A Comparative Study of PSO, Genetic Algorithm, and Grid Search for LSTM Hyperparameter Optimization in Coal Production Forecasting

M. Iqbal Nurrifki[1], Ghina Zulfa Inayah[1], Gabriel Saputra Tampubolon[2], Nada Lubna Alfahima[2]
Ufairanisa Najma Sabrina[2], Ryanaldi Robby Kusuma[2], Ave Regina[2]
e-mail: 5003221061@student.its.ac.id

***Abstract*— This research investigates the effectiveness of Long Short-Term Memory (LSTM) networks in predicting coal production in the United States by evaluating three hyperparameter optimization approaches: Particle Swarm Optimization (PSO), Grid Search, and Genetic Algorithm (GA). The analysis utilizes monthly coal production data spanning from 1973 to 2024, obtained from the U.S. Energy Information Administration (EIA). Each optimization strategy was employed to fine-tune critical LSTM parameters, including the number of hidden units, layer count, dropout rate, and learning rate. The models' accuracy was measured using Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). The results indicate that PSO consistently produced better forecasting accuracy than Grid Search and GA across different input sequence lengths. The most accurate model was achieved with a sequence length of 12, resulting in a test RMSE of 3555.7 and a MAPE of 6.28%, along with strong generalization and low overfitting. In addition, PSO demonstrated the most efficient runtime, requiring only 22.9 seconds, which was significantly faster compared to the other two methods. These results confirm that PSO is a reliable and computationally efficient approach for tuning LSTM models in time series forecasting, making it highly suitable for applications such as energy production monitoring and planning.**

***Index Term*—LSTM, hyperparameter optimization, PSO, Grid Search, Genetic Algorithm, coal production forecasting**

I. INTRODUCTION

A. Background

Coal remains a crucial component of electricity generation in the United States, even as the global transition toward cleaner energy sources continues. However, coal production is subject to considerable variability due to regulatory changes, shifts in market prices, and fluctuating demand, resulting in highly dynamic data that pose challenges for traditional statistical forecasting techniques [1].

One commonly used method for handling non-linear time series data is the Long Short-Term Memory (LSTM) model, which belongs to the family of Recurrent Neural Networks (RNNs) and is specifically designed to capture long-range dependencies within sequential data. LSTMs have shown significant success in several fields, such as forecasting commodity prices, electricity demand, and river stream flows. However, the effectiveness of LSTM models is greatly influenced by the careful selection of hyperparameters, including the number of hidden layers, time steps, and the learning rate. An improper choice of these settings can result in lower prediction accuracy or a slower model training process, which is why hyperparameter optimization techniques are crucial for enhancing the performance of LSTMs [2].

Various optimization algorithms have been used to overcome these challenges, including Grid Search, Particle Swarm Optimization (PSO), and Genetic Algorithm (GA). Each method has different characteristics in terms of search effectiveness and computational efficiency. Grid Search has advantages in exhaustive search but requires high computational time [3], while PSO and GA offer more efficient heuristic approaches but sometimes it is difficult to guarantee global convergence [4].

Considering the crucial role of hyperparameter optimization in enhancing the performance of LSTM models, and the distinct characteristics of each optimization method, a comparative analysis is needed to assess their relative effectiveness. This study focuses on the application of Grid Search, PSO, and GA to tune LSTM for coal production forecasting, evaluating model accuracy based on two key performance metrics: Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). The findings of this research are expected to offer empirical insights into the most suitable and accurate optimization method for hyperparameter tuning in coal production time series modeling.

B. Problem Statement

In the context of modeling coal production in the United States, which is characterized by dynamic and nonlinear data patterns, deep learning architectures such as Long Short-Term Memory (LSTM) are one of the promising approaches to prediction. However, the predictive performance of LSTM models is highly dependent on the proper configuration of hyperparameters. To that end, various optimization methods such as Particle Swarm Optimization (PSO), Grid Search, and Genetic Algorithm (GA) have been used to improve the accuracy of such models. While each method has shown potential in the forecasting field, there is still a lack of empirical studies that directly compare the effectiveness of these three approaches in optimizing LSTMs for coal production forecasting, particularly in the energy sector in the United States.

1. How do LSTM models tuned with PSO, Grid Search, and GA compare in terms of prediction accuracy for coal production in the United States, especially when evaluated using performance metrics such as Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE)?
2. Which hyperparameter optimization method (PSO, Grid Search, or GA) leads to the best overall predictive performance of the LSTM model?

C. Objective

This study aims to evaluate the effectiveness of LSTM models optimized through three distinct methods. The specific objectives of the study are as follows:

1. To analyze and compare the predictive accuracy of LSTM models in forecasting coal production in the United States by applying three different optimization techniques (Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Grid Search) using evaluation metrics such as Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE).
2. To identify the most effective LSTM model configuration for energy production forecasting systems by determining the best-performing optimizer among PSO, GA, and Grid Search.

