Optimizing Stock Price Forecasting using Elman RNN with Distributed Training and Hyperparameter Tuning

Anwar Rifai[1*], Ahmad Zubaid Muzzakki[1], Mohammad Syafrullah[2] and Riskiana Wulan[2]

[1] Faculty of Information Technology, Universitas Budi Luhur,
[2] Center for Artificial Intelligence Studies, Universitas Budi Luhur,
Jl. Ciledug Raya, Jakarta, Indonesia
anwar.rifai@budiluhur.ac.id

Abstract. The fluctuation of stock prices is influenced by various internal factors, such as tax policies and earnings per share, as well as external factors including economic conditions and political situations. These nonlinear and volatile characteristics present significant challenges for accurate prediction. This study applies a Recurrent Neural Network (RNN), specifically the Elman architecture, to forecast the daily closing prices of Bank Rakyat Indonesia (BBRI) stock using historical data from 2003 to 2024.To enhance computational efficiency, a distributed training strategy using data parallelism was employed, allowing faster model training on large-scale datasets. Additionally, hyperparameter tuning was carried out to optimize model performance. The best-performing model, optimized through extensive tuning, uses 9 time steps, 16 hidden neurons, a learning rate of 0.00011, ReLU activation, RMSProp optimizer, and Xavier Normal initialization. Evaluation results show that the model achieved a Mean Absolute Error (MAE) of 79.18 IDR, a Root Mean Square Error (RMSE) of 107.20 IDR, and a Mean Absolute Percentage Error (MAPE) of 1.58%. Furthermore, distributed training significantly accelerated the training process up to 33 times faster compared to conventional single-machine setups. These findings demonstrate the importance of distributed computing and thorough hyperparameter optimization in enhancing the performance of deep learning models for financial time series forecasting.

Keywords: Distributed training, financial forecasting, hyperparameter tuning, RNN, stock prediction

1 Introduction

Forecasting stock prices has long been a critical task in financial economics, decision sciences, and investment analysis. The accurate prediction of stock movements can provide significant advantages to investors, financial analysts, and policymakers. Stock price dynamics are inherently complex, driven by both micro-level internal factors, such as earnings per share (EPS), company revenue, dividend policies, and capital structure and macro-level external influences including inflation, interest rates, foreign

N. Alyani Ishak et al. (eds.), *Proceedings of the International Conference on Cross- Disciplinary Academic Research 2025 - Track 2 Advances in Business & Economics, Social Science, Communications & Media (ICAR-T2 2025)*, Advances in Economics, Business and Management Research 385,
https://doi.org/10.2991/978-94-6239-715-6_23

exchange fluctuations, global commodity prices, and political events (Mirza et al., 2025). These factors interact in non-linear and dynamic ways, leading to highly volatile price movements that are difficult to model using conventional methods (Karasan et al., 2025).

In the Indonesian context, the capital market has experienced rapid growth and democratization in recent years, with a surge of retail investors and increasing digital access to financial instruments. Among the stocks listed on the Indonesia Stock Exchange (IDX), Bank Rakyat Indonesia (BBRI) stands out as a flagship banking stock. As a state-owned enterprise with an extensive microfinance network and consistently strong financial performance, BBRI commands high investor confidence. Its stock is one of the most actively traded on the IDX, often used as a proxy for the performance of the national banking sector. BBRI's strategic position, historical stability, and relevance in Indonesia's economic development make it a compelling subject for predictive modeling (Napitupulu et al., 2021)

However, modeling and forecasting stock prices remains a formidable challenge. Traditional methods such as ARIMA, linear regression, and exponential smoothing often fail to capture the nonlinear dependencies and temporal irregularities embedded in financial time series. These methods typically assume stationarity and linear relationships, assumptions that do not hold in volatile market conditions (Divyashree et al., 2024). Moreover, they are often limited in their ability to incorporate long-term dependencies in sequential data.

