IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 December 2023, accepted 13 January 2024, date of publication 22 January 2024, date of current version 30 January 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3356553*

RESEARCH ARTICLE

# AutoCyclic: Deep Learning Optimizer for Time Series Data Prediction

**CHRISTIAN ARTHUR, NOVANTO YUDISTIRA, AND CANDRA DEWI**
Departemen Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya, Ketawanggede, Lowokwaru, Malang, East Java 65145, Indonesia
Corresponding author: Novanto Yudistira (yudistira@ub.ac.id)

This work was supported in part by Brawijaya University.

**ABSTRACT** Time series prediction poses a formidable challenge, marked by the inherent difficulty in capturing long-term dependencies and adapting to intricate data patterns. Existing methods, spanning statistical models and neural networks, often grapple with issues such as underfitting and overfitting. This study addresses these challenges by introducing Autocyclic Learning Rate (AutoCyclic), an innovative approach that seamlessly integrates cosine cyclic learning rates with considerations for autocorrelation and variance. AutoCyclic dynamically adjusts learning rates based on the characteristics of time series data, effectively mitigating challenges related to local minima and demonstrating robust adaptability to outliers. In evaluation across diverse datasets, including ETTm2, M4, and WindTurbine, AutoCyclic consistently outperforms traditional optimizers such as Adams Optimizer and Cosine Cyclic Learning Rate. The results underscore AutoCyclic's superior performance, showcasing its potential as a pivotal tool for enhancing predictive modeling in various time series forecasting scenarios. The groundbreaking nature of AutoCyclic lies in its ability to address the complexities of time series prediction, providing a valuable solution to the limitations faced by existing models. The study serves as a key contribution to the ongoing research in timeseries data prediction, with implications for improving the accuracy and efficiency of predictive models in diverse applications. For those interested in implementing AutoCyclic, the code is available at https://github.com/wtfish/AutoCyclic.

**INDEX TERMS** Autocorrelation, cosine cyclic learning rate, deep transformer, optimizer, time series, variance.

## I. INTRODUCTION

The problem of time series prediction is not a new challenge. Various approaches with statistical methods have been carried out using Seasonal AutoRegressive Integrated Moving Average (SARIMA) and Autoregressive Integrated Moving Average ARIMA to predict stocks [1], [2]. ARIMA models are used specifically for short-range predictions and have the disadvantage of being difficult to confirm the performance of long-term series. Especially for highly volatile data such as financial data and influenced by several factors such as government policies, global economic conditions, and other disasters. Research by [3] who studied forecasting with ARIMA managed to get MAPE accuracy of up to 38%. Another statistical model used for comparison is the Bayesian model, which is often pitted against the latest developments in Artificial Neural Networks (ANN). In [4], a Bayesian model was employed for prediction with an accuracy of up to 78%. However, these statistical approaches often produce inaccurate predictions due to issues such as under-fitting and over-fitting, making them unreliable for fluctuating data scenarios.

Given the limitations of statistical models, ANN has become a popular choice for forecasting [5]. The study by [6] utilized Elliot Wave Indicators and stand-alone technical analysis to predict the stock prices of five companies. The results obtained with ANN showed significantly higher accuracy compared to statistical models, reaching 93.83%. This suggests that with input and model tuning, performance is likely to be further improved. Various architectures and tuning techniques have been developed, such as Convolutional Neural Networks (CNN), Long Short-Term Memory

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Gaggero.

(LSTM), and Gated Recurrent Units (GRU), as discussed by [7], [8], and [9]. In [9],a novel model combining CNN + SVM was proposed for predicting global stock indices. Although the researchers claimed to maximize the capabilities of the model through historical down-sampling of input data, specific prediction results were not disclosed.

The LSTM and GRU models presented by [7] for forecasting the stock price of CNPC showed that LSTM outperformed GRU in terms of MSE, RMSE, and MAE. Despite the superiority of LSTM, the researchers revealed that GRU had a faster prediction time due to fewer internal ''gating'' mechanisms and parameters. Another development of LSTM is the Bidirectional LSTM (Bi-LSTM), a sequence processing model consisting of two LSTMs designed to process input in both forward and backward directions. The strength of Bi-LSTM lies in its ability to better understand the context from the past and the future, making its predictions more relevant [10], [11]. Bi-LSTM has been successfully applied to the forecasting of stock prices, as evidenced by [12], and achieved the best performance with an R-squared value of 95.93%.

In the evolution of predictive models such as LSTM, CNN, and GRU, a new approach has emerged that combines Transformer technology with attention mechanisms. Initially introduced by Vaswani et al [13]. in 2017 as a revolutionary model for natural language processing, Transformer has been widely applied in various fields, including time series analysis and prediction. The main advantage of Transformer is its ability, especially in NLP, to understand long-range relationships between words in a text. In the context of time series data, this can be interpreted as an understanding of long-term dependencies, with claimed faster training times due to parallel execution. The core innovation of the Transformer architecture lies in its attention mechanism, enabling selective weighting of information across the entire input sequence. Recent empirical research, as documented in publications [14], [15], [16], demonstrates the superior performance of Deep Transformer (DT) models with attention mechanisms in time series prediction tasks compared to traditional RNN architectures like LSTMs, CNNs, and GRUs.

Determining a learning rate is certainly an important part of tuning a deep learning model. With the wrong learning rate setting, a model will not get its local minimum. A high lr value will cause the model to ''jump over'' from global minima, while an lr value that is too small will make the model stuck in local minima [17]. Research by [18] revealed that the use of a decay learning rate can improve the performance and convergence speed of various models such as VGG and RESNET for CIFAR-10 photo classification.

Therefore, a precise combination is needed to prevent time series prediction models from getting trapped in local minima. For this reason, the significance of Deep Neural Networks (DNN) with Cyclic Learning Rate (CLR) is widely recognized [19]. Additionally, there is an advancement beyond the standard CLR, namely, the innovative maximum-minimum cosine approach, which is designed to determine an acceptable range for the maximum and minimum learning rates used during the training process. By implementing CLR, the learning rate during model training can be dynamically increased throughout the training cycle. This results in a significant acceleration of convergence as the model rapidly explores relevant parameter regions. While this approach appears quite powerful, it is important to note that variance also influences the performance of the optimizer.

In their study, Ruder [20] revealed that in Stochastic Gradient Descent (SGD), high data variance leads to a highly fluctuating objective function. In ''batch gradient descent'' optimization methods, the convergence process to the minimum value of parameters occurs within a basin. Conversely, in SGD, the fluctuations allow jumpings to new potential local minima, which may be better. On the other hand, this makes the convergence process to the exact minimum value more complex because SGD tends to overshoot the actual minimum value. In other words, while SGD fluctuations can aid in finding better local minima, they can also make it challenging to reach the exact minimum value.

### A. CONTRIBUTION

This research focuses on developing the previously static CLR based on statistical calculations by relating it to the nature of the data to find the global minimum. Autocorrelation cyclic learning rate (AutoCyclic) is a solution to solve this problem, the learning rate will be dynamically adjusted by taking into account the autocorrelation value and variance of a batch of data in the training phase. AutoCyclic adapts to various data patterns which will increase and decrease according to the variance value, thus overcoming the limitations of the usual CLR method.

In the performance evaluation testing of the AutoCyclic method, there are 3 data with different characteristics that will be tested to determine whether autocCyclic can work with complex data patterns. The first data is ETTm2 data which has a seasonal data pattern, WindTurbine which has a high overall variance value, and M4 competition data which has a complex data pattern with a lot of noise. These differences will be tested to measure the capabilities of AutoCyclic using MAE and MAPE metrics. The tests conducted are shown to demonstrate AutoCyclic's ability to deal with various forecasting objectives in the real world.

## II. RELATED WORK

### A. CYCLIC LEARNING RATE

Research by Smith became a pioneer in the initial concept of CLR [18]. In his method, there are several bounds hyperparameters for creating waves that must be set, namely *base_lr*, *max_lr*, and *step_size*. *base_lr* will be obtained from the learning rate range test, then the determination of *max_lr* and *step_size* will be determined according to

the experimental results. Initially, CLR was used to classify images with CIFAR-10 and CIFAR-100 datasets

Expanding upon CLR, Loshchilov et al. proposed a modified approach incorporating cosine-based warm restarts [21]. This method emphasizes partial warm restarts and enhances the anytime performance of stochastic gradient descent, particularly in deep learning tasks. Empirical evidence on datasets like CIFAR-10 and CIFAR-100 underscores the effectiveness of warm restarts in mitigating poor function landscapes and highlights the evolutionary nature of learning rate optimization.

