space result as present in Table 2.

Table 2. PSO hyperparameter tuning search space

Parameter	Search Space Result
Batch Size	100
Epoch	50
Hidden Layer	2
Loss Function	MSE
Neuron	32
Optimizer	Rmsprop

Figure 8 provides a valuable comparative assessment of MAPE for various prediction methods in two scenarios: "Without Smoothing" and "Smoothing with Optimum Alpha," highlighting the impact of smoothing techniques, specifically optimized with an alpha value, on predictive accuracy. In both scenarios, MAPE values indicate that LSTM consistently outperforms other methods, exhibiting the lowest MAPE values (3.9065%) and demonstrating exceptional predictive accuracy. Conversely, Bi-LSTM continuously records the highest MAPE values (7.6464%), suggesting lower predictive accuracy regardless of smoothing. Overall, Figure 8 underscores the significance of optimizing smoothing techniques with an alpha value to enhance predictive accuracy in data analysis and forecasting tasks. Although the average increase in MAPE value across all methods was a modest 0.1385%, LSTM consistently proves to be the most accurate method. At the same time, Bi-LSTM always lags in predictive accuracy in both scenarios. These findings emphasize the importance of judiciously applying smoothing techniques for improved predictive performance.

Figure 9 compares RMSE values for various prediction methods under two scenarios: "Without Smoothing" and "Smoothing with Optimum Alpha." A consistent trend observed in the data is the slight reduction in RMSE values across all methods when "Smoothing with Optimum Alpha" is applied, indicating improved prediction accuracy through smoothing. LSTM consistently outperforms other methods by achieving the lowest RMSE values in both scenarios (0.0624 and 0.0621), underscoring its accuracy. In contrast, Bi-LSTM consistently exhibits the highest RMSE values,

suggesting lower prediction accuracy regardless of smoothing (0.1252 and 0.1228). In summary, Figure 9 emphasizes the positive impact of smoothing techniques, particularly when optimized with an alpha value, on enhancing the predictive accuracy of these methods. The average decrease in RMSE of all methods after using smoothing with optimum alpha is 0.0061, this shows that smoothing with optimum alpha can detect outliers better and more sensitively. LSTM is the most accurate method, while Bi-LSTM consistently demonstrates the least accuracy, highlighting the significance of thoughtful smoothing application in data analysis and forecasting tasks.