In recent years, the emergence of machine learning (ML) and deep learning (DL) has significantly transformed the landscape of stock market prediction. These models are capable of learning hidden patterns and complex relationships from large volumes of data. Among them, Recurrent Neural Networks (RNNs) have shown particular promise for time series forecasting due to their ability to capture temporal dependencies by maintaining memory across time steps (Khan et al., 2025). Within the RNN family, the Elman RNN first introduced by Jeffrey Elman in 1990 employs a context layer that stores the hidden state from previous time steps, making it suitable for sequential modeling such as stock price prediction (Harbaoui & Elhadjamor, 2024).

Despite their advantages, RNNs present several practical challenges. First, training RNNs is computationally intensive, especially with large datasets and complex model architectures. The iterative nature of time-series learning, combined with the need for fine-tuning hyperparameters, often results in long training times and significant demands on computing resources. Second, improper tuning of model parameter such as learning rate, number of hidden neurons, choice of optimizer, and weight initialization strategy can lead to underfitting, overfitting, or vanishing gradients, thus diminishing model performance (Hao et al., 2025).

However, a critical gap remains in the efficient optimization of these models. Although gated architectures like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) are widely used to mitigate the vanishing gradient problem, they are computationally expensive and may be excessive for datasets where simpler architectures could suffice if properly tuned. The standard Elman RNN, while faster during inference, is notoriously difficult to train effectively due to its sensitivity to hyperparameter

initialization. Existing studies often overlook the potential of optimizing simple Elman RNNs to achieve a balance between speed and accuracy.

To address these issues, distributed training has emerged as a powerful solution. By leveraging multiple computational nodes (e.g., CPUs, GPUs, or distributed systems), training workloads can be parallelized and executed simultaneously, significantly reducing the time required to train deep learning models. This is especially effective in large-scale environments where datasets are extensive and models are deep and complex (Dehghani & Yazdanparast, 2023). Among distributed training strategies, data parallelism where data is split among worker nodes, and gradients are averaged across workers offers an efficient balance between implementation complexity and scalability (Xie et al., 2025).

Additionally, hyperparameter optimization plays a crucial role in achieving high model accuracy. Unlike model parameters that are learned during training, hyperparameters are manually set and can dramatically influence learning outcomes (Kaissar et al., 2025). The number of time steps, learning rate, optimizer type (e.g., SGD, Adam, RMSProp), activation function (e.g., ReLU, tanh), number of hidden neurons, and weight initialization strategy are some of the key hyperparameters that need careful tuning to ensure convergence and prevent learning pitfalls such as exploding or vanishing gradients (Wakimoto et al., 2024). Hyperparameter optimization is often performed using grid search, random search, or more advanced methods like Bayesian optimization or genetic algorithms.

This study focuses on optimization efficiency. We propose a hybrid approach that integrates the Elman Recurrent Neural Network, distributed training, and systematic hyperparameter tuning to forecast the daily closing stock price of Bank Rakyat Indonesia (BBRI) using historical data from 2003 to 2024. By combining the temporal modeling power of Elman RNN with the scalability of distributed training and the precision of hyperparameter optimization, we aim to develop a predictive model that is both accurate and computationally efficient.

We evaluate the model using standard metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) to assess prediction accuracy. Furthermore, we compare the training time performance with and without distributed training to quantify the efficiency gains. The outcome of this study is expected to provide both theoretical insights and practical tools for applying deep learning in financial forecasting, particularly in emerging markets like Indonesia, where computational resources may be limited and time-to-result is critical for investment decisions.

2 Method

2.1 Data Collection

The dataset used in this research was obtained from the Yahoo! Finance platform, which provides comprehensive historical stock price information for both national and international companies. Specifically, this study focuses on the historical stock prices of Bank Rakyat Indonesia (BRI) for the period from November 10, 2003, to June 16,

2024, comprising a total of 5,102 records. The dataset includes key financial indicators such as *Open*, *High*, *Low*, *Close*, and *Volume*. The data was retrieved in CSV format using the yfinance Python library, which offers convenient programmatic access to the Yahoo Finance API for financial market data acquisition. The downloaded data was then structured and preprocessed for further analysis.

