Optimizing Gated Recurrent Unit (GRU) for Gold Price Prediction: Hyperparameter Tuning and Model Evaluation on Historical XAU/USD Data

Abdul Faqih[1]*, Muhammad Jauhar Vikri[2], Ita Aristia Sa'ida[3]
Department of Informatics Engineering[1], [2], [3]
University of Nahdlatul Ulama Sunan Giri
Bojonegoro, Indonesia
abdulfaqih7777@gmail.com [1], vikri@unugiri.ac.id [2], itaaristia@unugiri.ac.id [3]

Abstract*— This study investigates the use of a Gated Recurrent Unit (GRU) model with a four-layer architecture for daily gold price closing prediction, motivated by the model's ability to effectively capture temporal dependencies in time series data. Gold price forecasting is highly challenging due to its volatility and external factors, making it an important area of research for investors and financial analysts. By systematically optimizing hyperparameters through 72 combinations of epochs, batch size, GRU layer units, and dropout rates, the study identifies the optimal configuration (100 epochs, batch size of 16, 256 units, dropout rate 0.1) based on MSE performance on validation data. The best model achieved MAE of 25.76, MSE of 954.97, and RMSE of 30.90, after inverse transformation on test data. These results highlight the potential of the GRU model in accurately forecasting gold prices, with implications for financial decision-making . However, the prediction error suggests that further improvements could be made by incorporating external factors or exploring advanced model architectures.

***Keywords*— Gated Recurrent Unit (GRU), Gold Price Prediction, Hyperparameter Optimization, Time Series**

I. INTRODUCTION

Investment activities are an integral part of the modern economy, where individuals and institutions allocate capital with the aim of generating returns or preserving asset value in the future. A variety of asset classes are available to investors, ranging from stocks, bonds, real estate, to commodities, each with distinct risk profiles and potential returns. Among the many investment options available, individuals tend to choose investments that offer higher returns [1]. One attractive option is gold, which is not only valued as a raw material for jewelry and technology, but also widely recognized as a vital investment instrument. Gold serves as a store of value *(safe haven)* especially during times of global economic turmoil, as well as a hedge against inflation and currency devaluation [2].

Price fluctuations in XAU/USD are driven by factors such as changes in currency exchange rates, interest rate policies (especially those of central banks like the U.S. Federal Reserve), inflation rates, and geopolitical tensions [3]. The dynamics of physical gold supply and demand also contribute to these fluctuations, making it crucial for investors to accurately predict price movements [4]. The complex and interrelated nature of these factors makes forecasting gold prices a challenging yet valuable task.

This study aims to address the challenge of predicting daily gold prices by implementing the Gated Recurrent Unit (GRU) model, which is well-suited for capturing the temporal dependencies in financial time series data. By evaluating various hyperparameter configurations, this research seeks to improve the accuracy of gold price predictions and provide a more reliable tool for investors.

To support investment decision-making amid gold price volatility, various studies have been conducted to develop accurate prediction methods. [5]This study compares the performance of models based on Recurrent Neural Networks (RNN), including LSTM and GRU, in forecasting economic and financial data in Indonesia, such as the IHSG, export value, and GDP. The results of this study indicate that the GRU model performs best overall and is more stable than RNN and LSTM on the data. On average, GRU recorded the smallest MAPE values on IHSG (Index Harga Saham Gabungan) data (0.3695%), export data (7.36%), and GDP data (1.77%). However, this study does not specifically focus on gold price prediction. The study also suggests increasing the number of scenarios with other combinations of hyperparameters and using model search techniques [6].

Specifically implements and compares GRU, Bi-GRU, LSTM, and Bi-LSTM for global gold price prediction. This study uses historical gold price data from *Yahoo Finance* and explores various optimization techniques, *batch sizes*, and *time steps*. The results of this study indicate that the Bi-GRU model with *Adam optimization*, a *batch size* of 8, and a *time step* of 20 provides the best performance for global gold price prediction, with an MSE value of 4.1153, an RMSE value of 2.0286, an MAE value of 1.5881, and a MAPE value of 0.8857%. Although relevant to the topic of gold price prediction using GRU and its variations, this study did not use the best hyperparameter selection [7].

