Comparative Analysis of Deep Learning Methods for Predicting the Value of the Standard & Poor's Global Supply Chain Intelligence (S&P GSCI) Nickel Stock Index

[1]Ragada Rahmansyah, [2*]Anik Vega Vitianingsih, [3]Rusdi Hamidan, [4]Anastasia Lidya Maukar, [5]Yoyon Arie Budi Suprio
[1,2,3]Informatics Department, Universitas Dr. Soetomo, Surabaya, Indonesia
[4]Industrial Engineering Department, President University, Bekasi, Indonesia
[5]Informatics Department, STIKOM PGRI Banyuwangi, Indonesia
Email: [1]ragaskal123@gmail.com, [2]vega@unitomo.ac.id, [3]rusdi@unitomo.ac.id, [4]almaukar@president.ac.id, [5]yoyonstikom@gmail.com

Article Info
Article history:
Received Mar 11th, 2025
Revised Apr 30th, 2025
Accepted May 26th, 2025

Keyword:
Forecasting
GRU
LSTM
Nickel
RNN

ABSTRACT
The development of information technology has opened up new opportunities in stock market forecasting, especially in nickel commodities, which are increasingly strategic in the global energy transition. This study uses a Deep Learning Method as Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and a Gated Recurrent Unit (GRU) to forecast the movement of the S&P GSCI Nickel stock index value. Yahoo Finance time series data for the years 2018–2024 are used in the dataset. The study's findings are used to evaluate each model's capacity to forecast changes in nickel stock prices. The RNN model is used in this study because it can work with sequential information, while LSTM works with three memory gates (input, forget, output), and GRU works with 2 gates, namely update and reset. Mean Absolute Percentage Error (MAPE) presents the results of open and closed variable forecasting errors with the lowest average for the RNN model of 2.08%, the LSTM model of 2.505%, and the GRU model of 1.505%. This study is expected to contribute to investor decision-making and the identification of the most accurate forecasting model for the nickel stock index.

Corresponding Author:
Anik Vega Vitianingsih
Informatics Department, Universitas Dr. Soetomo
Semolowaru Street No.84, Menur Pumpungan, Sukolilo District, Surabaya, East Java 60118
Email: vega@unitomo.ac.id

DOI: http://dx.doi.org/10.24014/ijaidm.v8i2.36129

1. INTRODUCTION

Information technology is not difficult to access because it has become a part of people's life [1]. The field of information technology is constantly evolving and becoming more advanced; as a result, it has the potential to bring about a multitude of benefits that the lives of individuals can benefit from [1]. One of the efforts to utilize information technology is currently being implemented in forecasting[2]. In order to meet the demand for goods or services, forecasting involves calculating the number of future requirements, including quantity, quality, time, and location [2]. Forecasting has many purposes, including projecting future values, one of which is to analyze the stock market [2]. One of the stocks now the target of investors is nickel due to the increasing global demand for nickel commodities caused by the transformation of fossil energy utilization towards environmentally friendly electrical energy using batteries with nickel raw materials [3].

Indonesia is one of the largest nickel producers globally, and its strategic role should be taken into account in the global market to meet the growing demand for nickel [4]. In recent years, nickel has increasingly been considered a viable alternative in the transition from fossil fuels to environmentally friendly energy sources. One notable use is in the transportation manufacturing industry, where nickel serves as a key raw material for electric vehicle batteries [5]. This surge in demand has attracted strong investor interest in the nickel sector, particularly in the stock market. One prominent platform reflecting nickel's market performance is the S&P (Standard & Poor's), which provides index-based assessments of real assets. Since acquiring the Global Supply Chain Intelligence (GSCI) from Goldman Sachs in 2007, S&P has maintained the GSCI as a benchmark for global commodity futures contracts. In practice, the S&P GSCI Nickel index demonstrates considerable price fluctuations, which are influenced by various factors including technical production issues, global economic conditions, and national downstream industrial policies promoted by the Indonesian government. These conditions complicate accurate prediction of nickel stock index movements [6]. Furthermore, selecting the most suitable forecasting model and optimizing hyperparameters for machine learning methods such as Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) remains a complex challenge. Currently, there is no specialized application or model to support investors in forecasting the S&P GSCI Nickel index. This highlights the urgency and relevance of research that applies and compares these algorithms in order to assist decision-making in nickel stock investments.