2.2 Preprocessing

The preprocessing phase includes several essential steps: data normalization, the sliding window method, and dataset splitting.

Data Normalization. Data normalization is the process of rescaling raw data so that it falls within a certain range, typically between 0 and 1 or -1 and 1. This step is crucial in machine learning to ensure that features with large numeric ranges do not disproportionately influence the model compared to those with smaller ranges. The normalization applied in this study uses the Min-Max scaling technique, calculated as shown in Equation (1):

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{1}$$

Sliding Window. The sliding window technique is used to convert time-series data into a supervised learning format. It works by creating a window of a fixed size that moves across the dataset. Each window contains a sequence of consecutive data points used as input, and the next immediate point is used as the target output. This approach helps the model recognize patterns and trends in sequential data, which is critical in time series forecasting tasks.

Train-Test Split. In this stage, the dataset is partitioned into three separate subsets, each serving a specific role in the model development process:

Training Set: This subset, comprising 70% of the total data, is used to train the model. The model learns the underlying patterns and trends in the data through iterative updates of its internal parameters.

Validation Set: Consisting of 15% of the data, the validation set is utilized to fine-tune the model's hyperparameters such as the number of hidden layers, learning rate, or batch size and to monitor the model's performance during training. This set helps prevent overfitting by evaluating how well the model generalizes to data it hasn't seen during weight updates.

Test Set: The remaining 15% of the data is reserved as the test set. It is strictly used for the final evaluation of the trained model. Since the test data remains completely unseen during both training and validation, it provides an unbiased estimate of the model's ability to generalize to real-world data.

This 70:15:15 split ratio is strategically chosen to balance the need for sufficient training data with reliable validation and testing phases. This chronological split is critical in financial time series to prevent "look-ahead bias" or data leakage, ensuring that the model is strictly trained on past data to predict future values.

2.3 Model Initialization

Before training begins, a set of hyperparameters must be carefully defined to guide the model's learning process. The sequence length determines how many previous time steps are considered in making predictions, while the number of input neurons corresponds to the dimensionality of each input feature. The hidden neurons control the model's ability to capture complex temporal patterns, and the output neurons typically represent the predicted value at each time step. A suitable learning rate is chosen to balance the speed and stability of learning, while the number of epochs defines how many times the model will iterate over the entire training dataset. The choice of activation function introduces non-linearity to help the model capture intricate relationships in the data. Additionally, an optimizer is selected to minimize the loss function effectively, and a weight initialization strategy is applied to provide a good starting point for training. These initial configurations are critical to ensure the model learns efficiently and generalizes well to unseen data.

2.4 Data Parallelization

Data parallelization refers to distributing subsets of the dataset across multiple computing units (workers), allowing simultaneous processing. This strategy significantly accelerates the training process, particularly when handling large-scale datasets, by utilizing available CPU resources to process multiple batches in parallel. In this study, data parallelization was implemented using the MPI4Py framework, which enables message passing interface (MPI)-based communication in Python for parallel computing tasks. The training and validation processes were executed across four compute nodes and one head node, each equipped with an NVIDIA GeForce GTX 1080 Ti. Each node managed 4 to 8 worker units, collaboratively handling data loading, model training, and evaluation to enhance computational efficiency and reduce overall training time.

2.5 Training Process

The training phase is conducted in three main stages: forward pass, backward pass, and weight updates.

Forward Pass. In this stage, the hidden state h_t is calculated based on the input x_t and the weight parameters. Then, the output y_t is produced. The prediction error is computed by comparing y_t with the actual target using a loss function. The formula (2) and (3) are used in this step.

$$h_t = f(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h) \quad (2)$$

$$y_t = W_{hy} \cdot h_t + b_y \quad (3)$$

Where:
f is the activation function,
W_{xh}, W_{hh}, W_{hy} are weight matrices,
b_h, b_y are biases.