A separate study evaluated the efficacy of various deep learning algorithms, including *Artificial Neural Network* (ANN), *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), and *Long Short-Term Memory* (LSTM), in predicting gold prices utilizing a dataset sourced from Kaggle. The study determined that the CNN model

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i2.2352, Copyright ©2025
Submitted : May 1, 2025, Revised : May 11, 2025, Accepted : May 14, 2025, Published : May 26, 2025

utilizing *Adam optimization* and the MSE loss function exhibited optimal performance in predicting gold prices within the used dataset. The CNN model achieved the smallest MAE value of 0.004848717761305338, the smallest MSE value of 4.3451079619612133e-05, and the smallest RMSE value of 0.006591743291392053. Although comparing RNN and LSTM, this study did not explore the performance of GRU in depth and recommends the use of a larger and more recent dataset as well as exploration of other models such as Transformer for further research.

Based on this literature review, there is a research gap focusing on gold price prediction in Indonesia using the GRU model. Although the GRU model has shown good performance in forecasting economic time series data in Indonesia and global gold price prediction, its in-depth application for specific gold price prediction in Indonesia with adequate *hyperparameter* exploration still requires further research. Therefore, This study seeks to develop a predictive model for gold closing prices (XAU/USD) utilizing the Gated Recurrent Unit (GRU) architecture, while systematically assessing the effects of diverse *hyperparameter* combinations: *epochs* (50, 100, 150), *batch sizes* (16, 32, 64), *neuron counts* (50, 100, 128, 256 units), *dropout rates* (0.1, 0.0), and a *learning rate* of 0.0001 on model performance. The performance of each combination will be measured and compared using the *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), and *Root Mean Squared Error* (RMSE) metrics on the test data to identify the most optimal configuration.

II. METHODOLOGY

The research process was carried out in several stages, starting with the collection of gold price data (XAU/USD) from TradingView, a popular financial data platform. Preprocessing steps were then performed, including normalization using the *Min-Max Scaler* to scale the data within the [0,1] range. A time series sequence was created with a timestep of 30, representing the past 30 days of price data for each prediction. Next, the dataset was split into training (80%) and testing (20%) sets, ensuring that the data followed a temporal order without shuffling, to maintain the integrity of time series forecasting. The GRU model with a four-layer hidden architecture was then applied, utilizing the Adam optimizer and MSE loss function. Various *hyperparameter* values were tested, including epochs (50, 100, 150), *batch sizes* (16, 32, 64), number of neurons (50, 100, 128, 256 units), *dropout* (0.1, 0.0), and *learning rate* (0.0001).

The choice of these hyperparameters was based on preliminary experiments and existing literature that indicated these values would provide a good balance between training time and model performance. For instance, the choice of batch size and number of epochs was influenced by previous research in financial time series forecasting, which suggested these values as optimal for minimizing overfitting and improving generalization. Despite the benefits of automated *hyperparameter* optimization methods like Grid Search or Random Search, they were not employed in this study due to time constraints and computational resource limitations. Instead, a more manual approach was taken to test a broad range of hyperparameters, ensuring that the model's performance could be thoroughly evaluated with reasonable computational effort.

Validation during training was carried out using a 20% split of the training data to monitor model performance and prevent overfitting through Early Stopping. Final evaluation was conducted on the test dataset, using the MAE, MSE, and RMSE metrics to compare the performance of different hyperparameter configurations. This research method is presented in the form of a flowchart in Fig. 1.