Numerous studies have explored the implementation of Deep Learning (DL) algorithms in forecasting stock prices and economic indicators, particularly emphasizing models such as RNN, LSTM, and GRU due to their proven capability in capturing temporal dependencies in sequential data. Arfan and ETP [7], for example, investigated the performance of LSTM in predicting the stock prices of several sharia-compliant Indonesian companies, namely PT Aneka Tambang Tbk, Erajaya Swasembada Tbk, Kalbe Farma, Semen Indonesia, and Wijaya Karya. Their study reported highly accurate results, with Mean Absolute Percentage Error (MAPE) values ranging from 1.51% to 2.66%, demonstrating LSTM's suitability for modeling non-linear time series in stock forecasting. In contrast, Pakaja et al [8] utilized the Autoregressive Integrated Moving Average (ARIMA) model on five major Indonesian stocks Adaro Energy Indonesia Tbk (ADRO), Aneka Tambang Tbk (ANTM), Bank Central Asia Tbk (BBCA), Astra International Tbk (ASII) and Bank Negara Indonesia (Persero) Tbk (BBNI) and obtained more varied MAPE results, with values peaking at 7.93%, indicating limitations in ARIMA's ability to handle volatile and dynamic stock data. Meanwhile, Yulisa et al [9] conducted a comparative study between LSTM and GRU for forecasting Indonesia's oil and gas exports. The results indicated that LSTM outperformed GRU slightly, achieving a MAPE of 12.8% compared to GRU's 13.3%, suggesting both models are viable but LSTM offers a marginal edge in accuracy. Furthermore, Ciptaswara [10] developed a hybrid model combining ARIMA with GRU to predict palm oil prices, yielding a remarkably low MAPE of 0.0117, which reflects the potential of hybrid approaches in enhancing prediction performance. Alim [11] assessed LSTM, RNN, and ARIMA on LQ45 index data and confirmed the superiority of deep learning models LSTM achieved MAPE values of 0.048 and 0.033, outperforming RNN (0.0601 and 0.0295) and ARIMA (0.7 and 0.9), thus validating LSTM's robustness in capturing long-term dependencies in financial time series. In addition, Saputra et al [12] and Rijal et al [13] focused on forecasting Indonesia's export values of oil and gas using GRU and LSTM. Saputra et al found GRU yielded a MAPE of 13.38%, while Rijal et al demonstrated LSTM's stronger accuracy at 6.23% for oil and gas and 8.18% for non-oil and gas, significantly outperforming the traditional Vector Autoregression (VAR) model, which resulted in MAPE values of 18.4% and 32.1%, respectively. These collective findings underline the effectiveness of deep learning, particularly LSTM, in financial forecasting tasks. However, despite the widespread use of RNN, LSTM, and GRU in various financial forecasting contexts, there is a noticeable absence of studies applying these methods specifically to the S&P GSCI Nickel index. Therefore, this study aims to fill that gap by conducting a comparative analysis of RNN, LSTM, and GRU models in forecasting the S&P GSCI Nickel index, contributing a novel application of machine learning to commodity index forecasting.

The purpose of this research is to compare the RNN, LSTM and GRU forecasting models using the S&P GSCI Nickel index dataset with Open and Close variables, to identify which model is the best and determine the optimal hyper-parameter combination.

2. RESEARCH METHOD

Forecasting the open and close values of S&P GSCI Nickel using The RNN, LSTM, and GRU methods uses several visualized processes based on the flow in Figure 1.

The initial phase of this study involved collecting historical data on the S&P GSCI Nickel index, specifically the daily opening and closing prices from 2008 to 2024, retrieved from the Yahoo Finance platform. Yahoo Finance is a widely recognized source for financial datasets, offering comprehensive information on stock performance, market trends, and economic indicators. After data acquisition, the preprocessing stage was carried out to enhance data quality and ensure compatibility with the machine learning models. This phase included handling missing values, converting date formats into appropriate time-series indices, and applying normalization using the Min-Max Scaler method to scale the input features within the range of 0 to 1. This normalization was essential to reduce the impact of differing value scales and to improve model convergence.

Following pre-processing, the dataset was divided into training and testing subsets using a chronological split to preserve the time-dependent structure of the data. Feature selection was performed to identify and retain the most relevant attributes specifically, the normalized opening and closing prices-serving as inputs for the predictive models.

The core modeling phase involved the implementation of three deep learning architectures: RNN, LSTM and GRU. These models were selected due to their ability to process sequential data and learn temporal dependencies, which are crucial for time-series forecasting. Each model was constructed using a standard architecture consisting of input, hidden, and output layers, with variations in internal memory mechanisms tailored to each algorithm's design.

Hyper-parameter tuning was conducted through a grid search approach to optimize critical parameters such as the number of hidden units, number of epochs, batch size, activation function. This optimization aimed to improve the generalization ability and predictive accuracy of each model. Once the training phase was complete, model performance was evaluated using the Mean Absolute Percentage Error (MAPE) metric, which provides a clear indication of forecast accuracy by comparing predicted values against actual observations. The model with the lowest MAPE value was considered the most effective in capturing the trend and volatility of the S&P GSCI Nickel index.

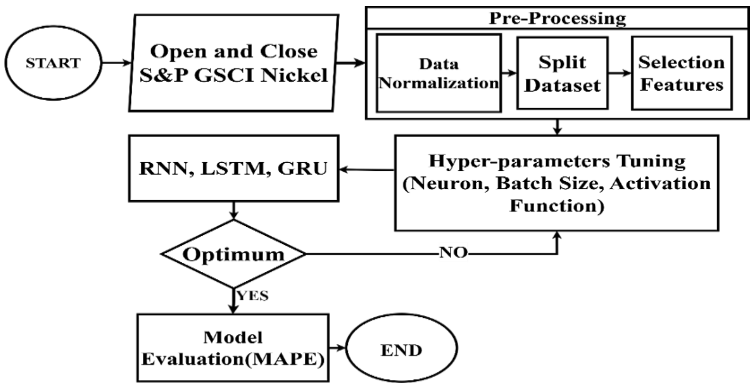

Figure 1. Flowchart System

2.1 Forecasting

Forecasting is predicting a time series's future using information from the past based on the time series and several related variables [14]. Forecasting is an integral part of decision-making in setting goals and objectives for an organization or company [15]. As management seeks to reduce reliance on uncertainty and adopt a more scientific approach to handling situations, the forecasting demand is growing [2]. Because every organization can estimate the impact of good and bad, such as scheduling, forecasting is important [15].

