MATEMATIKA, MJIAM, Volume 41, Number 2, 151–166

A Volatility-Responsive LSTM Approach for Predicting KLCI Closing Prices Using Dynamic Optimizer Switching

[1,2]Abang Mohammad Hudzaifah Abang Shakawi* and [2]Ani Shabri

[1]Department of Mathematical Sciences, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Malaysia

[2]Centre for Pre-University Studies Universiti Malaysia Sarawak, 94300, Sarawak, Malaysia

*Corresponding author: asamhudzaifah@unimas.my

Article history
Received: 12 June 2024
Received in revised form: 15 March 2025
Accepted: 30 May 2025
Published on line: 1 August 2025

Abstract In financial time series forecasting, the ability of models to adapt to changing market conditions is critical for improving prediction accuracy. This study explores a novel approach to optimizing Long Short-Term Memory (LSTM) models by dynamically adjusting optimizers based on market volatility, specifically using the Average True Range (ATR) as a volatility indicator. Traditional optimizers like Adaptive Moment Estimation (Adam), Root Mean Squared Propagation (RMSprop), and Stochastic Gradient Descent (SGD) each offer distinct advantages under different market conditions; however, their effectiveness is limited when applied uniformly throughout the training process. To address this limitation, a dynamic optimization strategy was proposed, that switches between optimizers during the training process based on ATR values, enhancing the model's adaptability. This method was applied to predict the Kuala Lumpur Composite Index (KLCI) closing prices, showing improved prediction performance over conventional models that rely on a single optimizer. This adaptive approach offers a robust solution for stock index prediction in volatile markets, contributing to the broader field of financial forecasting through the integration of volatility-driven learning techniques.

Keywords Bursa Malaysia; Dynamic Optimizer Switching; LSTM; Machine learning; Time series; Volatility-Responsive LSTM.

Mathematics Subject Classification 37M10, 62P20, 68T99, 91B84.

1 Introduction

Financial markets are characterized by their dynamic and volatile nature, posing significant challenges for accurate prediction and analysis [1]. The Kuala Lumpur Composite Index (KLCI), representing the performance of the Malaysian stock market, is no exception. Traditional statistical models often struggle to capture the non-linear patterns and abrupt changes present in financial time series data. To address these limitations, machine learning approaches,

particularly Long Short-Term Memory (LSTM) networks, have garnered substantial attention due to their proficiency in modelling sequential data and capturing long-term dependencies [2].

LSTM networks, a specialized form of recurrent neural networks (RNN), are particularly adept at handling sequential data with long-term dependencies, making them well-suited for financial time series forecasting [3]. Unlike traditional neural networks, which struggle with vanishing or exploding gradients over extended sequences, LSTMs utilize a unique architecture that includes memory cells and gating mechanisms. These mechanisms enable LSTMs to retain and prioritize important information over multiple time steps, allowing them to capture underlying patterns in stock prices or other time-dependent data.

Within these networks, optimization plays a pivotal role, governing how effectively the model converges toward an optimal solution. Optimizers in machine learning can generally be categorized into two main types: metaheuristic optimizers and gradient-based optimizers. Metaheuristic optimizers, such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA), explored the solution space through population-based approaches that mimic natural processes and does not rely on gradient information [4]. They employ mechanisms such as mutation, crossover, and swarm intelligence to explore and exploit the solution space, often making them suitable for highly non-convex problems or those with numerous local optima [5]. These methods are particularly useful for optimizing complex, non-convex problems, but they tend to be computationally intensive and can require significant time to converge to optimal solutions.

In contrast, gradient-based optimizers, such as Adaptive Moment Estimation (Adam), Root Mean Squared Propagation (RMSprop), and Stochastic Gradient Descent (SGD), utilize gradient information to update model parameters iteratively [6]. They are particularly efficient for large datasets and high-dimensional parameter spaces, making them the preferred choice for training neural networks, including LSTM models. These optimizers are designed to converge quickly to local minima by leveraging the gradient of the loss function, facilitating the fine-tuning of model weights [7].

A variety of gradient optimizers exist beyond the commonly used Adam [8], SGD [9], and RMSprop [10], each offering unique methods to enhance convergence and generalization. AdaGrad, short for Adaptive Gradient Algorithm, tailors learning rates for individual parameters based on historical gradient information, favoring larger updates for infrequent parameters, which proves useful in LSTM tasks like NLP where certain tokens occur more often than others [11]. While beneficial for sparse and high-dimensional data, AdaGrads rapid decay in learning rate may lead to slower training.

AdaDelta, an extension of AdaGrad, addresses AdaGrads rate decay by retaining a history of gradients within a sliding window [12]. This adaptation allows learning rates to adjust dynamically without a global setting, beneficial for nonstationary data, although it requires extra memory to store past gradients. Nadam (Nesterov-accelerated Adam) combines Adam's adaptivity with Nesterov momentum, leading to faster convergence, especially in complex LSTM models [13]. While computationally intensive, Nadam's efficiency is often offset by its acceleration of convergence.

AMSGrad modifies Adam's update rules to correct its convergence in noisy gradient environments, ensuring stable learning by maintaining a maximum of past squared gradients [14]. This optimizer offers more stability, albeit sometimes at the expense of speed compared to Adam. AdaBound introduces bounds to learning rates, effectively blending adaptive methods with

SGD [15]. Initially behaving like Adam, it gradually transitions to SGD, creating smoother convergence at the cost of added tuning for the transition phase. Radam [16], a modified Adam, includes a rectification term to stabilize early-stage training, useful for LSTMs facing high gradient variance, while Yogi [17] and NovoGrad [18] enhance gradient handling for nonstationary and large-scale data, respectively. Ranger combines Radam's stability and Lookahead's exploratory updates, offering accelerated, stable learning, particularly for NLP [19]. Several optimizers like SGDW [20], AdaMax [21], and Fromage [22] modify weight decay or adapt norms to enhance stability and generalization, each with unique strengths suited to varying LSTM model scales and data sparsity.