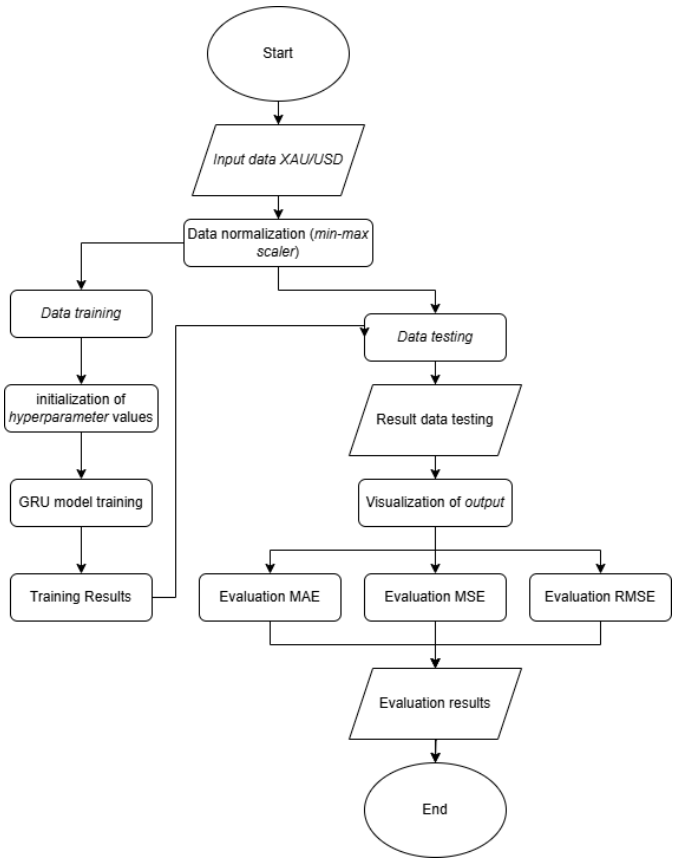

Fig 1. Research Method

A. Dataset

This analysis utilizes historical data on the price of gold traded against the US dollar (XAU/USD). This data was obtained from the TradingView platform (www.tradingview.com), a popular online platform that provides real-time charting and financial market analysis tools. The data includes time, opening price, high price, low price, and closing price. The data used spans from November 2006 to March 2025, with some of the data presented in Table 1. Utilizing large and diverse datasets is crucial in financial analysis as they enable more accurate predictions and better generalization of models. Integrating deep learning and big data algorithms significantly enhances the accuracy of financial risk behavior predictions, highlighting the importance of extensive datasets in financial forecasting [8].

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i2.2352, Copyright ©2025
Submitted : May 1, 2025, Revised : May 11, 2025, Accepted : May 14, 2025, Published : May 26, 2025

TABLE I. XAU/USD DATASET FROM 2006 TO 2025

Time	Open	High	Low	Close
2006-09-21 21:00:00	584.2	592.7	584.2	589.65
2006-09-24 21:00:00	589.65	592	582.3	590.7
2006-09-25 21:00:00	590.7	594	586.1	591.8
2006-09-26 21:00:00	591.8	603.65	589.8	603
2006-09-27 21:00:00	603	607.1	600.4	601.1
2006-09-28 21:00:00	601.1	603.8	594.4	598.25
2006-10-01 21:00:00	598.25	604.15	594.75	596.7
2006-10-02 21:00:00	596.7	598.5	573.8	574.8
…	…	…	…	…
2025-03-02 22:00:00	2873.14	2895.21	2859	2892.985
2025-03-03 22:00:00	2892.7	2927.91	2881.98	2917.85
2025-03-04 22:00:00	2917.885	2929.98	2894.35	2919.265
2025-03-05 22:00:00	2918.685	2926.72	2891.15	2911.1
2025-03-06 22:00:00	2912.935	2930.54	2896.91	2909.55
2025-03-09 21:00:00	2912.41	2918.385	2896.71	2904.68

B. *Data Normalization*

Data normalization is an important technique in machine learning and time series analysis, especially when using gradient-based models such as GRU. Normalization aims to minimize errors, ensure data is scaled and easy to process in models. In this case study, we use the min-max scaling technique with an interval of [0,1]. The formula for normalization is shown in equation (1). Reference [9] data normalization techniques, such as *Min-Max scaling* and *Z-normalization*, have been explored to enhance model performance. The results of normalization with *min-max scaler* are shown in Fig. 2.

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

Explanation :
X_{scaled} = value of the data after scaling.
X = original value of the data.
X_{min} = minimum value of the feature in the dataset.
X_{max} = maximum value of the feature in the dataset.

The results of data normalization are shown in the following Fig 2:

Fig 2. Data after data normalization process

C. *Data Splitting*

Maintaining temporal order during data splitting is crucial for time series forecasting models to prevent data leakage and ensure realistic performance evaluation, as highlighted [8]. The constructed dataset (x and y) is partitioned into training and testing subsets, with an allocation of 80% for training and 20% for testing. The division is performed sequentially *(shuffle=False)* to maintain the time sequence, in accordance with the characteristics of time series data. The data division visualization is presented in Fig. 3.