D. Limitation

To ensure clarity and focus, this study was conducted under several defined limitations, as outlined below:

1. The data utilized in this study comprises monthly coal production records from 1973 to 2024 in the United States, sourced from reputable institutions such as the U.S. Energy Information Administration (EIA).
2. This research centers around a specific deep learning model, Long Short-Term Memory (LSTM), and assesses its performance after being optimized with three distinct hyperparameter tuning methods: Particle Swarm Optimization (PSO), Grid Search, and Genetic Algorithm (GA).
3. The evaluation of model performance was carried out using statistical measures, including Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE).

E. Controbution

This study aims to contribute to the growing body of research on the application of deep learning in energy forecasting by offering both theoretical and practical advancements. The contributions of this research can be categorized into the following three areas:

1. Theoretical Contribution

   This study provides new insights into the effectiveness of different hyperparameter optimization techniques—Particle Swarm Optimization (PSO), Grid Search, and Genetic Algorithm (GA)—in enhancing the performance of Long Short-Term Memory (LSTM) networks, specifically in modeling coal production data.
2. Methodological Contribution

   It introduces a systematic comparison of three distinct optimization methods applied to the same deep learning model (LSTM), offering a structured evaluation of their impact on prediction accuracy using metrics such as Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE).
3. Practical Contribution

   The research proposes a predictive modeling framework that can serve as a reference for developing data-driven coal energy monitoring and planning systems, thereby supporting more informed decision-making in industrial operations and energy policy formulation.

II. RELATED WORK

Chung and Shin (2018) introduced a hybrid model that merges genetic algorithms (GA) with LSTM networks for forecasting stock market trends. In their approach, GA automatically configured the LSTM model by selecting parameters like the time window size and the number of hidden units. This method proved effective for analyzing random and non-stationary financial time series. The genetic algorithm allowed for a more structured tuning process, removing the need for labor-intensive manual experimentation. Testing on KOSPI index data revealed that the GA-LSTM model achieved better predictive accuracy than traditional benchmarks [5].

Xu et al. (2022) presented a flood forecasting method that uses an LSTM model optimized with Particle Swarm Optimization (PSO). This research focused on modeling rainfall-runoff dynamics in two river basins in China, aiming to enhance both accuracy and training speed. PSO was used to fine tune crucial LSTM parameters such as neuron count, time steps, and batch size. Their results showed that the PSO-LSTM model delivered more stable and accurate forecasts than conventional models, while also being more efficient to train making it well-suited for real-time forecasting applications [6].

Mubarak et al. (2024) developed a hybrid CNN-BiLSTM-AR model for forecasting electricity prices one day ahead and explored the role of hyperparameter optimization (HPO) methods in improving model accuracy and efficiency. In order to optimize the parameters of the deep learning model, this study uses three optimization techniques: Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Random Search (RS). It shows that PSO consistently produces the lowest error (RMSE and MAE) and a relatively efficient convergence time. By demonstrating that the choice of HPO approaches can have a substantial impact on prediction performance and computational complexity in LSTM-based time series models, this paper provides a significant contribution. As a result, this method is a crucial point of reference in our study for evaluating how well different optimizers perform on LSTM models when it comes to predicting coal production in the US [7].

III. METHODOLOGY

A. Theoretical Background and Related Concepts

1) Recurrent Neural Networks (RNN)

A Recurrent Neural Network (RNN) refers to a specialized architecture within artificial neural networks that is structured to effectively handle data characterized by sequential or temporal dependencies. This model is particularly well-suited for tasks where the order and context of input data points are crucial, enabling it to capture patterns over time through its internal memory mechanism. Its primary advantage lies in its ability to retain historical information through internal memory that is integrated into its network structure. Unlike feedback loops, RNNs incorporate a looping mechanism that allows data from previous time steps to be recycled and used as contextual information for processing subsequent inputs. This architectural feature makes RNNs highly effective for modeling temporal dynamics and dependencies over both short and long-time horizons. Consequently, RNNs are highly suitable for applications such as natural language processing, time series prediction, and sequential pattern analysis [8].

At each time step t, the RNN takes an input vector x_t along with the hidden state from the previous time step h_{t-1}, and computes the new hidden state h_t using the following equation:

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h$$

where σ represents a nonlinear activation function (commonly *tanh* or *ReLU*), W_{xh} and $W_{hh}$are weight matrices, and b_h is the bias vector. The output of the RNN at time step t is then calculated as:

$$y_t = \phi(W_{hy}h_t + b_t)$$

RNNs are widely employed in applications such as time series forecasting, speech recognition, and natural language processing due to their capacity to detect patterns in sequential data. However, traditional RNN models face significant challenges in capturing long-term dependencies because of the vanishing gradient problem during training using the Backpropagation Through Time (BPTT) algorithm [9]. To overcome this limitation, more sophisticated architectures like Long Short-Term Memory (LSTM) have been introduced. These models incorporate gating mechanisms that enable them to retain and transmit relevant information across longer sequences, improving their capacity to capture long-term dependencies more effectively.

2) *Long Short-Term Memory (LSTM)*

The Long Short-Term Memory (LSTM) model, an improved form of RNN, was proposed by Hochreiter and Schmidhuber in their 1997 *Neural Computation* paper. It is stated that RNNs such as BPTT and RTRL have difficulty maintaining error flow in the long run. This shortcoming can be overcome by LSTM which is designed with many activation functions in its intermediate layer with complex operations on various gates [10]. The network in LSTM is a neural network with nodes that are not wrapped over time, so it has the possibility to utilize previous calculations. The configuration of the LSTM network through several hyperparameters correctly so that it has a good effect on the performance of the model. LSTM is very sensitive to network parameters that will affect the performance of the model such as the number of hidden layers, the number of unit cells in each layer, the activation function, the size of the input history time, and so on [11].