Beyond individual optimizers, some techniques incorporate switching mechanisms, adjusting strategies during training to address specific training phases. Dynamic switching optimizers take this concept a step further by combining the strengths of multiple optimizers within a single training session. Methods like SWATS (Switching from Adam to SGD) allows a model to transition from one optimization technique to another based on training progression, adapting to different stages of the learning process [23]. Early-stage training, which typically benefits from adaptive optimizers like Adam, often prioritizes fast convergence, while later stages may require simpler optimizers like SGD for better generalization. By allowing for an adaptive switch between optimizers, dynamic switching can harness the strengths of various optimization strategies, adjusting to changing data patterns and training dynamics [24].

In scenarios involving frequent shifts in data patterns, such as financial markets or other time-series data, optimizers that can adapt to certain conditions become especially valuable. State-sensitive optimization techniques adjust learning rates or optimization strategies based on data fluctuations, aiming to stabilize the training process. For instance, optimizers like AdaSecant [25] use gradient variance as a signal to adapt their behavior, reducing the learning rate during periods of high volatility or switching to a simpler optimizer when fluctuations are detected. These state-driven approaches aim to improve convergence stability and ensure more consistent performance in environments where data patterns are prone to sudden shifts.

However, while both dynamic switching and state-sensitive optimizers have shown promise in handling complex datasets, there remains a gap in integrating volatility sensitivity directly into adaptive LSTM models [26]. Current techniques often apply these optimizers in isolation or within standard architectures, overlooking the potential benefits of directly aligning LSTM model parameters with volatility-driven optimizer behaviours. This research gap suggests an opportunity for developing a volatility-driven approach to LSTM optimization, where model adjustments are dynamically informed by volatility measures, offering more robust and responsive model performance in unpredictable data environments.

Market volatility, reflected in fluctuations in asset prices, significantly impacts the performance of predictive models. High-volatility periods often lead to more rapid price changes, requiring models to adjust their learning strategies accordingly. Conversely, low-volatility periods demand a more conservative approach to avoid overfitting. This makes predicting financial market behaviour particularly challenging due to the unpredictable nature of volatility and the non-linear patterns in the data. Volatility is also a key reflection of market sentiment, capturing the collective perception of investors during periods of uncertainty or stress. Incorporating volatility into stock index forecasting models helps to identify these critical periods, providing deeper insights into investor behaviour and market sentiment, which can anticipate potential market trends and shifts [27]. As volatility often accompanies significant price movements,

integrating it into forecasting models allows for better adaptation to sudden changes in market conditions. This leads to more accurate predictions, especially during volatile periods, which are crucial for making timely and effective decisions [28].

Hence, this study introduced a dynamic optimizer-switching strategy for LSTM models based on volatility patterns, as measured by the Average True Range (ATR). ATR is a valuable tool for capturing dynamics of market volatility, providing a quantitative measure of price fluctuations over time. By leveraging ATR values at each training epoch, this approach dynamically adjusts the model's optimizer to better suit the current volatility regime, enhancing the model's ability to learn effectively under different market conditions. This method aims to improve the prediction accuracy of LSTMs, particularly in volatile financial environments, and is tested on the Kuala Lumpur Composite Index (KLCI) closing price data.

2 Methodology

2.1 Data Collection and Preparation

This study utilized the daily closing prices of the KLCI from January 2, 2018, to January 31, 2023, comprising a total of 1,225 observations obtained from the Yahoo Finance website. The dataset was divided into two segments: the training set, spanning from January 2, 2018, to June 29, 2022 (90% of the data), and the testing set, covering the period from June 30, 2022, to January 31, 2023 (10% of the data).

2.2 Average True Range

ATR is a flexible measure of volatility that quantifies the average magnitude of price fluctuations over a given timeframe [29]. Unlike traditional measures that only consider closing prices, ATR incorporates both intraday price highs and lows, providing a more comprehensive view of market volatility [30]. It is particularly valuable for setting stop-loss levels and determining the potential range of price movements within a given trading session or timeframe.

The ATR is calculated as the mean of the absolute differences between consecutive daily closing prices over a specified period as follows:

$$\mathrm{ATR} = \frac{\sum_{i=1}^{n} \mathrm{TR}_i}{p}, \tag{1}$$

where

- TR_i represents true range of each day i, and
- p represents the number of periods.

2.3 LSTM Model

The LSTM model represents a variation of RNN that incorporates a sequence of iterative computational components. LSTMs are particularly well-suited for tasks involving time series analysis, natural language processing, and other applications where temporal dependencies are crucial [31]. LSTMs incorporate unique memory cells that enable them to maintain information

over long periods. As illustrated in Figure 1, an LSTM unit consists of a memory cell, an input gate, an output gate, and a forget gate, which function collaboratively to process and preserve information over temporal periods [32].