Building upon CLR, Wen et al. proposed the max-min cosine cyclic learning rate scheduler (MMCCLR) for automatic learning rate range identification in new datasets [22]. On datasets related to defect diagnosis, the suggested snapshot ensemble convolutional neural network (SECNN) employing MMCCLR performs better. This development validates the possibility of autonomous learning rate adaptation in group education.

Obaidat investigated cyclic learning rate optimization and suggested the trapezoidal cycle pattern as the optimal configuration [19]. With fewer iterations and faster convergence, this method replaces a fixed learning rate with a triangularly changing rate during each training cycle. Superior stability and accuracy were found in experiments conducted on VGG and RESNET networks for the CIFAR-10 and CIFAR-100 datasets as compared to standard CLR. This adaptability in deep neural network training is improved by this flexibility within reasonable bounds.

The dropCyclic learning rate schedule was a further development in 2022 [23]. This approach includes a learning rate decrease at every epoch with the goal of navigating to a new local minimum in the next cycle. Evaluations across diverse datasets and comparisons with the proposed method demonstrated the improvement classification accuracy of dropCyclic.

### B. ADAM OPTIMIZER

Bianchi et al. found that the Adam optimizer was the most effective method for training parameters on RNN, LSTM, and GRU [24]. This paper emphasizes the importance of the L2 norm of weights as an optimal regularization approach, shedding insight on the efficacy of optimization strategies.

Further study has explored how optimization techniques impact the accuracy of models in image classification tasks [25], [26], with an emphasis on the application of various optimizers such as SGD and Adam. Test results employing the Adam optimizer showed improved accuracy at the training, test, and validation stages on the COVIDx CT, CIFAR-10, CIFAR-100, and skin cancer datasets.

This study continues the previous investigation of the influence of optimization methods on model accuracy in image classification, with a focus on optimizers such as SGD and Adam. Notably, research like the one given in [27] demonstrates the adaptability of the Adam optimizer. In this paper, the writer presents a unique hybrid model, WT-Adam-LSTM, for accurate power price forecasting, exhibiting Adam's versatility beyond picture classification applications. Adam's success in forecasting is further demonstrated in studies such as [28], where it outperformed various baseline models, and in the research on ''The Forecast of Coal Price Based on LSTM with Improved Adam Optimizer'' [29], which highlights the practical uses of Adam optimizer in refining models. These examples demonstrate the Adam optimizer's adaptability and efficiency in improving accuracy across a variety of forecasting applications and datasets, extending its usefulness beyond picture categorization scenarios. Given its versatility, it is necessary to evaluate and compare its performance in terms of cycle learning rates. This comparison analysis will help us gain a better knowledge of the Adam optimizer's performance in diverse optimization settings, particularly in the dynamic context of cyclic learning rates, hence expanding our grasp of its capabilities and optimal utilization.

### C. VARIANCE IN OPTIMIZER

The significance of variance in optimization techniques was emphasized by Ruder [20], which prompted the creation of new optimizers including RMSprop, Adamax, and Adams. This work highlights how variation in stochastic gradient descent facilitates a more thorough search of local minimum points, offering a more nuanced perspective of the advantages and disadvantages of variance in optimization techniques.

Exploring the impact of variance in optimization techniques, highlighted by Ruder [20], is crucial to enhancing the performance of optimizers. Ruder emphasized the importance of variance in the creation of advanced optimizers like RMSprop, Adamax, and Adams [30]. This emphasizes the relevance of variance in stochastic gradient descent by providing a deeper knowledge of its benefits and trade-offs, ultimately leading to more effective discovery of local minima.

The direct application of Stochastic Variance Reduction (SVR) techniques to deep learning optimization has encountered difficulties due to the recent challenges detailed in the study by [31]. While this study did not reject the possibility of variance in the optimization process, it did emphasize the need for additional research. This involves looking into adaptive SVR applications, meta-level tweaks to learning rates, scaling matrices, and possible hybridizations with other optimization approaches like Adagrad and ADAM.

In addition, a study on double descent in deep neural network (DNN) test error revealed a significant correlation between variance and test error. This understanding inspired the creation of a new metric, Optimization Variance (OV), which measures the diversity of model updates induced by stochastic gradients. Surprisingly, OV acquired only from the training set has a significant relationship with the test error, implying that variance is important in predicting a DNN's generalization ability. This finding provides new possibilities

for early halting without the need for a specific validation set and emphasizes the role of variance in optimizing the training process [32]. In summary, recent research stresses the importance of variance in increasing the functionality and efficiency of optimization algorithms, particularly in the dynamic terrain of deep learning.

### D. AUTOCORRELATION

The relevance of autocorrelation in refining forecasting models provides an opportunity to integrate these insights with cutting-edge DL optimization techniques, particularly cyclic learning rates. The synergies between autocorrelation-guided model construction and advanced DL optimizers suggest a promising avenue for future research, offering the potential for more efficient and accurate deep learning-based forecasting systems.

In a study by [33], the crucial role of Autocorrelation Functions (ACF) and Partial Autocorrelation Functions (PACF) in optimizing Artificial Neural Network (ANN) models is highlighted. The research proposes enhancements to DL optimization paradigms, specifically cyclic learning rate (CLR), by incorporating insights derived from autocorrelation functions. The utilization of more compact ANN structures, guided by autocorrelation functions, is recommended as a method to improve DL optimization without compromising accuracy.

Similarly, in agriculture [34], the application of autocorrelation functions aligns with the goal of refining DL optimization for crop yield forecasting. Autocorrelation assists in selecting optimal macro parameters for ANN models, illustrating a correlation between autocorrelation-informed model construction and advanced DL optimization techniques, such as cyclic learning rates. This approach aims to leverage autocorrelation-derived insights to fine-tune DL optimizers, thereby enhancing the overall performance of agricultural production forecasting models.

In the field of natural gas load forecasting [35], where autocorrelation functions play a crucial role, the superior performance of prediction models opens avenues for combining these insights with cutting-edge DL optimization strategies. Autocorrelation's effectiveness in improving forecasting accuracy aligns with the objectives of DL optimizers, like CLR, which aim to determine optimal learning rates for neural network training. The outcomes of this study could inspire further exploration of seamlessly integrating autocorrelation-informed insights into DL optimization frameworks, fostering the development of more robust and accurate deep learning-based forecasting systems.

Additionally, recent innovations in long-term series forecasting, such as the Autoformer model proposed by Wu et al. [36], introduce a novel decomposition architecture incorporating an Auto-Correlation mechanism. This addresses challenges in handling complex temporal patterns and information utilization bottlenecks faced by traditional Transformer models. Autoformer outperforms previous Transformer-based models in efficiency and accuracy, showcasing remarkable performance across diverse practical applications. These advancements underscore the potential of autocorrelation in shaping the future landscape of deep learning-based forecasting methodologies.

## III. PROPOSED AUTOCYCLIC OPTIMIZER FOR TIME SERIES DATA PREDICTION

In the development of learning rate optimization for neural networks in this study, the proposed model was the AutoCyclic Optimizer For Time Series Data Prediction. Unlike previous approaches, this model automatically determines the learning rate using the variance obtained from autocorrelation. Smith [18] introduced the concept of CLR and inspired numerous subsequent studies. However, this study took an innovative step by leveraging autocorrelation-informed insights to create a more sophisticated adaptive approach.

Previous studies, including Wen et al. [22] and Loshchilov et al. [21], examined the use of cyclic learning rate approaches to improve model performance. However, in this latest advancement, the Autocyclic Optimizer For Time Series Data Prediction uses autocorrelation and variance to automatically adjust the variance settings and increase the learning rate.

### A. LEARNING RATE RANGE TEST

The LR range test, an important deep learning approach, involves a systematic investigation of learning rate values over a brief training session to determine appropriate lower and upper limits. Beginning with a low learning rate, the procedure entails gradual changes until a point of performance plateau is achieved, with the upper bound carefully determined right before deterioration starts. In our study, we improved this process by repeatedly refining the learning rate within the specified range, aiming for an ideal value that is perfectly aligned with the complexities of our neural network design and dataset features. This thorough fine-tuning guarantees a highly efficient and personalized optimization process, which improves our model's resilience to the unique difficulties provided by training dynamics.