Backward Pass (Backpropagation Through Time). After calculating the error from the forward pass, Backpropagation Through Time (BPTT) is applied to compute the gradients across time steps. The objective of BPTT is to optimize the weights and biases to minimize the loss function.

Model Update. Weight and bias updates are performed using the Gradient Descent method. This algorithm seeks to find the optimal parameters θ that minimize the cost function $J(\theta)$. The parameter update step is given by Equation (4)

$$\theta = \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \tag{4}$$

Where:

α is the learning rate,
$\frac{\partial J(\theta)}{\partial \theta}$ is the gradient of the cost function.
This update process continues iteratively until convergence.

2.6 Testing

The trained model is then tested using the testing dataset, which was not used during training. The model makes predictions on this unseen data, and the results are compared with actual values. The goal of testing is to assess the model's generalization ability, not just its ability to memorize training data.

The performance is evaluated using the following metrics:
Mean Absolute Error (MAE) – Equation (5):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{5}$$

Root Mean Squared Error (RMSE) – Equation (6):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{6}$$

Mean Absolute Percentage Error (MAPE) – Equation (7):

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n} \left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{7}$$

If the evaluation results are not satisfactory, hyperparameter tuning is performed to improve model accuracy.

2.7 Forecasting

In the final phase, the trained model is used to make stock price forecasts. The model utilizes either the latest data from the training set or newly obtained input data to make

future predictions. This forecasting process relies on the patterns and temporal relationships learned during the training phase

3 Result and Discussion

This section discusses the results obtained from the implementation of the Elman Recurrent Neural Network (RNN) model for predicting the daily closing prices of Bank Rakyat Indonesia (BBRI). The discussion is presented in a descriptive and analytical manner, covering the model's prediction accuracy, training efficiency under distributed settings, the role of hyperparameter tuning, and a comparison with findings from recent studies.

3.1 Prediction Accuracy and Error Analysis

The predictive performance of the Elman Recurrent Neural Network (RNN) model was evaluated using three commonly accepted metrics: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE). *These* metrics allow for a comprehensive assessment of the model's capability to forecast BBRI's daily closing stock prices accurately and consistently. Table 1 show the prediction Accuracy using RNN and RNN combined with hyperparameter optimalization.

Table 1. Prediction Accuracy of Elman RNN Model on BBRI Stock Data

Metric	Without hyperparameter	Using Hyperparameter
MAE	1237.27IDR	79.18 IDR
RMSE	1460.30 IDR	107.20 IDR
MAPE	24%	1.58%

The results indicate that the model achieved high forecasting accuracy across all three evaluation metrics after using hyperparamter. The MAE of 79.18 IDR implies that, on average, *the* model's predictions deviate from the actual closing price by approximately 79 Rupiah. This level of precision is favorable given the volatility typical in daily stock trading.

The RMSE value of 107.20 IDR confirms the model's robustness in minimizing larger prediction errors. Since RMSE penalizes large deviations more heavily than MAE, a relatively low RMSE signifies that the model is not only accurate on average but also stable across different time periods.

Meanwhile, the MAPE of 1.58% illustrates excellent relative accuracy. In financial time series forecasting, a MAPE below 2% is generally considered outstanding, especially in daily price prediction where market noise and rapid fluctuations are common. This finding aligns with Bansal et al. (2022), who state that models achieving a MAPE in the 1% to 5% range are highly reliable for short-term predictions in volatile financial environments. Compared to baseline unoptimized LSTM models which often

hover around 2-3% MAPE due to overfitting on smaller datasets (Harbaoui & Elhadjamor, 2024), our optimized Elman RNN demonstrates superior performance. This suggests that a simpler architecture, when rigorously tuned via distributed search, can outperform more complex models in specific high-frequency trading scenarios.