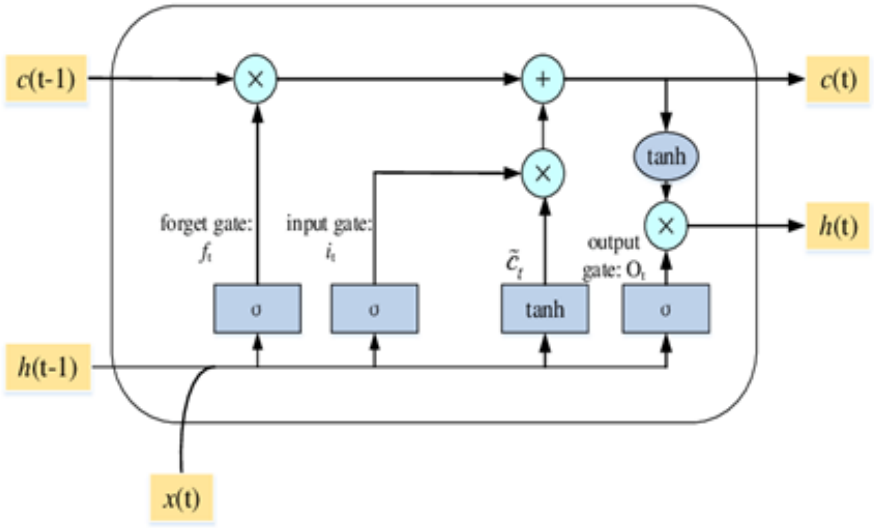


Figure 1: Architecture of the LSTM unit

The forget gate f_t dictates the degree to which the previous cell state c_{t-1} is discarded, computed as $f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$. The input gate i_t determines what new information is retained in the cell state, given by $i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$, and the candidate cell state $\tilde{c}_t$ is calculated using $\tilde{c}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right)$, with hyperbolic tangent function. The current cell state C_t is updated using the formula $C_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$. Finally, the output gate O_t governs the output of the LSTM cell, calculated as $O_t = \sigma\left(W_O \cdot [h_{t-1}, x_t] + b_O\right)$, and the hidden state h_t is derived from $h_t = O_t \odot \tanh\left(c_t\right)$. Here, W_f, W_i, W_c and W_O represents the weight matrices that connect the input to the respective gates while b_f, b_i, b_c and b_O represents the bias terms that are added to the respective gates to adjust the activations, helping the LSTM model account for inherent shifts in the data and improve the gating mechanism's flexibility. The Hadamard product, $\odot$ is an element-wise multiplication, allowing the model to selectively control the flow of information. The σ (sigma) symbol represents the sigmoid activation function, which is used in the input, forget, and output gates to control the flow of information by producing values between 0 and 1, determining how much information should be passed through or discarded.

2.4 Optimizers

2.4.1 SGD Optimizer

SGD updates model parameters using a single or a few randomly chosen data samples, unlike the traditional gradient descent which uses the full dataset for each update [9]. This approach reduces computation time and introduces stochasticity, which can help the optimizer escape local minima in the error surface. SGD updates the models parameters by applying the gradient of the loss function with respect to the weights. The formula for the weight update at time

step t is:

$$w_{t-1} = w_t - \eta \nabla L\left(w_t\right), \tag{2}$$

where

- w_t represents the weight at time t,
- η represents the learning rate, and
- $\nabla L\left(w_t\right)$ represents the gradient of the loss function with respect to the weight.

2.4.2 RMSprop Optimizer

RMSprop adapts the learning rate for each parameter based on recent gradients, where it divides the learning rate by the root of the average of recent squared gradients, rather than accumulating the squares of all previous gradients [33]. This helps prevent the learning rate from shrinking too quickly, allowing the optimizer to continue learning over longer periods. The update rule is:

$$w_{t+1} = w_t \frac{\eta}{\sqrt{E\left[\nabla L\left(w_t\right)^2\right] + \epsilon}} \nabla L\left(w_t\right), \tag{3}$$

where

- w_t represents the weight at time t,
- η represents the learning rate,
- $\nabla L\left(w_t\right)$ represents the gradient of the loss function with respect to the weight,
- $E\left[\nabla L\left(w_t\right)^2\right]$ represents the moving average of squared gradients, and
- ϵ represents a small constant to prevent division by zero.

2.4.3 Adam Optimizer

Adam is a combination of RMSprop and momentum, using both first and second moments of the gradients to adapt the learning rate [8]. This dual mechanism helps the optimizer adjust learning rates based on the magnitude of past gradients, leading to more stable convergence. The update rule is:

$$m_t = \beta_1 m_{t-1} + \left(1 - \beta_1\right) \nabla L\left(w_t\right), \tag{4}$$

$$v_t = \beta_2 v_{t-1} + \left(1 - \beta_2\right) \nabla L\left(w_t\right)^2, \tag{5}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{6}$$

$$w_{t+1} = w_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{7}$$

where

- w_t represents the weight at time t,

- η represents the learning rate,
- ϵ represents a small constant to prevent division by zero,
- m_t and v_t represents the first and second moments (means) of the gradients, and
- β_1 and β_2 represents hyperparameters controlling the decay rates of the moment estimates.

This study utilized the daily closing prices of the KLCI from January 2, 2018, to January 31, 2023, comprising a total of 1,225 observations obtained from the Yahoo Finance website. The dataset was divided into two segments: the training set, spanning from January 2, 2018, to June 29, 2022 (90% of the data), and the testing set, covering the period from June 30, 2022, to January 31, 2023 (10% of the data).

2.5 LSTM Model with ATR-Based Dynamic Optimizer Switching Mechanism

The adaptive optimizer function selects an appropriate optimizer for each epoch by analyzing the ATR values in the context of recent data. Initial hyperparameters such as learning rate, optimizer, and batch size are set, with the optimizer chosen adaptively during training based on ATR values. The ATR values are computed at each epoch over a sliding window of the most recent data points. Based on the ATR value, the optimizer is dynamically changed according to the following rules:

- High Volatility (ATR > a): The optimizer is switched to RMSprop, which is known for handling noisy, non-stationary data effectively.
- Low Volatility (ATR < b): The optimizer is switched to SGD, which performs well in stable market conditions by reducing overfitting.
- Moderate Volatility (b $\leq$ ATR $\leq$ a): The optimizer defaults to Adam, which adapts well to various situations with its balance of speed and stability.