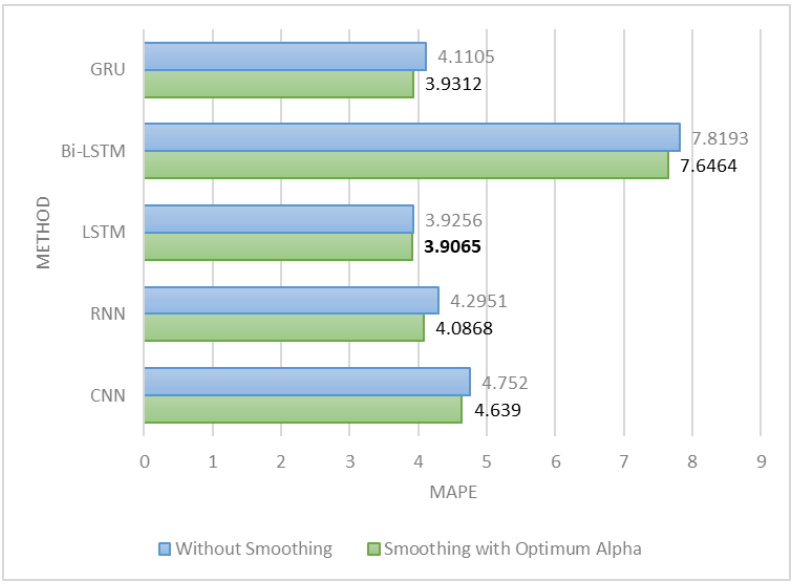

Fig. 8. MAPE evaluation result

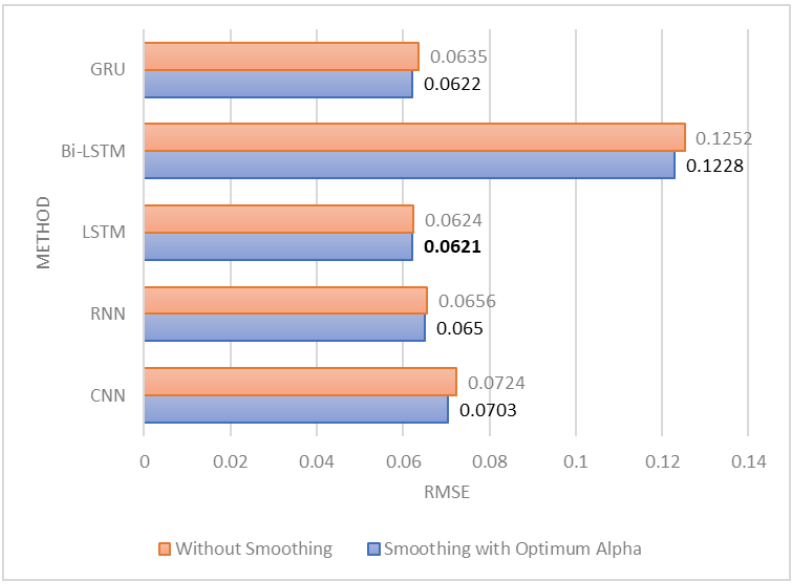

Fig. 9. RMSE evaluation result

Figure 10 presents a comparative R^2 values analysis for various prediction methods in two scenarios. The data consistently reveals that R^2 values improve across all forms when "Smoothing with Optimum Alpha" is applied, indicating a superior fit to the dataset. LSTM consistently outperforms other methods by achieving the highest R^2 values in both scenarios (0.9021 and 0.9027), confirming its strong alignment with the data. In contrast, Bi-LSTM continuously records the lowest R^2 values, indicating a relatively weaker fit regardless of smoothing (0.6042 and 0.6195). Basically, Figure 10 underscores the positive impact of optimizing smoothing techniques with an alpha value on improving the goodness of fit. LSTM consistently excels at fitting the data, while Bi-LSTM

consistently demonstrates the weakest fit in both scenarios. These findings stress the importance of judiciously applying smoothing techniques to enhance the performance of these methods in data analysis and forecasting.

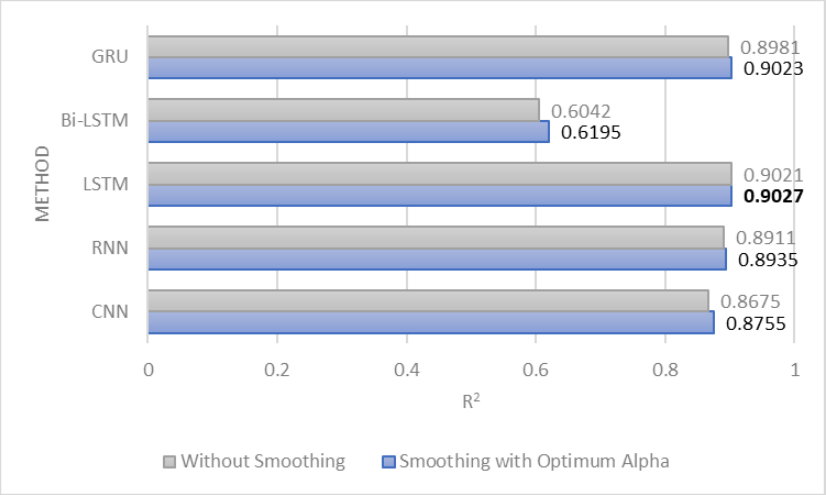

Fig. 10. R^2 evaluation result

Figure 11 provides insights to assess how the application of smoothing techniques, optimized with an alpha value, impacts the computational efficiency of these methods when handling data. The data reveals that, in most cases, "Smoothing with Optimum Alpha" leads to reduced computational times compared to the "Without Smoothing" scenario. This suggests that smoothing can improve the computational efficiency of these methods. CNN consistently shows shorter computational times in both scenarios, highlighting its efficiency. Conversely, Bi-LSTM and GRU require more time for computations, particularly without smoothing. These findings emphasize the importance of considering computational efficiency when choosing prediction methods for data analysis and forecasting tasks.