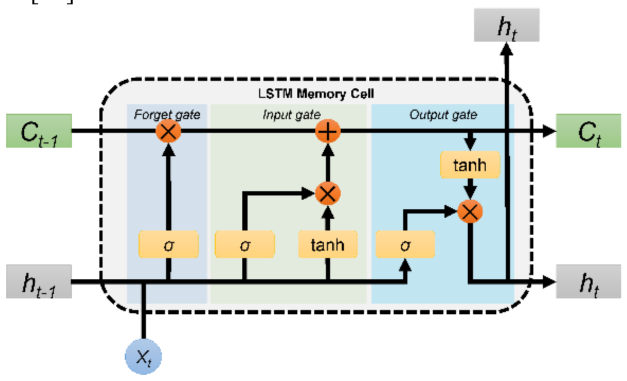

Figure 1. LSTM Structure

The components of LSTM consist of cells, input gate, output gate, and forget gate. Cells in LSTM function to store values at changing time intervals, so the network has the possibility to store information over a long period of time [12]. Each LSTM unit consists of the following parts.

1. Cell state ($\tilde{C}_t$): stores long-term memory.
   $\tilde{C}_t = \tanh(w_C x_t + u_c h_{t-1} + b_c)$
2. Input gate (i_t): controls how much new information from input is added to memory.
   $i_t = \sigma(w_i + u_i h_{t-1} + b_i)$
3. Forget gate (f_t): decides which information should be discarded from memory.
   $f_t = \sigma(w_o x_t + u_o h_{t-1} + b_0)$
4. Output gate (o_t): determines how much of the current memory is passed to the output hidden stat (h_t)
   $o_t = \sigma(w_o x_t + u_o h_{t-1} + b_0)$
   $h_t = o_t \otimes \tanh(C_t)$
5. Memory Update. The last memory C_t is updated with the following equation.
   $C_t = (f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t)$

3) *Grid Search*

Grid Search (GS) is a method that involves discretizing the range of each hyperparameter (HP) and systematically evaluating all possible combinations of their values. For numerical and integer HPs, the values are typically spaced evenly within predefined boundaries. The number of distinct values tested for each HP is referred to as the grid resolution. For categorical HPs, either all possible values or a selected subset may be considered [13]. The procedure for choosing hyper-parameters and evaluating the model is carried out through a grid search method with cross-validation.

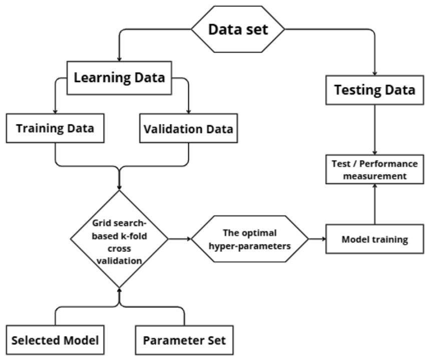

Figure 2. The Process of Hyper-parameter Selection and Model Evaluation with Grid Search

Grid Search is one of the most commonly used techniques for hyperparameter tuning in machine learning models. This method works by dividing the search space into fixed combinations of values for each hyperparameter, then exhaustively evaluating all these combinations [14]. For example, if there are three hyperparameters with five candidate values each, a total of 125 configurations would need to be tested. Although Grid Search is easy to implement and can be parallelized, it becomes highly inefficient in high-dimensional search spaces due to the exponential growth of combinations a problem commonly referred to as the *curse of dimensionality*.

4) *Particle Swarm Optimization (PSO)*

Particle Swarm Optimization (PSO) is a population-based metaheuristic algorithm introduced by Kennedy and Eberhart in 1995. This method is designed to mimic the dynamics of social behavior seen in the collective movements of organisms such as birds in a coordinated search for food sources. Through the principle of collaboration between individuals in a population, PSO enables efficient search for optimal solutions in complex search spaces. Since its introduction, the algorithm has been widely used in various optimization applications, especially in dealing with nonlinear and multidimensional global solution search problem [15].

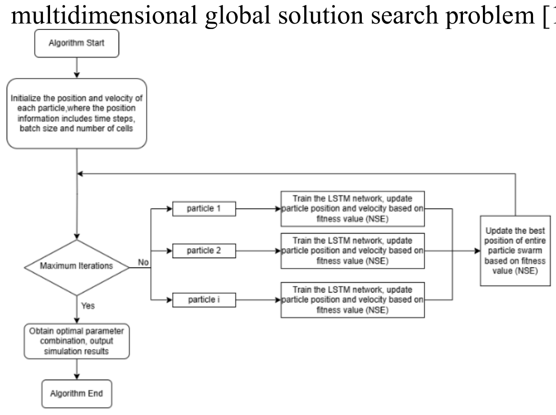
Algorithm Start
Initialize the position and velocity of each particle,where the position information includes time steps, batch size and number of cells
Maximum Iterations
No
Yes
particle 1
particle 2
particle i
Train the LSTM network, update particle position and velocity based on fitness value (NSE)
Train the LSTM network, update particle position and velocity based on fitness value (NSE)
Train the LSTM network, update particle position and velocity based on fitness value (NSE)
Update the best position of entire particle swarm based on fitness value (NSE)
Obtain optimal parameter combination, output simulation results
Algorithm End