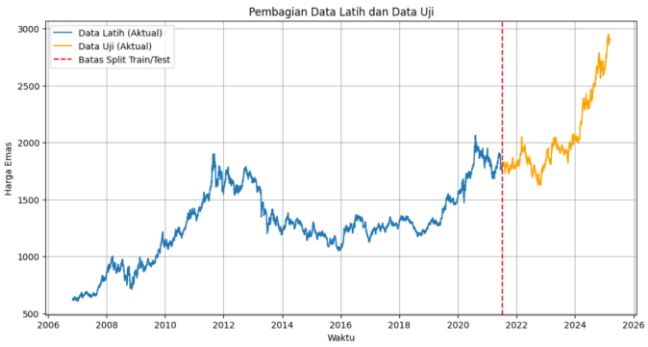

Fig 3. Training and Testing Data Split

D. *GRU Model*

The *Gated Recurrent Unit* (GRU) is a *Recurrent Neural Network* (RNN) architecture developed to mitigate the vanishing gradient issue commonly faced by conventional RNNs, particularly when processing sequential data with long-term dependencies [6].GRU can be considered an evolution of simpler and more efficient RNNs, with the ability to learn and retain information from data sequences over longer periods. The primary function of GRU in neural networks is to process sequential data by retaining relevant information from previous time steps while filtering out irrelevant or outdated information. This allows the network to understand the temporal context in the data and make more accurate predictions.

The GRU structure simplifies the recurrent unit by introducing two main gates, the *update gate* and the *reset gate*. The *Update gate* (z_t) controls the proportion of information from the previous hidden state (h_{t-1}) that is passed on to the current hidden state (h_t). Mathematically, the update gate is

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i2.2352, Copyright ©2025
Submitted : May 1, 2025, Revised : May 11, 2025, Accepted : May 14, 2025, Published : May 26, 2025

calculated using the following formula:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (2)$$

Explanation:
σ = sigmoid function
x_t = current input
h_{t-1} = previous hidden state
W_z and U_z = weight matrices
b_z = bias for the update gate.

The *reset gate* (r_t) determines how much of the previous hidden state is ignored in the calculation of the current hidden state, calculated using the formula:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (3)$$

Explanation:
σ = sigmoid function
x_t = current input
h_{t-1} = previous hidden state
W_r and U_r = weight matrices
b_r = bias for the reset gate.

The internal architecture of the GRU entails computing the candidate *hidden state* ($\tilde{h}_t$), which is derived from the current input and the preceding *hidden state*, adjusted by the *reset gate*. The candidate concealed state is calculated using the subsequent formula:

$$\tilde{h}_t = tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \quad (4)$$

Explanation:
$tanh$ = hyperbolic tangent function
r_t = reset gate
h_{t-1} = previous hidden state
W_h and U_h = weight matrices
b_h = bias for the candidate hidden state
$\odot$ = denotes element-wise multiplication.

An illustration of GRU model hidden state computation is shown in Fig. 4.

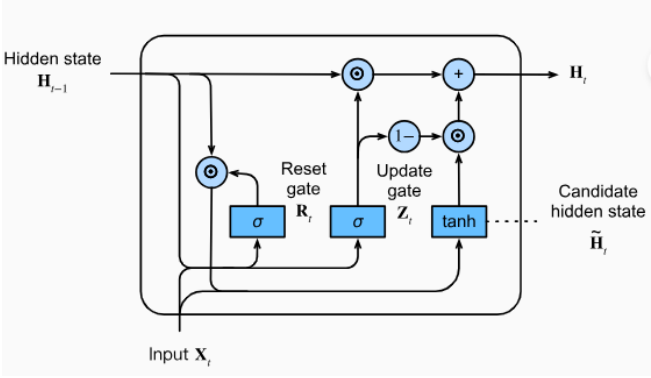

Fig 4. GRU Model Hidden State Computation

At this stage, the optimal GRU model architecture is built. The model is configured as a sequential model, meaning layers are added sequentially. The prepared GRU model is trained using the training data. The model specifications are as follows:

1. Optimizer: Adam
2. Hidden layers: 4
3. Number of Neurons: 50, 100, 128, 256 units
4. Timestep: 30
5. Epochs: 50, 100, 150 (with early stopping)
6. Batch Size: 16, 32, 64
7. Dropout: 0.1, 0.0
8. Learning rate: 0.0001

E. Testing Process

The testing process is carried out after the computation process on the training data. The trained model is then implemented using the testing data to obtain the prediction results.

