

RESEARCH ARTICLE

A Denoising Time Window Algorithm for Optimizing LSTM Prediction

SIHAO WAN 

College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China


e-mail: workflow.wan@gmail.com

ABSTRACT Realistic anomalous data noise is usually locally distributed in nature. To address the challenge posed by clustered anomalous noise in time series data, which not only reduces the accuracy of forecasting models but also presents a unique challenge due to its clustered nature and existing methods' limitations in handling such noise, this study introduces a novel denoising algorithm leveraging an optimized Long Short-Term Memory (LSTM) network, named 'Denoising Time Window Optimized LSTM' (DTW-LSTM), which innovatively identifies and rectifies clustered anomalous noise in time series data, significantly enhancing prediction accuracy by adeptly balancing the denoising process and data integrity preservation, as evidenced by comprehensive experimental validations. The algorithm employs a sliding time window mechanism to analyze the original series and its Fourier approximation, identifying clustered anomalies through a clustering model. It then uses a symmetrical data mixing and crossover technique to enhance the data quality in noisy segments. This process preserves the integrity of time series data by maintaining a balance between the original and fitted series, leading to a significant improvement in prediction accuracy. Experimental results indicate that the algorithm can adeptly discern the presence of clustered anomalous noise within the current time window and effectively rectify sections of data containing such noise. By surpassing the performance benchmarks set by contemporary forecasting models, this algorithm has established itself as the new SOTA method in the field of time series forecasting. Each sub-model contributes to the superior accuracy and reliability of the algorithm.

INDEX TERMS Time series forecasting, data preprocessing, sampling, sliding time window mechanism, neural network.

I. INTRODUCTION

Meteorological forecasting holds paramount importance. Beyond typical weather predictions, it extends to studying climate system alterations which subsequently impact societal sectors like transportation and finance. Atmospheric science necessitates understanding factors such as temperature, precipitation, atmospheric pressure, and wind strength. Of these, temperature is a pivotal parameter with profound implications. Variations in temperature directly affect daily life and, in extreme cases, may even pose hazards like fires. Considering the essence of meteorological predictions lies in forecasting certain quantities over time, it closely aligns with time series prediction problems [1].

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

Time series forecasting is a method that deciphers the inherent patterns of sequences evolving over time to predict future values. This approach finds wide applications in meteorology, transportation, chemical engineering, and financial analysis, among others. Conventional mathematical techniques often employ the Grey Model (GM) to discern the dynamic behavior within sample data, aiding predictive accuracy [2]. However, with recent advances in computational capabilities, intelligent algorithms rooted in machine learning and deep learning are challenging these traditional forecasting methods. Recurrent Neural Networks (RNNs) embed temporal concepts into network designs, enhancing adaptability [3]. Further advancements led to the Long Short-Term Memory (LSTM) model by Hochreiter and Schmidhuber, which addresses gradient explosion, gradient vanishing, and long-term memory limitations inherent in

traditional RNNs, making it more adept at handling prolonged time series data [4]. Presently, LSTM models are prevalent in domains like natural language processing and time series forecasting.

However, time series often exhibit noise that manifests as anomalies, possibly resulting from factors like environmental shifts, ecosystem disturbances, or abnormal human activities. Such anomalies might cluster temporally. For instance, data noise during periods around cold waves may substantially exceed that in regular intervals. Cheng et al. explored anomaly detection in multivariate time series, introducing a graph-based algorithm that discerns dependencies between variables using kernel matrix alignment. Their findings suggest this alignment mitigates noise, retaining only those anomalies in the target series that can be explained by others, such as ecological disturbances in geoscientific data [5]. Another study delved into cross-modal transfers in abnormal gait pattern recognition, highlighting the challenges of handling complex health informatics datasets, especially 4-D human motion data (xyz-t) encoding spatial-temporal information. By analyzing these data, one can derive indicators of abnormal gaits in medical applications [6].

Given that time series typically comprise noise, conventional least squares regression might not offer high predictive accuracy. Modern techniques leverage machine learning or deep learning to better extract features and predict from noisy time series. This work aims to meticulously identify and rectify clustered noise anomalies in time series using a fusion of least squares regression and a sliding noise-filtering temporal window for enhanced feature extraction and prediction for deep learning models. The algorithm detects noise via clustering, refines local sequences through a data stacking model, and precisely amends the original series. This study is based on Daily Average Temperature (DAT) as a dataset, which is fitted using Fourier series to obtain the fitting function. This is because fitting the DAT data using a function shaped like a Fourier series gives better access to its periodic characteristics through the symmetry of the time series. Notably, while the time series is fitted using Fourier series in this study, if the feature space before and after the series is the same, the proposed method can adapt to other data forms, endorsing its broad applicability [7].

This study's novel contributions include:

- 1) Emphasizing the clustered distribution of anomaly noise, the approach employs sliding temporal windows to meticulously analyze local sequences within the time frame. This method eradicates the necessity for holistic sequence amendments, and when the window unmodified, a window progresses by merely a singular time step, thereby keenly capturing anomalies and bolstering the stability of clustering centroid coordinates based on repeated part. When the window modified, algorithm will ignore the repeated part by processing the length of the window in order to increase the computational speed.

- 2) The model introduces a data stacking strategy that not only contemplates the intrinsic attributes of the original series but also integrates the smoothness and periodicity facets of the Fourier-transformed sequences. This symmetrical synergistic combination propels the generation of a sequence that closely mirrors the substantive characteristics of the original.
- 3) Leveraging a multi-objective optimization technique within the repair model, the methodology ensures meticulous value rectification. By harnessing an adaptive repair strategy, each time window is independently addressed, safeguarding its unique features and preventing undue influence from overarching data, thereby augmenting the sequence's receptivity by LSTM models.

The remainder of this paper is structured as follows: Section II reviews relevant literature; Section III details our proposed algorithm; Section IV shows the analyses under the process of our algorithm; Section V presents simulation experiments among various groups; and Section VI concludes and outlines future research directions.

II. RELATED WORK

While traditional time series analysis methods such as the Autoregressive Moving Average model (ARIMA) and the Error, Trend, and Seasonality model (ETS) have their advantages, they often struggle with complex nonlinear patterns. In contrast, time series forecasting based on deep learning has garnered widespread attention in recent years across various application domains. Particularly, Recurrent Neural Networks (RNN) and Long Short-Term Memory networks (LSTM) have become popular tools for time series forecasting due to their ability to capture long-term dependencies in time series data. These models can identify dynamic changes in data over extended periods, and they have been widely applied in areas such as natural language processing and financial forecasting.

A. RELATED WORK IN TIME SERIES FORECASTING

Time series forecasting aims to leverage temporal dependencies in historical data, using statistical or machine learning models to predict numerical variations at future time points. This encompasses a broad range of applications, including financial market fluctuations, meteorological changes, and stock price trends, offering data-driven predictability for decision-making. Researchers strive to optimize model accuracy and robustness while accounting for trends, cyclical patterns, and seasonality in time series to achieve more precise future predictions. In previous studies, traditional time series forecasting methods, such as ARIMA, were employed to capture autoregressive, differencing, and moving average characteristics of data, modeling trends and seasonality. For instance, Interrupted Time Series (ITS) is considered one of the best designs to establish causal relationships when randomized controlled trials are infeasible. This approach

mainly tracks outcomes before and after interventions. One study focused on using the ARIMA model for interrupted time series analysis to assess large-scale health intervention measures [8]. The ARIMA model has performed well in many practical applications [9], [10], [11].

However, its sensitivity to data stationarity and parameter selection might limit its performance on complex datasets [12]. Recently, there's a trend in the academic community towards adopting machine learning and deep learning approaches for time series forecasting. Efforts in the academic community are geared towards integrating deep learning approaches with classic or traditional algorithms and models to achieve superior results [13], [14], [15]. It achieved the best Average Ranking (AR) among the methods employed. Additionally, the results showed that models based on multi-head attention were the second-best, highlighting the predictive power of the attention mechanism in time series forecasting. Another research introduced a deep learning hybrid model based on LSTM and the Artificial Bee Colony algorithm (ABC). It explored various facets of market sentiment, utilizing the robust big data platform Hadoop ecosystem and its services to compute sentiment polarity indices. The ABC algorithm was apt for selecting hyperparameters for the deep LSTM model, balancing the exploration and exploitation issue. Processing a vast amount of multi-dimensional reviews from social media posed a significant challenge. The performance of the ABC-LSTM hybrid model was compared with other primary models and hybrids using evolutionary algorithms like Differential Evolution (DE) and Genetic Algorithm (GA). Performance analysis affirmed that, combined with sentiment polarity, the ABC-optimized LSTM surpassed other models in prediction accuracy [16]. Another study proposed a deep learning model based on PSO to evolve the initial weights of LSTM and fully connected layers. Further, the research introduced an adaptive approach using the velocity of particles to enhance the inertia coefficient of PSO. The method combined adaptive PSO and the Adam optimizer to train LSTM. The adaptive PSO sought to evolve the initial weights in the LSTM network and fully connected layers. The research also compared the predictive efficacy of the proposed method with a hybrid LSTM model based on the genetic algorithm, the Elman neural network, and the standard LSTM. Experimental outcomes showcased that the model successfully achieved optimal initial weights and biases for LSTM and the fully connected layer, boasting superior predictive accuracy than other models [17]. One study employed random search and a metaheuristic method based on SARS-Cov-2 virus propagation to obtain the best values for hyperparameters, using the optimal LSTM to predict power demand for a 4-hour forecasting period. Results and discussions were based on nine and a half years of Spanish electricity data measured at ten-minute intervals [18]. The academic community also hopes to obtain more precise results by improving the structure of deep learning models and achieve models with faster

computational times. One paper focused on several severe challenges with the Transformer model when forecasting long sequence time series (LSTF), such as quadratic time complexity, high memory usage, and inherent limitations of the encoder-decoder architecture. They designed an efficient transformer-based model for LSTF, named Informer, and conducted extensive experiments on four large-scale datasets. The outcomes suggested that Informer significantly outperformed existing methods, presenting a novel solution to the LSTF issue [19]. Similar studies have been conducted in the following articles [20], [21], [22], [23], [24].

B. RELATED WORK IN DENOISING AND SLIDING TIME WINDOW ALGORITHM

Denoising techniques are pivotal in digital signal processing, image processing, and communication fields. The primary goal is to recover original information from signals or images contaminated by noise. Noise can be attributed to various reasons like device imperfections, channel interference, or errors during data acquisition. Earlier denoising techniques often employed filters like the Wiener and Gaussian filters. With the advancement in computational capabilities and theoretical progress, a plethora of sophisticated denoising methods emerged, encompassing wavelet transforms, Total Variation (TV) regularization, sparse representation, and dictionary learning, to name a few. One research introduced an autoregressive model for multivariate probabilistic time series forecasting, named TimeGrad. This model samples from the data distribution at each time step by estimating its gradient. It combines the advantages of autoregressive models and possesses the flexibility of EBMs, serving as a universal high-dimensional distribution model while achieving autoregressive denoising [25]. Another Research [26] optimises the prediction effect of RNN,LSTM,GRU and Transformer using look-back window technique and compares it, the result proves the effectiveness of Transformer. There are also areas where better time series prediction has been achieved based on denoising algorithms [27]. In recent studies, deep learning techniques, especially Convolutional Neural Networks (CNN), have been extensively employed for denoising tasks [28]. Owing to their potent representation learning capabilities and adaptability, these methods have achieved unprecedented performance on multiple standard datasets. Furthermore, unsupervised and semi-supervised learning methods have also garnered attention in the denoising domain, as they can be trained without clean labels [29], [30].

The Sliding Time Window algorithm is a technique commonly used for processing streaming data. Its primary objective is to compute or statistically analyze a specific metric in continuously updating data streams, maintaining focus on data over the most recent period. This algorithm is applicable in many real-time data analysis and monitoring scenarios, such as network traffic monitoring, real-time log analysis, and sensor data processing. Its fundamental

principle involves maintaining a fixed-length time window that continually slides forward with time, discarding old data and incorporating new data. The goal is to maintain real-time computation and focus solely on recent data, better adapting to the evolution of data streams. Several studies in reality are based on the Sliding Time Window algorithm. For example, One paper introduced two novel sliding window-based CSP techniques, SW-LCR, and SW-Mode. Both methods were independent of selecting specific time points, offering additional advantages to generic feature extraction techniques like CSP, enhancing their performance [31]. A similar approach was used in the following study to investigate the effect of time windows and denoising on time series [32], [33], [34].

In summary, from past research in time series algorithm studies, it's evident that the selection or optimization of algorithms or models, the scale of algorithmic time complexity, hyperparameter setting or optimization, and the correlation and interpretability of steps in algorithm processing will profoundly impact the algorithm's performance. These factors can be further categorized into two aspects: effectively capturing and extracting time series data features and balancing the algorithm's time complexity with prediction accuracy. As evident, employing various time series neural network algorithms for time series forecasting has been validated as an effective method by numerous scholars. Many have attempted innovations in modifying time window length, the number of hidden neurons, batch size, learning rate, biases, etc., applying them in their respective study domains. Some scholars also tried modifying network topologies, yielding excellent results in the time series forecasting domain. However, to our knowledge, hardly any scholars have attempted parallel computation by introducing its fitted series and time window or optimized original time series data features. Similarly, few scholars have tried to determine time window status based on the time series's aftereffect. Based on these observations, the study has proposed a denoising time window-optimized LSTM forecasting algorithm. It categorizes similar time windows into one class using a clustering model, uses a repair model to perform multi-objective optimization between the original time series and fitted series, and employs a data stack model for cross-mixing both series. This strengthens beneficial data features for time series forecasting and reduces cluster-based anomalous noise.

III. PROPOSED METHODOLOGY

In order to improve the accuracy of the time series forecasting, this paper employs the least squares regression method to generate a fitted time series corresponding to the original time series. Furthermore, the study introduces a sliding denoising time window to eliminate cluster-based anomaly noises. This preprocessing facilitates subsequent deep learning models in feature extraction and prediction. The introduction of this time window is structured into three sections:

The decision-making model that determines whether the windowed data contains cluster-based anomalous noise through clustering algorithms. The data stacking model designed for optimized local sequence processing. The repair model that corrects the original time series once the windowed data is identified to contain clustered anomalous noises. For clarity, relationships between these models are detailed as follows:

- 1) **Clustering Model:** This model creates state points for the local data of each time window. The relationship between the state points and the clusters is then determined through clustering algorithms, assessing whether the current window contains clustered anomalous noises.
- 2) **Repair Model:** Using the target series and the fitted series as multiple objectives (or the target series and the original series, depending on the context), this study employs a machine learning approach to iteratively optimize the local original series by denoising and return the denoised results to the local data.
- 3) **Data Stacking Model:** By intermixing the original series with the fitted series, the study aims to extract features from both and merge them into a target series, which is then fed into the repair model. The intermixing strategy is context-dependent.