Figure 3. Flowchart of PSO Algorithm in LSTM Model

In the Particle Swarm Optimization (PSO) framework, the search for an optimal solution is carried out collectively by a group of agents referred to as particles that navigate within an N-dimensional search space [16]. Each particle represents a candidate solution and is characterized by three main attributes that guide its movement throughout the optimization process: position, velocity, and personal best position. These attributes are mathematically defined as follows:

$$X_i = (x_{i1}^t, x_{i2}^t, \dots, x_{iN}^t)^T$$

$$V_i = (v_{i1}^t, v_{i2}^t, \dots, v_{iN}^t)^T$$

$$P_i = (p_{i1}^t, p_{i2}^t, \dots, p_{iN}^t)^T$$

$$P_g = \left(p_{g1}^t, p_{g2}^t, \dots, p_{gN}^t\right)^T$$

Where:
X_i : Position vector of the particle, representing a potential solution;
V_i : Particle's velocity;
P_i : Personal best position found so far by the particle;
P_g : Global best position discovered by the entire swarm.

The velocity and position of each particle are update at each iteration according to the following equations:

$$v_i^{t+1} = w.v_i^t + c_1.r_1.(p_i^t - x_i^t) + c_2.r_2.(g^t - x_i^t)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

Here, w denotes the inertia weight that controls the momentum of the particle; c_1 and c_2 are the cognitive and social learning factors, respectively; r_1 and r_2 are uniformly distributed random values in the range [0, 1]. The first component reflects self-confidence, while the second encourages learning from others.

In this study, PSO is employed to optimize the hyperparameters of LSTM models, specifically the number of hidden units, number of layers, dropout rate, and learning rate. Each particle encodes a different hyperparameter combination, and the prediction performance of the corresponding deep learning model is evaluated using a fitness function based on RMSE and MAPE. The PSO algorithm iteratively updates the position of each particle in the parameter space to minimize prediction error, thereby identifying the most effective hyperparameter settings for accurate time series forecasting. The hyperparameter optimization process using the PSO algorithm in the LSTM model can be explained more clearly through the following flowchart.

5) *Genetic Algorithm*

The Genetic Algorithm (GA) is a form of metaheuristic method influenced by the principles of biological evolution and natural selection. Initially proposed by John Holland in 1975, it was further developed by David Goldberg [17], GA is part of the evolutionary algorithm family and is commonly used for solving complex optimization problems, particularly in situations where optimal solutions are difficult to determine analytically or through traditional methods.

In general, GA operates by creating an initial population of individuals (candidate solutions), each represented by a chromosome. The chromosome consists of multiple genes that represent the solution's parameters. This population is evaluated based on an objective function, known as the fitness function. The individuals with the highest fitness values are considered the best and are more likely to be chosen for reproduction [17].

The three key operators in GAs are:

1. Selection: Choosing the top individuals from the population based on their fitness for reproduction.
2. Crossover: Combining two parent individuals to generate new offspring, passing on characteristics from both parents.
3. Mutation: Introducing small random changes to the genes within chromosomes to preserve diversity within the population and avoid early convergence.

The GA process is iterative, continuing until a stopping condition is fulfilled, such as reaching a set number of generations or achieving the targeted fitness level. Each iteration represents a new generation, which is expected to have solutions of higher quality than the previous generation [17].

Thus, GA can be seen as an adaptive, population-based optimization method that works well in large, complex solution spaces. Its strength lies in its ability to explore and exploit solutions effectively without needing derivative information (gradients) from the objective function.

6) *MinMax Scaler*

The goal of the scaler is to rescale all data to the same range. Data scaling is used to prevent dominant patterns of coal production behavior from influencing the characteristics of smaller coal production. The actual patterns of coal production behavior that generally apply will be provide using the same range of values. To maintain coal production in line with actual production patterns, the following is the approach from MinMax scaler.

$$x'_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Coal data values are scaled using this method into a range of 0 to 1. We applied the MinMax scaler method to ensure that all coal production values are normalized to the same scale.

7) Time Complexity

Time complexity is the computation time calculated based on the number of steps of the Turing machine to produce $\varepsilon = 2^{-n}$ approximations of the output, allowing the use of advice strings that depend on the input size n (non-uniform) (Hemmerling, 2000). According to Hemmerling (2000), for $r \in R$ in an Oracle Turing Machine (OTM) with a binary convergent sequence φ and n∈N is defined as follows.

$$time_M(r, n) = sup\{time_M(\varphi, n) : \varphi \in CF(r)\}$$

With the value of $time_M(r, n)$ having recursive properties and being in N if M computes a real function on r.