2.2 MinMax Scaler

MinMax Scaller normalization is a technique that transforms data into a scale between 0 and 1 [16]. This procedure, which can be computed using Equation (1), is known as MinMax normalization because it places the results within a consistent range [16]. Since **Xmax** is the largest data value, **Xmin** is the smallest, and **X** is the actual data.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

2.3 RNN Model

RNN is defined as learning that uses a cycle model of repeated connections in a network of nodes so that it can capture sequence dynamics [17]. At first, this may seem counterintuitive. However, the forward architecture of neural networks ensures a clear computational sequence. The repeating connections are well-defined, leaving no room for ambiguity. Each step uses the same basic parameters, and the RNN remains constant throughout the time and stages of the sequence [17]. Each step uses the same fundamental parameters, maintaining the RNN's consistency across time and sequence stages [17]. The RNN operates through a series of sequential computational steps, each defined by specific mathematical Equations [14]. The following analysis breaks down the core components and their several steps:

Step-1: Neural Network Layer Initialization. The process begins with initializing the network's memory mechanism, where the first hidden state is computed. This step combines the input data with the network's parameters to establish an initial representation of the sequence described in Equation (2). The hidden state $\mathrm{h(t)}$ stores information from the current timestep, acting as the network's memory. The weight matrix Wxh dictates the impact of the input $\mathrm{x(t)}$ on the hidden state while $\mathrm{b_h}$ serves as a bias term to fine-tune the calculation.

$$\mathrm{h(t) = \tan h(Wxh \cdot x(t) + b_h)} \tag{2}$$

Step-2: Hidden State Processing. After initialization, the hidden state is updated at each timestep by integrating information from both the previous hidden state and the current input. This allows the network to maintain temporal dependencies in sequential data, which can be calculated by Equation (3). Where the matrix Whh defines how the weight hidden to the hidden state from the previous timestep $\mathrm{h(t-1)}$ affects the current hidden state. The term $\mathrm{Wxh \cdot x(t)}$ captures the contribution of the new input and $\mathrm{b_h}$ remains a biased term that fine-tunes the transformation.

$$\mathrm{h(t) = tanh(Whh \cdot h(t-1) + Wxh \cdot x(t) + bh)} \tag{3}$$

Step-3: Output Generation. Once the hidden state is processed, the final step is to generate an output based on the current hidden state. This transformation maps the hidden representation into Equation (4). The output y(t) represents the network's prediction at time t, derived from the hidden state h(t)using the weight matrix Why, which maps hidden representations to output space. The bias term by adjusts the output before the final prediction.

$$\mathrm{y(t) = Why \cdot h(t) + by} \tag{4}$$

2.4 LSTM Model

LSTM is one of the architecture types of artificial neural networks (ANNs) in natural language processing and machine learning[18]. LSTMs are designed to address the problem of long-term dependency of data sequences [14]. Computer logic gates inspire the LSTM design with a new component called a memory cell, which functions as a classic long-term memory. The LSTM has several gates with different functions, such as adding, reading, and forgetting information from this memory cell. This memory cell acts as a gradient highway, allowing gates to pass through the network relatively unimpeded [14]. LSTM is the main innovation that avoids the loss of gradients in RNNs [14]. According to an analysis of the error flow in existing RNNs, which forms the basis of the LSTM architecture, present architectures cannot handle long-time lags since the backpropagated error would either increase or decrease exponentially [19]. The following stages can be used to describe how LSTM operates [14].

Step-1: LSTM Cell Initialization. The process begins with initializing the network's memory mechanism, where the first hidden and cell states are computed. Unlike standard RNNs, LSTMs maintain a memory cell Ct that helps regulate the flow of information through different gates. As explained in Equation (5), Both the current input and the previous concealed state's data are read by the input gate. Where the input gate is located the cell status determines the amount of extra information that should be added. In the input xt and the prior hidden state $\mathrm{H_{t-1}}$, the weights are defined by the matrices Wxi and Whi, respectively. The bias term b_i ensures additional flexibility in the computation.

$$\mathrm{It = \sigma(Wxi \cdot xt + Whi \cdot H_{t-1} + bi)} \tag{5}$$

Step-2: Processing in Hidden States. The forget gate chooses which data from the prior cell state should be removed to preserve pertinent information. The sigmoid activation function, which produces values between 0 and 1, is used to calculate it, as shown in Equation (6). After computing the forget gate, the new candidate cell state is updated using the input gate It, and a new candidate value $\mathrm{\tilde{C}}$ is computed using the tanh activation function in Equation (7). The final cell state Ct is then updated using Equation (8). Where the forget gate Ft controls the retention of information from the previous cell state $\mathrm{C_{t-1}}$, while the input gate It determines how much of the new candidate's value $\mathrm{\tilde{C}t}$ should be added. The element-wise multiplication $\odot$ ensures selective updating of memory.