The framework of the sub-model algorithms is illustrated in Figure 1.

This framework outlines the algorithm's execution flow and the relationships between the sub-models. The algorithm starts with an initialization phase which involves data preprocessing, algorithm parameter setting, and Fourier series fitting of the original sequence to generate a fitted sequence of the same length. The original sequence and the fitted sequence are then passed into the denoising sliding time window model for noise removal. The denoising process is iterative. The position of each iteration in a high-dimensional state space is calculated and classified using a clustering classifier. If the position is outside the range defined by the centroid, the data is first synthesized through the data stacking model, then the repair model is applied to correct the local original sequence within this time window, reducing the noise in the sequence. If the position is within the range of the centroid, no correction is made and the time window continues to slide until the end of the sequence. The final sequence is input into an LSTM network for training, prediction, and testing.

A. DESIGN OF THE CLUSTERING MODEL

Uniformly identifying the anomalous time series noise and processing it is one of the important goals of this study to improve the accuracy of time series prediction. During the sliding of the denoising time window designed by the algorithm, it continuously identifies whether the local data inside contains clustered abnormal noise. If such noise is detected, denoising is performed on the local data. The

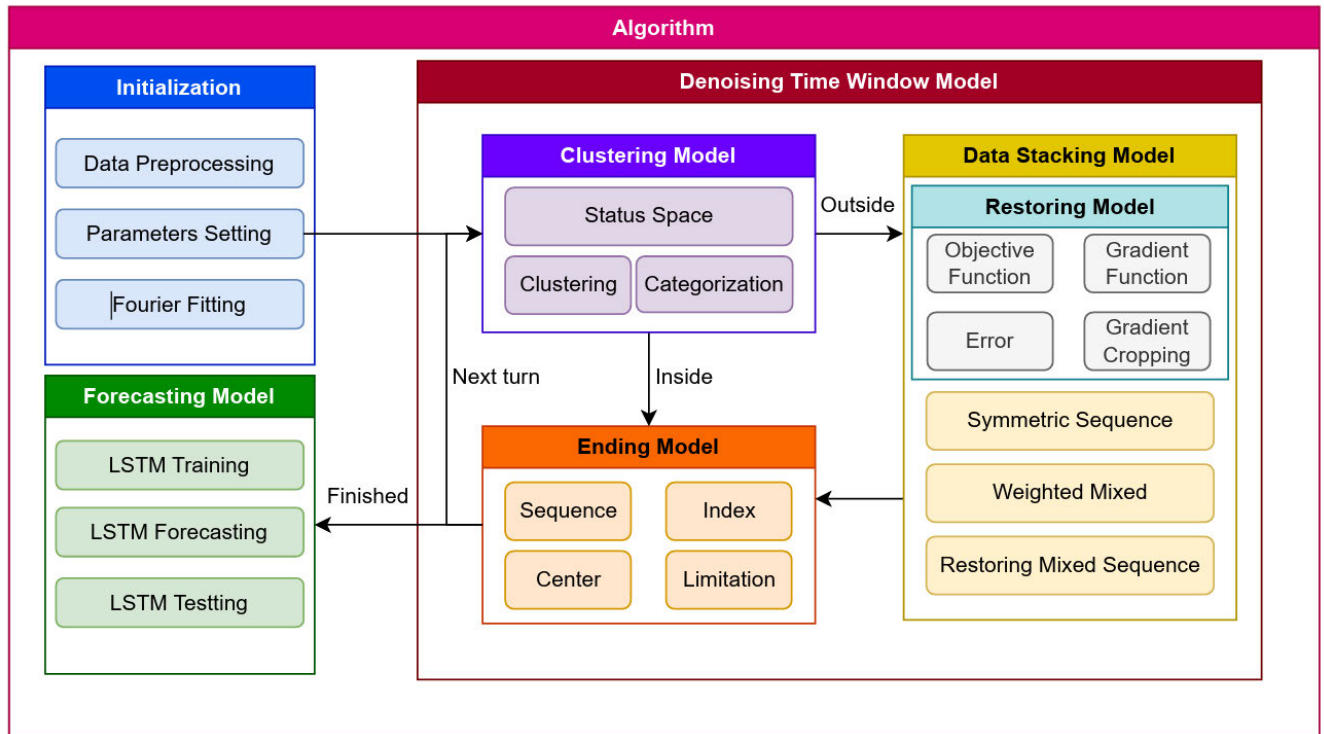


FIGURE 1. The framework of the sub-model algorithms.

clustering model aims to map local data from each time window to a high-dimensional space using its data features. By evaluating the Euclidean distance between the state point and the centroid, the position of the state point in the state space can be determined, achieving identification of the clustered abnormal noise. Thus, the clustering model elaborated in this paper is utilized to determine whether there's a need to correct the original sequence within the time window. Here, it is assumed that τ represents the number of dimensions in this high-dimensional space.

During the sliding process of the time window, the i -th round of the time window conducts calculations on the original and fitted sequences within the window. Based on these calculations, a position P_i in the τ -dimensional space corresponding to this time window is generated. As the time window slides, the number of points in the τ -dimensional space keeps increasing. It's worth noting that, considering the clustering model's requirement to identify and judge whether the time window is abnormal, the clustering calculation for each time window follows the procedure of "calculate P_i - judge abnormality - merge P_i ". The calculation of P_i will be based on the statistical relationship between the local original sequence and the fitted sequence in the time window, and the absolute value of each indicator will serve as the size of the current dimension of P_i . The abnormality judgment operation requires a geometric analysis of the point P_i corresponding to the current time window with the previous point set S_{i-1} (excluding P_i) under the τ -dimensional space.

Specifically, the model first calculates the unique clustering centroid P_C of S_{i-1} , and then constructs a τ -dimensional space H_i^τ with P_C as the center and d_i^τ as the Euclidean distance. The geometric relationship between P_i and H_i^τ is used to determine whether the current P_i is abnormal. Finally, regardless of whether the time window is abnormal, the current P_i will be merged into the point set to form S_i , and participate in the calculation of the next round of the time window.

In the clustering process, the coordinates of the centroid are computed to minimize the Euclidean distance from all data points. Consequently, the objective function represents the sum of squared Euclidean distances of all points within the set S_{i-1} to the cluster centroid. This is given by Eq.(1):

$$J(P_C) = \sum_{j=1}^{i-1} \|P_j - P_C\|^2 \tag{1}$$

where P_j represents a point in the τ -dimensional space ranging from index 1 to i , and $J(P_C)$ is the objective function acting on P_C . Given that the coordinates of the points are represented by a vector of length τ , assuming $P_j = (P_{j1}, P_{j2}, \dots, P_{j\tau})$ and $P_C = (c_1, c_2, \dots, c_\tau)$, the Euclidean distance between the two points is Eq.(2):

$$\|P_j - P_C\|^2 = \sum_{k=1}^{\tau} (P_{jk} - c_k)^2 \tag{2}$$

The objective function is expanded further as Eq.(3):

$$J(P_C) = \sum_{j=1}^{i-1} \sum_{k=1}^{\tau} (P_{jk} - c_k)^2 \quad (3)$$

To obtain the clustering center of the entire point set S_i , this objective function $J(P_C)$ needs to be minimized. Differentiating this objective function with respect to each dimension c_k of the center point P_C yields as Eq.(4):

$$\frac{\partial J(P_C)}{\partial c_k} = \frac{\partial}{\partial c_k} \sum_{j=1}^{i-1} \sum_{k=1}^{\tau} (P_{jk} - c_k)^2 = -2 \sum_{j=1}^{i-1} (P_{jk} - c_k) \quad (4)$$

To minimize the above objective function $J(P_C)$, set Eq.(5):

$$\frac{\partial J(P_C)}{\partial c_k} = 0 \quad (5)$$

The expression for the coordinate of the center point in the k -th dimension is Eq.(6):

$$c_k = \frac{1}{i-1} \sum_{j=1}^{i-1} P_{jk} \quad (6)$$

Thus, the coordinates of the center point can be derived as Eq.(7):

$$P_C = \left(\frac{1}{i-1} \sum_{j=1}^{i-1} P_{j1}, \frac{1}{i-1} \sum_{j=1}^{i-1} P_{j2}, \dots, \frac{1}{i-1} \sum_{j=1}^{i-1} P_{j\tau} \right) \quad (7)$$

This mathematically proves that a valid set S_{i-1} (for $i > 1$) has a unique clustering center P_C . Considering that the set should be generalized to S_{i-1} (where i belongs to the set of positive integers), the model will make a special judgment for $i = 1$, at which point the cluster center is the point itself. According to the above calculation method, a stable solution P_C can be obtained as the clustering center coordinate for the i -th round. This stable clustering center will well represent the previous entire sequence. Assuming the boundary of the τ -dimensional space and the Euclidean distance from the clustering center are represented by δ_i^τ , the Euclidean distance between the newly added point P_i and the clustering center P_C can be expressed as Eq.(8):

$$d_i^\tau = \sqrt{\sum_{k=1}^{\tau} \|P_{ik} - c_k\|^2} \quad (8)$$

If the newly added high-dimensional space point satisfies $d_i^\tau > \delta_i^\tau$, it is determined that the newly added point P_i is outside the current high-dimensional space H_i^τ , and the algorithm will correct the data within that time window. Otherwise, the inequality $d_i^\tau \leq \delta_i^\tau$ holds, thus ignoring the correction requirement for this time window. Given that the corrective time window's effect range corresponds to the

entire time interval of the time window, if the time window length is denoted by L , the sliding step length ΔL of the current time window should satisfy as Eq.(9):

$$\Delta L = \begin{cases} 1, & \text{if } d_i^\tau \leq \delta_i^\tau \\ L, & \text{else} \end{cases} \quad (9)$$

This step length means that when the data in the time window does not require correction, the window slides by only one time step. However, after the original sequence data in the time window is corrected, the window slides by the entire window length. The advantages of this algorithm design are:

- 1) Once the entire original sequence in the time window is corrected, the corrected data is not corrected again. This accelerates the sliding speed of the time window and reduces the overall time complexity of the algorithm.
- 2) If the time window is not corrected, to capture cluster anomalies as sensitively as possible, the time window is set to move only one time step. This ensures that potential cluster anomalies are not missed during the window's movement.
- 3) If the position corresponding to the time window in the high-dimensional state space falls within the range of the centroid, only one time step corresponding to the time series will have $len - 1$ points repeating with the previous state. This implies that the new state space point is a neighbor of the previous state point and will likely be calculated inside the high-dimensional space H_{i+1}^τ similarly to the previous point. This recursive relationship enhances the stability of the centroid coordinate x_c . This stability ensures that the centroid P_C more precisely represents the state of the sequence, thereby optimizing the clustering algorithm's classifier classification results.

B. MODEL DESIGN FOR ANOMALY RECTIFICATION

To enhance the accuracy of identifying clustered anomalous noise within a time window sequence, a rectification model has been proposed. The primary objective of this model is to adjust the assignment sequence res , making its values progressively converge towards a given target sequence, denoted as $target$. This approach introduces the notion of "selectivity", intending to adopt a multi-objective strategy in sequence rectification. Directly assigning the anomalous segment values to the target sequence would strip it of the characteristic smoothness and periodicity derived from its fitted sequence, ultimately impeding the predictive capabilities of the LSTM model. Conversely, assigning the anomalous segment to its Fourier-fitted sequence would lose the intrinsic data features of the dataset under consideration, which is equally detrimental to time series forecasting.

Throughout the rectification process, the assignment sequence res gradually approaches the $target$ sequence via machine learning techniques. However, with each iteration, a Fourier series noise is introduced to reduce the fitting degree

between *res* and *target*, while simultaneously enhancing the curve’s smoothness and periodicity. These attributes are more readily captured by subsequent LSTM networks for prediction.

During the iterative process of *res* converging to *target*, the cost function is denoted as the Mean Squared Error (MSE), which captures the discrepancy between the current sequence and the target sequence. With every iteration, the goal is to minimize this discrepancy. Here, assumed that the sequence *res* corresponds to the value R at each time point, and the sequence *target* corresponds to the value T at each time point, the sequence *fourier* corresponds to the value F at each time point. For the j^{th} time step within the window of the e^{th} iteration, The cost function can be represented as $K^{(e)}$ in Eq.(10):

$$K^{(e)} = \frac{1}{L} \sum_{j=1}^L \|R_j^{(e-1)} - T_j^{(e-1)}\|^2 \quad (10)$$

The gradient of this function with respect to $R_j^{(e-1)}$ is Eq.(11):

$$\frac{\partial K^{(e)}}{\partial R_j^{(e-1)}} = \frac{2}{L} \sum_{j=1}^L (R_j^{(e-1)} - T_j^{(e-1)}) \quad (11)$$

To enhance the noise introduced by the Fourier series, the algorithm optimizes the repair values of the assignment sequences. The algorithm defines an incremental sequence, A , indicating the proportion of the repair at each time step in the time sequence repair process. The purpose of designing this sequence is that it possesses memory. In each machine learning iterative process, it is essential to repair the time sequence based on the repair situation of the previous iteration. The increment sequence allows for the preservation of the changes from the last time step. The increment value for a time step in the current round is determined by the increment value of the same time step in the previous round and the gradient of this time step in the current round. This approach ensures that the repair of each time step converges. For the j^{th} time step within the window of the e^{th} iteration, its value is assigned as Eq.(12), where α refers to the learning rate:

$$A_j^{(e)} = A_j^{(e-1)} - \alpha \frac{\partial K^{(e)}}{\partial R_j^{(e-1)}}, \quad \forall j = 1, 2, \dots, L \quad (12)$$

Considering the noise addition process of the Fourier series, the repair value $\gamma(j, e)$ should not exceed the difference between the current ‘res’ sequence and the Fourier sequence. Otherwise, the repair value will cause the ‘seq’ to cross the Fourier series, thereby reducing the noise impact added by the Fourier series. $\gamma(j, e)$ can only take the maximum value among them as the actual repair size as Eq.(13):

$$\gamma_j^{(e)} = \|R_j^{(e-1)} - F_j\| \cdot \max(A_j^{(e)}, 1) \quad (13)$$

Finally, the actual repair value is assigned to the sequence *res* to be processed. After the repair, the output *res* is the result, expressed as Eq.(14):

$$R_j^{(e)} = R_j^{(e-1)} + (2\delta_{(R_j^{(e-1)} \leq F_j^{(e-1)})} - 1)\gamma_j^{(e)} \quad (14)$$

Here, δ is a Kronecker delta function. It is 1 when the subscript is true, and 0 otherwise. The algorithm will iterate several times and determine whether a stable solution for the cluster center coordinates has been obtained based on two factors as Eq.(15) and Eq.(16):

1. The number of iterations reaches the maximum allowable limit:

$$e > e_0 \quad (15)$$

2. The objective function converges:

$$K^{(e)} < \epsilon \quad (16)$$

where ϵ is a small number. When the objective function is less than this value, it is determined that the cost function has converged. The aforementioned equations indicate that the repair process for each time step is independent. This approach attenuates the internal linkage between time steps, enhancing the repair independence of each time step. As a result, the internal linkage among time steps during the time window repair process is solely determined by the cost function. The unified objective for all time steps during the iteration process is to move the sequence as close as possible to the target sequence. However, during this movement, each iteration is influenced by the noise introduced by the Fourier series. This approach accomplishes the multi-objective optimization task for the assignment sequence, *seq*, by absorbing as much of the data characteristics of the target sequence as possible, as well as the smoothness and periodicity of the Fourier series. Consequently, this is beneficial for time series prediction.