This dynamic optimizer-switching strategy can be mathematically expressed as:

$$\text{Opt}_t = \begin{cases} \text{RMSprop} & \text{if } \text{ATR}_t > a \\ \text{Adam} & \text{if } b \leq \text{ATR}_t \leq a \\ \text{SGD} & \text{if } \text{ATR}_t < b \end{cases} \tag{8}$$

where

- Opt_t represents the optimizer selected for epoch t based on the ATR value at the start of the epoch

The following pseudocode in Figure 2 illustrates the adaptive switching strategy implemented in this studys optimization process. It details how the model dynamically selects between different optimizers such as Adam, RMSprop, and SGD based on predefined conditions that reflect training phase requirements.

Algorithm 1 ATR-Based Dynamic Optimizer Switching for LSTM Training

Require: T: Total epochs, N: Total data points, ATR Values, Thresholds for volatility

for epochs $e = 1$ to T **do**
    Calculate dynamic window size $W \leftarrow \lfloor N/T \rfloor$
    Calculate mean ATR over current window:

$$\text{ATR} = \frac{1}{W} \sum_{i=(e-1)W+1}^{eW} \text{ATR}_i$$

    if ATR_e = High threshold **then**
        Select optimizer: RMSprop
    else if ATR_e = Low threshold **then**
        Select optimizer: SGD
    else
        Select optimizer: Adam
    end if
    Recompile model with chosen optimizer
    Train model on training data for one epoch
    Track training and validation loss
end for
return Trained LSTM model, performance metrics

Figure 2: Pseudocode for Dynamic Optimizer Switching in LSTM

2.6 Model Evaluations

The forecasting models were evaluated using two performance measures: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{9}$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{10}$$

The true value is represented by y_i while $\hat{y}_i$ denotes the estimated value. A smaller MAE and RMSE indicate a more accurate prediction model.

3 Results and Discussion

3.1 Volatility Analysis and Market Movement

The analysis of the ATR values alongside the closing price reveals important insights into the relationship between volatility and stock price behaviour. As shown in Figure 3, the closing price

(blue line) fluctuates throughout the period, with a sharp decline observed around early 2020, likely corresponding to the global market crash due to the COVID-19 pandemic. In contrast, the ATR values (red line), which measure market volatility, remain relatively low during periods of stability but show a significant spike during this market downturn. This indicates that volatility increases when the market experiences large price movements, particularly during periods of uncertainty. After the sharp rise in ATR during early 2020, the volatility values gradually decrease, reflecting market stabilization. This reinforces the rationale for incorporating ATR-based dynamic optimizer switching for LSTM training, as it allows the model to better account for volatility-driven market shifts.

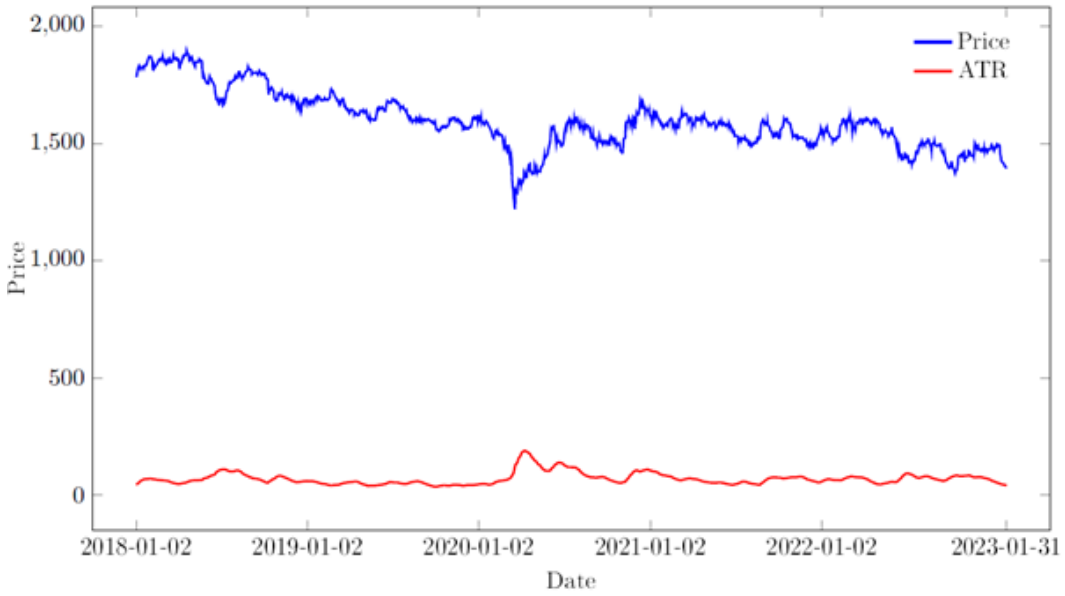


Figure 3: Comparison of ATR Values and KLCI Prices from January 2018 to January 2023

3.2 Hyperparameter Setting on LSTM Model

The hyperparameter values were empirically tuned through an iterative process in order to achieve optimal model performance during the training phase. Following multiple experimental iterations, the hyperparameter settings for both the LSTM model incorporating an ATR-based dynamic optimizer switching mechanism and the standard LSTM model include a single hidden layer with 200 neurons. A dropout rate of 0.2 is applied to prevent overfitting. The models use a timestep of 20, a batch size of 64, and are trained for 50 epochs. The activation function employed is the hyperbolic tangent (tanh), while the recurrent function is the sigmoid function. The tanh function helps the model learn complex, non-linear patterns in stock market data by mapping inputs to a range of $[-1, 1]$. This makes it well-suited for capturing the inherent volatility and fluctuations in financial time series data. The mean squared error serves as the loss function. Three separate standard LSTM models are used in this study, each employing a different optimizer: Adam, SGD, and RMSprop. These models will serve as benchmarks to compare against the LSTM model that employs an ATR-based dynamic optimizer switching mechanism.