$$\mathrm{Ft = \sigma(Wxf \cdot xt + Whf \cdot H_{t-1} + bf)} \tag{6}$$

$$\tilde{C}t = \tanh(Wxc \cdot xt + Whc \cdot H_{t-1} + bc) \quad (7)$$

$$Ct = Ft \odot C_{t-1} + It \odot \tilde{C}t \quad (8)$$

Step-3: Output Generation. The output gate controls how much cell state information should be passed to the hidden state. This transformation is computed as described in Equation (9) and (10). The output gate Ot determines how much information from the cell state Ct should be passed to the hidden state Ht. The final output yt is obtained using the weight matrix Wy, which maps the hidden representation to the output space while acting as a bias term to refine the final prediction. The three gates that are most important in information processing are the input, forget, and output gates. To regulate the information that flows across the network, the weight parameters are essential, with the weight parameters for hidden-to-input represented by W_{hi}, hidden-to-forget represented by W_{hf}, and hidden-to-output W_{ho}.
To regulate the information that flows across the network, the weight parameters are essential. The weight parameters for the hidden-to-input are represented by W_{hi}, , the hidden-to-forget by W_{hf}, , and the hidden-to-output by W_{ho}.

$$Ot = \sigma(Wxo \cdot xt + Who \cdot Ht - 1 + bo) \quad (9)$$

$$Ht = Ot \odot \tanh(Ct) \quad (10)$$

Finally, the network output at time t is computed using Equation (11). Each computational step involves the hidden state at the previous time, representing the network's short-term memory. The weight parameter U connects the input directly to the hidden layer, while the input at any given time and the bias (b) influence the computation. The result is the current hidden state (h), which reflects the complex transformation of the input information based on the advanced gate mechanism in the LSTM architecture.

$$yt = Wy \cdot Ht + by \quad (11)$$

2.5 GRU Model

GRU is an RNN model variant with a simpler structure than LSTM [14]. GRU has two main gates, the update gate and the reset gate, without using a long-term memory component and a hidden state to disseminate information [14]. The advantage of GRU is that it is simpler and only requires a smaller number of parameters, thus making GRU faster to train and more efficient use of computing resources [14]. GRU can be described in the following steps:
Step-1: Reset Gate Computation. The process begins with the reset gate, which determines how much information from the past should be forgotten. This gate computes a weighted sum of the current input xt and the H_{t-1} variable is a previous hidden state, using the respective weight matrices Wxr and Whr, followed by a bias term br. A sigmoid activation function σ, which produces values ranging from 0 to 1, is applied to the result, thereby controlling the amount of the previous hidden state that is preserved. The output Rt is explained in Equation (12).

$$Rt = \sigma(Wxr \cdot xt + Whr \cdot Ht - 1 + br) \quad (12)$$

Step-2: Update gate computation. The update gate determines the amount of fresh data that should be kept in the hidden state. It mixes the prior hidden state H_{t-1} with the current input xt, using the weight matrices Wxh and Whu and adding a bias term bu. The output is again passed through a sigmoid activation function σ, which produces a value between 0 and 1. The output of the update gate, Ut, controls how much of the previous hidden state H_{t-1} will be retained, and how much of the new candidate's hidden state will be included. The Equation for the update gate is computed in Equation (13). Update the computation of the gate. The update gate determines the amount of fresh data that should be kept in the hidden state. It mixes the prior hidden state H_{t-1} with the current input xt.

$$Ut = \sigma(Wxh \cdot xt + Whu \cdot H_{t-1} + bu) \quad (13)$$

Step-3: Candidate hidden states computation. The Candidate of the hidden state $\widetilde{H}t$, is calculated. This stage determines how much more data could be added to the hidden state. The input xt and the previous hidden state H_{t-1} (which was reset by the reset gate Rt) are combined using weighted sums utilizing the matrices Wxh and Whh, and then a bias term bh is added to determine the candidate's hidden state. The outcome is run through the tanh activation function to make sure the results fall inside a range that can be used for additional computations. It is possible to calculate the candidate's hidden state $\widetilde{H}t$ using Equation (14).

$$\widetilde{H}t = \tanh(Wxh \cdot xt + Whh \cdot (Rt \odot H_{t-1}) + bh) \quad (14)$$

Step-4: Update on Hidden State. Calculating the updated hidden state Ht, which combines the candidate's hidden state $\widetilde{H}t$ with the prior hidden state H_{t-1}, is the last step. While the remainder of the candidate's hidden state $\widetilde{H}t$ is included. The update gate Ut decides how much of the prior hidden state should be retained. The element-wise multiplication $\odot$ is used to accomplish this. Equation (15) can be used to determine the updated hidden state.

$$Ht = Ut \odot Ht - 1 + (1 - Ut) \odot \widetilde{H}t \quad (15)$$

2.6 Grid Search Hyper-Parameter Tuning

Grid search means trying all possible combinations of the hyper-parameters of interest [18]. Grid search is a hyperparameter optimization technique that systematically explores all possible combinations of predefined hyperparameters to get the best configuration for a machine learning model by creating a discrete grid throughout the search space, with each point representing a unique combination of hyperparameter values [19].