Figure 1 shows an illustrative example of the LR range test (LRRT) based on Smith's methodology [18]. Examining Figure 1, it becomes apparent that the model initiates convergence promptly. Consequently, it is justified to set the base learning rate ( $n_{\text{init}}$) at 0.001. Additionally, observations have shown that beyond a learning rate of 0.006, the accuracy improvement becomes erratic and eventually experiences a decline. Therefore, it is rational to set the maximum learning rate ($n_{\max}$) as 0.006. These insights derived from LRRT play a crucial role in setting appropriate boundary values for effective training of the neural network.

Unlike conventional LR range tests, Figure 2 shows our approach involves a meticulous examination of the learning rate over a comprehensive spectrum, ranging from $10^{-6}$ to $10^{-1}$. This nuanced evaluation entails running the model for 100 epochs, allowing the learning rate to increment linearly

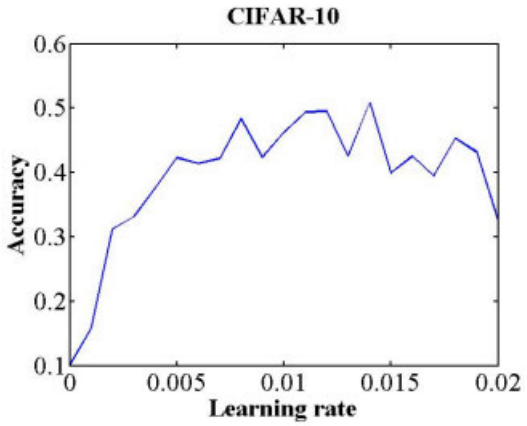


**FIGURE 1.** Learning range test with CIFAR-10 dataset.

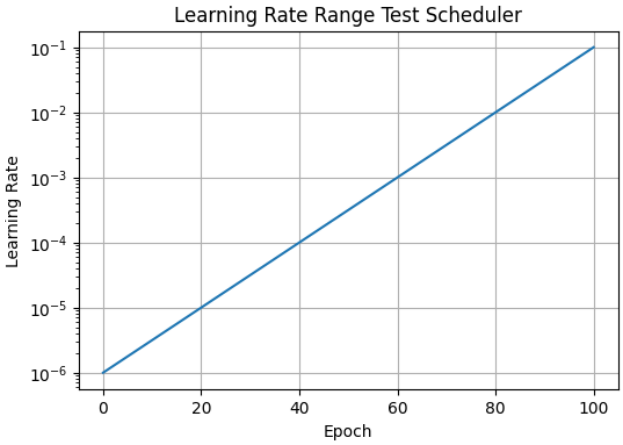


**FIGURE 2.** Learning range test scheaduler.

within the specified range. This tailored LR range test proves to be exceptionally beneficial, particularly when confronted with the challenges posed by different datasets.

### B. COSINE CYCLIC LEARNING RATE SCHEDULE

Cyclic cosine annealing (CCA) is the primary learning strategy in the snapshot ensemble method used during model training. CCA enables a faster reduction in the learning rate process involves a 100-epoch curve with five cycles denoted as $M_1, M_2, \ldots M_5$, representing distinct models for each local minimum, as illustrated in Figure 1. The computation of the CCA follows Equation 1.

$$n = \frac{n_{\text{init}}}{2}(\cos\frac{\pi \mod \left(t-1, \frac{T}{M}\right)}{[T/M]}) + 1) \tag{1}$$

In this context, $n$ represents the learning rate at the current iteration, $n_{\text{init}}$ denotes the initial learning rate, $t$ signifies the current iteration count, $T$ is the total number of iterations, and $M$ is the total number of cycles. In the framework of Cyclic Cosine Annealing (CCA), only the initial learning rate ($n_{\text{init}}$) needs to be adjusted, while the other hyperparameters remain constant. Consequently, inappropriate adjustments of the learning rate can lead to non-convergence of the training process, resulting in varying local minima after each cycle.

### C. VARIANCE

Variance is a statistical concept that has an important role in analyzing and describing the distribution or variation of data in a set of observations. This concept provides a quantitative way to measure how far or how close the data points in a data set are to their mean value [36]. Here is the formula for variance in Equation 2.

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n} \tag{2}$$

Variance, represented by $var(X)$, measures the spread of values in the input vector X from its mean ($\bar{X}$). In this formula, $\bar{X}$ is the average of the input vector, and $X_{\text{i}}$ is the value at index position $i$ in the vector X. The variance provides information about the variation of values in the vector. The higher the variance, the greater the spread; conversely, the lower the variance, the closer the values are to the mean. The variance is an important statistical metric in the analysis of data distribution characteristics [37].

### D. SIGMOID ACTIVATION FUNCTION

The sigmoid activation function is used not only as a key element in artificial neural networks, but also as a tool to reduce the range of data values and perform normalization. Normalization is an important process in data processing to ensure that each feature or variable has a uniform range of values, which facilitates model training and improves its performance [5]. The following Equation 3 below is a formula for the sigmoid activation function.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

The sigmoid function, denoted by $sigmoid(x)$, is a mathematical function that converts the input variable $x$ to a range of values between 0 and 1. In this formula, $e$ represents Euler's Number ($\approx 2.71828$). The sigmoid function is widely used in mathematical modeling, especially in the context of neural networks and statistics. This function produces an output with an S-shaped curve that maps input values to the corresponding probability intervals or activation scores. The sigmoid function is valuable for normalization or activation in machine learning models.

### E. AUTOCORRELATION

Autocorrelation is a crucial statistical technique employed in time series analysis to elucidate the degree of correlation existing between values within a time series at specific time intervals [38]. This method delves into the intricate patterns of interrelationships among observations at various time points, thereby facilitating the uncovering of latent structures and trends concealed within the temporal data. Autocorrelation, defined by Equation 4, quantifies the similarity between values at different periods in time, significantly contributing

to the analysis and interpretation of temporal patterns within a dataset. This statistical method is notably useful for revealing hidden dependencies and cyclic patterns that would otherwise go unnoticed, giving analysts and researchers with a full understanding of the dynamics inherent in time series data.

$$autocorr(k) = \frac{\sum_{t=k+1}^{n}(Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2} \quad (4)$$

A measure of the degree of correlation between a series value at a particular time, $Y_t$, and its value at a prior time, $Y_{(t\text{-}k)}$, is autocorrelation, denoted by $autocorr(k)$ at lag $k$. $n$ is the total number of data in the series, and $\bar{Y}$ is the average of the whole time series in this formula. To assess how much the values are connected, the number of lags, denoted by $k$, establishes the separation between two observed time observations. In addition to offering a summary of the patterns of interactions between observations in a time series, this autocorrelation formula can shed light on the internal organization and temporal dependencies of time series data.

### F. PROPOSED AUTOCYCLIC COSINE CYCLIC LEARNING RATE SCHEDULE

This paper presents the AutoCyclic Optimizer for Time Series Data Prediction, a novel approach that uses variance from autocorrelation to automatically calculate the learning rate in the quickly developing field of neural network optimization. Unlike traditional methods, our work incorporates knowledge from autocorrelation into an advanced adaptive strategy, motivated by Smith's introduction of CLR [18]. Although previous research has looked into ways to improve model performance using cyclic learning rates, our work stands out because it integrates autocorrelation considerations in a unique way. The next part offers a detailed pseudocode illustration, dubbed Algorithm 1, that clarifies the complex functions of the AutoCyclic Optimizer.