The function $f : R \to R$ can be computed approximatively using the time limit $t : N \to N$ if there is an OTM with M [18]. According to Hemmerling (2000), since there is an OTM with M, then M computes f approximatively and all n and r in the domain will satisfy the time with the following equation.

$$time_M(r, n) \le t(n)$$

The general time complexity TTT of Particle Swarm Optimization (PSO) can be represented as follows.

$$T = O(N * I * D * C)$$

Let N denote the number of particles (swarm size), I the total iterations, D the dimensionality of the search space, and C the computational cost of evaluating the fitness function. Overall, PSO is widely regarded as an efficient optimization algorithm [19].

B. Dataset Description

The dataset used in this study comprises historical monthly records of coal production in the United States, measured in thousands of tons, and sourced from the U.S. Energy Information Administration (EIA). Spanning from January 1973 to April 2025, the dataset forms a continuous time series consisting of over 628 monthly observations. A key advantage of this dataset is its completeness and consistency, with no missing values, ensuring high reliability for modeling purposes.

Due to its extended temporal coverage and consistent observation intervals, the dataset is highly suitable for training deep learning models such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which rely on detecting temporal patterns within sequential data. The dataset captures both seasonal variations and long-term production trends, reflecting real-world complexity in energy output. Therefore, it serves as a strong benchmark for assessing the performance of hyperparameter optimization using Particle Swarm Optimization (PSO) in the context of time series forecasting.

In addition, the dataset embodies long-term structural shifts in U.S. energy production, making it a valuable test case for evaluating forecasting models under realistic and evolving economic and environmental conditions.

C. Data Preprocessing

The data preprocessing stage is the first step taken to prepare the data that will be used in this research. This process includes data cleaning, normalization, and splitting. In the data cleaning stage, identification of possible null values in the dataset is carried out. This step is important to ensure that the data used is consistent and reliable in the analysis and modeling process.

After the data is cleaned, a normalization process is carried out which is useful for equalizing the scale between variables and maintaining the consistency of values in the dataset. Normalization is done using the Min-Max Scaler method, which will change the values in the dataset into a range between 0 and 1. The final stage of preprocessing is data division. The dataset is divided into two parts, namely 90% training data and 10% testing data. This process aims to allow the model to be trained with the majority of data and evaluated for performance using data that has never been seen before.

D. Predictive Models Used

This study utilizes the Long Short Term Memory (LSTM) model to perform time series prediction. To optimize the performance model, three hyperparameter optimization methods were applied: Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Grid Search. All the methods used the same search space the following parameters:

1. Number of hidden units: 16 to 64
2. Number of LSTM layers: 1 to 2
3. Dropout rate: 0.2 to 0.3
4. Learning rate: 0.0001 to 0.01

In all experiments, the following parameters were fixed:

1. Sequence length: 12
2. Number of epochs: 200
3. Batch size: 32

E. Performance Evaluation Method

In the context of predictive model optimization, loss function error also referred to as the discrepancy between actual response values and predictions, is a function used to assess model performance. The loss function errors used to forecast coal production are Mean Absolute Percentage Error (MAPE) and Rooted Mean Square (RMSE). The RMSE and MAPE values are calculated using the following formulas.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Where:
n : Number of data;
y_i : Actual of the ith data;
$\hat{y}_i$: Prediction of the ith data.

When utilizing the model, RMSE and MAPE serve as metrics to show the degree of risk and opportunity. The maximum percentage of opportunity that may be attained with the forecasting model is represented by the accuracy derived from MAPE, which is the upper limit of model accuracy. The distance that is closest to the real value is the result of normalized absolute distance, or MAPE (Manhattan distance).

IV. EXPERIMENT AND RESULT

A. Comparison Analysis and Result

After preprocessing, the dataset was split into training and testing sets and modeled using LSTM. Each model was trained with early stopping, which also served as the fitness function in PSO, Grid Search, and GA for hyperparameter tuning (hidden size, number of layers, dropout rate, learning rate). Model performance was evaluated using RMSE and MAPE on both training and testing sets to assess accuracy and generalization.

Table 1 offers a detailed summary of the experimental outcomes across three sequence length settings: 12, 24, and 36. For each sequence length, the table highlights the best-performing hyperparameter configurations identified by each optimization method, along with their associated RMSE and MAPE values. This comparative analysis facilitates a clearer understanding of which optimization technique most effectively enhances LSTM performance in time series forecasting tasks.

Table 1. Model Performance Comparison of Hyperparameter Optimization Methods on LSTM