Collectively, these results demonstrate that the Elman RNN when supported by appropriate data preprocessing, optimized architecture, and sufficient historical depth can effectively model the nonlinear and sequential patterns in stock price behavior. The high accuracy achieved confirms its potential application in real-time decision-making tools within financial markets.

3.2 Training Efficiency with Distributed Learning

Another important contribution of this research lies in the implementation of data-parallel distributed training. Fig 1. shows the visualization of the work unit. When training was performed on a single CPU, the model took approximately 3960 seconds to complete 100 epochs. In contrast, the distributed training setup using four CPUs reduced the training time to about 120 seconds, marking a 33x acceleration in computational time.

This dramatic improvement is aligned with the findings of Patil & Chickerur (2023), who reported that distributed learning techniques, particularly data-parallelism, are effective in speeding up deep neural network training without sacrificing model performance. In our study, the accuracy metrics between the distributed and single-CPU versions of the model were nearly identical, confirming that the speed-up did not introduce degradation in learning quality.

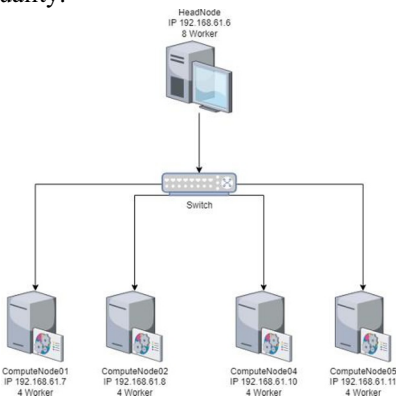

Fig 1. Visualization of the work unit

Such improvement in training speed is particularly beneficial in real-world financial modeling scenarios where quick model updates are often required in response to new data or shifting market conditions.

3.3 Influence of Hyperparameter Optimization

This study also emphasizes the critical role of hyperparameter optimization in enhancing model performance. As illustrated in Figure 2, a clear improvement is observed

after applying hyperparameter optimization techniques. The optimization was conducted using a grid search approach, in which various combinations of hyperparameters were systematically evaluated. To facilitate clarity and readability, the specific values explored during this process are summarized in Table 2 below

Table 2. Hyperparameter Search Space

Hyperparameter	Values Explored
Time Steps	3, 5, 7, 9
Hidden Layer Neurons	8, 16, 32, 64
Learning Rate	0.001, 0.0005, 0.00011
Optimizer	RMSProp, Adam
Activation Function	ReLU, Tanh

This structured approach allowed the identification of the optimal combination of pa- rameters that significantly improved the accuracy and stability of the model during training and validation phases.

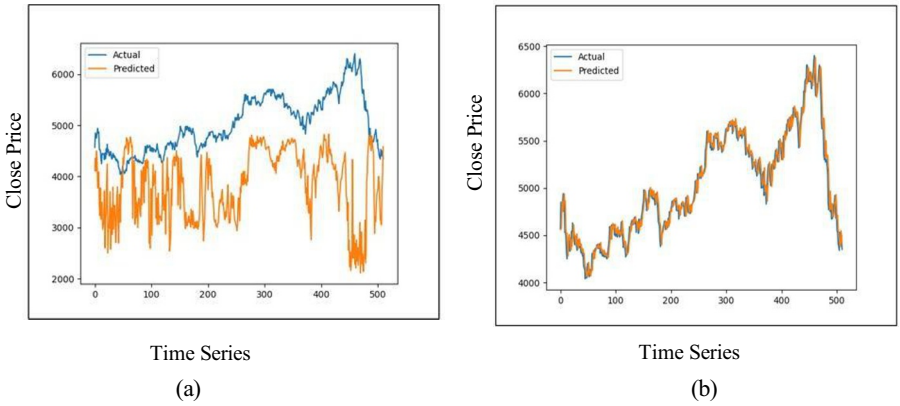

Fig 2. Before Hyperparameter Optimization (a) After Hyperparameter Optimization

It was observed that the best-performing configuration 9 time steps, 16 hidden neurons, a learning rate of 0.00011, ReLU activation, RMSProp optimizer with $\beta = 0.9$, and Xavier Normal weight initialization achieved the lowest MAPE. In contrast, increasing the number of neurons beyond 32 often led to overfitting, and higher learning rates caused convergence instability. The relatively low number of hidden neurons (16) required for optimal performance validates our hypothesis that for BBRI stock data, a

concise model prevents the memorization of noise, unlike deeper architectures which are prone to overfitting.