**Algorithm 1** AutoCyclicLR(base_lr, max_lr, step_size, batch_data)

**procedure** AutoCyclicLR(base_lr, max_lr, step_size, batch_data)
  $vars \leftarrow []$
  **for** *items* in *batch_data* **do**
    $autocorrs \leftarrow$ autocorr(*items*)
    $output \leftarrow$ sigmoid(nan_to_num($autocorrs$), $nan = 0$))
    $step\_var$.append(variance($output$))
  **end for**
  $batch\_variance \leftarrow$ mean($step\_var$)
  $cycle \leftarrow$ floor$(1 + \frac{\text{last_epoch}}{2 \times \text{step_size}})$
  $x \leftarrow$ abs$(\frac{\text{last_epoch}}{\text{step_size}} - 2 \times cycle + 1)$
  $lr \leftarrow base\_lr + (max\_lr - base\_lr) \times \frac{1+\cos(\pi \times x)}{2} \times 1 + batch\_variance$
  **return** $lr$
**end procedure**

Specifically designed for neural network training, the *AutoCyclicLR* algorithm provides a dynamic method for adjusting the learning rate. It starts by making a blank list called vars in order to store variance values. Autocorrelation is calculated, the output is subjected to sigmoid processing, and the variance is inserted into the vars list after each iteration over batches of data (*batch_data*). Next, we compute *batch_variance*, which is the mean of these variances. By calculating the location inside the cycle, the method ascertains the current cycle by using the *step_size* and *last_epoch* hyperparameters. After that, a cosine annealing schedule that takes into account the *batch_variance* is used to modify the learning rate ($lr$). As the procedure's output, the ultimate learning rate is then given back. This innovative approach intricately employs autocorrelation insights to dynamically modulate the learning rate throughout the training process, contributing to more nuanced and effective optimization of neural networks.

## IV. TIMESERIES DATASETS

### A. ETTM2 DATASET

The ETTm2 dataset, compiled by Beijing Guowang Fuda Science & Technology Development Company [39], covers two years, from July 1, 2016, to June 26, 2018, and captures the electricity distribution dynamics at 15-minute intervals. With 69,680 data points, it includes parameters such as High Useful Load, High Useless Load, Middle Useful Load, Middle Useless Load, Low Useful Load, Low Useless Load, and Oil Temperature. Reflecting electricity usage patterns influenced by weekdays, holidays, seasons, and temperature, the complexity of the dataset is highlighted by seasonal patterns and a substantial variance value of 141,288. This emphasizes the necessity for advanced modeling and optimization techniques in the electricity distribution domain. Figure 3 provides a visual representation of the data series.

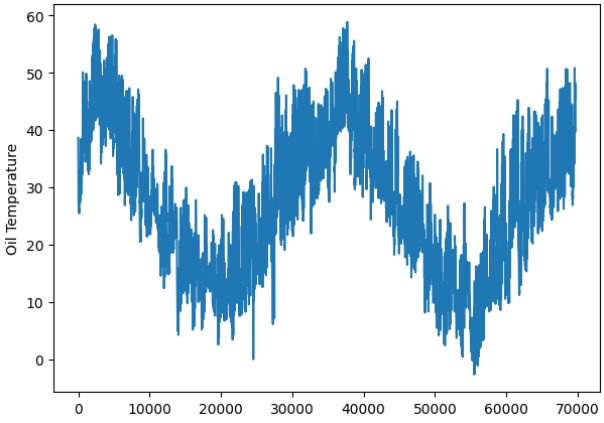


**FIGURE 3.** Oil temperature feature series visualization.

### B. WIND TURBINE DATASET

The following data from a wind turbine contains various data about the turbine speed, rotor, and the energy generated as

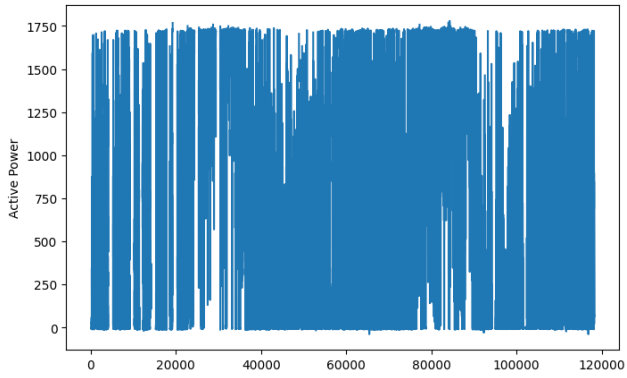


**FIGURE 4.** ActivePower feature series visualization.

Active Power. The data was recorded from January 2018 to March 2020. The data was recorded at intervals of every 10 minutes, so there are 118,224 data points. It is known that the date has a fairly high variance of 373,653.637. The following Figure 4 visualizes the Active Power series.

### C. M4 DATASET

The M4 competition is a pivotal challenge in time series Forecasting and serves as a benchmark for evaluating various models in real-world scenarios. Understanding its development and findings is crucial for navigating contemporary forecasting techniques. The M4 dataset, known for its noisy nature and high ambient variance of 48,236,882.24078, presents a significant challenge. Figure 5 visually captures the complexity of the M4 data series.

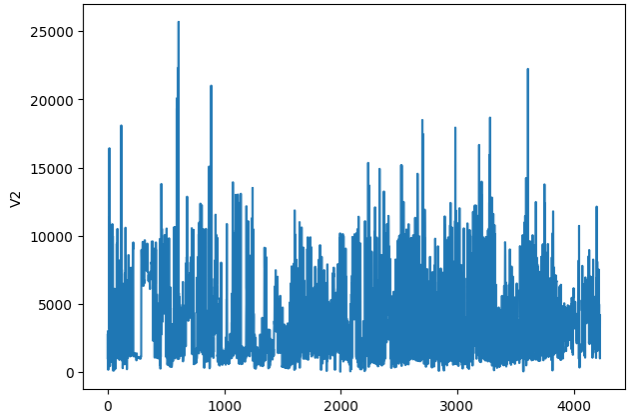


**FIGURE 5.** V2 feature series visualization.

## V. EXPERIMENTAL RESULTS AND DISCUSSION

We have demonstrated the effectiveness of the proposed autocorrelation-based cyclic learning rate (AutoCyclic) and compared it with two learning rate methods, namely the Adams optimizer and cosine cyclic learning rate, on three time series datasets. For the backbone of the model to be used for prediction, Deep Transformer, we then compared two other architectures namely LSTM and RNN.

All experiments were trained and evaluated under Windows using an Intel(R) Core-i7-10750H CPU @ 2.60GHz, 16GB RAM, and a GeForce GTX 1650 GPU with 4GB of RAM. We implemented all models and AutoCyclic methods based on the Torch deep learning framework and the Pytorch Forecasting library.

### A. EVALUATION METRICS

In the context of this research, two essential evaluation metrics, Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE), were employed to assess the performance of the forecasting models. MAPE, calculated as the average percentage difference between the predicted ($\bar{y}_t$) and observed ($y_t$) values over a given period $T$, is expressed by the Equation 5.

$$\text{MAPE} = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \times 100 \tag{5}$$

This metric provides a relative measure of accuracy, offering insights into the average percentage difference between predicted and actual values. Meanwhile, MAE, computed as the average absolute difference between predicted and observed values, is represented by the Equation 6.

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_t| \tag{6}$$

Furthermore, Root Mean Squared Error (RMSE) was employed as a loss function for model training. RMSE assesses the square root of the average of the squared differences between predicted and observed values over a given period $T$ is represented by Equation 7.

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2} \tag{7}$$

The use of MAPE and MAE is justified by their capacity to offer a comprehensive analysis of predicting accuracy that takes absolute and relative mistakes into account. MAE concentrates on the absolute size of mistakes, whereas MAPE offers information on the accuracy percentage-wise. As a loss function, RMSE makes sure that the total squared error is minimized during training, which promotes model convergence. The goal of this combination of assessment measures and loss function is to optimize the model for precise predictions in real-world settings and to thoroughly examine the forecasting performance.

### B. TRAIN SETTING

#### 1) DATA RATIO AND NUMBER OF EXPERIMENTS

In this research, we divided the dataset into three parts, namely, training and validation, with a ratio of 80%:20%, which was normalized within the range [-1,1]. The testing data comprised 120 data points. To standardize the Cyclic Learning Rate pattern, each data point underwent 50 steps during each training iteration, with batch sizes for M4,

ETTm2, and WindTurbine set to 67, 1114, and 1510, respectively. Additionally, we subsequently trained the model on the training and validation sets using the optimal configuration and then evaluated its performance on the test set. This two-phase approach ensures a comprehensive assessment of the model's generalization capabilities across different datasets.

#### 2) BACKBONE ARCHITECTURES

In this study, we used three different deep transformer model architectures, each tailored to optimize its respective dataset, along with the determination of optimal AutoCyclic hyperparameters. Since each dataset exhibits unique characteristics, we utilized three different models to account for these variations. In the "TESTING ON A DIFFERENT MODEL" sub-section, a comparative analysis of models was performed using a single-layer LSTM/RNN with 16 neurons. Table 1 provides an overview of the hyperparameters of the Deep Transformer models utilized in the study.