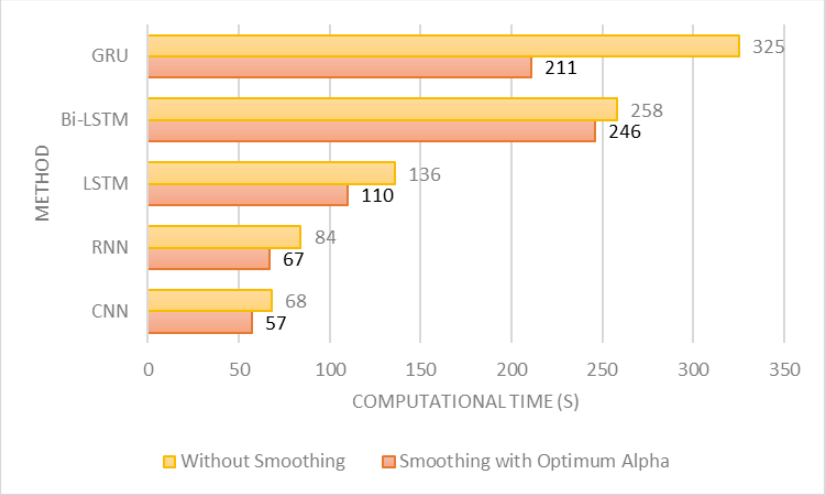

Fig. 11. Computational time evalution result

Overall, the use of an optimum alpha value can significantly enhance the forecasting results for energy data in all DL method. In this study, the LSTM model consistently stands out as the top choice, yielding the lowest MAPE and RMSE values while achieving the highest R^2 value. For computation time, LSTM is also in the middle, not too fast and not too long. This indicates that LSTM not only provides a high level of prediction accuracy but also offers the best fit to the existing data compared to the other evaluated methods.

The implications of these findings in the field of energy are that selecting the suitable model or method, especially when using an optimum alpha value, can significantly improve the accuracy of predictions in energy resource planning and management. In the energy sector, more accurate predictions can have a positive impact on optimizing energy usage, reducing waste, and supporting environmental sustainability. Furthermore, the use of DL and optimized methods like LSTM in energy forecasting opens up opportunities to develop more intelligent and more efficient solutions for energy supply management, especially in situations where energy sustainability and efficiency are becoming increasingly crucial.

IV. Conclusions

In conclusion, this study underscores the significant impact of optimizing smoothing techniques with an optimum alpha (α) value on enhancing the accuracy of energy usage forecasting using DL models. Among the models tested, LSTM consistently outperforms others, displaying the lowest MAPE (3.9065%) and RMSE (0.0621) values and the highest R^2 (0.9027), making it the top choice for accurate predictions. Notably, the application of optimum alpha values has proven to be more successful in terms of improving prediction accuracy across various metrics. Computational efficiency is also a critical consideration, with CNN demonstrating shorter computation times (57s). Limitations of this research include the specific dataset used, which may not be entirely representative of all energy usage scenarios, and the computational resources required for LSTM. Future research should explore the generalizability of these findings across diverse energy datasets and further investigate the computational optimization of LSTM. These findings have crucial implications for energy resource management, as more accurate predictions can aid in optimizing energy usage, reducing waste, and supporting environmental sustainability, emphasizing the relevance of thoughtful model selection and hyperparameter tuning.

Declarations

Author contribution

All authors contributed equally as the main contributor of this paper. All authors read and approved the final paper.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare no known conflict of financial interest or personal relationships that could have appeared to influence the work reported in this paper.

Additional information

Reprints and permission information are available at http://journal2.um.ac.id/index.php/keds.

References

[1] A. Sharif, S. Kocak, H. H. A. Khan, G. Uzuner, and S. Tiwari, "Demystifying the links between green technology innovation, economic growth, and environmental tax in ASEAN-6 countries: The dynamic role of green energy and green investment," Gondwana Res., vol. 115, pp. 98–106, Mar. 2023.

[2] P. Ma, S. Cui, M. Chen, S. Zhou, and K. Wang, "Review of Family-Level Short-Term Load Forecasting and Its Application in Household Energy Management System," Energies, vol. 16, no. 15, p. 5809, Aug. 2023.

[3] L. Malka, F. Bidaj, A. Kuriqi, A. Jaku, R. Roçi, and A. Gebremedhin, "Energy system analysis with a focus on future energy demand projections: The case of Norway," Energy, vol. 272, p. 127107, Jun. 2023.

[4] S. Kapp, J.-K. Choi, and T. Hong, "Predicting industrial building energy consumption with statistical and machine-learning models informed by physical system parameters," Renew. Sustain. Energy Rev., vol. 172, p. 113045, Feb. 2023.

[5] Y. Zou, R. V. Donner, N. Marwan, J. F. Donges, and J. Kurths, "Complex network approaches to nonlinear time series analysis," Phys. Rep., vol. 787, pp. 1–97, Jan. 2019.