These observations align with the conclusions made by Rahamathulla & Ramaiah (2025) who emphasized that hyperparameter tuning is not only necessary for performance optimization but also for model stability and generalization. The careful design of this tuning process is therefore considered one of the key success factors in the model's performance.

To further validate the outcomes, the results were compared with previous studies involving Elman RNN in similar contexts. For example, Moonlight et al (2023) implemented Elman RNN for stock prediction on Indonesian banks and reported MAPE values ranging from 3% to 11%, depending on the stock and configuration used. Compared to their results, the 1.58% MAPE achieved in this study indicates a significant improvement in prediction accuracy.

Likewise, Zulfa et al., (2021) applied Elman RNN for predicting ocean current velocity and reported MAPE values around 3.12%. Although in a different domain, the comparison reinforces the general effectiveness of Elman RNNs for temporal forecasting when combined with systematic hyperparameter tuning and efficient training mechanisms.

4. Conclusion

This study presented an Elman Recurrent Neural Network (RNN) model for forecasting the daily closing prices of Bank Rakyat Indonesia (BBRI) stock, integrating two key enhancements: systematic hyperparameter tuning and distributed training via data parallelism. The model was trained on historical data spanning from 2003 to 2024, and its performance was evaluated using three standard metrics: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE).

The results indicate that the model achieved strong predictive accuracy, with an MAE of 79.18 IDR, an RMSE of 107.20 IDR, and a MAPE of 1.58%. These results suggest that the Elman RNN is capable of modeling complex temporal patterns in stock price time series data. The statistical validation of the model's predictions further confirmed their significance. Additionally, the use of distributed training dramatically reduced the model's training time by a factor of 33 demonstrating the practicality and scalability of this approach for time-sensitive financial applications.

The study also highlighted the critical role of hyperparameter optimization. It was shown that even slight changes in learning rate, number of neurons, or activation function can significantly influence model performance. Through careful tuning, the model was able to achieve a high level of accuracy without overfitting, despite its relatively simple architecture.

While the model performed well under historical conditions, it is important to note that deep learning models are inherently data-driven and may not fully capture unpredictable market shocks, such as geopolitical crises or sudden regulatory changes. Therefore, predictions generated by such models should be used as supporting tools, supplemented with expert judgment and external analysis.

For future research, it is recommended to explore more advanced RNN variants, such as Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU), and com-

pare their performance under similar distributed training settings. Additionally, incorporating macroeconomic indicators or sentiment analysis data could further enhance prediction robustness and interpretability. Finally, real-time implementation of this model on streaming financial data would be a valuable next step toward practical deployment in algorithmic trading systems.

References

Bansal, M., Goyal, A., & Choudhary, A. (2022). Stock Market Prediction with High Accuracy using Machine Learning Techniques. *Procedia Computer Science*, *215*, 247–265. https://doi.org/10.1016/j.procs.2022.12.028

Dehghani, M., & Yazdanparast, Z. (2023). From distributed machine to distributed deep learning: a comprehensive survey. Journal of Big Data, 10(1), 158. https://doi.org/10.1186/s40537- 023-00829-x

Divyashree, S., Joshua, C. J., Md, A. Q., Mohan, S., Abdullah, A. S., Mohamad, U. H., Innab, N., & Ahmadian, A. (2024). Enabling business sustainability for stock market data using machine learning and deep learning approaches. Annals of Operations Research, 342(1), 287–322. https://doi.org/10.1007/s10479-024-06118-x