**TABLE 1. Hyperparameters of the deep transformer for each dataset.**

| Hyperparameter | Dataset | | |
|---|---|---|---|
| | M4 | ETTm2 | WindTurbine |
| Dmodel | 32 | 64 | 256 |
| Encoder | 2 | 1 | 2 |
| Decoder | 4 | 1 | 4 |
| FFN | 30 | 100 | 30 |
| PRE-LAYER | 30 | 100 | 30 |
| POST-LAYER | 30 | 100 | 30 |
| TIMELAG | 4 | 4 | 21 |

### C. AUTOCYCLIC HYPERPARAMETERS

#### 1) M4 DATASET

The model specified in Table 1 for the M4 dataset underwent a thorough evaluation using the AutoCyclic learning rate approach to determine the optimal hyperparameters. The initial phase involved the Learning Rate Range Test shown in Figure 2, which identifies the base_lr or $n_{\text{init}}$ as 0.00019179102616724887. Attention then shifted to two critical hyperparameters, max_lr or $n_{\text{max}}$ and step_size, which are key to influencing model performance. Systematic testing explored various combinations to identify configurations that yielded the most favorable results. This rigorous evaluation ensures the model's adaptability to the distinctive characteristics of the M4 dataset, improving overall performance.

Table 2 presents the outcomes of the experiments where two hyperparameters max_lr or $n_{\text{max}}$ and step_size, were explored through various combinations. The determination of the optimal values, guided by the criterion of the lowest Mean Absolute Percentage Error (MAPE), revealed that 8 is the optimal value for max_lr or $n_{\text{max}}$ and 15 for step_size. Similarly, if the lowest Mean Absolute Error (MAE) was prioritized, the optimal values were 7 for max_lr or $n_{\text{max}}$ and 15 for step_size. The selection of the best hyperparameters values was a balanced consideration of both the MAPE and MAE, resulting in the selection of 7 for max_lr or $n_{\text{max}}$ and 15 for step_size.

**TABLE 2. Test results for max_lr and step_size hyperparameters for the M4 dataset.**

| Max_lr multiplier | step_size | Loss | |
|---|---|---|---|
| | | MAPE | MAE |
| 3 | 10 | 72.27391 | 7532.672 |
| 3 | 25 | 66.39518 | 7531.819 |
| 3 | 50 | 62.29197 | 7522.836 |
| 5 | 10 | 64.57704 | 7573.069 |
| 6 | 15 | 56.51266 | 7504.31 |
| 7 | 10 | 56.58286 | 7511.828 |
| **7** | **15** | **55.71032** | **7495.095** |
| 7 | 25 | 65.14588 | 7525.491 |
| 7 | 50 | 57.43141 | 7510.586 |
| 8 | 15 | 55.21423 | 7520.991 |
| 8 | 25 | 62.98857 | 7532.86 |

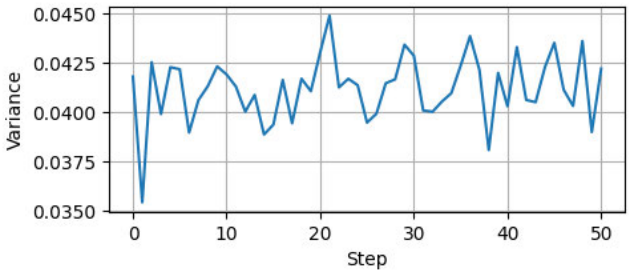


**FIGURE 6. Batch variance of M4 dataset for each step in one epoch.**

Figure 6 shows the calculated variance at each step. Following the successful implementation of autocyclic, the next step was to train with autocyclic, where the variance in the image was calculated according to Algorithm 1. This algorithm entails extracting the autocorrelation values for each batch, computing their variances, and subsequently taking the average. In Algorithm 1, the variance values presented in Figure 6 correspond to the values derived from line 8, specifically *batch_variance*.The calculated variances provide valuable insights into the fluctuation patterns during the training process, contributing to a comprehensive understanding of the autocyclic training dynamics. The visualization of these variance values aids in interpreting the impact of autocyclic training on the performance of the model.

The outcomes of the learning rate pattern, are presented in Figure 7. Aligned with the predetermined $n_{\text{init}}$, $n_{\text{max}}$, and step_size values, the pattern displays two distinct peaks. Despite resembling a typical wave, each step had unique values. For example, the first peak (step 14) had a value of 0.0013872499, while the second peak (step 44) had a value of 0.0013911905417252786. This alignment corresponds to Figure 6, where step 14 had a lower batch variance compared to step 44. This confirms that the learning rate adjusts according to its batch variance, highlighting the interplay between learning rate adjustments and variance dynamics. These insights contribute to a more comprehensive

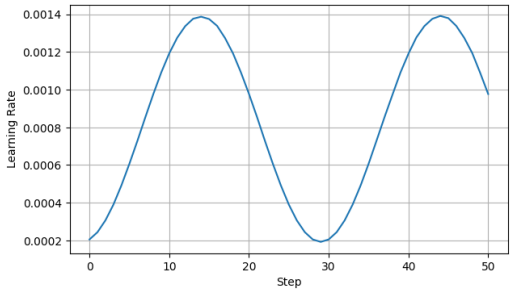


**FIGURE 7.** Learning rate pattern generated by autoCyclic on M4 data.

analysis of the AutoCyclic training process and improve our understanding of the intricate relationship between learning rate adjustments and training dynamics.

### 2) ETTM2 DATASET

The ETTm2 model in Table 1 underwent AutoCyclic learning rate testing, optimizing base_lr at 0.000689261 through the Learning Rate Range Test in Figure 2. Additionally, max_lr and step_size were tested to ensure refined AutoCyclic hyperparameters for optimal model performance in subsequent training phases.

**TABLE 3.** Test results for max_lr and step_size hyperparameters for the ETTm2 dataset.

| Max_lr multiplier | step_size | Loss | |
|---|---|---|---|
| | | MAPE | MAE |
| 3 | 10 | 0.73218 | 0.31897 |
| **3** | **25** | **0.3694** | **0.1637** |
| 3 | 50 | 0.40779 | 0.18103 |
| 5 | 10 | 0.44377 | 0.19828 |
| 6 | 15 | 0.6511 | 0.28448 |
| 7 | 10 | 0.61526 | 0.26724 |
| 7 | 15 | 0.57439 | 0.25862 |
| 7 | 25 | 0.38522 | 0.17241 |
| 7 | 50 | 0.4276 | 0.18966 |
| 8 | 15 | 0.69485 | 0.30172 |
| 8 | 25 | 0.38855 | 0.17241 |

The results of testing 2 hyperparameters, max_lr or $n_{\max}$ and step_size, with various combinations are presented in Table 3. Unlike M4, in this test, both max_lr and step_size exhibited consistently low values for both MAPE and MAE losses. Therefore, the values of 3 for the max_lr multiplier and 25 for step_size were chosen.

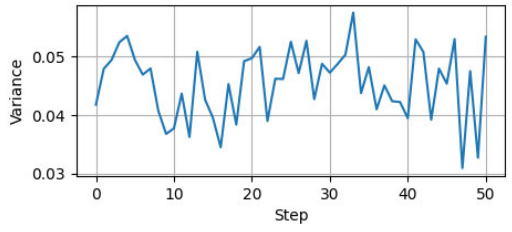


**FIGURE 8.** Batch variance of the ETTm2 dataset for each step in one epoch.

In Figure 8, the visual representation of variance calculation results at each training step is depicted. Following the autocyclic process, training involved the extraction of autocorrelation values, which are crucial for measuring the correlation between the current batch and preceding batches. The subsequent computation of batch variances using learning rate and batch variance highlights how the learning rate adapts to and influences the temporal dependencies within each batch. This nuanced approach, which considers both autocorrelation and batch variance, adds sophistication to the training dynamics, contributes to a more comprehensive evaluation of the adaptation of the learning rate to temporal characteristics, and emphasizes their dual importance in optimizing the training process for improved model performance.