C. DATA STACKING MODEL DESIGN

To improve the denoising effect of time series data, relying solely on features derived from the original time series and its fitted sequence might be insufficient. Therefore, the study introduces a data stacking model, aiming to enhance the efficacy of the rectification model, thereby optimizing the LSTM model’s predictive performance. The fundamental philosophy behind this design is to derive a sequence enriched with more distinctive features, which can then act as a target sequence guiding the alignment of the sequence under rectification.

The operational procedure of the data stacking model is iterative. It comprises two primary steps:

- 1) Initially, the sequence awaiting rectification is rectified using both the original and Fourier series sequences, generating three rectified sequences.
- 2) These rectified sequences are then integrated to produce a composite sequence, which emerges as the time series result of a single stacking cycle. This newly

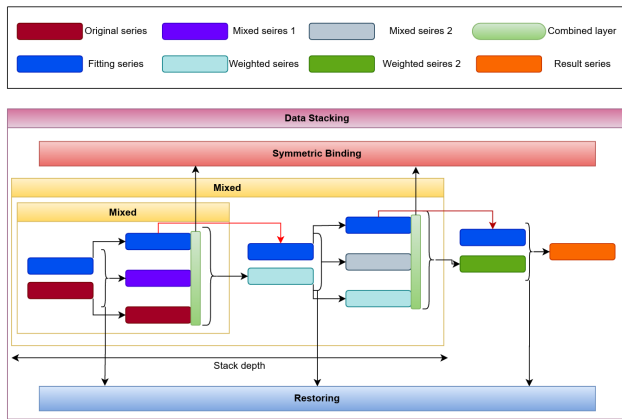


FIGURE 2. Logic diagram of the data stacking algorithm.

formed time series can again be stacked with the original or fitted sequence to intensify the stacking depth.

The sequence on top of the data stack is input into the rectification model as the target sequence, wherein it, along with the fitted sequence, acts as dual objectives for denoising. Such a way of designing the data stack adopts the idea of symmetry. Considering that neural networks are able to capture the data characteristics of the time series during training, a smooth fitted series can reduce the interference of clustered anomalous noise on the model training in the local region of the time series. However, excessive use of the fitted sequence approximation instead of the original time series loses the data characteristics of the original time series. Based on this, from the perspective of data mixing, the study designed the target sequence to be symmetric with respect to the original and fitted sequences.

Empirical evidence suggests that adopting a data stacking approach enhances the algorithm's ability to unearth salient features of the time series data. The advantage of this hybrid rectification model lies in its simultaneous consideration of the raw sequence's data features and the smoothness and periodicity of the Fourier transformed sequence. This symmetry perspective facilitates more effective rectification of the sequence in question. Throughout the operation of the data stacking model, the study continues to employ an adaptive rectification strategy. This strategy ensures that each timestamp is treated individually during rectification, fostering independence in the rectification of each timestamp, and circumventing the potential loss of their inherent characteristics due to the influence of overarching data patterns. Figure 2 shows the design of the data stacking model, which combined the idea of symmetry.

In actual stacking operations, the stacked sequences can be further stacked, increasing data processing depth. Sequence weighting choices are flexible. In addition to the current composite sequence and the sequence from the previous iteration, one can opt to use either the original or the fitted sequence as the third sequence for blending. The

Algorithm 1 DTW-LSTM for Denoising and Forecasting

Require: Original time sequence y_{Origin} , length of sequence len , Test sequence y_{Real}

Ensure: Future data y_{Pred} , evaluation indicators: $EVAL$

```

1:  $y_{Seq} \leftarrow init(0)$ ,  $idx, num \leftarrow 0$  {Initialization}
2: Preprocessing the time sequence, setting parameters:
   window length  $L$  and weight array  $w$ , fitting time
   sequence  $y_{Fourier}$ , creating status space  $H_0^T$ , and points
   set  $S \leftarrow \emptyset$ 
3: while  $idx < L$  do
4:    $idx \leftarrow idx + 1$ 
5: end while
6: while  $idx < len$  do
7:    $num \leftarrow num + 1$ ,  $area \leftarrow [idx, idx + L - 1]$ 
8:    $P_i^T \leftarrow CreatePoint(y_{Origin}(area), y_{Fourier}(area))$ 
9:   if  $num < \mu$  or  $P_i^T \notin H_{i-1}^T$  then
10:     $y_{Seq}(area) \leftarrow y_{Origin}(area)$ ,  $idx \leftarrow idx + 1$ 
11:   else
12:     $y_{Combined} \leftarrow Restoring(y_{Origin}, y_{Fourier})$ 
13:    for  $i$  in range(1, size(w)) do
14:       $y_{Mixed} \leftarrow w(i) \times [y_{Origin}, y_{Combined}, y_{Fourier}](i)$ 
15:    end for
16:     $y_{Combined} \leftarrow Restoring(y_{Mixed}, y_{Fourier})$ 
17:    for  $i$  in range(1, size(w)) do
18:       $y_{Mixed} \leftarrow w(i) \times [y_{Mixed}, y_{Combined}, y_{Fourier}](i)$ 
19:    end for
20:     $y_{Seq}(area) \leftarrow Restoring(y_{Mixed}, y_{Fourier})$ 
21:     $idx \leftarrow idx + L$ 
22:   end if
23:    $S \leftarrow P_i^T, H_i^T \leftarrow CreateStatusSpace(S)$ 
24: end while
25:  $y_{Pred} \leftarrow LSTM\_Forecast(y_{Seq})$ 
26:  $EVAL \leftarrow Evaluation(y_{Real}, y_{Pred})$ 

```

blending process might require some finetuning; however, the essential consideration is the desired feature shift direction. If more fitted sequence results are sought, the third sequence can be chosen as the fitted one, increasing the stacking depth. Conversely, if preserving the characteristics of the original data is prioritized, the third sequence can be the original. It's also feasible for the original and fitted sequences to alternate. It's worth noting that a deeper stack doesn't necessarily equate to superior predictive results. For instance, if the feature offset direction is set towards the fitted sequence in each iteration, the impact of stacking on the original sequence diminishes over iterations. The time complexity, however, grows exponentially, which isn't conducive to practical predictions. Therefore, the selection of symmetry objects, the size of mixing weight vectors and the number of mixing times in the symmetry process will affect the actual prediction.

In our experiments, the optimal configuration was a double-layered Fourier structure with a stack depth of two, consistently biasing the resultant sequence towards the

fitted one. Other configurations tested include double-layered original sequence structures and combined original-Fourier sequences, with the latter two yielding similar prediction outcomes. Nevertheless, none of them outperformed the double-layered Fourier structure. Our comparative experiments are primarily based on this configuration. The following section provides a pseudocode representation of the primary algorithm.

The pseudocode mentioned employs a two-layer Fourier structure. In this context, idx signifies the pointer location within the algorithm, num denotes the current time window's index, L is the length of the time window, len indicates the length of the time series training set, and μ represents the minimum time window index necessary for a legitimate clustering model execution.

The algorithm operates as follows:

- 1) Initial preprocessing takes place, leading to the formation of the first time window.
- 2) Between lines 6 to 8, the algorithm increments the time window index, retrieves the corresponding time interval 'area', and extracts the state point P_i^T based on this interval from both the original and the fitted series.
- 3) Line 9 of the algorithm evaluates the clustering condition. This assessment requires both $num < \mu$ and P_i^T not belonging to H_{i-1}^T .

Given that the initial high-dimensional state space lacks point set S , the algorithm introduces a minimal time window index μ to integrate the state point P_i^T directly into the high-dimensional state space. Generally, μ is set to 3, and this paper validates with $\mu = 3$. The sphere H_{i-1}^T is composed of the first $i - 1$ points from point set S_{i-1} , centered at P_C with a Euclidean radius defined by 'Threshold'. If P_i^T belongs to H_{i-1}^T , it is perceived as consistent with the state of the time series, implying no cluster anomalies, and thus, the local data remains unchanged as per line 10. Otherwise, the local data undergoes modification, and lines 12-21 are executed.

Line 10 assigns the original time series corresponding to the time interval $area$ directly to the result series, advancing the time window by one step. Lines 12-19 first repair the original and fitted series to produce a combined sequence. Subsequently, these three sequences are linearly combined using a weight vector to produce a mixed series. This mixed series is once again repaired and linearly combined with the fitted series, resulting in a second combined and mixed series. Lastly, the post-mixed series is mended with the fitted series, assigned to the result series, and the time window shifts by its length to prevent redundant modifications, optimizing algorithmic efficiency.

At the end of each time window iteration, line 23 incorporates the state point P_i^T into the point set. This updated point set S_i then undergoes clustering to generate a new centroid and high-dimensional sphere H_i^T , which will be pivotal in the clustering analysis of the subsequent time window. Once the sliding of the time window concludes, the resulting series is fed into an LSTM for predictions, yielding several statistical metrics as the forecast outcome.

While this paper predominantly employs a two-layer Fourier structure for experiments, the algorithm's processing logic follows suit. Nevertheless, this structure is adaptable. For instance, if one desires to process the algorithm with a two-layer original series structure, lines 12 and 16 should replace the fitted series with the original series. Furthermore, the operations between lines 16 and 19 are essentially a second stack operation based on lines 12 and 15. The algorithm offers flexibility in controlling stack depth, allowing it to be adjusted for different data feature offsets and stack depths. The clustering algorithm's specifics (forming state points and state spaces) align with the K-Means approach, and thus, are not elaborated further in this paper.

D. ARCHITECTURE OF LONG SHORT-TERM MEMORY

LSTM (Long Short-Term Memory) is a specialized form of Recurrent Neural Network (RNN), characterized by its ability to store information over extended time periods through its multiple homogeneous units. Distinct from traditional RNNs, an LSTM model encompasses three control gates, namely the input gate, output gate, and forget gate. Together, these compound control gate units facilitate the reading, writing, and resetting of all neurons within the network.

A significant enhancement of LSTM over previous RNNs is the introduction of a softmax layer, utilizing the log-likelihood function as the loss function for error backpropagation. This attribute grants LSTM the capacity to selectively transmit information.

The forget gate, denoted by f , governs the degree to which x_t is forgotten, retaining a part that subsequently feeds into the input gate. Within the input gate i , x_t is processed through the combined influence of tanh and sigmoid activation functions to preserve the vector, eventually outputting the current value through the output gate h . The expressions for these operations are among Eq.(17) to (22):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (17)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (18)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (19)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (20)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (21)$$

$$h_t = o_t \cdot \tanh(C_t) \quad (22)$$

Here, x represents the input vector of the LSTM unit; f , i , and o denote the forget gate, input gate, and output gate respectively; h symbolizes the output vector of the cell; C signifies the cell state. Subscripts t indicate the time instant; σ and \tanh represent the sigmoid and tanh activation functions; W stands for the random weight matrix, and b is the bias vector. The architecture of the LSTM algorithm is showed in Figure 3.

In the architecture of LSTM, the crux resides in retaining the memory of the cell state C at time t , modulated by the forget gate and input gate. The input gate determines whether to allow or inhibit the updating of the cell state; the output gate

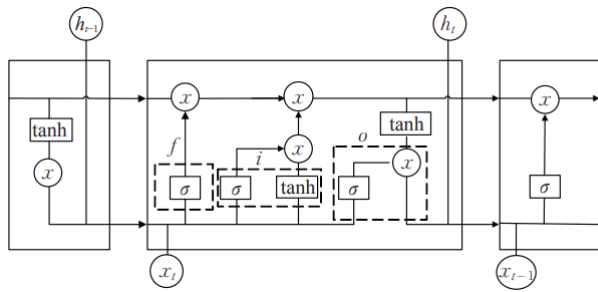


FIGURE 3. Depicts the topological structure of the LSTM.

controls the output and transfer of the cell state to the next unit; the forget gate enables the cell to remember or discard its prior state. Backpropagation remains the prevalent method for training this structure.

IV. ALGORITHMIC EFFECT ANALYSIS

A. DATASETS, METRICS, AND CONFIGURATIONS

This study utilizes atmospheric temperature data from Beijing, recorded at 2 meters above the ground, spanning from 2006 to 2022. This symmetric form of the periodic time series favours Fourier series for fitting. However, as mentioned earlier, our method will work for any function that can be regressed excellently. According to the dataset:

- 1) From January 1st, 2006 to January 19th, 2010, temperature measurements were taken four times daily at intervals of 6 hours: specifically at 2:00, 8:00, 14:00, and 20:00.
- 2) From June 20th, 2010 to December 31st, 2022, the frequency of data collection increased to eight times daily at intervals of 3 hours: specifically at 2:00, 5:00, 8:00, 11:00, 14:00, 17:00, 20:00, and 23:00.

In the preprocessing stage, daily temperature readings were averaged. For the period before January 19th, 2010, the four daily readings were summed and then averaged, while for the subsequent period, the eight daily readings were summed and averaged. This process yielded the average daily temperature. After preprocessing, there were five instances of missing values (NaN). For these instances, the temperature values from the preceding and following days were averaged to impute the missing data.

For the purpose of this study, the dataset was divided into training and testing subsets. The period from 2006 to 2018 was designated as the training set, while the years 2019 to 2022 formed the testing set. Experimental results were derived exclusively from the testing set.

The accuracy metrics employed in this study adhere to internationally accepted evaluation standards. These metrics include: Mean Absolute Error (MAE), Sum Squared Error (SSE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). Their respective formulas are presented in the following table.

In the aforementioned formulas, consider a sequence variable with a data length of n , where the corresponding natural number i traverses from 1 to n . Here, y_i represents the actual value, and \hat{y}_i denotes the predicted value. A smaller error value indicates higher model accuracy. Additionally, for the definition in high-dimensional space, this study adopts eight metrics to calculate their respective positions in the eight-dimensional state space for each time window. In the subsequent dimensional calculations, y_1 refers to the original time series y_{origin} , and y_2 denotes its approximated time series $y_{fourier}$. The symbol σ represents the standard deviation of data within a local interval. In the autocorrelation function, the lag time is one time step. Table 1 presents the detail of the equations.

All experiments in this study were conducted on a Windows 10 64-bit operating system, powered by an Intel i7-10750H 2.60GHz processor and an NVIDIA GeForce GTX 1650 Ti graphics card. The experimental environment was based on Tensorflow-2.10.0.