F. Output Visualization

The visualization of the GRU model's prediction results is performed to compare the actual gold price movement with the gold price predicted by the GRU model. A line plot is used to represent these two time series. The X-axis of the graph represents time, taken from the *'time'* column in the original dataset, for the testing data segment. The Y-axis represents the gold price in its original scale. The output visualization is presented in Fig. 5 as follows:

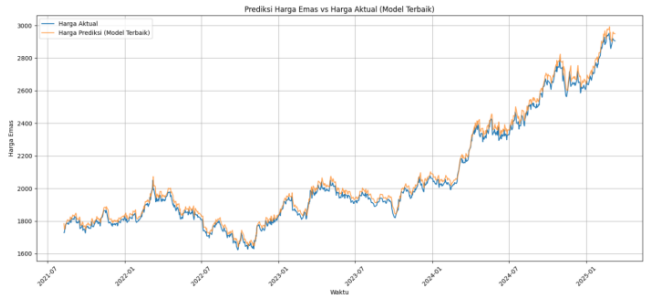

Fig 5. Visualization of Actual vs Predicted Prices

G. Evaluation

To measure the performance of the GRU model in predicting XAU/USD prices, the following evaluation metrics will be used:

1. *Mean Absolute Error* (MAE), This metric computes the average of the absolute prediction errors. MAE offers a summary of the mean prediction error expressed in the original price units. The formula for *Mean Absolute Error* (MAE) is as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|yi - \hat{y}i| \quad (5)$$

Where:
n is the total number of predictions.
y_i is the actual XAU/USD price at time i.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i2.2352, Copyright ©2025
Submitted : May 1, 2025, Revised : May 11, 2025, Accepted : May 14, 2025, Published : May 26, 2025

$\hat{y}i$ is the predicted XAU/USD price by the model at time *i.*

MAE is readily interpretable as its outcome is expressed in the same units as the original data. Nevertheless, MAE does not assign greater significance to substantial errors, rendering it less responsive to outliers in comparison to MSE and RMSE. [10]

2. *Mean Squared Error* (MSE), computes the average of the squared prediction errors. MSE exhibits more sensitivity to substantial errors than MAE due to the squaring of errors. The formula for Mean Squared Error (MSE) is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(yi - \hat{y}i)^2 \quad (6)$$

Where:
n, $y_i, \hat{y}i$ have the same meanings as in the MAE formula.

MSE penalizes large errors more than MAE, which is beneficial if the model needs to avoid large errors. However, the MSE result is in squared units of the original data, making interpretation more difficult [11]

3. *Root Mean Squared Error* (RMSE) is the square root of *Mean Squared Error* (MSE). RMSE provides the error value in the same units as the original data, facilitating interpretation in contrast to MSE. Furthermore, as it represents the square root of the Mean Squared Error, the *Root Mean Squared Error* (RMSE) is particularly sensitive to significant errors. The formula for RMSE is:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(yi - \hat{y}i)^2} \quad (7)$$

Where:
MSE = calculated as in the previous MSE formula.

RMSE penalizes large errors more heavily and is more sensitive to outliers compared to MAE. This metric is commonly used for comparing the performance of different models[10]

III. Discussion And Results

The historical gold data (XAU/USD) obtained from TradingView, spanning from November 21, 2006, to March 9, 2025, serves as the dataset for this study. The original dataset consists of 4782 data points (rows). After performing preprocessing steps as outlined in Section 3 (Methodology), including data normalization using the *Min-Max Scaler* and the formation of time series data sequences with a timestep of 30, the number of data samples ready for training and testing the model is 4752 samples (i.e., the original data minus the lookback).

The GRU model used in this study has a Sequential architecture consisting of four hidden layers, each being a GRU layer with 50, 100, 128, and 256 units. The first GRU layer receives input with dimensions (30, 1), reflecting the 30 timesteps and 1 feature (normalized closing price). The first three GRU layers are configured with *return_sequences=True*, so that each layer produces a full output sequence, which then becomes the input for the subsequent GRU layer. The fourth GRU layer uses return_*sequences=False* because it is the last GRU layer before the output Dense layer. Its output is averaged into a single vector of size 50, representing the temporal features of the 30-step input sequence. The output layer is a single Dense layer that predicts a single price value (the next normalized price).

The model is compiled with the *Adam optimizer*, known for its efficiency in handling sparse or noisy data, and uses Mean Squared Error (MSE) as the loss function, which is commonly used for regression problems. A summary of the model architecture and the number of trainable parameters is shown in the table 2.