Seq Length	Optimizer	Best Configuration	RMSE	MAPE
12	Grid Search	Hidden Size : 16 Number Layers : 1 Dropout : 0.2 Learning Rate : 0.001	Train : 6201.33 Test : 4111.68	Train : 6.51% Test : 7.44%
	Genetic Algorithm	Hidden Size : 25 Number Layers : 2 Dropout : 0.2 Learning Rate : 0.0005	Train : 8088.76 Test : 5156.41	Train : 8.53% Test : 9.95%
	PSO PSO	Hidden Size : 41 Number Layers : 1 Dropout : 0.3 Learning Rate : 0.0006	Train : 6190.9 Test : 3555.7	Train : 6.51% Test : 6.28%
24	Grid Search	Hidden Size : 16 Number Layers : 1 Dropout : 0.3 Learning Rate : 0.001	Train : 6397.88 Test : 6156.23	Train : 6.65% Test : 12.08%
	Genetic Algorithm	Hidden Size : 32 Number Layers : 1 Dropout : 0.3 Learning Rate : 0.003	Train : 7344.96 Test : 7125.59	Train : 7.59% Test : 13.7%
	PSO	Hidden Size : 44 Number Layers : 1 Dropout : 0.2 Learning Rate : 0.0002	Train : 6178.24 Test : 4633.64	Train : 6.43% Test : 8.67%
36	Grid Search	Hidden Size : 16 Number Layers : 1 Dropout : 0.25 Learning Rate : 0.001	Train : 5706.13 Test : 6638.75	Train : 6.04% Test : 13.6%
	Genetic Algorithm	Hidden Size : 44 Number Layers : 1 Dropout : 0.23 Learning Rate : 0.003	Train : 7547.11 Test : 7571.61	Train : 7.63% Test : 15.34%
	PSO	Hidden Size : 47 Number Layers : 1 Dropout : 0.23 Learning Rate : 0.006	Train : 5792.92 Test : 4163.86	Train : 5.98% Test : 7.94%

The results presented in the table clearly indicate that Particle Swarm Optimization (PSO) consistently outperforms both Grid Search and the Genetic Algorithm (GA) across all examined sequence lengths. Specifically, for the sequence length of 12, PSO achieves the best performance with the lowest Root Mean Square Error (RMSE) on the test dataset, recorded at 3555.7, and the lowest Mean Absolute Percentage Error (MAPE) at 6.28%. This optimal result is attained using a configuration comprising 41 hidden units, a single LSTM layer, a dropout rate of 0.3, and a learning rate of 0.0006. These findings suggest that PSO is particularly efficient in exploring the hyperparameter space and converging on configurations that yield higher predictive accuracy.

A similar trend is observed for the sequence length of 24, where PSO again delivers superior results, with a test RMSE of 4633.64 and a MAPE of 8.67%. These values are markedly better than those obtained through Grid Search or GA. Furthermore, for the longest sequence length of 36, PSO continues to lead, achieving a test RMSE of 4163.86 and a MAPE of 7.94%. In contrast, both alternative optimization strategies yield higher error metrics and greater variability in prediction accuracy, reinforcing the robustness of PSO in fine-tuning LSTM model parameters for time series forecasting.

As a result of a thorough evaluation across all experimental conditions, the configuration obtained through Particle Swarm Optimization (PSO) with a sequence length of 12 is identified as the most appropriate model for deployment. This selection is supported by the configuration's consistently strong performance in both Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE), along with its relatively simple model architecture, which consists of a single LSTM layer. This simplicity contributes to its computational efficiency and facilitates practical implementation.

The optimal hyperparameter setting includes 41 hidden units, one LSTM layer, a dropout rate of 0.3, and a learning rate of 0.0006. Among all evaluated models, this configuration achieved the lowest test error, yielding an RMSE of 3555.7 and a MAPE of 6.28 percent. In addition, the close alignment between training and testing metrics indicates that the model does not suffer from overfitting. The training MAPE stands at 6.51 percent, which is very close to the 6.28 percent obtained on the test set. Similarly, the RMSE values show a consistent pattern between the two datasets. These outcomes highlight the model's strong generalization ability, confirming the robustness and reliability of the PSO-optimized configuration for real-world forecasting applications.

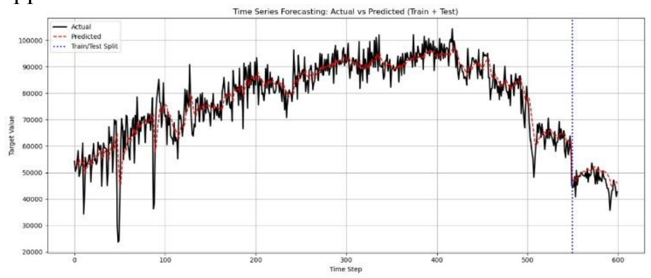

Figure 4. Plot of Actual vs Predicted in Training and Testing Data

Figure 4 further demonstrates this by illustrating the comparison between actual and predicted values throughout both the training and testing phases. The predicted values exhibit a strong alignment with the actual data over time, showing only slight deviations—particularly during the testing period. This close correspondence visually reinforces the evaluation metrics discussed earlier and offers a clear, intuitive indication of the model's accuracy and its ability to generalize effectively.

Although the predictive accuracy of the LSTM model has been validated using metrics such as Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE), successful implementation in practical contexts also demands careful consideration of computational efficiency, particularly in terms of processing time. In real-world applications where large datasets are involved or frequent model updates are necessary, the duration required for hyperparameter tuning can directly influence the performance and adaptability of the forecasting system. To address this aspect, the study expands its evaluation by comparing the time complexity of several hyperparameter optimization methods, including Particle Swarm Optimization (PSO), Grid Search, and Genetic Algorithm (GA). This comparative analysis is intended to identify which method achieves the most effective balance between forecasting accuracy and time efficiency, thereby offering practical insights for improving the operational feasibility of coal production forecasting models.