B. FITTING EFFECT ANALYSIS

This study forecasts the temperature data of Beijing, recorded 2 meters above the ground, from 2006 to 2022. To further evaluate the algorithm's effectiveness in time series processing, this study addresses and analyzes the phenomenon of cluster-based anomalous noise. Initially, for both the pre- and post-denoised time series and their corresponding Fourier series, visualizations are generated using MATLAB. Subsequently, by calculating the differences between the values of the pre- and post-denoised time series and their corresponding Fourier series over the time interval, the study examines the cluster-based anomalous noise in the post-denoised time series. In the visual representations, the blue curve signifies the actual temperature time series of Beijing from 2006 to 2022, 2 meters above the ground. The red curve represents the Fourier series approximation of the blue curve within that time frame. The yellow curve illustrates the improved temperature time series after denoising the data corresponding to the blue curve.

Regarding parameter configuration, the expression for the Fourier series is given by the Eq.(23), with the fitting parameters are detailed in the Table 2. In this equation, c represents the constant term, and a_i and b_i are the Fourier coefficients for the cosine and sine terms, respectively, and w is the angular frequency.

$$F(t) = c + \sum_{i=1}^8 (a_i \cos(iwt) + b_i \sin(iwt)) \quad (23)$$

Figure 4 reveals that the experimental Fourier series fitting yields satisfactory results due to the incorporation of eight terms each in the sine and cosine functions.

Following the algorithmic refinement of the original sequence, the newly derived sequence and the original time series were compared against the Fourier series to compute their respective errors. Specifically, by subtracting the Fourier

TABLE 1. Combined and transposed evaluation index formula.

| Attribute | Formula | Meaning |
|-----------|---|--------------------------------|
| MAE | $\frac{1}{n} \sum_{i=1}^n \hat{y}_i - y_i $ | Mean Absolute Error |
| SSE | $\sum_{i=1}^n (\hat{y}_i - y_i)^2$ | Sum of Squared Errors |
| MSE | $\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$ | Mean Squared Error |
| RMSE | $\sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2}$ | Root Mean Squared Error |
| MAPE | $\frac{100\%}{n} \sum_{i=1}^n \left \frac{\hat{y}_i - y_i}{y_i} \right $ | Mean Absolute Percentage Error |
| x_{i1} | $\frac{\sum_{j=idx}^{idx+L-2} (y_1(j) - \bar{y}_1)(y_1(j+1) - \bar{y}_1)}{\sum_{j=idx}^{idx+L-1} (y_1(j) - \bar{y}_1)^2}$ | Auto-correlation |
| x_{i2} | $\frac{1}{L} \sum_{j=idx}^{idx+len-1} y_1(j)$ | Moving Average |
| x_{i3} | $\sqrt{\frac{1}{L} \sum_{j=idx}^{idx+L-1} (y_1(j) - \bar{y}_1)^2}$ | Original Standard Deviation |
| x_{i4} | $\sqrt{\frac{1}{L} \sum_{j=idx}^{idx+L-1} (y_2(j) - \bar{y}_2)^2}$ | Fitted Standard Deviation |
| x_{i5} | $\frac{1}{L} \sum_{j=idx}^{idx+L-1} \left(\frac{y_1(j) - \bar{y}_1}{\sigma} \right)^4$ | Original Kurtosis |
| x_{i6} | $\frac{1}{L} \sum_{j=idx}^{idx+L-1} \left(\frac{y_2(j) - \bar{y}_2}{\sigma} \right)^4$ | Fitted Kurtosis |
| x_{i7} | $\frac{1}{L} \sum_{j=idx}^{idx+L-1} (y_1(j) - \bar{y}_1)(y_2(j) - \bar{y}_2)$ | Covariance |
| x_{i8} | $\frac{\sum_{j=idx}^{idx+L-1} y_1(j) - \sum_{j=idx}^{idx+L-1} y_2(j)}{len}$ | Deviation Degree |

TABLE 2. Parameters after processing the original dataset.

| Parameters | Values | Parameters | Values |
|------------|----------|------------|---------|
| w | 0.9078 | c | 13.75 |
| a_1 | -0.07389 | b_1 | 0.01613 |
| a_2 | -0.2069 | b_2 | 0.0252 |
| a_3 | 0.0845 | b_3 | -0.3116 |
| a_4 | 0.2382 | b_4 | -0.1607 |
| a_5 | 0.06081 | b_5 | -0.1522 |
| a_6 | 0.5811 | b_6 | -0.294 |
| a_7 | -12.6 | b_7 | 8.2 |
| a_8 | -0.07564 | b_8 | 0.5436 |

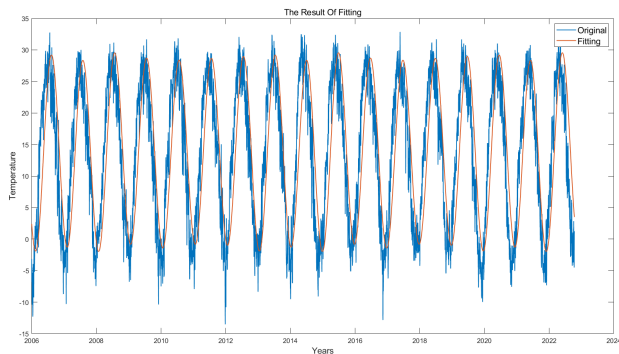


FIGURE 4. Fitting effect diagram.

series from either the algorithmically derived sequence or the original series, the following experimental results were obtained as follows.

In the Figure 5, the blue curve represents the absolute error between the Fourier series sequence and the original sequence, while the red curve represents the absolute error between the Fourier series sequence and the reconstructed sequence. As observed from Figure 5, the new time series derived from the original data through algorithmic processing

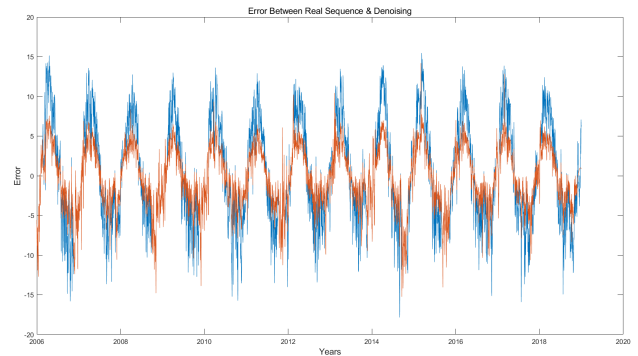


FIGURE 5. After processing, the numerical difference between the time series and the fitted series.

TABLE 3. Error metrics for original and processed time series with fitted fourier series.

| Sequence Type | MAE | SSE | MSE | RMSE | MAPE |
|---------------|--------|--------|---------|--------|----------|
| Original | 5.4035 | 193454 | 40.7695 | 6.3851 | 372.3611 |
| Processed | 1.4950 | 14857 | 3.1311 | 1.7695 | 102.9463 |

closely aligns with the fitted Fourier series in terms of numerical values. However, examining the absolute value of the numerical differences and their volatility reveals that the post-processed time series does not entirely conform to the Fourier series fitting. The experimental results lead to the conclusion that the denoising time window algorithm processes the original time series using a multi-objective optimization approach. The algorithm continuously seeks optimization between the original and fitted time series, maximizing the extraction of data features from both, enhancing the smoothness and periodicity of the time series, reducing the cluster-based anomalous noise inherent in the original sequence, and minimizing prediction errors. The following Table 3 presents the relative errors between the respective time series in the training set and the Fourier series.

This error demonstrates that the denoised and corrected algorithmic sequence is closer to the Fourier series fitting of the original sequence than the original sequence itself. This indicates that the denoising process overall reduces the occurrence of significant errors, leading to a more stable data set and a smoother graphical representation of the original time series. As will be discussed later, such stable time series fluctuations are advantageous for global time series forecasting.

C. DENOISING EXPERIMENTAL ANALYSIS

The algorithm employs a sliding time window to denoise the initially inputted time series. During the sliding process, state points in a high-dimensional space are continuously generated based on the current local data. This facilitates the determination of the presence of cluster-based anomalous noise within the local data of the time window using a clustering model. Given that the algorithm will repair the local sequences identified as having cluster-based anomalous noise, this paper presents both the results post-clustering and post-repair. The clustering section delineates the distribution of values for each dimension in the high-dimensional state space and the positions of their centroids. Additionally, a two-dimensional clustering diagram is showcased following principal component analysis of the high-dimensional state space. The repair section illustrates the time series post-algorithmic repair and juxtaposes it against the original and fitted sequences, emphasizing the effects of the repair.

1) CLUSTERING RESULTS ANALYSIS

An eight-dimensional state space was designed in the algorithm. After each iteration update of the time window, state points are generated based on the current data from the original and fitted sequences, which are then utilized in calculating the state space that includes the centroid. The Euclidean distance between the centroid and the current state point is computed and compared against a threshold to determine if the point lies inside or outside the state space. For points inside, the time window simply shifts by one time step, ensuring spatial proximity between consecutive state points. For points outside, the data within the current time window undergoes repair. In an experimental group with a clustering threshold of 5 and a time window length of 5, the following diagrams Figure 6 were generated. These diagrams contain eight axes, where each axis uses blue vertical lines to represent specific values for that dimension across different rounds. A red cross on each axis marks the centroid for that dimension.

Using principal component analysis (PCA), the original eight-dimensional state space is reduced to a two-dimensional space. This two-dimensional space is visualized using a scatter plot that maps all the points (including the centroid) from the eight-dimensional state space. Furthermore, a distance curve portrays the Euclidean distance between state points and the centroid across iterations. By using a clustering threshold as the division criterion, groups of diagrams are

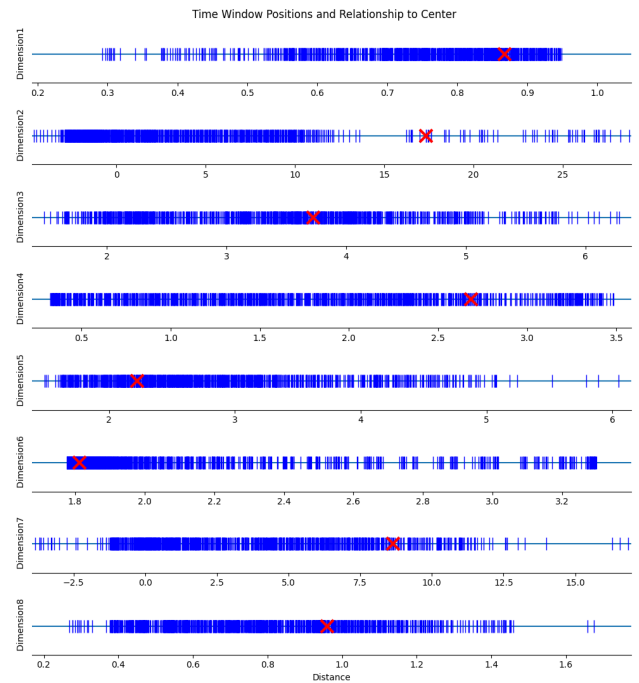


FIGURE 6. Dimensional data and central points.

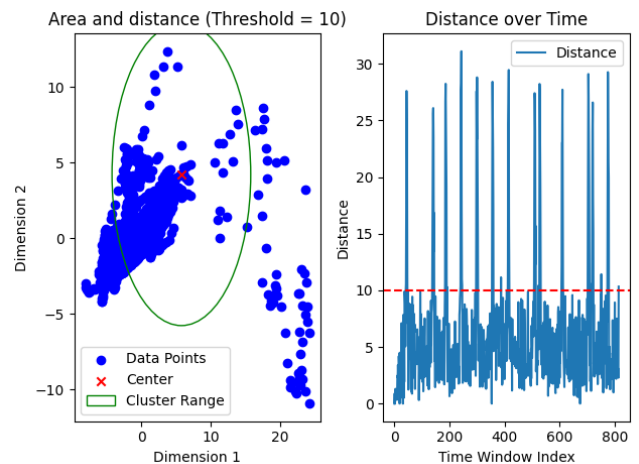


FIGURE 7. Cluster quality and euclidean distance presentation.

consolidated under the same threshold, as demonstrated below.

In the Figure 7 The eight-dimensional state space points are projected onto a plane following a reduction to two-dimensional coordinates using principal component analysis. From the diagrams, it is evident that the algorithm enhances the clustering effect of the point set through its clustering model and time step movement strategy, rendering the centroid more stable. Additionally, points around the centroid are categorized into two types based on a threshold delineation: those within the state space, where local data in the time window remains unchanged, and those outside, where modifications to the local data are necessary. It should be noted that the choice of such thresholds should ideally

ensure that the entire cluster range contains most of the points in the sample space, while a few points are exposed outside the range. This approach allows for the adjustment of a small number of local anomalous noise data while retaining most of the original data characteristics.

2) REPAIR RESULTS ANALYSIS

In the process of time series rectification, the study places a premium on understanding the ramifications of varying time window lengths on the outcomes of time series. Utilizing a clustering threshold of 10, and employing an incremental step of 5, the study systematically investigates the impact of differing time window lengths. Specifically, the study records both the count of these windows and the count of consecutive windows. Here, the term “consecutive window” denotes a scenario wherein, following an initial classification of a window as standard, the window merely shifts by a single time step. Based on these enumerated relationships, our experiment further calculates the ratio of consecutive window count to the total window count. A higher ratio suggests that the algorithm introduces minimal modifications to the initial series, indicating that the inherent features of the original series are substantially potent. Such an observation implies that this parameter set could potentially be more effective on the given dataset.

Considering that the length of the time series in our dataset remains constant, Figure 8 elucidates that, with the augmentation of the time window length, both the count of time windows and consecutive time windows manifest a declining trend. Moreover, the trend in the count of consecutive windows is almost parallel to that of the overall window count. The right-side graph delineates the ratio between the two metrics, showcasing that as the time window length burgeons from 5 to 25, the ratio experiences a gradual upswing. However, during the interval where the time window length lies between 25 and 50, this ratio remains relatively stable, subsequently beginning to taper off post 50. Such observations prompt us to infer that, with a clustering threshold of 10, time window lengths ranging between 25 and 50 appear to be the most efficacious.

D. EXPERIMENTAL RESULTS ANALYSIS

As illustrated in Figure 9, the time series, after undergoing the denoising process, more closely resembles the Fourier series compared to the original time series. The original series exhibited more pronounced discrepancies in its features. In contrast, the processed series has attenuated these exaggerated characteristics while retaining certain features essential for time series forecasting. Therefore, during forecasting, the processed time series is less influenced by high-variance features, ensuring enhanced temporal continuity for long-term predictions.