TABLE 1. GRU Model Architecture

Layer (type)	Output Shape	Param
gru (GRU)	(None, 30, 50)	7.950
gru_1 (GRU)	(None, 30, 50)	15.300
gru_2 (GRU)	(None, 30, 50)	15.300
gru_3 (GRU)	(None, 50)	15.300
dense (Dense)	(None, 1)	51

To identify the most effective hyperparameter configuration, the GRU model was trained nine times, each with different combinations of *epochs* (50, 100, 150) and *batch sizes* (16, 32, 64). During the training process, the model's performance on a small portion of the training data (set aside as a validation split, accounting for 20% of the training data) was monitored. The training loss and validation loss were recorded at each epoch.

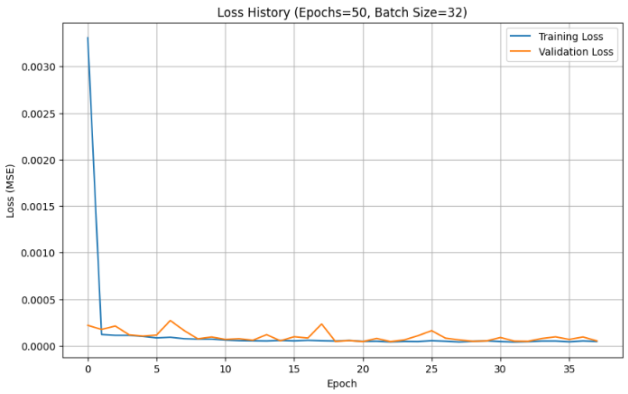

Fig 6. An Example of a Loss History Plot

Fig. 6 presents several examples of *loss history plots* during training. Ideally, both the training loss and validation loss should decrease as the epochs progress, indicating that the model is learning from the data. However, if the validation loss begins to increase while the training loss continues to decrease, this is an indication of overfitting—the model memorizes the training data but loses its ability to generalize to unseen data.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i2.2352, Copyright ©2025
Submitted : May 1, 2025, Revised : May 11, 2025, Accepted : May 14, 2025, Published : May 26, 2025

In this context, the role of Early Stopping becomes crucial. Early Stopping is a regularization technique that monitors performance metrics (in this case, validation loss) on the validation set. If the monitored metric does not show improvement (or even worsens) over a specified number of consecutive epochs (*patience = 15*), the training process is automatically halted, and the best model weights from the previous epoch are restored. This prevents the model from training too long and overfitting, thus improving its generalization ability on new data (test data). The *"Epochs Completed"* column in Table 3 reflects the actual number of *epochs* completed by the training before Early Stopping was activated, which is often less than the nominal *epoch* count (50, 100, 150) specified.

TABLE 2. 72 hyperparameter Combination Experiment

Epoch	Batch Size	MAE	MSE	RMSE	Training Epochs Completed
50	16	0.006941	0.000089	0.009451	30
50	32	0.007080	0.000096	0.009809	50
50	64	0.008334	0.000134	0.011584	50
100	16	0.008420	0.000140	0.011827	28
…	…	…	…	…	…
100	32	0.008514	0.000143	0.011949	44
100	64	0.007486	0.000109	0.010457	81
150	16	0.008577	0.000148	0.012154	31
150	32	0.007239	0.000101	0.010043	43
150	64	0.006961	0.000091	0.009555	64

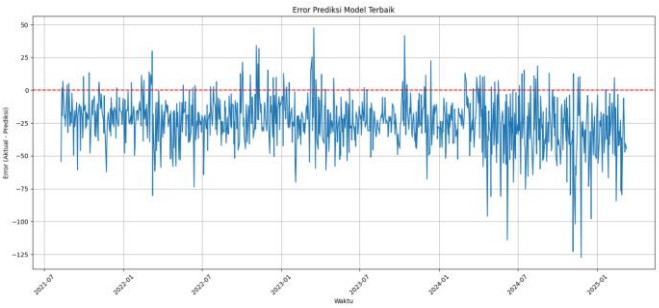

Fig 7. Prediction Model Error Graph

Based on the results from 72 experiments, the *hyperparameter* combination that resulted in the lowest normalized MSE value on the internal validation set (monitored by Early Stopping and confirmed by final evaluation on the test set) was: *Epochs*=100 (although it was stopped earlier by Early Stopping at *epoch* 83), *Batch Size*=16, *Units*=256, and *Dropout Rate*=0.1. The model, retrained with this optimal configuration, was then evaluated on the test dataset. Fig. 7 presents a graph of testing data errors with the best hyperparameter values. After reversing the prediction results to the original price scale through inverse transformation using the same scaler, the following performance metrics were obtained: MAE of 25.761967, MSE of 954.970235, and RMSE of 30.902593. Fig. 8 visualization comparing the actual and predicted prices on the test data demonstrates the ability of this optimal model to capture the general trend of gold price movement.