2.7 MAPE

MAPE is a method for calculating a forecasting model’s performance value MAPE measures the average percentage error between the actual and predicted values, considering the magnitude of the error relative to the true value taken without regard to sign [20][15]. MAPE presents the prediction error as a percentage, allowing users to understand how large the prediction error is compared to the size of the true value. The prediction model is more accurate when the MAPE value is lower. MAPE is widely used in various industries, such as business, economics, and data science, to evaluate the performance of predictive models and improve future prediction strategies [15]. The MAPE equation can be described in Equation (16) [15].

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right| \times 100\% \quad (16)$$

3. RESULTS AND ANALYSIS

The research utilizes a web crawling methodology to collect S&P GSCI Nickel data from Yahoo Finance using Python libraries, as presented in Table 1. The analysis encompasses weekly data spanning from 2008 to 2024, specifically focusing on the opening and closing values of the S&P GSCI Nickel index. The dataset comprises 856 data points, providing a comprehensive temporal coverage of price movements throughout this period. These observations capture the dynamic behaviour of nickel prices in the global commodity market over a 16-year timeframe.

Table 1. Open and Close S&P GSCI Nickel

Date	Open	Close
2008-01-01	614,05	611,67
2008-01-08	616,09	615,28
2008-01-15	612,57	597,16
2008-01-22	576,76	603,29
………	……..	……..
2024-04-16	596.95	589.96
2024-04-23	589.96	592.55
2024-04-30	592.5	576.95
2024-05-07	577.14	579.50
2024-05-14	579.5	591.04

3.1 Data Normalization

Before being applied to the dataset model, it must first be normalized according to the format of the three models and scaled with a scale of 0 to 1 using MinMax Scaler. The following is the result of the dataset after normalization, which can be seen in Table 2.

Table 2. Normalized Dataset

Date	Open	Close
2008-01-01	0.594259	0.589203
2008-01-08	0.597419	0.594782
2008-01-15	0.591966	0.566776
2008-01-22	0.536491	0.576251
2008-01-29	0.579619	0.574844
2008-01-01	0.594259	0.589203

3.2 Grid Search Hyper-Parameter Tuning

This study employed grid search for hyper-parameter tuning, a systematic approach to selecting the optimal set of parameters for the model. In this case, the hyperparameters tuned include the number of neurons in each layer, batch size, and activation function. The candidate parameters used for this study are in Table 3.

Table 3. Candidate Hyper-Parameter

Neuron	Activation Function	Batch Size
[32, 64, 32]	Relu	16
[64, 128, 64]	Tanh	32
[128, 256, 128]	Sigmoid	64

After Hyper-parameter tuning is done, training will be carried out on each forecasting model. An example of the forecasting result graph can be seen in Figure 2, the forecasting graph of the three models with architectures combining 64,128,64 neurons with a batch size of 32 and using the RELU activation function.

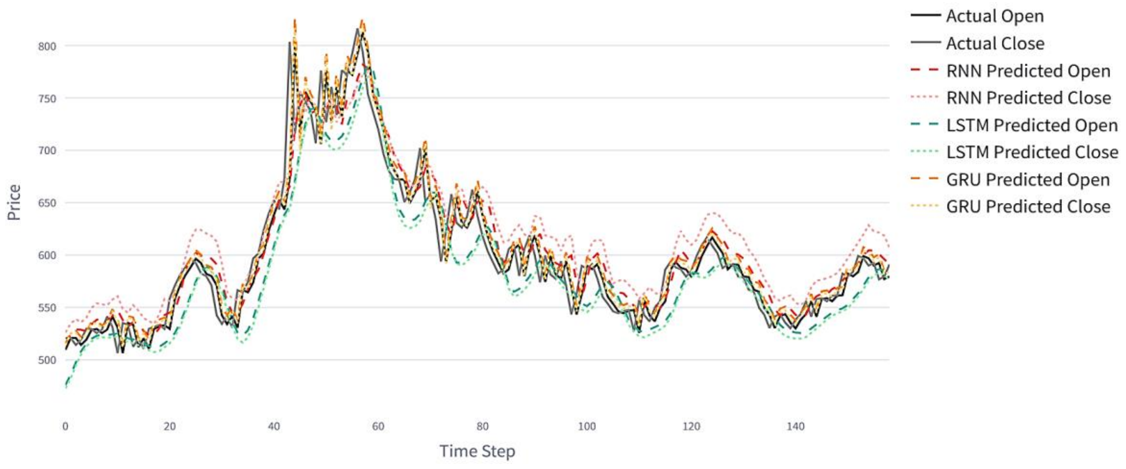

Figure 2. The Result of Model Prediction

Hyperparameter tuning experiments on the three models will be conducted from the candidate hyperparameters in Table 1. The MAPE result of the three models can be summarized in Table 4. RNN development with [32, 64, 32] neurons, tanh activation function, and 16 batch size presents the lowest average MAPE value of 2.08%. LSTM with the architecture of neurons [128, 256, 128], tanh activation function, and batch size 16 achieved a MAPE value of 2.505%. The RNN model, which combines [64, 128, 64] neurons with tanh activation function and batch size 64, has the highest average MAPE value of 7.02%. The GRU model, with the architecture of neurons [128, 256, 128] and the RELU activation function and batch size of 16, obtained the lowest MAPE value of 1.505%, which means that this architecture is the best in this study.