The results depicted in Figure 10 follow the algorithm’s execution flow. The first graph depicts the training and prediction processes of LSTM, where the blue segment represents the training set inputted into LSTM, and the red

segment showcases LSTM’s forecasted results. It is evident from the first graph that the predicted series, compared to the original, has significantly reduced cluster-based noise. This is manifested in the smoother forecasted series with reduced volatility and more prominent periodic features. The second graph presents the discrepancies between the forecasted results derived from this training set and the test set. The forecasted curve appears to circumvent some of the impacts brought about by cluster-based noise, particularly evident in the series’ extreme peaks and troughs where the forecast tends to converge towards the median. While this approach may not be optimal for predicting certain extreme scenarios (for instance, predictions can be less accurate for some cluster-based anomalies), the overall forecast converging towards the center enhances the accuracy of the entire prediction curve.

V. COMPARATIVE EXPERIMENTS AND ANALYSIS

During the comparative experiments, this study employed five sets of experiments to validate the efficacy of the proposed algorithm in time series forecasting.

Firstly, the algorithm was juxtaposed with various conventional forecasting models in terms of prediction accuracy. These models encompassed the Fourier series fitting prediction model, Holt-Winters prediction model, SARIMA prediction model, ETS prediction model, LSTM prediction model, and the post-fitted LSTM prediction model. The parameters for these traditional forecasting algorithms were fine-tuned based on the optimal configurations determined through experimental evaluations on the same dataset used in this study. This study will present both the best and worst prediction results of the proposed algorithm, comparing them with the control groups.

Secondly, by adjusting each of the three variables within the algorithm, different prediction accuracy outcomes were observed, showcasing the influence of these internal variables on the forecasting performance. These variables included the length of the sliding time window, the Euclidean distance threshold between the clustering model’s centroid and the state space boundary, and the combinations of weights and stacking structure in the data stacking model. Furthermore, the study has tested its performance with the state-of-the-art(SOTA) methods.

Lastly, an ablation study was conducted. By sequentially omitting the clustering model, the repair model, and the data stacking model, and retaining the other two models for prediction, the study demonstrated that each segment of the algorithm positively contributes to the final prediction outcome.

A. PREDICTION ACCURACY COMPARISON AMONG VARIOUS MODELS

This study utilized Fourier series regression, Holt-Winters, SARIMA, and LSTM to make predictions based on a daily regression dataset. The outcomes from these techniques were subsequently contrasted with the results from the model introduced in this paper. For the Holt-Winters model, the

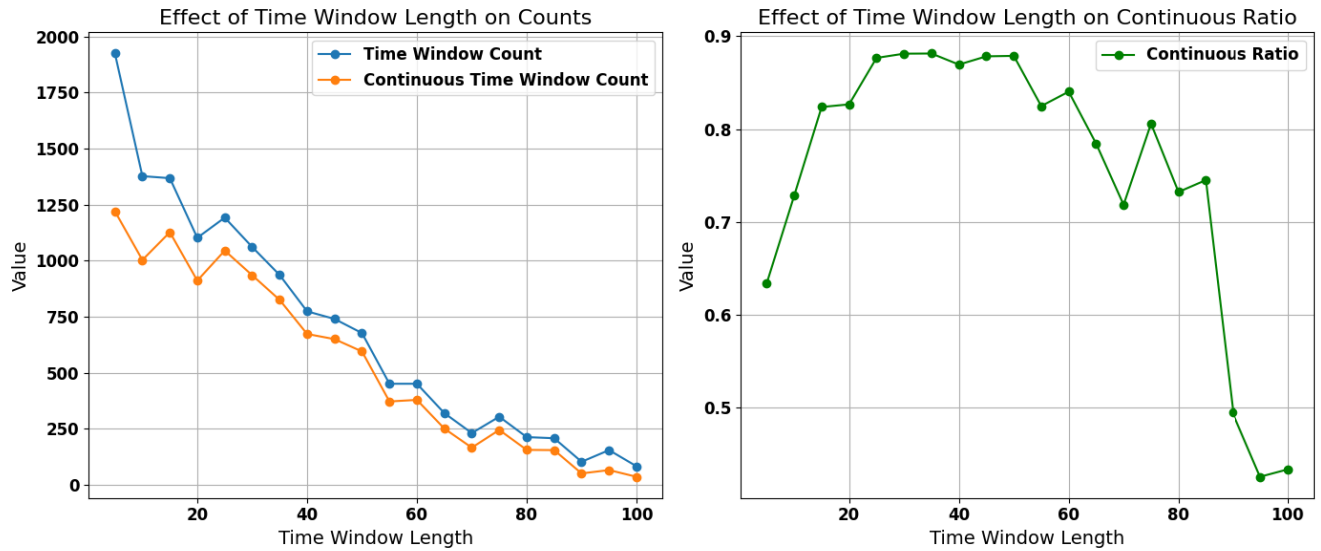


FIGURE 8. Restoring figure about the time series.

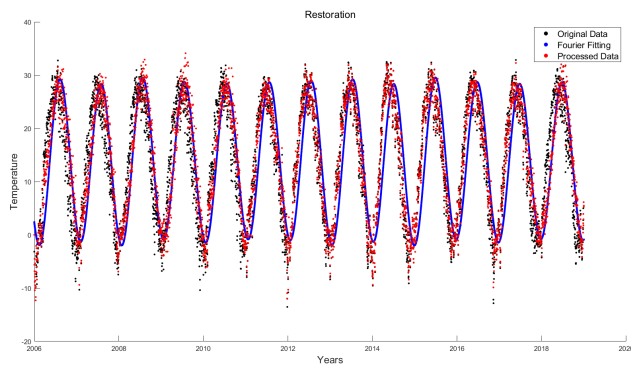


FIGURE 9. Comparison between the original time series before and after denoising and its fourier series.

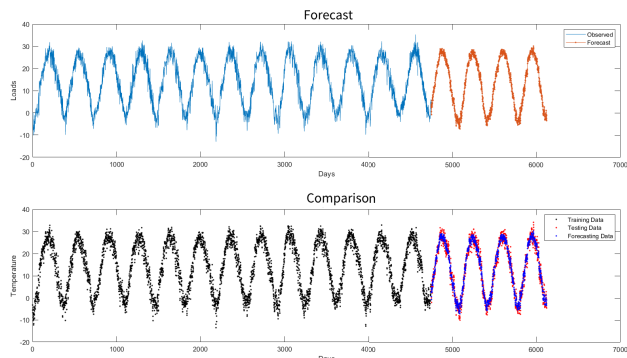


FIGURE 10. Comparison between forecast results and the test set.

study established the parameters as $\alpha = 0.05$, $\beta = 0.05$, and $\gamma = 0.05$. In the SARIMA model, it was postulated that the seasonal cycle adheres to an annual cycle, with a seasonal differentiation value of 1 and $P = 3$, $Q = 3$. The SVR employed the RBF function. Within the LSTM network

architecture, the number of neurons was set at 288, iterating 300 times, with an initial $\alpha = 0.005$. The learning decline period was 125, and the learning rate reduction factor was 0.2. The loss function was based on MAE. By adjusting these parameters, the following graph illustrates the comparative results of different time series prediction algorithms.

From the Figure 11, it can be discerned that all four traditional forecasting methods can aptly predict future cyclical time series. In the first diagram, the Fourier series demonstrates reduced sensitivity to peak and trough values of the cyclical time series. This limitation could potentially stem from the functional expression of the Fourier series. Employing a higher-order Fourier series might mitigate this phenomenon, but there's a risk of overfitting. The Holt-Winters model, depicted in the second diagram, slightly underperforms in capturing the characteristics of the cyclical time series, resulting in a more substantial amplitude compared to the test data. The third and fourth diagrams, particularly the one employing LSTM, offer commendable predictive results, with the LSTM rendering almost perfect fitment of the cyclical time series. Among these methods, LSTM boasts the highest accuracy.

In the Figure 12 above, the red curve (or set of points) represents the actual DAT test set time series data. The blue curve (or set of points) signifies the time series predicted by the model proposed in this paper, having processed 13 years of training set data (from 2006 to 2018) and then forecasted using the LSTM model. The green curve (or set of points) represents the time series directly predicted by the LSTM model after processing the DAT training set data. The right side of the graph showcases magnified portions of the graph based on test set years from 2019 to 2022. It's evident from the visualization that, generally, the blue curve is closer to the red curve compared to the green curve. This suggests

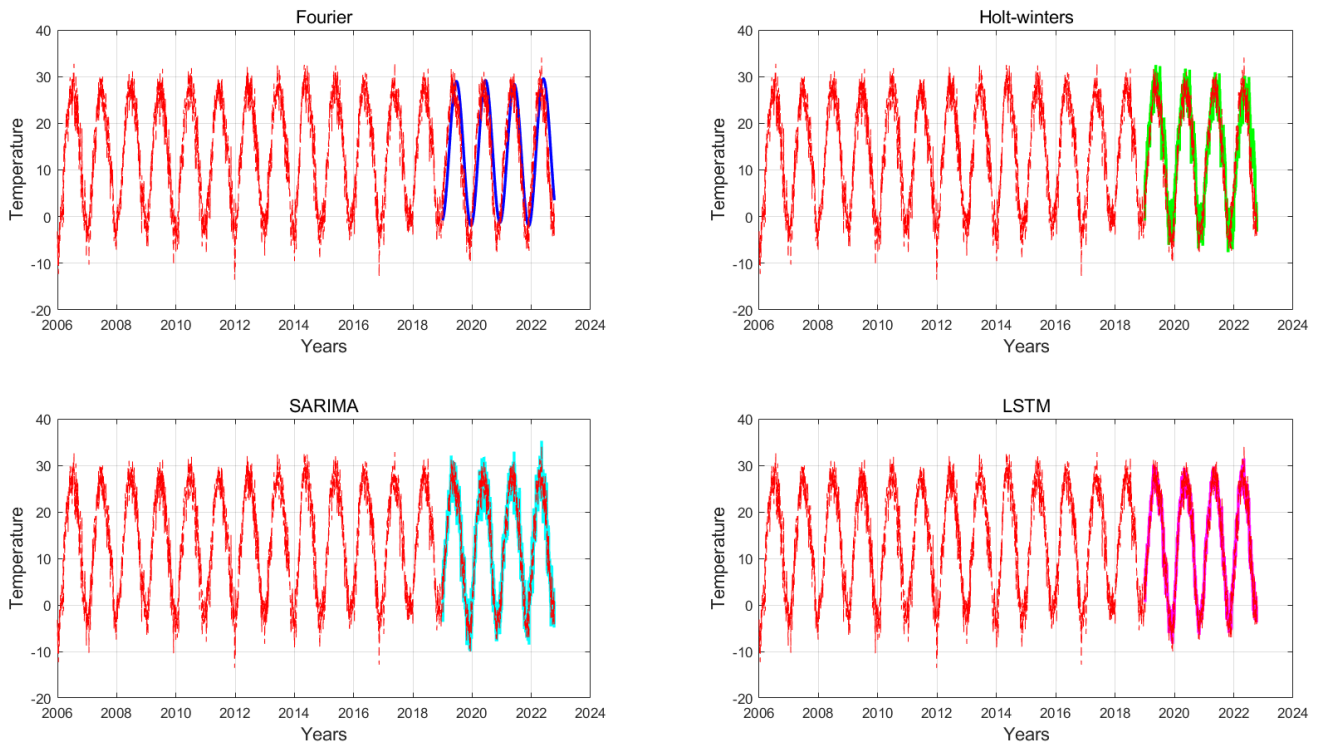


FIGURE 11. Comparison of prediction outcomes among different time series prediction algorithms.

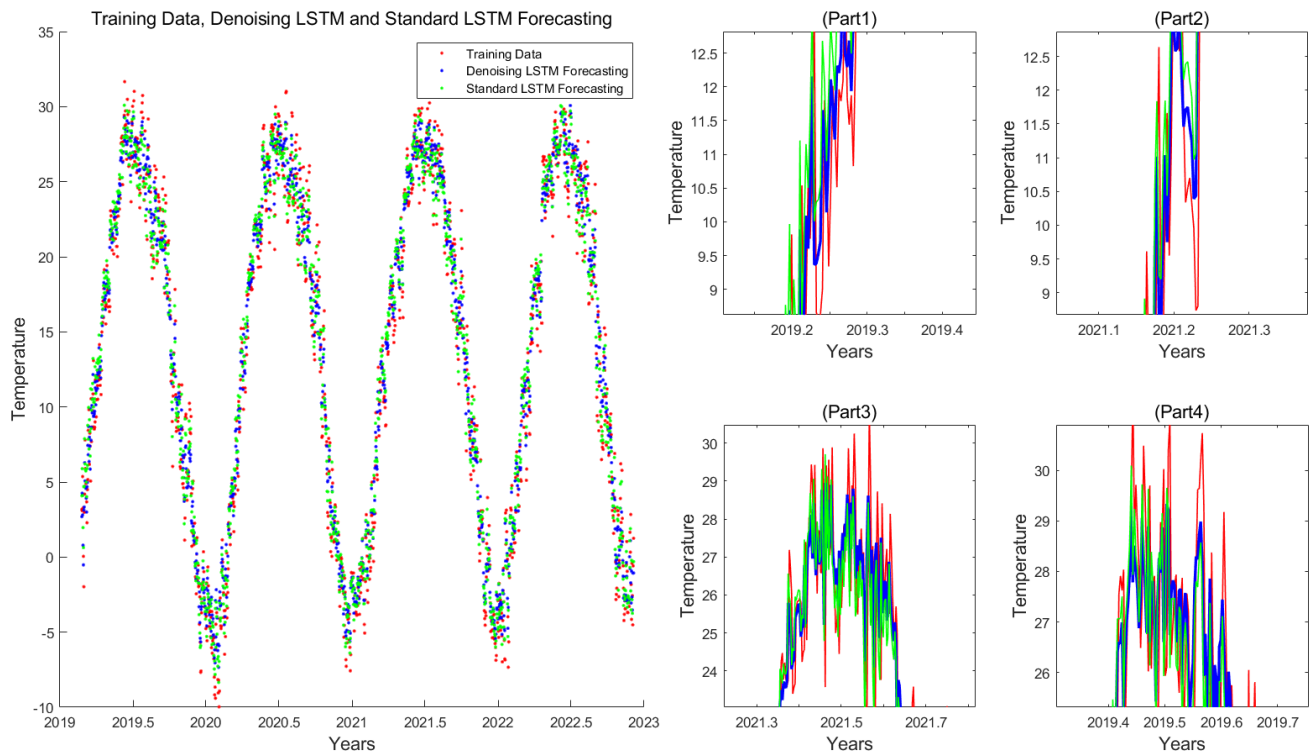


FIGURE 12. In the test set, a comparison between the algorithm presented, the original LSTM model, and the test data.

that the sequence predicted post-processing by our algorithm has a higher precision than the traditional LSTM. However,

there are sporadic regions, particularly those prone to cluster noise, where the predictive performance is marginally

TABLE 4. Comparison of evaluation indicators for different time series forecasting algorithms.

| Methods | MAE | SSE | MSE | RMSE | MAPE |
|-----------------------|---------------|---------------|---------------|---------------|----------------|
| Holt-Winters | 7.3206 | 102226 | 74.0770 | 8.6068 | 296.8287 |
| Fourier | 5.8099 | 61827 | 44.8016 | 6.6934 | 221.9773 |
| ETS | 3.4211 | 24715 | 17.9090 | 4.2319 | 121.7165 |
| SARIMA | 1.9168 | 8433.6 | 6.1113 | 2.4721 | 88.6367 |
| Standard LSTM | 1.7623 | 7172.7 | 5.2052 | 2.2815 | 59.6972 |
| Fourier-LSTM | 1.3672 | 4406.3 | 3.1977 | 1.7882 | 39.1381 |
| DTW-LSTM(Best) | 0.8863 | 1739.1 | 1.2620 | 1.1234 | 35.3521 |
| DTW-LSTM(Worst) | 1.7599 | 7035.2 | 5.1053 | 2.2595 | 54.2889 |

inferior to LSTM. Overall, the accuracy of our proposed algorithm significantly surpasses that of the conventional LSTM network, validating its superior predictive capability in comparison to the traditional LSTM model. Table 4 presents a comparison of the precision of various predictive algorithms.