3.3 Performance Comparison

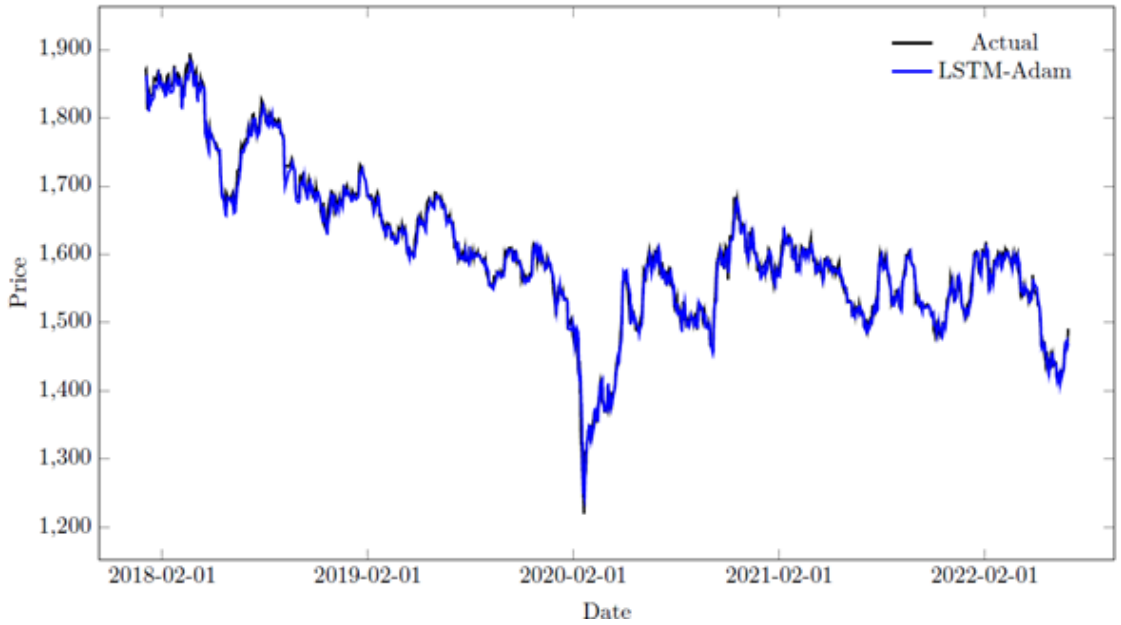
The ATR-based dynamic optimizer LSTM was compared against the standard LSTM model to gauge its predictive power and adaptability across different datasets. The results indicate that the dynamic optimizer switching mechanism consistently yielded lower MAE and RMSE values across training and testing dataset, as presented in Table 1.

Table 1: Forecasting performance of Dynamic Optimizer LSTM and other models

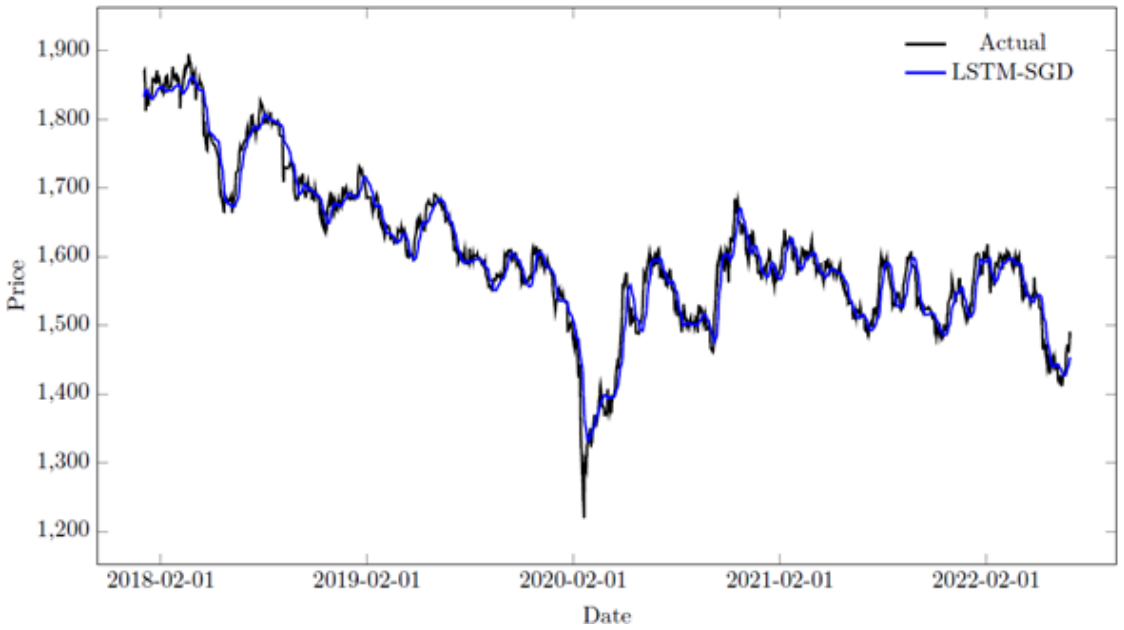
Model	**Training Datasets**		**Testing Datasets**	
	MAE	**RMSE**	**MAE**	**RMSE**
LSTM-SGD	16.25	23.26	13.65	18.08
LSTM-Adam	9.61	12.74	8.46	11.96
LSTM-RMSprop	9.30	12.67	9.36	12.45
ATR Dynamic LSTM	**9.02**	**12.17**	**8.25**	**11.66**