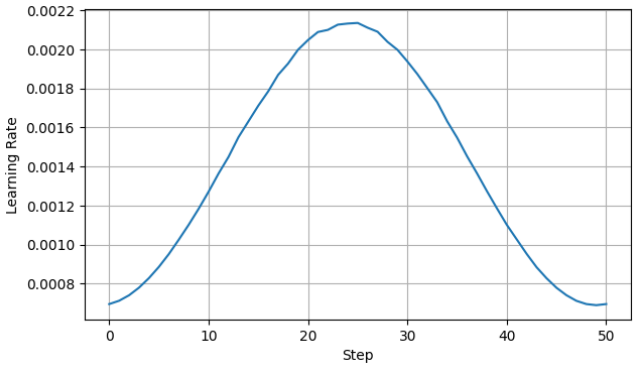


**FIGURE 9.** Learning rate pattern generated by autoCyclic on ETTm2 data.

The results of the AutoCyclic pattern are shown in Figure 9, forming a wave with a single peak based on the selected $n_{\text{init}}$, $n_{\max}$, and step_size values. The impact of fluctuating variance due to autocorrelation was evident in steps 20-30. This was consistent with Figure 8, where fluctuations were observed during steps 20-30.

### 3) WIND TURBINE DATASET

Similar to the M4 and ETTm2 datasets, the Wind Turbine model in Table 4 was tested with AutoCyclic learning rate to determine the optimal AutoCyclic hyperparameters. First, the model was subjected to the Learning Rate Range Test depicted in Figure 2, which resulted in the identification of the optimal learning rate, also referred to as base_lr or $n_{\text{init}}$, which is 0.0002782559402207. Subsequently, two additional hyperparameters were tested: max_lr or $n_{\max}$ and step_size.

Table 4 displays the results of testing two hyperparameters, max_lr or $n_{\max}$ and step_size, with various combinations. Similar to ETTm2, in this testing, both max_lr and step_size showed equally low values for both loss metrics, MAPE and MAE. Therefore, the values of 7 were chosen for the max_lr multiplier and 25 for the step_size.

Figure 10 illustrates the variance generated by row 8 in Algorithm 1. The resulting variance appeared to fluctuate but had a very small range. This was due to the non-fluctuating data, which caused the learning rate values to be less curly and more smooth.

**TABLE 4.** Test results for max_lr and step_size hyperparameters for Wind Turbine dataset.

| Max_lr multiplier | step_size | Loss | |
|---|---|---|---|
| | | MAPE | MAE |
| 3 | 10 | 19.5225 | 124.7979 |
| 3 | 25 | 19.2849 | 124.404 |
| 3 | 50 | 19.448 | 124.5555 |
| 5 | 10 | 18.852 | 123.1616 |
| 5 | 15 | 18.5694 | 122.7474 |
| 6 | 15 | 18.1661 | 121.606 |
| 7 | 10 | 18.5553 | 121.6565 |
| 7 | 15 | 18.998 | 122.323 |
| **7** | **25** | **14.690** | **117.545** |
| 7 | 50 | 18.7225 | 121.4747 |
| 8 | 15 | 18.4317 | 121.3737 |

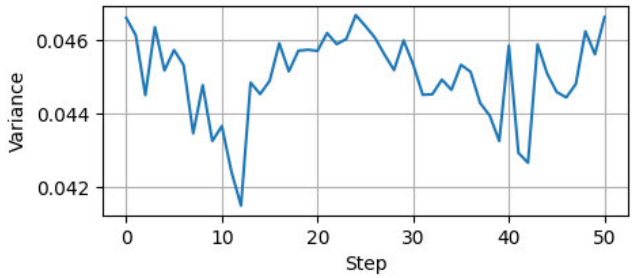


**FIGURE 10.** Batch variance of the wind turbine dataset for each step in one epoch.

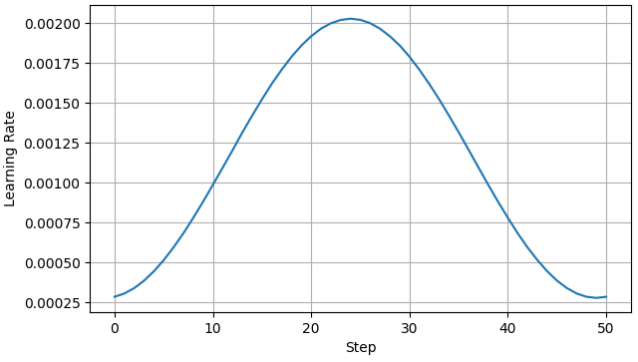


**FIGURE 11.** Learning rate pattern generated by autocyclic on wind turbine data.

The pattern of the learning rate, visualized in Figure 11 shows two troughs corresponding to the predetermined $n_{\text{init}}$, $n_{\text{max}}$, and step_size values. Despite the wave-like appearance, each step had a distinct value. For example, in the first trough, step 0 had a value of 0.000285145, while in the second trough, step 49 had a value of 0.00027825. This alignment is consistent with Figure 10, where step 14 had a batch variance of 0.04661, which was higher than step 49 with 0.04562.

### D. NUMBER OF INPUT FEATURES

Furthermore, the performance of the model was tested with varying counts of feature: 1 feature (univariate) and multivariate with 2 and 3 features. This study employs two datasets with multiple features: ETTm2 and Wind Turbine. The ETTm2 dataset comprises two variables, MULL and HULL, with correlations of 0.5 and 0.34, respectively, with the predicted feature (Oil Temperature). Meanwhile, in the Wind Turbine dataset, two features, WindSpeed and GeneratorWinding1Temperature, had strong correlations with Active Power, with values of 0.94 and 0.93, respectively.

**TABLE 5.** Testing the number of model features with ETTm2 data.

| Input Features | Loss | |
|---|---|---|
| | MAPE | MAE |
| **UNIVARIATE** | **0.3694** | **0.1637** |
| MULTIVARIATE (2) | 0.76216 | 0.33621 |
| MULTIVARIATE (3) | 0.45697 | 0.19828 |

Table 5 presents the outcomes of testing the Deep Transformer model with AutoCyclic over different numbers of input variables. For the multivariate setting with two features, namely oil temperature and MULL, the results showed significantly worse performance compared to all the loss values obtained in the univariate case. Similarly, when three features were included, the performance of the model, as evidenced by higher MAPE and MAE values, was inferior to the two-feature configuration, reinforcing the conclusion that the univariate approach focusing on the oil temperature feature yielded more favorable results.

**TABLE 6.** Testing the number of model features with Wind Turbine data.

| Input Features | Loss | |
|---|---|---|
| | MAPE | MAE |
| **UNIVARIATE** | **14.690** | **117.545** |
| MULTIVARIATE (2) | 16.50704 | 119.91919 |
| MULTIVARIATE (3) | 18.71872 | 122.46465 |

The results of testing the Deep Transformer with AutoCyclic on the wind turbine data are presented in Table 6. Similar to the ETTm2 data, models incorporating multiple features tended to degrade the overall model performance. The loss values of MAPE and MAE for multi-feature configurations proved to be worse than those for univariate settings, and increasing the number of features further worsened the model performance.

### E. TESTING ON A DIFFERENT MODEL

The comparison results in Table 7 illustrate the performance of different models, including Deep Transformer (DT), LSTM, and RNN, each paired with AutoCyclic, Cosine Cyclic Learning Rate (CLR), and Adam Optimizer. The results showed that AutoCyclic consistently achieved the lowest MAPE and MAE values across different scenarios. Cosine CLR performed well for DT and LSTM models but gave poorer results for RNN compared to Adam Optimizer. Adam Optimizer consistently ranked last in performance across all scenarios.

Figure 12 shows the prediction results of the M4 test data with 120 data points using three different models, DT, LSTM, and RNN, each employing the best-performing scheduler.

**TABLE 7.** Comparison of M4 data test results with other models.

| Model | Loss | |
|---|---|---|
| | MAPE | MAE |
| **DT + AutoCyclic** | **55.71** | **7495.09** |
| DT + Cosine CLR | 62.62547 | 7566.913 |
| DT + Adam Optimizer | 72.00092 | 7.628 |
| **LSTM + AutoCyclic** | **98.40144** | **8.250** |
| LSTM + Cosine CLR | 117.823 | 8.288 |
| LSTM + Adam Optimizer | 120.6024 | 8.182 |
| **RNN + AutoCyclic** | **116.1231** | **7.856** |
| RNN + Cosine CLR | 121.732 | 8.240 |
| RNN + Adam Optimizer | 103.9726 | 8.194 |

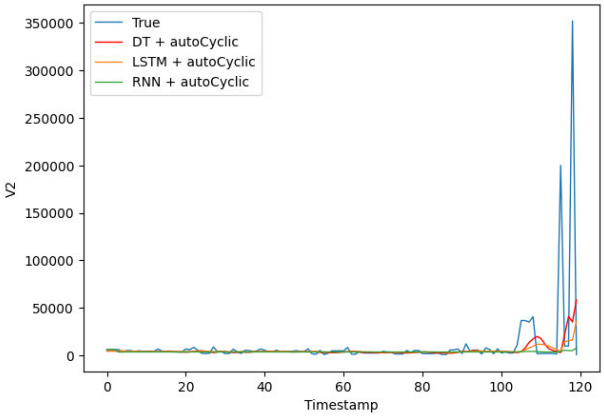


**FIGURE 12.** Test data prediction results with different models for M4 dataset.

It is evident that DT outperformed the other two models, approaching the original values, followed by LSTM and RNN.