From the aforementioned results, it is evident that the Fourier series regression, being the initial regression model, requires improvements in its accuracy. One plausible reason might be the Fourier series function's limited capability in fitting time series with significant fluctuations. The SARIMA algorithm demonstrates superior capability in extracting seasonal features, thus outperforming the Holt-Winters algorithm in accuracy. LSTM, representing a deep learning forecasting method, surpasses the precision of the previously mentioned conventional techniques. The algorithm proposed in this study, when juxtaposed with other models mentioned in the experiments, showcases the least error and optimal fitment. This corroborates the model's enhanced efficacy in forecasting cyclical time series datasets, especially those with pronounced fluctuations reminiscent of Fourier series.

B. COMPARISON OF PREDICTION ACCURACY BASED ON TIME WINDOW LENGTH

Observations from the previous experiments suggest that different time window hyperparameters inevitably influence the final prediction outcomes. This can be attributed to the varying data volume acquired from local data during the same iteration, owing to different time window lengths. Considering daily temperature data, which is structured on a daily basis, this study evaluated several parameters for the time window length L . This was done to derive experimental results based on different time window hyperparameters, facilitating an analysis of the influence of the algorithm's time window on prediction outcomes. The stack model parameters for this comparative experiment were set as $w_1 = 0$, $w_2 = 0.225$, and $w_3 = 0.775$. A dual-layer Fourier series sequence stack structure was employed, with clustering thresholds set at both 2 and 10. The experimental results derived from adjusting these hyperparameters are illustrated in the Table 5 below.

As observed from Table 5, under a clustering threshold of 2, setting the time window length between 10 and 30 is advantageous for the final prediction outcome. Subsequently,

TABLE 5. Comparison of evaluation indicators for different time window hyperparameters ($\delta_j^T = 2$).

| L | MAE | SSE | MSE | RMSE | MAPE |
|-----|---------------|---------------|---------------|---------------|----------------|
| 5 | 1.2222 | 2919.0 | 2.1182 | 1.4554 | 47.8380 |
| 10 | 0.9426 | 1995.8 | 1.4484 | 1.2035 | 36.2136 |
| 20 | 0.9637 | 2090.6 | 1.5170 | 1.2317 | 45.1676 |
| 30 | 0.9457 | 1930.7 | 1.4011 | 1.1837 | 34.5615 |
| 40 | 1.4232 | 4349.7 | 3.1567 | 1.7767 | 55.3308 |
| 50 | 1.7580 | 6878.6 | 4.9916 | 2.2342 | 58.4887 |
| 60 | 1.5896 | 5675.5 | 4.1185 | 2.0294 | 52.8714 |
| 90 | 1.3815 | 4331.9 | 3.1435 | 1.7730 | 42.2973 |
| 100 | 1.2109 | 3260.2 | 2.3660 | 1.5382 | 40.3397 |
| 200 | 1.0634 | 2495.9 | 1.8112 | 1.3458 | 35.6176 |
| 300 | 1.1176 | 2811.9 | 2.0406 | 1.4285 | 36.1650 |
| 365 | 1.1304 | 2887.1 | 2.0953 | 1.4475 | 38.1584 |

TABLE 6. Comparison of evaluation indicators for different time window hyperparameters ($\delta_j^T = 10$).

| L | MAE | SSE | MSE | RMSE | MAPE |
|-----------|---------------|---------------|---------------|---------------|----------------|
| 5 | 1.6120 | 6151.4 | 4.4639 | 2.1128 | 53.7358 |
| 10 | 1.2467 | 3617.3 | 2.6250 | 1.6202 | 53.1846 |
| 20 | 1.2372 | 3450.6 | 2.5039 | 1.5824 | 47.6782 |
| 30 | 1.0343 | 2387.2 | 1.7324 | 1.3162 | 39.9951 |
| 40 | 1.0738 | 2681.6 | 1.9460 | 1.3950 | 47.2812 |
| 50 | 1.2585 | 3554.3 | 2.5792 | 1.6060 | 46.1925 |
| 60 | 1.2862 | 3713.3 | 2.6945 | 1.6415 | 47.4101 |
| 90 | 1.6012 | 5689.4 | 4.1286 | 2.0319 | 45.7276 |
| 100 | 1.7599 | 7035.2 | 5.1053 | 2.2595 | 54.2889 |
| 200 | 0.9731 | 2066.5 | 1.4996 | 1.2246 | 37.0096 |
| 300 | 1.1176 | 2811.9 | 2.0406 | 1.4285 | 36.1650 |
| 365 | 1.4050 | 4569.5 | 3.3160 | 1.8210 | 45.3307 |

the prediction error for various categories initially increases before decreasing.

Table 6 indicates that when the clustering threshold is set at 10, a time window length of 30 yields the most favorable prediction results. The subsequent prediction errors exhibit a pattern of initial increase followed by a decrease, similar to the trend observed in Table 6. Furthermore, prediction results corresponding to a clustering threshold of 10 generally underperform when compared to those with a clustering threshold of 2. The aforementioned data is graphically represented in the subsequent figures.

The initial segments of the graphical representation reveal significant oscillations in the error curve. The error typically peaks when the time window parameter ranges between 50 and 100. However, as the length of the time window progressively increases, there's a discernible decline in error. This prompts a hypothesis: shorter time windows might struggle in extracting valuable features. While certain lengths of time windows might effectively capture data characteristics, others may falter. As the time window expands, its capability to seize data features wanes, culminating in the error peaking when the time window size approximates 100. Considering the gradual extension of the scope captured by the time window as its length increases, the error tends to diminish as the time window continues to expand. Moreover, for this particular dataset, setting the time window length

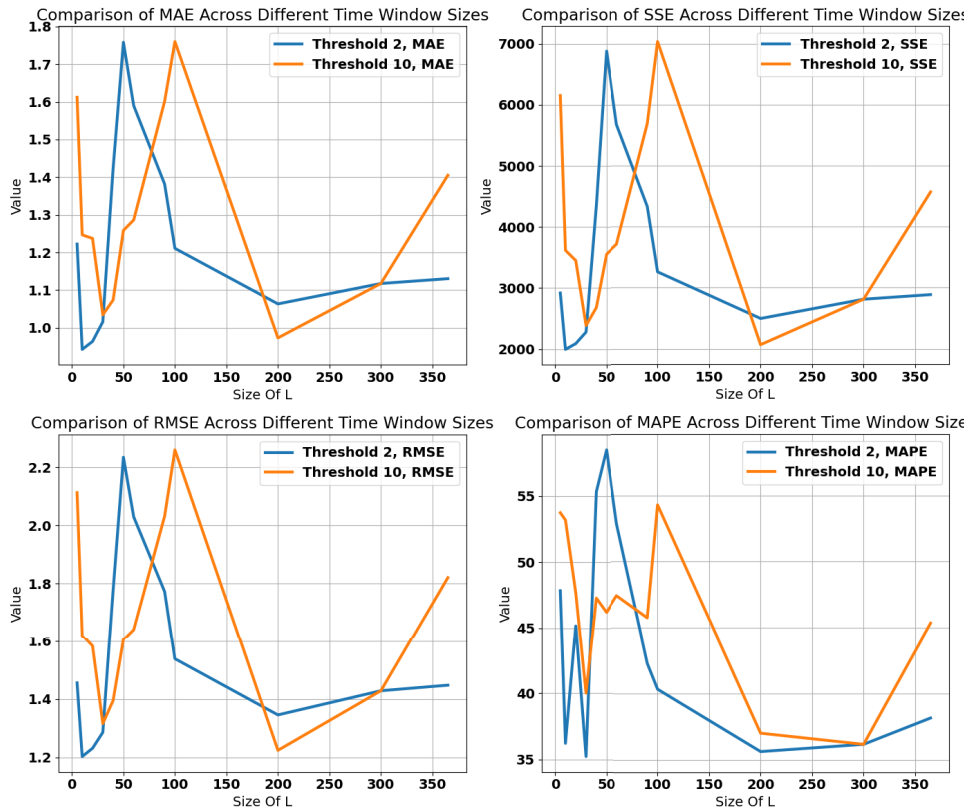


FIGURE 13. Impact of different time window lengths on time series prediction errors.

at 30 appears optimal, leveraging it as a hyperparameter results in commendable prediction outcomes and heightened accuracy. Additionally, as inferred from Figure 13, the overall shift to the right of the graphical representation with an increase in the clustering threshold suggests a potential need to concurrently enlarge the clustering threshold and time window length to enhance prediction outcomes. Furthermore, generally speaking, prediction outcomes corresponding to a clustering threshold of 2 surpass those associated with a threshold of 10.

C. COMPARISON OF PREDICTION ACCURACY BY CLUSTERING THRESHOLD

The threshold of the clustering model, as a hyperparameter, plays a pivotal role in influencing the prediction results. Given that the threshold represents the Euclidean distance between cluster centroids and the boundaries of the state space, the study adjusted this value to analyze its impact on prediction outcomes. As the clustering threshold decreases, the state space centered around the cluster centroid continuously contracts. Newly added state points are increasingly likely to be deemed as exhibiting cluster-based anomalous noise, leading to the rectification of the local sequence of the added time window, thus amplifying the efficacy of our algorithm. For this comparative experiment, the stack model parameters were set as $w_1 = 0$, $w_2 = 0.225$, and $w_3 = 0.775$, employing a two-layer Fourier series sequence stack structure

TABLE 7. Different clustering thresholds with a time window length of 10.

| $\delta\tau_i$ | MAE | SSE | MSE | RMSE | MAPE |
|----------------|---------------|---------------|---------------|---------------|----------------|
| 0 | 1.1445 | 2949.1 | 2.14 | 1.4629 | 37.2488 |
| 0.5 | 1.3784 | 4321.9 | 3.1364 | 1.771 | 43.6483 |
| 1 | 0.9722 | 2086.2 | 1.5138 | 1.2304 | 34.8542 |
| 1.5 | 0.9616 | 2073.6 | 1.5047 | 1.2267 | 36.1322 |
| 2 | 0.9426 | 1995.8 | 1.4484 | 1.2035 | 36.2136 |
| 2.5 | 1.222 | 3508.5 | 2.5459 | 1.5956 | 52.4161 |
| 3 | 1.2768 | 3885.2 | 2.8194 | 1.6791 | 56.736 |
| 4 | 1.2799 | 3792.3 | 2.7519 | 1.6589 | 57.8412 |
| 5 | 1.3165 | 4164.9 | 3.0224 | 1.7385 | 56.9341 |
| 10 | 1.2467 | 3617.3 | 2.625 | 1.6202 | 53.1846 |
| 15 | 1.4946 | 5118.4 | 3.7145 | 1.9273 | 59.142 |
| 20 | 1.5834 | 5705.4 | 4.1404 | 2.0348 | 62.0224 |
| 25 | 1.6493 | 6244.5 | 4.5313 | 2.1287 | 61.4095 |
| 30 | 1.6744 | 6478.1 | 4.701 | 2.1682 | 55.5864 |

with time window lengths of 10 and 30. The results derived from adjusting the clustering threshold hyperparameter are displayed in the subsequent Tables 7 and 8.

Table 7 reveals that for a time window length of 10, setting the clustering threshold between 1 and 2 yields optimal prediction outcomes. Thereafter, the prediction error consistently escalates with an increasing clustering threshold, implying that a larger threshold results in diminished prediction accuracy.

Table 8 indicates that for a time window length of 30, a clustering threshold ranging between 1.5 and 5 is more suitable. Specifically, when the clustering threshold

TABLE 8. Different clustering thresholds with a time window length of 30.

| δ_i^T | MAE | SSE | MSE | RMSE | MAPE |
|--------------|---------------|---------------|---------------|---------------|----------------|
| 0 | 1.2881 | 3702.6 | 2.687 | 1.6392 | 41.8974 |
| 0.5 | 1.2497 | 3474.2 | 2.5211 | 1.5878 | 40.0801 |
| 1 | 1.4175 | 4508.9 | 3.2721 | 1.8089 | 50.2226 |
| 1.5 | 1.0501 | 2458.6 | 1.7841 | 1.3357 | 37.0084 |
| 2 | 0.9457 | 1930.7 | 1.4011 | 1.1837 | 34.5615 |
| 2.5 | 1.0952 | 2665.7 | 1.9346 | 1.3909 | 42.1709 |
| 3 | 0.8863 | 1739.1 | 1.2620 | 1.1234 | 35.3521 |
| 4 | 1.1583 | 2930.3 | 2.1266 | 1.4583 | 43.1357 |
| 5 | 0.9256 | 1888.0 | 1.3701 | 1.1705 | 35.2053 |
| 10 | 1.0343 | 2387.2 | 1.7324 | 1.3162 | 39.9951 |
| 15 | 1.3142 | 4006.3 | 2.9037 | 1.7051 | 58.9708 |
| 20 | 1.6017 | 5877.6 | 4.2655 | 2.0653 | 61.6399 |
| 25 | 1.6268 | 6167.5 | 4.4758 | 2.1156 | 61.914 |
| 30 | 1.6524 | 6278.1 | 4.556 | 2.1345 | 56.6071 |

is relatively low, the impact of the clustering threshold on prediction accuracy under longer time window lengths exhibits more pronounced oscillations compared to shorter lengths. Furthermore, the optimal clustering threshold range expands in tandem with the increase in time window length, corroborating observations from the previous comparative experiments. The relationships between these hyperparameters and errors are graphically represented in the following figures.