Figure 4 shows the comparison of actual versus predicted values on the training dataset across the four optimizers: Adam, SGD, RMSprop, and the ATR-based dynamic optimizer. In this figure, each line represents the predictions made by a specific optimizer, plotted against the actual values to illustrate their alignment under different market conditions. The Adam optimizer's predicted line closely follows actual values in volatile phases due to its adaptive nature, while SGD's line appears smoother however exhibiting lagging in more volatile segments. RMSprop, balancing between stability and adaptation, typically shows intermediate tracking accuracy. Finally, the ATR-based dynamic optimizer switches between aggressive and conservative settings based on volatility, aiming to enhance prediction accuracy by aligning the model's responsiveness with the current market environment. This dynamic approach was able to capture both stable and volatile patterns in the training data, aiming for a balanced and flexible prediction trend across market regimes.

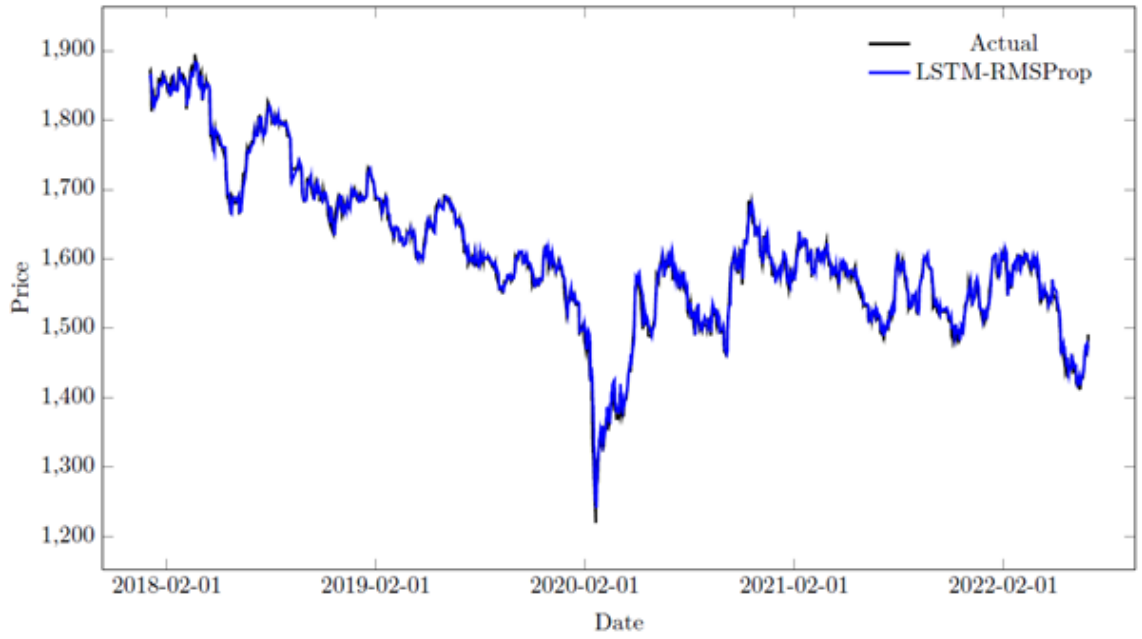
Figure 5 illustrates the comparison of actual versus predicted values on the testing dataset across the four optimizers: Adam, SGD, RMSprop, and the ATR-based dynamic optimizer. In this figure, the ATR-based dynamic optimizer outperforms the other three by maintaining closer alignment with actual values across both stable and volatile market conditions, thanks to its ability to adapt based on volatility. In contrast, SGD shows the weakest performance, with its predictions noticeably lagging and failing to capture rapid market shifts due to its fixed, conservative learning rate. RMSprop performs better than SGD, adapting moderately to changes but still struggling in high-volatility periods. Adam, while superior to SGD and RMSprop due to its adaptive learning rate, often shows over-responsiveness in volatile regions, leading to fluctuations that occasionally misalign with actual values. By comparison, the ATR-based dynamic optimizer adjusts effectively between aggressive and conservative settings, producing a more accurate and steady prediction line that closely follows actual trends, demonstrating a clear advantage in handling unpredictable market patterns.


Actual
LSTM-Adam
Price
1,900
1,800
1,700
1,600
1,500
1,400
1,300
1,200
2018-02-01
2019-02-01
2020-02-01
2021-02-01
2022-02-01
Date