Table 4. RNN, LSTM, and GRU Model Performance

Neurons	Activation	Batch Size	MAPE (%)		
			Open	Close	Average
		RNN Model			
[32, 64, 32]	Relu	16	1,65	2,52	2,085
[32, 64, 32]	Relu	32	1,92	3,04	2,48
[32, 64, 32]	Relu	64	3,16	3,87	3,515
[32, 64, 32]	Tanh	16	1,45	2,71	2,08

Neurons	Activation	Batch Size	MAPE (%)		
			Open	Close	Average
[32, 64, 32]	Tanh	32	2,53	3	2,765
[32, 64, 32]	Tanh	64	3,02	3,57	3,295
[64, 128, 64]	Relu	16	1,45	3,44	2,445
[64, 128, 64]	Relu	32	1,87	3,28	2,575
[64, 128, 64]	Relu	64	2,63	4,2	3,415
[64, 128, 64]	Tanh	16	2,73	3,84	3,285
[64, 128, 64]	Tanh	32	4,38	3,69	4,035
[64, 128, 64]	Tanh	64	6,91	7,13	7,02
[128, 256, 128]	Relu	16	2,27	2,94	2,605
[128, 256, 128]	Relu	32	2,45	3,53	2,99
[128, 256, 128]	Relu	64	3,36	5,01	4,185
[128, 256, 128]	Tanh	16	3,88	3,79	3,835
[128, 256, 128]	Tanh	32	3,42	7,35	5,385
[128, 256, 128]	Tanh	64	3,29	3,97	3,63
[32, 64, 32]	Sigmoid	16	2,56	3,26	2,91
[32, 64, 32]	Sigmoid	32	3,76	4,3	4,03
[32, 64, 32]	Sigmoid	64	3,04	3,72	3,38
[64, 128, 64]	Sigmoid	16	2,22	3	2,61
[64, 128, 64]	Sigmoid	32	2,24	3,05	2,645
[64, 128, 64]	Sigmoid	64	3,28	3,57	3,425
[128, 256, 128]	Sigmoid	16	2,16	3,01	2,585
[128, 256, 128]	Sigmoid	32	2,45	3,39	2,92
[128, 256, 128]	Sigmoid	64	3,29	4,06	3,675
LSTM Model					
[32, 64, 32]	Relu	16	3,35	3,73	3,54
[32, 64, 32]	Relu	32	4,26	4,93	4,595
[32, 64, 32]	Relu	64	5,18	5,61	5,395
[32, 64, 32]	Tanh	16	2,66	3,5	3,08
[32, 64, 32]	Tanh	32	4,31	5,31	4,81
[32, 64, 32]	Tanh	64	5,03	5,56	5,295
[64, 128, 64]	Relu	16	2,96	3,77	3,365
[64, 128, 64]	Relu	32	3,37	4,04	3,705
[64, 128, 64]	Relu	64	4,66	5,17	4,915
[64, 128, 64]	Tanh	16	2,58	3,51	3,045
[64, 128, 64]	Tanh	32	3,21	3,94	3,575
[64, 128, 64]	Tanh	64	4,17	4,95	4,56
[128, 256, 128]	Relu	16	2,99	3,34	3,165
[128, 256, 128]	Relu	32	3,16	3,95	3,555
[128, 256, 128]	Relu	64	4,25	4,81	4,53
[128, 256, 128]	Tanh	16	1,86	3,15	2,505
[128, 256, 128]	Tanh	32	3,01	4,29	3,65
[128, 256, 128]	Tanh	64	3,78	4,57	4,175
[32, 64, 32]	Sigmoid	16	4,3	5,41	4,855
[32, 64, 32]	Sigmoid	32	4,15	4,66	4,405
[32, 64, 32]	Sigmoid	64	5,43	5,97	5,7
[64, 128, 64]	Sigmoid	16	3,24	3,98	3,61
[64, 128, 64]	Sigmoid	32	5,22	5,94	5,58
[64, 128, 64]	Sigmoid	64	4,71	5,3	5,005
[128, 256, 128]	Sigmoid	16	3,53	4,67	4,1
[128, 256, 128]	Sigmoid	32	4,1	4,81	4,455
[128, 256, 128]	Sigmoid	64	4,05	4,52	4,285
GRU Model					
[32, 64, 32]	Relu	16	0,63	2,51	1,57
[32, 64, 32]	Relu	64	2,76	3,72	3,24
[32, 64, 32]	Tanh	16	0,73	2,83	1,78
[32, 64, 32]	Tanh	32	1,37	2,81	2,09
[32, 64, 32]	Tanh	64	2,6	3,58	3,09
[64, 128, 64]	Relu	16	1,05	2,33	1,69
[64, 128, 64]	Relu	32	1,75	3,15	2,45
[64, 128, 64]	Relu	64	2,17	3,33	2,75
[64, 128, 64]	Tanh	16	1,28	2,58	1,93
[64, 128, 64]	Tanh	32	1,5	2,87	2,185
[64, 128, 64]	Tanh	64	2,42	3,12	2,77
[128, 256, 128]	Relu	16	0,63	2,38	1,505
[128, 256, 128]	Relu	32	1,38	2,64	2,01
[128, 256, 128]	Relu	64	2,74	3,65	3,195
[128, 256, 128]	Tanh	16	0,74	3,32	2,03
[128, 256, 128]	Tanh	32	0,89	2,65	1,77
[128, 256, 128]	Tanh	64	2,22	3,29	2,755
[32, 64, 32]	Sigmoid	16	1,3	2,53	1,915
[32, 64, 32]	Sigmoid	32	2,09	3,13	2,61
[32, 64, 32]	Sigmoid	64	3,03	3,74	3,385