Table 2 Comparison Result of Time Complexity

Seq Length	Grid Search (s)	Genetic Algorithm (s)	PSO (s)
12	45.4	40.1	22.9
24	49.7	27.9	24.9
36	67.3	68.2	37.7

Table 2 provides a comparative analysis of the time complexity for three hyperparameter optimization techniques: Grid Search, Genetic Algorithm (GA), and Particle Swarm Optimization (PSO), applied to the LSTM model across various sequence lengths. The results indicate that PSO consistently demonstrates the lowest computational time in all scenarios. For a sequence length of 12, PSO required only 22.9 seconds, compared to 45.4 seconds for Grid Search and 40.1 seconds for GA. A similar trend is observed for the sequence lengths of 24 and 36, where PSO again outperformed the other methods with execution times of 24.9 and 37.7 seconds, respectively. These differences in computational time across methods are further illustrated in Figure 5 for clearer comparison.

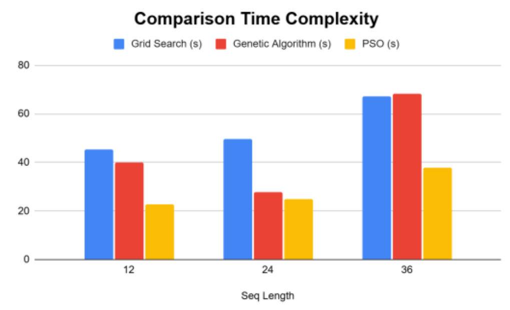

Figure 5. Time complexity comparison of Grid Search, Genetic Algorithm, and PSO

In contrast, Grid Search exhibited the highest time complexity overall, which is expected due to its exhaustive nature in evaluating all possible parameter combinations. GA performed better than Grid Search but was still less efficient than PSO in most cases. These findings highlight the computational advantage of PSO in terms of optimization speed, making it a more practical choice for real-time or large-scale forecasting applications where efficiency is critical.

V. CONCLUSION AND FUTURE WORK

A. Conclusion

This study comprehensively evaluates the performance of LSTM models in predicting coal production in the United States by comparing three optimization methods: PSO, Grid Search, and GA. The main findings consistently show that PSO provides superior prediction performance compared to Grid Search and GA for all sequence lengths tested. Specifically, the LSTM model configuration optimized with PSO at a sequence length of 12 achieves the best results, with an RMSE of 3555.7 and a MAPE of 6.28% on the test data. This optimal configuration uses 41 hidden units, 1 LSTM layer, a dropout rate of 0.3, and a learning rate of 0.0006.

In addition to accuracy, the close alignment between training and testing metrics confirms that the PSO-optimized model does not suffer from overfitting and demonstrates strong generalization capabilities. The simplicity of this model architecture, which uses only one LSTM layer, also contributes to computational efficiency and ease of practical implementation. Overall, the predictive modeling framework developed in this study offers a valuable reference for creating data-driven coal energy monitoring and planning systems, thereby supporting more informed decision-making in industrial operations and energy policy formulation. Future research could explore the integration of other deep learning models or ensemble methods to potentially enhance prediction accuracy and robustness.

B. Limitations and Suggestions for Future Research

The results obtained in the comparative analysis of PSO, Genetic Algorithm, and Grid Research for LSTM hyperparameter optimizer on coal data and its forecasting produce quite good results and answer the problem formulation. However, in the analysis conducted in this study there are still limitations that have not been covered.

First, the analysis conducted in this study only uses one type of deep learning model, namely Long Short-Term Memory (LSTM), without comparing it with other architectures such as Gated Recurrent Unit (GRU), Temporal Convolutional Networks (TCN), or Transformer. This makes this analysis limited in understanding the relative performance of various models in the context of coal production time series data. In fact, these architectures are also suitable for time series data, especially in handling long-term dependencies and complex variations in the data. Furthermore, the model built in this analysis only relies on historical production data, without considering external variables that have the possibility of affecting the prediction in real terms. Possible external variables include economic indicators, regulatory policies, or weather factors. Therefore, based on the limitations of this analysis, it is expected that future studies will explore and compare the performance of these models. By doing so, the most optimal approach can be obtained in the context of coal production forecasting.

Second, the static dataset in this analysis evaluates the performance of the model by dividing the data into training and testing data. However, in reality or field practice, data is dynamic and continuously updated. Then, this approach has not described the needs of a prediction system that is suitable for data changes. Therefore, based on the limitations of this analysis, it is expected that in future analyses to improve the real-world application, and consider the implementation of online learning or incremental learning that will make the model adjust in real-time to new data, without the need to completely retrain.

Third, the analysis in this study uses three hyperparameter optimization methods, namely PSO, Grid Search, and Genetic Algorithm (GA). The results obtained show that PSO has advantages in terms of accuracy and time efficiency. However, the analysis in this study does not include more modern optimization methods such as Bayesian Optimization or Hyperband which have greater potential in the efficiency of parameter space exploration. Furthermore, despite analyzing the computation time, this evaluation still produces results that are limited to scenarios with LSTM architecture and certain dataset sizes. Therefore, based on the limitations of this analysis, it is expected that in future analysis to consider applications that have a large scale or frequent updates, which certainly require further testing to ensure the scalability and time efficiency of the methods used.