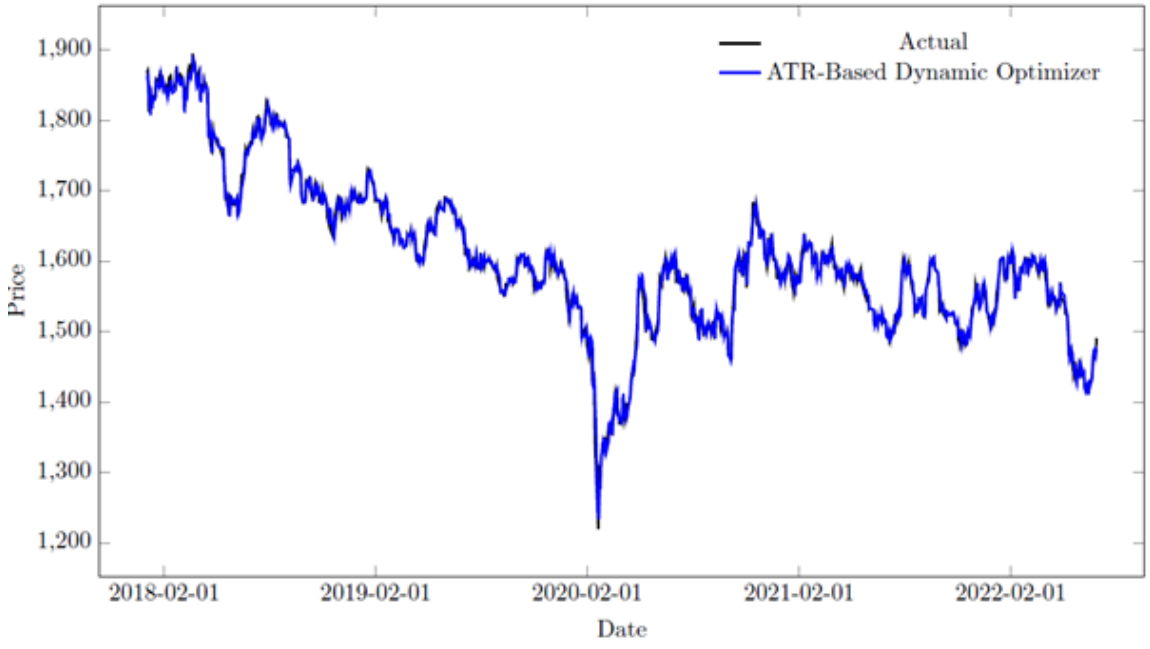
(a)


Actual
LSTM-SGD
Price
1,900
1,800
1,700
1,600
1,500
1,400
1,300
1,200
2018-02-01
2019-02-01
2020-02-01
2021-02-01
2022-02-01
Date

(b)



(c)



(d)

Figure 4: Actual vs. Predicted Values on Training Dataset Across Optimizers: (a) Adam, (b) SGD, (c) RMSprop, and (d) ATR-Based Dynamic Switching

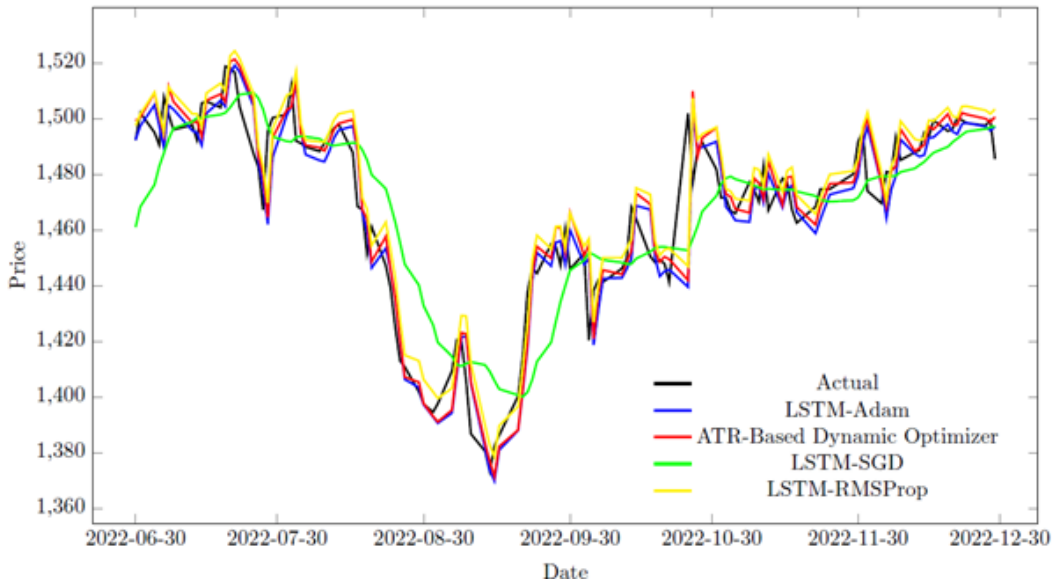


Figure 5: Actual vs. Predicted Values on Testing Dataset Across Optimizers: Adam, SGD, RMSprop, and ATR-Based Dynamic Switching

The KLCI index experienced a drastic drop in 2022, driven by external factors such as political upheaval and the economic recovery phase [34], as shown in Figure 5. This decline is effectively managed by the dynamic adjustment mechanism, which adapts the model's hyperparameters based on market conditions. During sharp market drops, the mechanism adjusts to more aggressive settings, enabling the model to respond quickly to sudden changes, while in stable periods, it uses conservative settings to avoid overreacting to minor fluctuations. This adaptability enhances the model's predictive accuracy and robustness, highlighting the strength of a volatility-driven approach in capturing and responding to significant market shifts.

4 Conclusion

In conclusion, this study demonstrates the effectiveness of a volatility-driven approach in enhancing LSTM model performance for stock index prediction. By incorporating dynamic adjustments based on market volatility, the model achieves a balanced responsiveness, adapting its hyperparameters to align with both stable and volatile conditions. The ATR-based dynamic optimizer switching, in particular, shows notable improvements in prediction accuracy compared to traditional optimizers like Adam, SGD, and RMSprop. The ability of the dynamic optimizer to shift between conservative and aggressive settings ensures better alignment with market trends, especially during drastic index movements, as seen in the KLCI's fluctuations in 2022. These findings underscore the value of incorporating market-driven adjustments in predictive models, offering a robust framework for more reliable stock index forecasting in varying economic environments. Future research may explore extending this approach to other financial indices and examining additional volatility measures to further optimize model adaptability and performance.

Acknowledgments

The authors would like to thank the Ministry of Higher Education Malaysia and Universiti Malaysia Sarawak for financial support of a PhD candidate and for supporting the publication fees..