Neurons	Activation	Batch Size	MAPE (%)		
			Open	Close	Average
[64, 128, 64]	Sigmoid	16	2,08	2,88	2,48
[64, 128, 64]	Sigmoid	32	1,7	2,92	2,31
[64, 128, 64]	Sigmoid	64	2,22	3,22	2,72
[128, 256, 128]	Sigmoid	16	1,91	2,61	2,26
[128, 256, 128]	Sigmoid	32	1,85	3,15	2,5
[128, 256, 128]	Sigmoid	64	2,35	3,9	3,125

The result RNN, LSTM, and GRU models for predicting the S&P GSCI Nickel index revealed that the GRU model with [128, 256, 128] neurons, relu activation function, and batch size 16 achieved the lowest MAPE of 1.505%, outperforming both LSTM (2.505%) and RNN (2.08%) models. GRU's superior performance can be attributed to its architectural advantages in handling time series data through efficient update and reset gates, while smaller batch sizes across all models suggested better adaptation to commodity price volatility. These findings extend previous research on deep learning for financial forecasting, challenging assumptions about gated architectures while providing practical implications for traders and analysts in commodity markets. Despite strengths in comprehensive hyper-parameter optimization, limitations include focus on a single commodity index and reliance on MAPE as the primary metric, suggesting future research directions such as evaluating performance across different prediction horizons, implementing ensemble approaches, incorporating additional external features, developing adaptive models for different market regimes, and exploring newer architectures like Transformer-based models.

4. CONCLUSION

GRU is the most effective model for predicting the S&P GSCI Nickel index prices, achieving the lowest MAPE of 1.505% with an architecture of [128, 256, 128] neurons, RELU activation function, and batch size 16. LSTM, which got a MAPE of 2.505% with a similar architecture but using tanh activation, comes in second. The RNN model, which combines [64, 128, 64] neurons with tanh activation function and batch size 64, has the highest average MAPE value of 7.02%. RNN is the third one in the series. According to the research findings, the RELU activation function consistently performs better than tanh and sigmoid activations, particularly when applied to small batch sizes. Smaller batch sizes tend to give better results. For future development, researchers should explore these deep learning models (RNN, LSTM, and GRU) with a focus on optimizing computational efficiency to handle larger data sets while maintaining or improving prediction accuracy, which could involve applying parallel processing techniques, optimizing model architecture, and exploring advanced pre-processing methods. Researchers could also develop a practical mobile-based application to help investors easily determine the optimal time to invest in the future.

REFERENCES

[1] A. Arfan and L. ETP, “Perbandingan Algoritma Long Short-Term Memory dengan SVR Pada Prediksi Harga Saham di Indonesia,” Petir, vol. 13, no. 1, pp. 33–43, 2020, doi: 10.33322/petir.v13i1.858.

[2] F. Pakaja, A. Naba, and Purwanto, “Peramalan Penjualan Mobil Menggunakan Jaringan Syaraf Tiruan dan Certainty Factor,” Eeccis, vol. 6, no. 1, pp. 23–28, 2012, doi: 10.21776/jeeccis.v6i1.162.

[3] S. S. Naibaho and S. R. Simangunsong, Indonesia Gudang Nikel, Indonesia Memimpin Transisi Energi, vol. 2, no. 28. 2022. doi: 10.33088/jspi.v2i1.

[4] D. N. Tsirwiyati, “Kebijakan Larangan Ekspor Nikel Indonesia,” J. Huk. Respublica Fak. Huk. Univ. Lancang Kuning, vol. Xi, no. 231, pp. 1–12, 2023, doi: 10.31849/respublica.v22i2.13468.

[5] R. Ciptaswara, “Implementasi Hilirisasi Mineral Dan Batu Bara Dalam Rangka Mewujudkan Kedaulatan Energi Dan Daya Saing Industri Nasional,” vol. 34 no 2, pp. 521–558, 2022, doi: 10.22146/mh.v34i2.3490.

[6] R. D. Martono, “Analisis Pengaruh Harga Komoditas Dunia Terhadap Pergerakan Indeks Harga Saham Gabungan (IHSG), Indeks Lq 45 , Dan Jakarta Islamic Index (JII) Di Bursa Efek Indonesia (BEI),” no. 106081002483, pp. 1–106, 2019.