Table 8 compares DT, LSTM, and RNN models with different schedulers, including AutoCyclic, Cosine Cyclic Learning Rate, and Adams Optimizer. AutoCyclic consistently achieved the lowest MAPE and MAE values in all scenarios. Although Cosine CLR worked well for DT and RNN, LSTM performed worse than Adams Optimizer, which consistently ranked last.

**TABLE 8.** Comparison of ETTm2 data test results with other models.

| Model | Loss | |
|---|---|---|
| | MAPE | MAE |
| **DT + AutoCyclic** | **0.3694** | **0.1637** |
| DT + Cosine CLR | 0.62342 | 0.27586 |
| DT + Adam Optimizer | 0.65649 | 0.28448 |
| **LSTM + AutoCyclic** | **0.636762** | **0.28333** |
| LSTM + Cosine CLR | 0.833927 | 0.375 |
| LSTM + Adam Optimizer | 0.76203 | 0.34166 |
| **RNN + AutoCyclic** | **0.57514** | **0.25833** |
| RNN + Cosine CLR | 0.62034 | 0.2666 |
| RNN + Adam Optimizer | 0.8123 | 0.3666 |

In Figure 13, the prediction results for the ETTm2 dataset are meticulously presented, showing the outcomes achieved with the optimal scheduler for each model. Unmistakably, the Deep Transformer model stood out once again, demonstrating superior performance compared to the other two

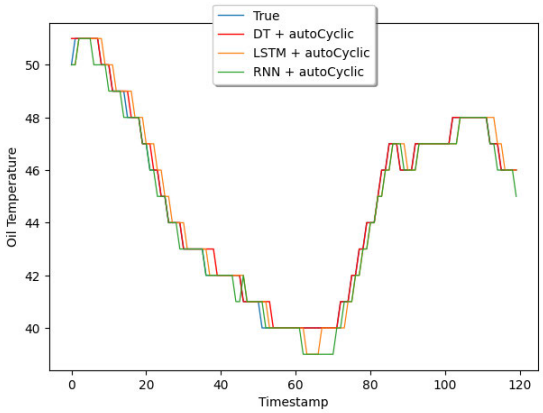


**FIGURE 13.** Test data prediction results with different models for ETTm2 dataset.

models, which exhibited less effective predictive capabilities. Notably, the alternative models exhibited mispredictions in the 60-70 timestamp range, resulting in elevated loss values. The graphical representation in Figure 13 provides a comprehensive visual insight into the comparative performance of these models on the ETTm2 dataset, with the Deep Transformer model notably excelling across various timestamps. This nuanced analysis affords a deeper understanding of the model's strengths and weaknesses, particularly in handling specific timestamp ranges, and underscores the significance of choosing an appropriate scheduler to enhance overall predictive accuracy.

Table 9 provides a detailed and comprehensive comparison of multiple models, with AutoCyclic consistently demonstrating superior performance over alternative optimizers across all models. A notable observation was the contrast in results compared to the findings in the two preceding datasets, where Cosine CLR consistently produced more favorable outcomes than Adam Optimizer across all models, confirming the relatively less effective performance of the latter. This nuanced exploration underscores the robustness of AutoCyclic and highlights the varying impact of optimization strategies across diverse datasets, providing valuable insights into the understanding of model behavior in different contexts.

**TABLE 9.** Comparison of Wind Turbine data test results with other models.

| Model | Loss | |
|---|---|---|
| | MAPE | MAE |
| **DT + AutoCyclic** | **14.69** | **117.545** |
| DT + Cosine CLR | 16.6725 | 120.01 |
| DT + Adam Optimizer | 22.13762 | 121.323 |
| **LSTM + AutoCyclic** | **41.2335** | **131.141** |
| LSTM + Cosine CLR | 52.42012 | 129.725 |
| LSTM + Adam Optimizer | 58.51616 | 130.292 |
| **RNN + AutoCyclic** | **36.4345** | **128.617** |
| RNN + Cosine CLR | 39.1811 | 130.208 |
| RNN + Adam Optimizer | 42.7806 | 130.449 |

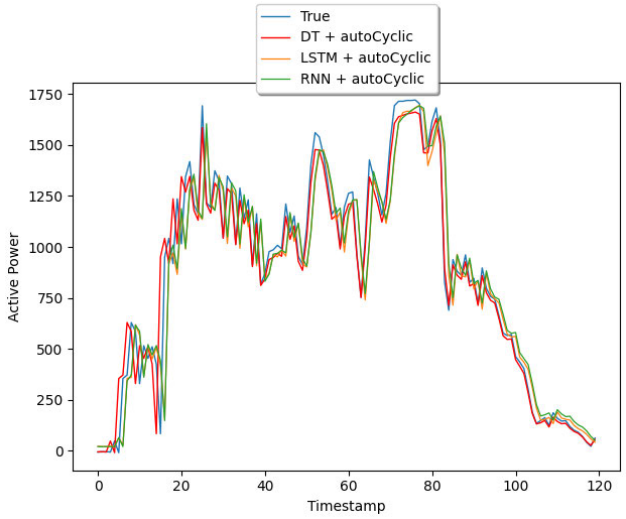


**FIGURE 14.** Test data prediction results with different models for wind turbine dataset.

Figure 14 shows the prediction results for the wind turbine dataset using the best Scheduler for each model. The three models, namely Deep Transformer (DT), LSTM, and RNN, showed prediction outcomes with remarkably close values, indicating a subtle variance in their performance. This marginal difference suggests that the models were closely aligned in their predictive capabilities when utilizing the identified optimal Scheduler configurations. The narrow spread in prediction results emphasizes the nuanced differences between the models and highlights the challenge of discerning significant variations in their performance on the wind turbine dataset.

## VI. CONCLUSION

In conclusion, this study highlights the exceptional proficiency of AutoCyclic to leverage autocorrelation to discern temporal patterns in data. Through a comprehensive comparison involving models such as Deep Transformer, LSTM, and RNN, alongside optimization methods such as AutoCyclic, Cosine Cyclic Learning Rate (CLR), and Adams Optimizer, AutoCyclic consistently outperforms its counterparts. The study, which spans diverse scenarios and datasets like M4, ETTm2, and wind turbine, underscores AutoCyclic's adaptability and reliability in addressing challenges associated with traditional optimization techniques. The quantitative results in Tables 7, 8, and 9 reveal autocyclic's consistently superior performance in achieving the lowest Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE) values, surpassing both Cosine CLR and Adams Optimizer. The visual representations in Figures 12, 13, and 14 reinforce autocyclic's predictive prowess across diverse datasets. This study contributes significantly to the advancement of time series forecasting methodologies by emphasizing the effectiveness and practical applicability of AutoCyclic in addressing real-world forecasting challenges, with a specific emphasis on leveraging autocorrelation.

Given the prevailing preference for transformer-based models in time series forecasting, this study strongly recommends exploring autocyclic within optimized transformer models such as informer [39], fedformer [40], or autoformer [36]. Such exploration aims to establish superior benchmarks, emphasizing the crucial role of autocorrelation in enhancing the effectiveness of autocyclic. This study paves the way for nuanced investigations of the interplay between autocorrelation and transformer-based forecasting, promising ongoing refinements in forecasting methodologies.

## ACKNOWLEDGMENT

The authors would like to thank the Artificial Intelligence Center, Universitas Brawijaya.

## REFERENCES

[1] S. Khanderwal and D. Mohanty, "Stock price prediction using ARIMA model," *Int. J. Marketing Hum. Resource Res.*, vol. 2, no. 2, pp. 98–107, 2021.