References

[1] Idrees, S. M., Alam, M. A. and Agarwal, P. A prediction approach for stock market volatility based on time series data. *IEEE Access*. 2019. 7: 17287-17298.

[2] Houdt, V. G., Mosquera, C. and Nápoles, G. A review on the long short-term memory model. *Artificial Intelligence Review*. 2020. 53(8): 5929-5955.

[3] Bharadiya, J. P. Exploring the use of recurrent neural networks for time series forecasting. *International Journal of Innovative Science and Research Technology*. 2023. 8(5): 2023-2027.

[4] Tang, J., Liu, G. and Pan, Q. A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. *IEEE/CAA Journal of Automatica Sinica*. 2021. 8(10): 1627-1643.

[5] Palle, R. R. Evolutionary Optimization Techniques in AI: Investigating Evolutionary Optimization Techniques and Their Application in Solving Optimization Problems in AI. *Journal of Artificial Intelligence Research*. 2023. 3(1): 1-13.

[6] Mehmood, F., Ahmad, S. and Whangbo, T. K. An efficient optimization technique for training deep neural networks. *Mathematics*. 2023. 11(6): 1360.

[7] Karthick, K. Comprehensive Overview of Optimization Techniques in Machine Learning Training. *Control Systems and Optimization Letters*. 2024. 2(1): 23-27.

[8] Kingma, D. and Ba, J. Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015) Ithaca, NY*. 2015. 1-15.

[9] Robbins, H. and Monro, S. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*. 1951. 22(3): 400-407

[10] Tieleman, T. and Hinton, G. Divide the gradient by a running average of its recent magnitude. *Coursera: Neural networks for machine learning*. 2017.

[11] Duchi, J., Hazan, E. and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*. 2011. 12(7): 2121-2159.

[12] Zeiler, M. D. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*. 2012.

[13] Dozat, T. Incorporating nesterov momentum into adam. *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016) San Juan, Puerto Rico*. 2016. 1–4.

[14] Tran, P. T. On the convergence proof of amsgrad and a new version. *IEEE Access*. 2019. 7, 61706-61716.

[15] Savarese, P. On the convergence of adabound and its connection to sgd. *arXiv preprint arXiv:1908.04457* 2019.

[16] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J. and Han, J. *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020) Virtual*. 2020, 1-13.

[17] Zaheer, M., Reddi, S., Sachan, D., Kale, S. and Kumar, S. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*. 2018. 31.

[18] Ginsburg, B., Castonguay, P., Hrinchuk, O., Kuchaiev, O., Lavrukhin, V., Leary, R. and Cohen, J. M. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks. *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020) Virtual*. 2020, 1-13.

[19] Jiang, J. R. and Lin, Y. T. Deep learning anomaly classification using multi-attention residual blocks for industrial control systems. *Sensors*. 2022. 22(23): 9084.

[20] Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *Proceedings of the 8th International Conference on Learning Representations (ICLR 2019) New Orleans, Louisiana*. 2019, 1-13.

[21] Wang, F., Wang, H., Zhou, X. and Fu, R. A driving fatigue feature detection method based on multifractal theory. *IEEE Sensors J*. 2022. 22(19): 19046–19059.

[22] Bernstein, J., Vahdat, A., Yue, Y. and Liu, M. Y. On the distance between two neural networks and the stability of learning. Advances in Neural Information Processing Systems. 2020. 33: 21370-21381.

[23] Keskar, N. S. and Socher, R. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*. 2017.

[24] Arnold, M., Fink, S. J., Grove, D., Hind, M. and Sweeney, P. F. A survey of adaptive optimization in virtual machines. *Proceedings of the IEEE*. 2005. 93(2): 449-466.

[25] Gulcehre, C., Moczulski, M. and Bengio, Y. Adasecant: robust adaptive secant method for stochastic gradient. *Proceedings of the International Joint Conference on Neural Networks (IJCNN) Anchorage, Alaska*. 2017, 1-13.

[26] Yao, Y., Zhao, Y. and Li, Y. A volatility model based on adaptive expectations: An improvement on the rational expectations model. *International Review of Financial Analysis*. 2022. 82, 102202.

[27] Hao, X., Ma, Y. and Pan, D. Geopolitical risk and the predictability of spillovers between exchange, commodity and stock markets. *Journal of Multinational Financial Management*. 2024. 73, 100843.

[28] Zhang, F., Zhang, Y., Xu, Y. and Chen, Y. Dynamic relationship between volume and volatility in the Chinese stock market: evidence from the MS-VAR model. *Data Science and Management*. 2024. 7(1), 17-24.

[29] Wilder, J. W. New concepts in technical trading systems. *Greensboro, NC: Trend Research*. 1978.

[30] Atkins, A., Niranjan, M. and Gerding, E. Financial news predicts stock market volatility better than close price. *The Journal of Finance and Data Science*. 2018. 4(2): 120-137.

[31] Mienye, I. D., Swart, T. G. and Obaido, G. Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. *Information*. 2024. 15(9): 1-34.

[32] Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*. 1997. 9(8): 1735-1780.

[33] Tieleman, T. and Hinton, G. Divide the Gradient by a Running Average of Its Recent Magnitude. *Coursera: Neural networks for machine learning*. 2012.

[34] Shakawi, A. M. H. A. and Shabri, A. Improving Prediction of Bursa Malaysia Stock Index Using Time Series and Deep Learning Hybrid Model. In: Saeed, F., Mohammed, F., Fazea, Y. (eds) *Advances in Intelligent Computing Techniques and Applications. Lecture Notes on Data Engineering and Communications Technologies* 2024. 210. Springer, Cham.