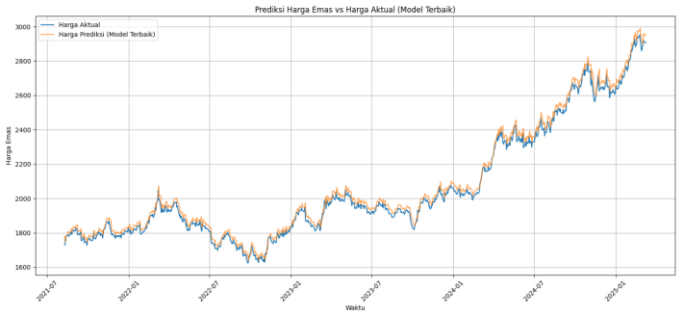

Fig 8. Visualization of Actual vs Predicted Prices

To validate the relative effectiveness of the GRU model, a comparison was made with a simple *Recurrent Neural Network* (RNN) model as the baseline model. Using the MAE metric, the optimized GRU model MAE = 0.017721 (normalized) was compared with the performance of the RNN model. The RNN model produced an MAE of 0.006556 (normalized) on the same test data. This comparison shows that the GRU model demonstrates superior performance compared to RNN, indicating that the gate mechanism in GRU (update gate and reset gate) is more effective in handling long-term dependencies and gradient flow in gold price time series data compared to the simpler RNN architecture. The following is a graph comparing MAE between GRU and RNN in Fig. 9.

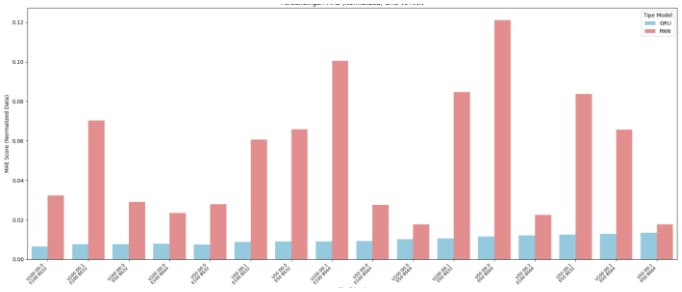

Fig 9. Comparison of MAE GRU and RNN

Although the GRU model shows promising results, the presence of prediction errors indicates that there are factors that cannot be fully modeled. Some potential causes of prediction errors include:

1. *Intrinsic Data Volatility*: Gold prices, like other financial assets, are inherently volatile and influenced by complex and often unpredictable factors. Periods of high volatility, during which prices can fluctuate significantly in a short period of time, pose a particular challenge for time series prediction models. Models may struggle to fully capture sudden spikes or drops in prices if such patterns are not sufficiently represented in the training data.
2. *External Economic Factors*: The models developed in this study only use historical price data as input. However, gold prices are strongly influenced by various external macroeconomic and geopolitical factors that are not explicitly included as features in the models. Factors such as central bank monetary policy (e.g., interest rate changes), exchange rates

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i2.2352, Copyright ©2025
Submitted : May 1, 2025, Revised : May 11, 2025, Accepted : May 14, 2025, Published : May 26, 2025

(particularly the US Dollar), inflation rates, global political uncertainty, industrial demand, and market sentiment can cause price movements that cannot be predicted based solely on historical data. The absence of these exogenous variables in the model may limit its predictive accuracy, especially when these external factors undergo significant changes.

Overall, these findings suggest that the GRU model is a valid and promising approach for gold price forecasting. However, to enhance the accuracy and robustness of the model in the future, considering the integration of relevant external features and exploring more advanced model architectures or hybrid approaches could be beneficial research directions. A more in-depth analysis of error characteristics, as discussed, is also important to guide further model development iterations.