Considering some of the limitations of the analysis in this study, it is hoped that future research can produce a prediction model that is not only more accurate and adaptive, but also efficient and can be applied in real-world or field scenarios, especially in the dynamic and complex coal sector.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to Mr. Muhammad Syukron from the Department of Statistics, Institut Teknologi Sepuluh Nopember (ITS), for the valuable guidance, insightful feedback, and continuous support throughout the development of this research. Their expertise and encouragement have been instrumental in the completion of this study.

REFERENCES

[1] J. Runge and R. Zmeureanu, “A review of deep learning techniques for forecasting energy use in buildings,” *Energies*, vol. 14, no. 3, p. 608, 2021. [Online]. Available: 10.3390/en14030608

[2] Hemmerling, Armin. (2000). On the Time Complexity of Partial Real Function. Research Article in Journal of Complexity, 16, 363-376. http://dx.doi.org/10.1006/jcom.1999.0539

[3] H. Chung and K. Shin, “Genetic algorithm–optimized long short-term memory network for stock market prediction,” *Sustainability*, vol. 10, no. 10, p. 3765, 2018. [Online]. Available: https://www.mdpi.com/2071-1050/10/10/3765

[4] H. Dhake, Y. Kashyap, and P. Kosmopoulos, “Algorithms for hyperparameter tuning of LSTMs for time series forecasting,” *Remote Sens.*, vol. 15, no. 8, p. 2076, 2023. [Online]. Available: 10.3390/rs15082076

[5] Lawi, A., Mesra, H., & Amir, S. (2022). Implementation of Long Short-Term Memory and Gated Recurrent Units on grouped time-series data to predict stock prices accurately. Journal of Big Data, 9(1), 89. https://doi.org/10.1186/s40537-022-00597-0

[6] Y. Xu, C. Hu, Q. Wu, et al., “Research on particle swarm optimization in LSTM neural networks for rainfall–runoff simulation,” J. *Hydrol.*, vol. 608, p. 127553, 2022. [Online]. Available: 10.1016/j.jhydrol.2022.127553

[7] H. Mubarak, A. Abdellatif, S. Ahmad, M. Z. Islam, S. M. Muyeen, M. A. Mannan, and I. Kamwa, “Day-ahead electricity price forecasting using a CNN-BiLSTM model in conjunction with autoregressive modeling and hyperparameter optimization,” *Int. J. Electr. Power Energy Syst.*, vol. 161, p. 110206, 2024. [Online]. Available: 10.1016/j.ijepes.2024.110206

[8] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” *arXiv preprint arXiv:1506.00019*, 2015. [Online]. Available: https://arxiv.org/abs/1506.00019

[9] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994. [Online]. Available: 10.1109/72.279181

[10] A. Lawi, H. Mesra, and S. Amir, “Implementation of Long Short-Term Memory and Gated Recurrent Units on grouped time-series data to predict stock prices accurately,” *J. Big Data*, vol. 9, no. 1, p. 89, 2022. [Online]. Available: 10.1186/s40537-022-00597-0

[11] H. Dhake, Y. Kashyap, and P. Kosmopoulos, “Algorithms for Hyperparameter Tuning of LSTMs for Time Series Forecasting,” *Remote Sens.*, 2023. [Online]. Available: 10.3390/rs15082076

[12] K. D. Rao, A. Ramakrishna, M. Ramesh, P. Koushik, S. Dawn, P. Pavani, T. S. Ustun, and U. Cali, “A Hyperparameter-Tuned LSTM Technique-Based Battery Remaining Useful Life Estimation Considering Incremental Capacity Curves,” *IEEE Access*, vol. 12, pp. 127259–127271, 2024. [Online]. Available: 10.1109/ACCESS.2024.3450871

[13] B. Bischl *et al.*, “Hyperparameter optimization: Foundations, algorithms, best practices and open challenges,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 13, no. 2, e1484, 2023. [Online]. Available: 10.1016/j.csite.2023.103813

[14] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[15] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. ICNN’95 - Int. Conf. Neural Netw*, vol. 4, pp. 1942–1948, 1995. [Online]. Available: 10.1109/ICNN.1995.488968

[16] R. K. Huda and H. Banka, “New efficient initialization and updating mechanisms in PSO for feature selection and classification,” *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3283–3294, 2020. [Online]. Available: 10.1007/s00521-019-04395-3

[17] T. Alam, S. Qamar, A. Dixit, and M. Benaida, “Genetic algorithm: Reviews, implementations and applications,” *Int. J. Eng. Pedagog.*, vol. 10, no. 6, pp. 106–117, 2020. [Online]. Available: 10.3991/ijep.v10i6.14567

[18] A. Hemmerling, “On the Time Complexity of Partial Real Function,” *J. Complexity*, vol. 16, pp. 363–376, 2000. [Online]. Available: 10.1006/jcom.1999.0539

[19] N. E. Udenwagu, A. A. Oni, and A. A. Ezenwoke, "Comparative study of different inertia weight strategies in particle swarm optimization based on actual computational time cost," *Int. J. Theor. Appl. Mech.*, vol. 9, pp. 1–13, 2025.