[7] G. Budiprasetyo, M. Hani’ah, and D. Z. Aflah, “Prediksi Harga Saham Syariah Menggunakan Algoritma Long Short-Term Memory (LSTM),” J. Nas. Teknol. dan Sist. Inf., vol. 8, no. 3, pp. 164–172, 2023, doi: 10.25077/teknosi.v8i3.2022.164-172.

[8] A. Y. Pramudia, “Implementasi Metode Autoregressive Integrated Moving Average (ARIMA) pada Aplikasi Peramalan Harga Saham Berbasis Website,” J. Ilm. Komputasi, vol. 22, no. 1, pp. 105–112, 2023, doi: 10.32409/jikstik.22.1.3335.

[9] N. Prissy, M. Al Haris, and P. R. Arum, “Peramalan Nilai Ekspor Migas Di Indonesia Menggunakan Model Long Short Term Memory Dan Gated Recurrent Unit Dengan Optimasi Nesterov Adam,” J Stat., vol. 16, no. 1, pp. 12–26, 2022, doi: 10.36456/jstat.vol16.no1.a6121.

[10] T. P. Shella, “Hybrid Autoregressive Integrated Moving Average (ARIMA) – Gated Recurrent Unit (GRU) Dalam Peramalan Harga Sawit Pada Pt. Sawit Sumbermas Sarana Di Kalimantan Tengah,” 2023.

[11] M. N. Alim, “Pemodelan Time Series Data Saham LQ45 dengan Algoritma LSTM, RNN, dan Arima,” Pros. Semin. Nas. Mat., vol. 6, pp. 694–701, 2023,

[12] I. A. Saputra, A. V. Vitianingsih, Y. Kristyawan, A. L. Maukar, and J. F. Rusdi, "Forecasting Model of Export and Import Value of Oil and Gas Using Gated Recurrent Unit Method," Teknika, vol. 13, no. 2, pp. 239–243, 2024, doi: 10.34148/teknika.v13i2.861.
[13] K. A. Rijal, A. V. Vitianingsih, Y. Kristyawan, A. L. Maukar, and S. F. A. Wati, "Forecasting Model of Indonesia's Oil & Gas and Non-Oil & Gas Export Value using Var and LSTM Methods," J. Teknol. dan Manaj. Inform., vol. 10, no. 1, pp. 59–69, 2024, doi: 10.26905/jtmi.v10i1.13127.
[14] M. Joseph, Modern Time Series Forecasting with Python. 2022.
[15] M. V. E. . Makridakis S., Wheelwright S.S., "Forecasting Methods and Applications)," 2008.
[16] A. C. M. and S. Guido, Introduction to Machine Learning with Python by, no. 0. 2016.
[17] J. M. Czum, "Dive Into Deep Learning," J. Am. Coll. Radiol., vol. 17, no. 5, pp. 637–638, 2020, doi: 10.1016/j.jacr.2020.02.005.
[18] K. Sofien, Deep Learning for Finance. 2024.
[19] J. Brownlee, "Long Short-Term Memory Networks With Python," Mach. Learn. Mastery With Python, vol. 1, no. 1, p. 228, 2017.
[20] P. Goodwin, J. K. Ord, L.-E. Öller, J. A. Sniezek, and M. Leonard, Principles of Forecasting: A Handbook for Researchers and Practitioners, vol. 18, no. 3. 2002. doi: 10.1016/s0169-2070(02)00034-1.

BIBLIOGRAPHY OF AUTHORS

Ragada Rahmansyah, born on July 23, 2001 Babat, Lamongan, East Java. He has studied at SD Negeri Banaran 1 (2006-2012), MTS Binjai Serbangan (2012-2015), and SMK N 3 (2016-2019). Currently, the author continues his education in the Informatic Department, Universitas Dr. Soetomo.

Anik Vega Vitianingsih is currently working as a lecturer in the Informatics Department at Universitas Dr. Soetomo in Surabaya, Indonesia. Within the realm of machine learning-based spatial analysis, the Author is interested in studying subjects related to geographic information systems.

Rusdi Hamidan is a lecturer at Universitas Dr. Soetomo., Surabaya. He is particularly interested in Web and Mobile Development, especially in Kotlin for Mobile Device Programming and modern Web Front-End Development. He also enjoys exploring API integration and Object-Oriented Programming and advanced software development concepts.

Anastasia Lidya Maukar is currently working as a lecturer at President University in Indonesia, where she teaches lectures on industrial management and information systems development. S.T., M.Sc., and M.MT. are the degrees that she also possesses. Both the University of Hertfordshire and the Institut Teknologi Sepuluh Nopember Surabaya (ITS) awarded their master's degrees in information systems development and industrial management, respectively. Her academic background includes both of these degrees. Information systems, databases, statistics, and production systems are all examples of topics that can have research applications.

Yoyon Arie Budi Suprio is a lecturer at the Informatics Department of STIKOM PGRI Banyuwangi, Indonesia. Teaches courses on Computer Networks and Security and includes data encryption