Hao, S., Li, H.-W., Ni, Y.-Q., Zhang, W., & Yuan, L. (2025). State estimation in structural dynamics through RNN transfer learning. Mechanical Systems and Signal Processing, 233, 112767. https://doi.org/10.1016/j.ymssp.2025.112767

Harbaoui, H., & Elhadjamor, E. A. (2024). Enhancing Stock Price Prediction: LSTM-RNN Fusion Model. Procedia Computer Science, 246, 920–929. https://doi.org/10.1016/j.procs.2024.09.511

Kaissar, A., Nassif, A. B., Soudan, B., & Injadat, M. (2025). Enhancing CNN-based network intrusion detection through hyperparameter optimization. Intelligent Systems with Applications, 26, 200528. https://doi.org/10.1016/j.iswa.2025.200528

Karasan, A., Alp, O. S., & Weber, G.-W. (2025). Machine learning approach to stock price crash risk. Annals of Operations Research. https://doi.org/10.1007/s10479-025-06596-7

Khan, S., Mazhar, T., Shahzad, T., Ali, T., Ayaz, M., Ghadi, Y. Y., Aggoune, E.-H. M., & Hamam, H. (2025). Optimizing load demand forecasting in educational buildings using quantum-inspired particle swarm optimization (QPSO) with recurrent neural networks (RNNs):a seasonal approach. Scientific Reports, 15(1), 19349. https://doi.org/10.1038/s41598-025-04301-z

Mirza, F. K., Pekcan, Ö., Hekimoğlu, M., & Baykaş, T. (2025). Stock price forecasting through symbolic dynamics and state transition graphs with a convolutional recurrent neural network architecture. Neural Computing and Applications, 37(20), 15855–15890. https://doi.org/10.1007/s00521-025-11325-z

Moonlight, L. S., Harianto, B. B., Suprapto, Y., & Faizah, F. (2023). Forecasting the Currency Rate of The Indonesian Rupiah (IDR) against the US Dollar (USD) Using Time Series Data and Indonesian Fundamental Data. International Journal on Advanced Science, Engineering and Information Technology, 13(2), 694–702. https://doi.org/10.18517/ijaseit.13.2.17944

Napitupulu, H., Hidayana, R. A., & Saputra, J. (2021). Determination of VaR on BBRI Stocks and BMRI Stocks Using the ARIMA-GARCH Model. Operations Research: International Conference Series, 2(3), 71–74. https://doi.org/10.47194/orics.v2i3.178

Patil, M. S., & Chickerur, S. (2023). Study of Data and Model parallelism in Distributed Deep learning for Diabetic retinopathy classification. Procedia Computer Science, 218, 2253– 2263. https://doi.org/10.1016/j.procs.2023.01.201

Rahamathulla, M. Y., & Ramaiah, M. (2025). Optimizing anomaly detection models for edge IIoT with an enhanced firefly algorithm-based hyperparameter tuning strategy. Results in

Engineering, 27, 105843. https://doi.org/10.1016/j.rineng.2025.105843
Wakimoto, S., Matsukawa, Y., & Aoki, H. (2024). New criteria to select reasonable hyperparameters for kinetic parameter estimation in distributed activation energy model (DAEM) by using neural network. Chemical Engineering Science, 285, 119597. https://doi.org/10.1016/j.ces.2023.119597
Xie, Z., Zhang, K., Chen, J., Lee, C.-G., & He, S. (2025). A distributed training method with intergenerational accumulation and cross-node random drop for mechanical fault diagnosis. Applied Soft Computing, 181, 113532. https://doi.org/10.1016/j.asoc.2025.113532
Zulfa, I. I., Novitasari, D. C. R., Setiawan, F., Fanani, A., & Hafiyusholeh, M. (2021). Prediction of Sea Surface Current Velocity and Direction Using LSTM. IJEIS (Indonesian Journal of Electronics and Instrumentation Systems), 11(1), 93. https://doi.org/10.22146/ijeis.63669