As illustrated in the above Figure 14, overall, an augmented threshold elevates the prediction error and curtails accuracy. Initially, the escalation in threshold leads to considerable fluctuations in the error curve. Given that a smaller threshold strengthens the constraints of the clustering model, thereby enhancing prediction accuracy, it's rational to assert that the clustering model positively impacts the overall time series prediction outcomes. Furthermore, based on data observations, the most favorable prediction outcomes emerge when the clustering model's threshold is set between 1 and 3, optimizing the precision of local data modifications. It should be noted that the size of the window or the choice of threshold is not fixed but should be appropriately adjusted based on the length, amplitude, frequency, and other data characteristics of the time series. In addition, if the initial clustering threshold hyperparameter is set too large, the tolerance for data during the denoising window sliding process will be higher, and the likelihood of modifying the data will be lower. A larger time window means that more local data information can be encompassed at the same time, and the high-dimensional features mapped by the time window between two adjacent time steps should be more similar, which will promote the clustering effect of the data to the center to a certain extent. Therefore, these hyperparameters can be effectively adjusted according to appropriate needs.

D. COMPARISON OF PREDICTION ACCURACY BASED ON DATA STACK WEIGHTS AND STRUCTURES

The data augmentation aspect of the algorithm is controlled by modifying the proportion of mixed weights, which influences the state of the data in the combined target

TABLE 9. Comparison of evaluation indicators under different weights in the dual-layer fourier structure.

| w_1 | w_2 | w_3 | MAE | SSE | MSE | RMSE | MAPE |
|-------|-------|-------|---------------|---------------|---------------|---------------|----------------|
| 0.8 | 0.1 | 0.1 | 1.0865 | 2606.8 | 1.8917 | 1.3754 | 31.5192 |
| 0.5 | 0.3 | 0.2 | 0.9943 | 2174.3 | 1.5778 | 1.2561 | 31.8716 |
| 0.5 | 0.2 | 0.3 | 1.0046 | 2221.3 | 1.6119 | 1.2696 | 37.9679 |
| 0.1 | 0.8 | 0.1 | 1.0008 | 2230.0 | 1.6182 | 1.2721 | 29.2458 |
| 0.3 | 0.5 | 0.2 | 1.0464 | 2410.7 | 1.7495 | 1.3227 | 33.2457 |
| 0.2 | 0.5 | 0.3 | 0.9506 | 1970.8 | 1.4313 | 1.1959 | 33.9627 |
| 0.1 | 0.1 | 0.8 | 0.9282 | 1931.3 | 1.4016 | 1.1839 | 40.4514 |
| 0.3 | 0.2 | 0.5 | 1.2090 | 3305.6 | 2.3988 | 1.5488 | 61.8287 |
| 0.2 | 0.3 | 0.5 | 0.9951 | 2167.3 | 1.5728 | 1.2541 | 42.8958 |
| 0.0 | 0.2 | 0.8 | 1.1143 | 2761.4 | 2.0039 | 1.4156 | 31.7667 |
| 0.0 | 0.5 | 0.5 | 1.0393 | 2376.4 | 1.7345 | 1.3132 | 35.8356 |
| 0.0 | 0.8 | 0.2 | 0.9085 | 1834.2 | 1.3310 | 1.1537 | 34.7408 |
| 0.2 | 0.0 | 0.8 | 0.9746 | 2077.1 | 1.5072 | 1.2277 | 34.9034 |
| 0.5 | 0.0 | 0.5 | 1.1860 | 3147.1 | 2.2837 | 1.5112 | 41.7328 |
| 0.8 | 0.0 | 0.2 | 0.9838 | 2117.3 | 1.5366 | 1.2396 | 48.0247 |
| 0.2 | 0.8 | 0.0 | 0.9243 | 1887.3 | 1.3696 | 1.1703 | 35.9736 |
| 0.5 | 0.5 | 0.0 | 1.0940 | 2633.5 | 1.9110 | 1.3824 | 44.5111 |
| 0.8 | 0.2 | 0.0 | 1.0506 | 2445.1 | 1.7745 | 1.3321 | 31.3137 |

sequence. This method of modification does not alter the overall data stacking mode of the algorithm but is a tuning approach realized by adjusting hyperparameters. This study presents two sets of weight hyperparameter experiments for comparison: one based on a two-layer Fourier structure and the other on a two-layer original sequence structure. On one hand, these experiments can reflect the impact of different data stacking offset directions on prediction results. On the other hand, they demonstrate the effects of various weight array settings on prediction outcomes. The weight array must satisfy the condition of the equation $\sum_{i=1}^3 w_i = 1$. Where n represents the number of sequences to be mixed. Our approach distinguishes the sequence into three types for weighting. Further innovations, such as the adoption of diversified sequence weighting formulas, can also be explored.

Table 9 suggests that under the two-layer Fourier structure data stacking model, setting the weight vectors as (0.1,0.1,0.8), (0,0.8,0.2), (0.2,0.5,0.3), and (0.2,0.8,0) yields favorable prediction results.

Table 10 indicates that setting the weight vector as (0.1,0.1,0.8) provides optimal prediction results, and the prediction outcomes of the two-layer Fourier structure are notably superior to those of the two-layer original data structure. The data from the tables are sorted according to the error reflected in the second table, from high to low, for each triplet. Subsequently, curves from both tables are plotted under the same triplet to highlight the predictive differences resulting from different stacking structures.

The graph reveals that for each triplet weight, the prediction error generated using a two-layer original sequence structure consistently exceeds that of the two-layer Fourier structure, albeit with a few exceptions. This suggests that the dataset is more amenable to data stacking using the two-layer Fourier structure.

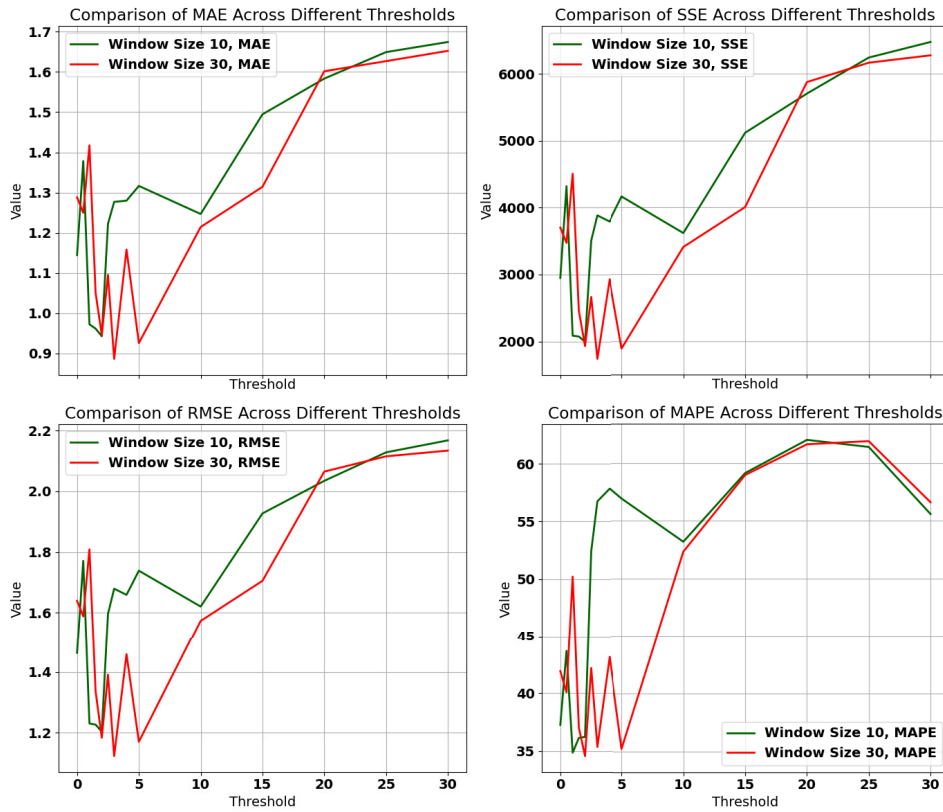


FIGURE 14. Impact of different clustering thresholds on time series prediction errors.

TABLE 10. Comparison of evaluation indicators under different weights in the dual-layer original sequence structure.

| w_1 | w_2 | w_3 | MAE | SSE | MSE | RMSE | MAPE |
|-------|-------|-------|---------------|---------------|---------------|---------------|----------------|
| 0.8 | 0.1 | 0.1 | 1.6958 | 6676.7 | 4.8453 | 2.2012 | 58.6482 |
| 0.5 | 0.3 | 0.2 | 1.6869 | 6506.7 | 4.7219 | 2.1730 | 60.3698 |
| 0.5 | 0.2 | 0.3 | 1.6842 | 6475.0 | 4.6989 | 2.1677 | 60.6270 |
| 0.1 | 0.8 | 0.1 | 0.9741 | 2096.9 | 1.5218 | 1.2336 | 37.5656 |
| 0.3 | 0.5 | 0.2 | 1.6241 | 6016.0 | 4.3656 | 2.0894 | 58.3320 |
| 0.2 | 0.5 | 0.3 | 1.0395 | 2443.2 | 1.7729 | 1.3315 | 37.5446 |
| 0.1 | 0.1 | 0.8 | 0.9620 | 2045.9 | 1.4847 | 1.2185 | 34.9190 |
| 0.3 | 0.2 | 0.5 | 1.6531 | 6234.6 | 4.5246 | 2.1271 | 59.9430 |
| 0.2 | 0.3 | 0.5 | 1.0334 | 2409.4 | 1.7485 | 1.3223 | 37.5432 |
| 0.0 | 0.2 | 0.8 | 1.1462 | 2934.9 | 2.1298 | 1.4594 | 33.1404 |
| 0.0 | 0.5 | 0.5 | 1.1347 | 2855.0 | 2.0719 | 1.4394 | 31.4285 |
| 0.0 | 0.8 | 0.2 | 1.0125 | 2276.5 | 1.6520 | 1.2853 | 37.5599 |
| 0.2 | 0.0 | 0.8 | 1.0935 | 2692.8 | 1.9541 | 1.3979 | 40.1541 |
| 0.5 | 0.0 | 0.5 | 1.6822 | 6488.4 | 4.7085 | 2.1699 | 56.8294 |
| 0.8 | 0.0 | 0.2 | 1.7221 | 6753.4 | 4.9009 | 2.2138 | 60.5001 |
| 0.2 | 0.8 | 0.0 | 1.1612 | 3070.2 | 2.2282 | 1.4927 | 40.7702 |
| 0.5 | 0.5 | 0.0 | 1.7058 | 6557.4 | 4.7585 | 2.1814 | 61.3314 |
| 0.8 | 0.2 | 0.0 | 1.6831 | 6471.7 | 4.6963 | 2.1671 | 58.0684 |

E. EXPERIMENT WITH SOTA

In addition to making predictions on the Beijing temperature dataset, this study also compares predictive accuracy with the current state-of-the-art (SOTA) in the field of time series forecasting across multiple datasets. A total of four SOTA methods were selected for comparison, along with

datasets related to tourism, sales, and food production to validate the accuracy of our algorithm. These datasets have different statistical properties due to inconsistent sources. The baselines for these four SOTA methods are all open source. The links to the datasets and baselines are shown in the table below. Initially, this study employs data science techniques to derive statistical characteristics of these datasets, followed by solving for their respective most fitting curves. Subsequently, both the original and fitted series are input into our program, with the original series also input into the SOTA programs. Finally, error analysis is conducted based on the five algorithms (four SOTA and our proposed algorithm) on the test set to obtain the final algorithm performance results.

The three datasets include predictions for real-world island tourist numbers, sales data for a company, and information on lemon harvests produced in Delhi from 2002 to 2021. The univariate time series images for these three types are shown below 16 and are all available on the Kaggle public dataset. It is noteworthy that this study only preprocesses non-zero values of the data. The reason for not conducting extensive preprocessing work is that the selected datasets are univariate time series, which can be directly imported and solved. To evaluate the differences in algorithm prediction effects, this study validates in three stages. Firstly, error analysis is conducted numerically based on common error metrics such as MAE. Subsequently, further validation is conducted

Comparison of Different Metrics for Tables 1 & 2

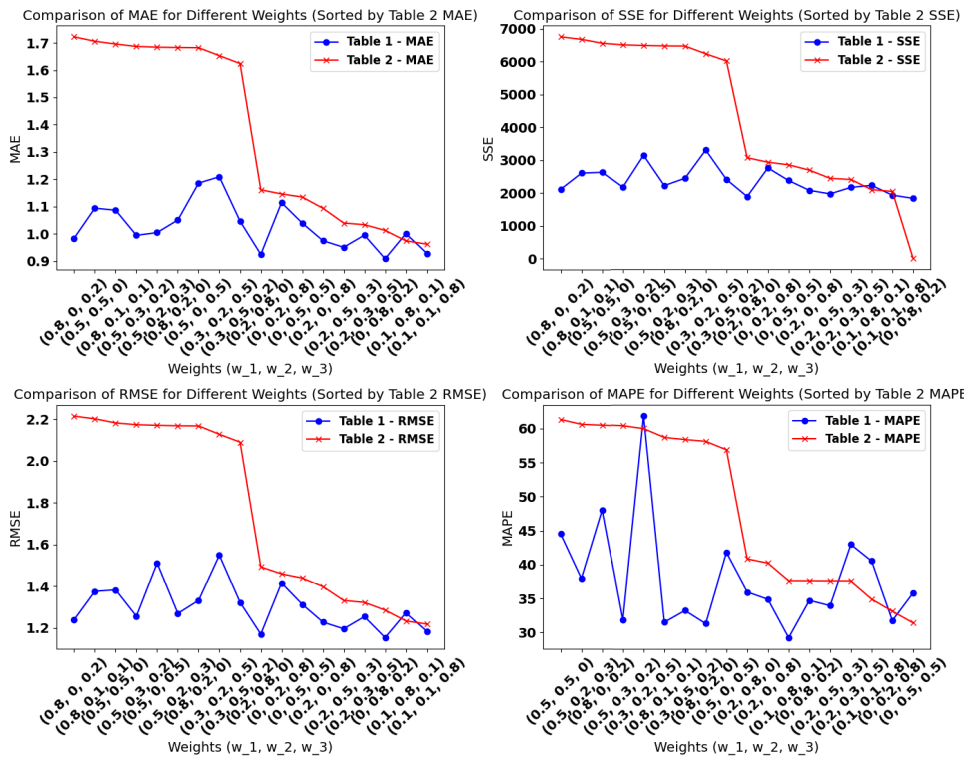


FIGURE 15. Influence of different data stacking methods and weight combinations on time series prediction error.