IV. CONCLUSION

This study successfully implemented and evaluated a *Gated Recurrent Unit* (GRU) model with a four-layer architecture for the task of daily gold closing price prediction. Through a systematic hyperparameter optimization process, testing 72 different combinations of epochs, batch size, units per GRU layer, and dropout rate, it was found that the configuration with 100 nominal epochs (stopped early at epoch 83 by Early Stopping), batch size of 16, 256 units, and dropout rate of 0.1 resulted in the best performance based on the MSE metric on the validation data. Evaluation on the test data showed that this optimal model was able to capture the main patterns and trends in the historical gold price data, achieving an RMSE value of around 30.90 after the scale was reversed. These results indicate that the GRU model, when properly configured and trained using techniques such as Early Stopping to prevent overfitting, is a promising approach for forecasting complex financial time series such as gold prices. However, the presence of prediction errors suggests that further improvements are still possible, such as through the addition of external features or exploration of more advanced model architectures

ACKNOWLEDGMENT

We would like to express our deepest gratitude to the faculty and staff at *Universitas Nahdlatul Ulama Sunan Giri* for their continuous support and guidance throughout this research. Their valuable insights and feedback have been instrumental in the development and completion of this study. We also extend our sincere thanks to TradingView for providing the historical gold price data, which formed the foundation of this research.

REFERENCES

[1] M. Fauzi, M. Jauhar Vikri, and S. Wahyudhi, 'Sistem PendukungcKeputusan Pemilihan Jenis Investasi MenggunakanxMetode Analytical Hierarchy Process (AHP)'.

[2] A. F. Yuliana and R. Robiyanto, 'PERAN EMAS SEBAGAI SAFE HAVEN BAGI SAHAM PERTAMBANGAN DI INDONESIA PADA PERIODE PANDEMI COVID-19', *Jurnal Ilmiah Bisnis dan Ekonomi Asia*, vol. 15, no. 1, pp. 1–11, Feb. 2021, doi: 10.32815/jibeka.v15i1.217.

[3] S. Novianto and H. Akbar Wibowo, 'The Implementation of Data Mining for Predicting XAU/USD Price Trends in the Forex Market on MetaTrader 5 using Naïve Bayes Method', *Intelmatics*, vol. 3, no. 2, pp. 85–90, Sep. 2023, doi: 10.25105/itm.v3i2.17199.

[4] R. Mattera, G. Athanasopoulos, and R. Hyndman, 'Improving out-of-sample forecasts of stock price indexes with forecast reconciliation and clustering', *Quant Finance*, vol. 24, no. 11, pp. 1641–1667, Nov. 2024, doi: 10.1080/14697688.2024.2412687.

[5] C. Alkahfi, A. Kurnia, and A. Saefuddin, 'Perbandingan Kinerja Model Berbasis RNN pada Peramalan Data Ekonomi dan Keuangan Indonesia', *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 4, pp. 1235–1243, Jul. 2024, doi: 10.57152/malcom.v4i4.1415.

[6] A. I. Putri, Y. Syarif, N. R. Aisyi, and N. Waeyusoh, 'Implementation of Gated Recurrent Unit, Long Short-Term Memory and Derivatives for Gold Price Prediction', vol. 2, no. 2, pp. 68–80, 2025, doi: 10.57152/predatecs.v2i2.1609.

[7] M. F. Julianto, M. Iqbal, W. F. Hidayat, and Y. Malau, 'PERBANDINGAN PENERAPAN ALGORITMA DEEP LEARNING DALAM PREDIKSI HARGA EMAS', *INTI Nusa Mandiri*, vol. 19, no. 1, pp. 71–76, Aug. 2024, doi: 10.33480/inti.v19i1.5559.

[8] A. Jamarani, S. Haddadi, R. Sarvizadeh, M. Haghi Kashani, M. Akbari, and S. Moradi, 'Big data and predictive analytics: A sytematic review of applications', *Artif Intell Rev*, vol. 57, no. 7, Jul. 2024, doi: 10.1007/s10462-024-10811-5.

[9] R. Chaudhuri, S. Deb, and H. Das, 'Noble Approach on Sensor Fused Bio Intelligent Path Optimisation and Single Stage Obstacle Recognition in Customized Mobile Agent', *Procedia Comput Sci*, vol. 218, pp. 778–787, Jan. 2023, doi: 10.1016/J.PROCS.2023.01.058.

[10] David Andrés, 'Error Metrics for Time Series Forecasting - ML Pills'. Accessed: May 01, 2025. [Online]. Available: https://mlpills.dev/time-series/error-metrics-for-time-series-forecasting/

[11] R. J. Hyndman and A. B. Koehler, 'Another look at measures of forecast accuracy', *Int J Forecast*, vol. 22, no. 4, pp. 679–688, Oct. 2006, doi: 10.1016/J.IJFORECAST.2006.03.001.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i2.2352, Copyright ©2025
Submitted : May 1, 2025, Revised : May 11, 2025, Accepted : May 14, 2025, Published : May 26, 2025