[6] A. Pranolo, Y. Mao, A. P. Wibawa, A. B. P. Utama, and F. A. Dwiyanto, "Robust LSTM With Tuned-PSO and Bifold-Attention Mechanism for Analyzing Multivariate Time-Series," IEEE Access, vol. 10, pp. 78423–78434, 2022.
[7] A. Pranolo, Y. Mao, A. P. Wibawa, A. B. P. Utama, and F. A. Dwiyanto, "Optimized Three Deep Learning Models Based-PSO Hyperparameters for Beijing PM2.5 Prediction," Knowl. Eng. Data Sci., vol. 5, no. 1, p. 53, Nov. 2022.
[8] J. Naskath, G. Sivakamasundari, and A. A. S. Begum, "A Study on Different Deep Learning Algorithms Used in Deep Neural Nets: MLP SOM and DBN," Wirel. Pers. Commun., vol. 128, no. 4, pp. 2913–2936, 2023.
[9] I. Koprinska, D. Wu, and Z. Wang, "Convolutional Neural Networks for Energy Time Series Forecasting," in 2018 International Joint Conference on Neural Networks (IJCNN), Jul. 2018, pp. 1–8.
[10] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent Neural Networks for Time Series Forecasting: Current status and future directions," Int. J. Forecast., vol. 37, no. 1, pp. 388–427, Jan. 2021.
[11] G. Bathla, R. Rani, and H. Aggarwal, "Stocks of year 2020: prediction of high variations in stock prices using LSTM," Multimed. Tools Appl., vol. 82, no. 7, pp. 9727–9743, Mar. 2023.
[12] M. Yang and J. Wang, "Adaptability of Financial Time Series Prediction Based on BiLSTM," Procedia Comput. Sci., vol. 199, pp. 18–25, 2022.
[13] A. N. . F. Faisal, A. Rahman, M. T. M. Habib, A. H. Siddique, M. Hasan, and M. M. Khan, "Neural networks based multivariate time series forecasting of solar radiation using meteorological data of different cities of Bangladesh," Results Eng., vol. 13, p. 100365, Mar. 2022.
[14] A. R. F. Dewandra, A. P. Wibawa, U. Pujianto, A. B. P. Utama, and A. Nafalski, "Journal Unique Visitors Forecasting Based on Multivariate Attributes Using CNN," Int. J. Artif. Intell. Res., vol. 6, no. 1, 2022.
[15] F. Kurniawan, S. Sulaiman, S. Konate, and M. A. A. Abdalla, "Deep learning approaches for MIMO time-series analysis," Int. J. Adv. Intell. Informatics, vol. 9, no. 2, p. 286, Jul. 2023.
[16] Y. Mao, A. Pranolo, A. P. Wibawa, A. B. Putra Utama, F. A. Dwiyanto, and S. Saifullah, "Selection of Precise Long Short Term Memory (LSTM) Hyperparameters based on Particle Swarm Optimization," in 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC), May 2022, pp. 1114–1121.
[17] X. Zhou, A. Pranolo, and Y. Mao, "AB-LSTM: Attention Bidirectional Long Short-Term Memory for Multivariate Time-Series Forecasting," in 2023 International Conference on Computer, Electronics & Electrical Engineering & their Applications (IC2E3), Jun. 2023, pp. 1–6.
[18] M. Elsaraiti, G. Ali, H. Musbah, A. Merabet, and T. Little, "Time Series Analysis of Electricity Consumption Forecasting Using ARIMA Model," in 2021 IEEE Green Technologies Conference (GreenTech), Apr. 2021.
[19] A. B. F. Khan, K. Kamalakannan, and N. S. S. Ahmed, "Integrating Machine Learning and Stochastic Pattern Analysis for the Forecasting of Time-Series Data," SN Comput. Sci., vol. 4, no. 5, p. 484, Jun. 2023.
[20] M. Skariah and C. D. Suriyakala, "Forecasting reservoir inflow combining Exponential smoothing, ARIMA, and LSTM models," Arab. J. Geosci., vol. 15, no. 14, p. 1292, Jul. 2022.
[21] A. P. Wibawa, A. B. P. Utama, H. Elmunsyah, U. Pujianto, F. A. Dwiyanto, and L. Hernandez, "Time-series analysis with smoothed Convolutional Neural Network," J. Big Data, vol. 9, no. 1, p. 44, Dec. 2022.
[22] V. Prema and K. U. Rao, "Development of statistical time series models for solar power prediction," Renew. Energy, vol. 83, pp. 100–109, Nov. 2015.
[23] S. Huber, H. Wiemer, D. Schneider, and S. Ihlenfeldt, "DMME: Data mining methodology for engineering applications – a holistic extension to the CRISP-DM model," Procedia CIRP, vol. 79, pp. 403–408, 2019.
[24] A. Tealab, H. Hefny, and A. Badr, "Forecasting of nonlinear time series using ANN," Futur. Comput. Informatics J., vol. 2, no. 1, pp. 39–47, 2017.
[25] K. APARNA, "Evolutionary computing based hybrid bisecting clustering algorithm for multidimensional data," Sādhanā, vol. 44, no. 2, p. 45, Feb. 2019.
[26] L. Vanneschi and S. Silva, "Particle Swarm Optimization," in Natural Computing Series, 2023, pp. 105–111.
[27] A. B. P. Utama, A. P. Wibawa, Muladi, and A. Nafalski, "PSO based Hyperparameter tuning of CNN Multivariate Time-Series Analysis," J. Online Inform., vol. 7, no. 2, pp. 193–202, 2022.
[28] M. Abo-Tabik, N. Costen, J. Darby, and Y. Benn, "Towards a Smart Smoking Cessation App: A 1D-CNN Model Predicting Smoking Events," Sensors, vol. 20, no. 4, p. 1099, Feb. 2020.
[29] W. J. Zhang, G. Yang, Y. Lin, C. Ji, and M. M. Gupta, "On Definition of Deep Learning," in 2018 World Automation Congress (WAC), Jun. 2018, pp. 1–5.
[30] D. A. Bashar, "Survey on Evolving Deep Learning Neural Network Architectures," J. Artif. Intell. Capsul. Networks, vol. 2019, no. 2, pp. 73–82, Dec. 2019.
[31] P. P. Shinde and S. Shah, "A Review of Machine Learning and Deep Learning Applications," in 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Aug. 2018, pp. 1–6.
[32] H. Apaydin, H. Feizi, M. T. Sattari, M. S. Colak, S. Shamshirband, and K. W. Chau, "Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting," Water (Switzerland), vol. 12, no. 5, pp. 1–18.
[33] A. Zanfei, B. M. Brentan, A. Menapace, M. Righetti, and M. Herrera, "Graph Convolutional Recurrent Neural Networks for Water Demand Forecasting," Water Resour. Res., vol. 58, no. 7, Jul. 2022.
[34] Z. Hu, J. Zhang, and Y. Ge, "Handling Vanishing Gradient Problem Using Artificial Derivative," IEEE Access, vol. 9, pp. 22371–22377, 2021.
[35] K. Smagulova and A. P. James, "A survey on LSTM memristive neural network architectures and applications," Eur. Phys. J. Spec. Top., vol. 228, no. 10, pp. 2313–2324, Oct. 2019.
[36] X. Meng, M. Liu, and Q. Wu, "Prediction of Rice Yield via Stacked LSTM," Int. J. Agric. Environ. Inf. Syst., vol. 11, no. 1, pp. 86–95, Jan. 2020.
[37] F. Shahid, A. Zameer, and M. Muneeb, "Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM," Chaos, Solitons & Fractals, vol. 140, p. 110212, Nov. 2020.
[38] H. Wang, Y. Zhang, J. Liang, and L. Liu, "DAFA-BiLSTM: Deep Autoregression Feature Augmented Bidirectional LSTM network for time series prediction," Neural Networks, vol. 157, pp. 240–256, Jan. 2023.
[39] Q. Cheng, Y. Chen, Y. Xiao, H. Yin, and W. Liu, "A dual-stage attention-based Bi-LSTM network for multivariate time series prediction," J. Supercomput., vol. 78, no. 14, pp. 16214–16235, Sep. 2022.