[2] D. S. Kumar, B. C. Thiruvarangan, A. Vishnu, A. S. Devi, and D. Kavitha, "Analysis and prediction of stock price using hybridization of SARIMA and XGBoost," in *Proc. Int. Conf. Commun., Comput. Internet Things (ICIoT)*, Mar. 2022, pp. 1–4.

[3] B. U. Devi, D. Sundar, and P. Alli, "An effective time series analysis for stock trend prediction using ARIMA model for nifty midcap-50," *Int. J. Data Mining Knowl. Manag. Process*, vol. 3, no. 1, pp. 65–78, Jan. 2013.

[4] L. S. Malagrino, N. T. Roman, and A. M. Monteiro, "Forecasting stock market index daily direction: A Bayesian network approach," *Exp. Syst. Appl.*, vol. 105, pp. 11–22, Sep. 2018.

[5] O. Bustos and A. Pomares-Quimbaya, "Stock market movement forecast: A systematic review," *Exp. Syst. Appl.*, vol. 156, Oct. 2020, Art. no. 113464.

[6] S. Lakshminarayanan, G. R. Weckman, A. Snow, and J. Marvel, "Stock market hybrid forecasting model using neural networks," in *Proc. IIE Annu. Conf.* May 2006, pp. 1–6.

[7] Y. Liu, "Stock prediction using LSTM and GRU," in *Proc. 6th Annu. Int. Conf. Data Sci. Bus. Analytics (ICDSBA)*, Oct. 2022, pp. 206–211.

[8] J. Kavinnilaa, E. Hemalatha, M. S. Jacob, and R. Dhanalakshmi, "Stock price prediction based on LSTM deep learning model," in *Proc. Int. Conf. Syst., Comput., Autom. Netw. (ICSCAN)*, Jul. 2021, pp. 1–4.

[9] J. Cao and J. Wang, "Stock price forecasting model based on modified convolution neural network and financial time series analysis," *Int. J. Commun. Syst.*, vol. 32, no. 12, p. e3987, Aug. 2019.

[10] S. Prakash, A. S. Jalal, and P. Pathak, "Forecasting COVID-19 pandemic using prophet, LSTM, hybrid GRU-LSTM, CNN-LSTM, bi-LSTM and stacked-LSTM for India," in *Proc. 6th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Mar. 2023, pp. 1–6.

[11] D. Park and D. Ryu, "Forecasting stock market dynamics using bidirectional long short-term memory," *J. Econ. Forecasting*, vol. 24, no. 2, pp. 22–34, Jun. 2021.

[12] Y. Loday, P. Apirukvorapinit, and P. Vejjanugraha, "Stock price prediction using modified bidirectional long short-term memory and deep learning models: A case study of Bhutan tourism corporation limited stock data," in *Proc. 8th Int. Conf. Bus. Ind. Res. (ICBIR)*, May 2023, pp. 645–650.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, 2017, pp. 1–11.

[14] B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *Int. J. Forecasting*, vol. 37, no. 4, pp. 1748–1764, Oct. 2021.

[15] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," 2020, *arXiv:2001.08317*.

[16] P. Lara-Benítez, L. G. Ledesma, M. Carranza-García, and J. M. Luna-Romera, "Evaluation of the transformer architecture for univariate time series forecasting," in *Proc. Conf. Spanish Assoc. Artif. Intell.*, Sep. 2021, pp. 106–115.
[17] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1310–1318.
[18] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 464–472.
[19] J. Li and X. Yang, "A cyclical learning rate method in deep learning training," in *Proc. Int. Conf. Comput., Inf. Telecommun. Syst. (CITS)*, Oct. 2020, pp. 1–5.
[20] S. Ruder, "An overview of gradient descent optimization algorithms," 2017, *arXiv:1609.04747*.
[21] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with restarts," 2016, *arXiv:1608.03983*.
[22] L. Wen, L. Gao, and X. Li, "A new snapshot ensemble convolutional neural network for fault diagnosis," *IEEE Access*, vol. 7, pp. 32037–32047, 2019.
[23] S. Noppitak and O. Surinta, "DropCyclic: Snapshot ensemble convolutional neural network based on a new learning rate schedule for land use classification," *IEEE Access*, vol. 10, pp. 60725–60737, 2022.
[24] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "An overview and comparative analysis of recurrent neural networks for short term load forecasting," 2017, *arXiv:1705.04378*.
[25] E. Hassan, M. Y. Shams, N. A. Hikal, and S. Elmougy, "The effect of choosing optimizer algorithms to improve computer vision tasks: A comparative study," *Multimedia Tools Appl.*, vol. 82, no. 11, pp. 16591–16633, Sep. 2022.
[26] R. Zaheer and H. Shaziya, "A study of the optimization algorithms in deep learning," in *Proc. 3rd Int. Conf. Inventive Syst. Control (ICISC)*, Jan. 2019, pp. 536–539.
[27] Z. Chang, Y. Zhang, and W. Chen, "Electricity price prediction based on hybrid model of Adam optimized LSTM neural network and wavelet transform," *Energy*, vol. 187, Nov. 2019, Art. no. 115804.
[28] Y. Ding, Y. Zhu, J. Feng, P. Zhang, and Z. Cheng, "Interpretable spatio-temporal attention LSTM model for flood forecasting," *Neurocomputing*, vol. 403, pp. 348–359, Aug. 2020.
[29] X. Liu, "Research on the forecast of coal price based on LSTM with improved Adam optimizer," *J. Phys., Conf.*, vol. 1941, no. 1, Jun. 2021, Art. no. 012069.
[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*.
[31] A. Defazio and L. Bottou, "On the ineffectiveness of variance reduced optimization for deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, 2019, pp. 1–11.
[32] X. Zhang, D. Wu, H. Xiong, and B. Dai, "Optimization variance: Exploring generalization properties of DNNs," 2021, *arXiv:2106.0171*.
[33] J. H. F. Flores, P. M. Engel, and R. C. Pinto, "Autocorrelation and partial autocorrelation functions to improve neural networks models on univariate time series forecasting," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2012, pp. 1–8.
[34] A. Rogachev and E. Melikhova, "Creating a neural network system for forecasting and managing agricultural production using autocorrelation functions of time series," in *Proc. ES Web Conf.*, vol. 164, 2020, p. 06005.
[35] X. Xiang, J. Shen, K. Yang, G. Zhang, J. Qian, and C. Zhu, "Daily natural gas load forecasting based on sequence autocorrelation," in *Proc. 37th Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Nov. 2022, pp. 1452–1459.
[36] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," 2021, *arXiv:2106.13008*.
[37] J. L. Fleiss, B. Levin, and M. C. Paik, *Statistical Methods for Rates and Proportions* (Wiley Series in Probability and Statistics). Hoboken, NJ, USA: Wiley, 2013.
[38] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications* (Springer Texts in Statistics). Berlin, Germany: Springer, 2005.
[39] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 12, hboxpp. 11106–11115.
[40] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 27268–27286.

**CHRISTIAN ARTHUR** is currently pursuing the bachelor's degree in computer science with Brawijaya University, Malang, Indonesia. He is actively exploring the dynamic field of deep learning. As he progresses in his academic journey and looks forward to graduating in the near future, his primary focus is on applying deep learning methodologies to forecasting. He is enthusiastic about contributing to the advancement of forecasting techniques by combining theoretical knowledge with practical applications to improve predictive modeling.

**NOVANTO YUDISTIRA** received the bachelor's and master's degrees in informatics engineering and master of computer science, in 2007 and 2011, respectively, and the Dr.Eng. degree in information engineering from Hiroshima University, in 2018. He is currently a Lecturer and a Researcher with Brawijaya University. Collaborating with AIST, Japan, in 2016, he continued as a Postdoctoral Fellow with RIKEN and Osaka University, from 2018 to 2020. He specializes in deep learning, multimodal computer vision, medical informatics, and big data analysis.

**CANDRA DEWI** received the B.Sc. degree in informatics from Institut Teknologi Sepuluh Nopember, in 2001, and the M.Sc. degree in information technology from IPB University, in 2008. She is currently pursuing the Ph.D. degree in computational biology with Brawijaya University. She has been with the Intelligent Computing Research Group, since 2012, exploring image processing and computer vision across various image types. Her research interests include machine learning applications, including pattern recognition and optimization, with a specialization in computer vision.