[40] C. Hu, S. Martin, and R. Dingreville, "Accelerating phase-field predictions via recurrent neural networks learning the microstructure evolution in latent space," Comput. Methods Appl. Mech. Eng., vol. 397, p. 115128, Jul. 2022.
[41] X. Wang, N. Xie, and L. Yang, "A flexible grey Fourier model based on integral matching for forecasting seasonal PM2.5 time series," Chaos, Solitons & Fractals, vol. 162, p. 112417, Sep. 2022.
[42] W. Sun and C. Huang, "A novel carbon price prediction model combines the secondary decomposition algorithm and the long short-term memory network," Energy, vol. 207, p. 118294, Sep. 2020.
[43] A. P. Wibawa, Z. N. Izdihar, A. B. P. Utama, L. Hernandez, and Haviluddin, "Min-Max Backpropagation Neural Network to Forecast e-Journal Visitors," in 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Apr. 2021, pp. 052–058.
[44] A. P. Wibawa, "Mean-Median Smoothing Backpropagation Neural Network to Forecast Unique Visitors Time Series of Electronic Journal," J. Appl. Data Sci., vol. 4, no. 3, pp. 163–174, Sep. 2023.
[45] Y. Yang, C. Yu, and R. Y. Zhong, "Generalized linear model-based data analytic approach for construction equipment management," Adv. Eng. Informatics, vol. 55, p. 101884, Jan. 2023, doi: 10.1016/j.aei.2023.101884.