TABLE 11. Datasets and baseline names and links.

| Name | URL |
|---------------------------|---|
| Tourist Numbers | https://www.kaggle.com/datasets/ayushnitb/tourist-numbers-univariate-forecasting-dataset |
| Sales Data | https://www.kaggle.com/datasets/vikramamin/holt-winters-forecasting-for-sales-data |
| Lemon Production in Delhi | https://www.kaggle.com/datasets/tharakinfinitly/lemon-production-in-delhi |
| Auto-period | https://github.com/akofke/autopperiod |
| Findfrequency | https://rdr.io/cran/forecast/man/findfrequency.html |
| MSTL | https://github.com/KishManani/MSTL |
| TBATS | https://rdr.io/cran/forecast/man/tbats.html |

using the Mann-Whitney U Test and the Permutation Test to verify that our algorithm indeed has certain performance improvements over traditional SOTA methods in terms of predictive performance. Below are the error comparisons between our algorithm and SOTA algorithms, as obtained through program processing. This study have selected “Auto-Period”, “FindFrequency”, “MSTL” and “TBATS” as the comparad SOTA methods, which are representative of clustering, denoising or prediction methods. Each experiment in this study used 1000 epochs and 1440 neurons. In order to enrich the parameter selection for this study, the specific parameters used in the experiments are given in the following table 12.

The comparison of prediction effects on the test set shows that, except for a significant gap in MAPE on the lemon production dataset compared to the best-performing MSTL

TABLE 12. Parameters of datasets.

| | L | δ_T^* | Training Size | Training Proportion |
|---------|-----|-----------------|---------------|---------------------|
| TourNum | 10 | 10^8 | 252 | 7/8 |
| Sale | 3 | 2×10^8 | 42 | 7/8 |
| Lemon | 5 | 2×10^6 | 200 | 5/6 |

TABLE 13. Error comparison for tourism data.

| | MAE | SSE | RMSE | MAPE |
|---------------|--------------|-------------------------|--------------|----------------|
| Auto-Period | 10975 | 6.3875×10^9 | 13782 | 30.0324 |
| FindFrequency | 18298 | 2.1377×10^{10} | 24368 | 44.9483 |
| MSTL | 36543 | 4.958×10^{10} | 37111 | 119.8206 |
| TBATS | 12833 | 1.0598×10^{10} | 17447 | 32.0364 |
| Ours | 10312 | 6.5605×10^9 | 13499 | 27.4273 |

method, our method achieved SOTA performance on almost all metrics across all datasets. In the tourism dataset, our

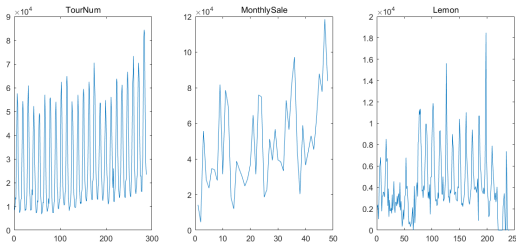


FIGURE 16. Public dataset with SOTA test.

TABLE 14. Error comparison for sales data.

| | MAE | SSE | RMSE | MAPE |
|---------------|--------------|--|--------------|----------------|
| Auto-Period | 20072 | 4.1595×10^9 | 26330 | 21.3455 |
| FindFrequency | 33989 | 8.8927×10^9 | 38498 | 39.4973 |
| MSTL | 78752 | 4.016×10^{10} | 81813 | 102.1755 |
| TBATS | 17712 | 3.7142×10^9 | 24880 | 21.1221 |
| Ours | 17089 | 2.3084×10^9 | 21487 | 19.6786 |

TABLE 15. Error comparison for production data.

| | MAE | SSE | RMSE | MAPE |
|---------------|------------|--|---------------|--|
| Auto-Period | 2147.9 | 3.5657×10^8 | 2985.7 | 1.5405×10^5 |
| FindFrequency | 2495.1 | 3.3828×10^8 | 2908.1 | 1.4602×10^5 |
| MSTL | 2192 | 3.6064×10^8 | 3002.7 | 2.6797×10^4 |
| TBATS | 6230.1 | 3.4372×10^9 | 9269.9 | 3.3186×10^5 |
| Ours | 993 | 4.6163×10^7 | 1074.3 | 4.0648×10^4 |

method also did not deviate significantly from Auto-Period in terms of the SSE metric. Given that these three datasets were fitted using Fourier series, the experiments demonstrate that these datasets do not follow a normal distribution, thus allowing the use of Mann-Whitney U Test and Permutation Test methods for verification. The purpose of these tests is to validate whether the prediction sequences generated by the study can produce significant differences compared to the prediction sequences from traditional SOTA methods. The following tables show the comparison between the prediction sequences from traditional SOTA methods and those generated by our proposed algorithm, with each table corresponding to a different dataset. On the left side of each table are the Mann-Whitney U Test results, and on the right are the Permutation Test results.

This study employs a hypothesis testing threshold of $P < 0.1$, meaning that when $P < 0.1$, the null hypothesis is considered true, indicating a significant difference between the two data sets. This, in turn, proves that the prediction sequences generated by the study’s method are significantly different from those produced by traditional SOTA methods. According to the calculations, the prediction sequences obtained through our algorithm on various datasets almost all show significant statistical differences compared to traditional SOTA methods, further validating the innovativeness of our algorithm in the field of univariate time series forecasting.

F. ABLATION EXPERIMENT

Ablation experiments are conducted by reducing the optimization steps related to the proposed algorithm based on the

original experiments. This approach aims to demonstrate the accuracy and necessity of each step designed in the algorithm. Three ablation experiments are selected: excluding the stacking model, excluding the clustering model, and excluding the repair model. In specific ablation tests:

The control group that excludes the stacking model only uses the original sequence, denoted as y_{Origin} and y_{Target} for correction after a local sequence in the time window is judged by the clustering model as needing repairs, without further data augmentation. The control group that excludes the repair model simply performs a simple weighted stack of the original data and Fourier series data and returns the weighted result when repairs are needed. The control group that excludes the clustering model sequentially inputs the entire sequence into the data stacking model and the repair model. Ultimately, the entire time sequence will be reorganized and repaired. For the ablation study, the procedures are as follows:

The control group excluding the stacking model, upon identifying local sequence anomalies via the clustering model, simply repairs the original sequence once. In this process, the new sequence continuously seeks optimization between the original and fitted sequences, returning the final new sequence to the local data. This procedure does not involve data stack levels and depths or cross-sequence mixing. The control group excluding the repair model, upon identifying local sequence anomalies via the clustering model, merely conducts a single cross-mixing process on the original and fitted sequences. In this cross-mixing process, only the average values of the two sequences at each time step are taken and returned to the local sequence without involving multi-objective optimization. The control group excluding the clustering model undergoes stacking and repair processing regardless of the state of the time window data. Ultimately, all sequence positions are sequences processed after stacking and repair. The parameters are set as $w_1 = 0$, $w_2 = 0.225$, $w_3 = 0.775$ adopting a two-layer Fourier series sequence stack structure, with a time window length of 30 and a clustering threshold of 2. Based on these standards, the following experimental results were obtained in the Tables among 19 to 21.

Numerous other ablation experiments were also conducted. Results show that the removal of each method in the ablation experiments consistently results in a decrease in accuracy. Without the stacking model, the data combination is relatively singular, causing the multi-objective optimization in the repair model to perform poorly. Without the repair model, the weighting method of the data stacking model is not suitable for changes at every time step, and the weighting weights cannot be adjusted in time, leading to a relatively singular stacking effect. Without the clustering model, the algorithm blindly repairs the original sequence, potentially destroying the data features of the original time sequence, which is not conducive to the LSTM model’s capture and training process. Among the three sub-models, the removal of the repair model has the most significant impact on

TABLE 16. Mann-whitney u test and permutation test comparison in tourism dataset.

| Method | (M-W) P-value | (M-W) Significance | (PT) P-value | (PT) Significance |
|---------------|---------------|--------------------|--------------|-------------------|
| Auto-Period | 0.086 | 1 | 0.110 | 0 |
| FindFrequency | 0.216 | 0 | 0.025 | 1 |
| MSTL | 0 | 1 | 0 | 1 |
| TBATS | 0.033 | 1 | 0.126 | 0 |

TABLE 17. Mann-whitney u test and permutation test comparison in sales dataset.

| Method | (M-W) P-value | (M-W) Significance | (PT) P-value | (PT) Significance |
|---------------|---------------|--------------------|--------------|-------------------|
| Auto-Period | 0.429 | 0 | 0.29 | 0 |
| FindFrequency | 0.125 | 0 | 0.021 | 1 |
| MSTL | 0.004 | 1 | 0 | 1 |
| TBATS | 0.930 | 0 | 0.974 | 0 |

TABLE 18. Mann-whitney u test and permutation test comparison in production dataset.

| Method | (M-W) P-value | (M-W) Significance | (PT) P-value | (PT) Significance |
|---------------|---------------|--------------------|--------------|-------------------|
| Auto-Period | 0.007 | 1 | 0.009 | 1 |
| FindFrequency | 0 | 1 | 0 | 1 |
| MSTL | 0 | 1 | 0 | 1 |
| TBATS | 0.607 | 0 | 0.010 | 1 |

TABLE 19. Comparison of evaluation indicators in ablation experiments (L = 30, δ_i^z = 2).

| Excluding Method | MAE | SSE | MSE | RMSE | MAPE |
|------------------|---------------|---------------|---------------|---------------|----------------|
| Unchanged | 0.9457 | 1930.7 | 1.4011 | 1.1837 | 34.5615 |
| Stack Model | 1.0508 | 2441.7 | 1.7718 | 1.3311 | 40.3167 |
| Repair Model | 1.6314 | 6177.4 | 4.4830 | 2.1173 | 55.2073 |
| Cluster Model | 1.3657 | 4210.7 | 3.0559 | 1.7481 | 43.8781 |

TABLE 20. Comparison of evaluation indicators in ablation experiments (L = 10, δ_i^z = 1).

| Excluding Method | MAE | SSE | MSE | RMSE | MAPE |
|------------------|---------------|---------------|---------------|---------------|----------------|
| Unchanged | 0.9722 | 2086.2 | 1.5138 | 1.2304 | 34.8542 |
| Stack Model | 1.3784 | 4502.3 | 3.2674 | 1.8076 | 63.0860 |
| Repair Model | 1.6219 | 6011.6 | 4.3627 | 2.0887 | 59.2975 |
| Cluster Model | 1.3540 | 4058.2 | 2.9450 | 1.7161 | 45.7916 |

TABLE 21. Comparison of evaluation indicators in ablation experiments (L = 45, δ_i^z = 15).

| Excluding Method | MAE | SSE | MSE | RMSE | MAPE |
|------------------|---------------|---------------|---------------|---------------|----------------|
| Unchanged | 1.1549 | 3164.3 | 2.2964 | 1.5154 | 53.2029 |
| Stack Model | 1.2315 | 3639.8 | 2.6413 | 1.6252 | 59.6414 |
| Repair Model | 1.6924 | 6525.7 | 4.7534 | 2.1761 | 60.1964 |
| Cluster Model | 1.3843 | 4337.6 | 3.1478 | 1.7742 | 43.7260 |

the predictive accuracy of the algorithm, followed by the clustering model and then the data stacking model.

VI. CONCLUSION AND FUTURE WORK

To enhance the predictive accuracy of time series forecasting methods, this paper introduces an optimized LSTM-based time series prediction algorithm, which addresses the capture and handling of cluster-shaped anomalous noise in the time series. The algorithm is designed to effectively capture and

rectify cluster-shaped anomalous noise, thereby continuously improving the overall characteristics of the time series. This enhances the LSTM model’s ability to learn from data features and consequently elevates the accuracy of time series predictions. Experimental investigations were conducted on surface temperature data from Beijing spanning from 2006 to 2022. The results demonstrate that the predictive algorithm, which integrates denoising and optimized time windows into the LSTM model, outperforms traditional mathematical and machine learning forecasting methods. this algorithm is simultaneously compared with four SOTA algorithms on three publicly available datasets for prediction error comparison and statistical testing, which signifies its substantial potential in the realm of time series forecasting.

Considering the comparative experiments in this study, which utilize a controlled variable method to explore the impact of hyperparameters on time series forecasting, the underlying mechanisms of various hyperparameters in the algorithm’s performance require further exploration. Moreover, for the design of hyperparameters, nonlinear programming could be incorporated to facilitate a global search for optimal solutions. Concerning the design of the clustering model, the introduction of weight vectors and alternative geometric distance calculation methods could enhance the computation of centroid distances within the state space boundaries. Additionally, the time window length and clustering threshold can be dynamically adjusted to adapt to changing time series patterns. These areas of study are slated as the focus for future investigations.

REFERENCES

[1] P. Hewage, A. Behera, M. Trovati, E. Pereira, M. Ghahremani, F. Palmieri, and Y. Liu, “Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station,” *Soft Comput.*, vol. 24, no. 21, pp. 16453–16482, Nov. 2020.

- [2] S. Guo, Y. Wen, X. Zhang, and H. Chen, "Runoff prediction of lower yellow river based on CEEMDAN-LSSVM-GM(1,1) model," *Sci. Rep.*, vol. 13, no. 1, p. 1511, Jan. 2023.
- [3] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*.
- [4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
- [5] H. Cheng, P.-N. Tan, C. Potter, and S. Klooster, "Detection and characterization of anomalies in multivariate time series," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2009, pp. 413–424.
- [6] X. Gu, Y. Guo, F. Deligianni, B. Lo, and S.-Z. Yang, "Cross-subject and cross-modal transfer for generalized abnormal gait pattern recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 546–560, Feb. 2021.
- [7] M. Wang, S. Wang, X. Wang, Y. Tian, Y. Wu, Y. Cao, J. Song, T. Wu, and Y. Hu, "The association between PM_{2.5} exposure and daily outpatient visits for allergic rhinitis: Evidence from a seriously air-polluted environment," *Int. J. Biometeorol.*, vol. 64, no. 1, pp. 139–144, Jan. 2020.
- [8] A. L. Schaffer, T. A. Dobbins, and S.-A. Pearson, "Interrupted time series analysis using autoregressive integrated moving average (ARIMA) models: A guide for evaluating large-scale health interventions," *BMC Med. Res. Methodol.*, vol. 21, no. 1, pp. 1–12, Dec. 2021.
- [9] U. M. Sirisha, M. C. Belavagi, and G. Attigeri, "Profit prediction using ARIMA, SARIMA and LSTM models in time series forecasting: A comparison," *IEEE Access*, vol. 10, pp. 124715–124727, 2022.
- [10] R. Kumar, P. Kumar, and Y. Kumar, "Multi-step time series analysis and forecasting strategy using ARIMA and evolutionary algorithms," *Int. J. Inf. Technol.*, vol. 14, no. 1, pp. 359–373, Feb. 2022.
- [11] S. K. Tiwari, L. Kumaraswamidhas, and N. Garg, "Comparison of SVM and ARIMA model in time-series forecasting of ambient noise levels," in *Advances in Energy Technology*. Cham, Switzerland: Springer, 2020, pp. 777–786.
- [12] Z. Ceylan, "Comparative analysis of deep learning and classical time series methods to forecast natural gas demand during COVID-19 pandemic," *Energy Sources, B, Econ., Planning, Policy*, vol. 18, no. 1, Dec. 2023, Art. no. 2241455.
- [13] K. Agarwal, L. Dheekollu, G. Dhama, A. Arora, S. Asthana, and T. Bhowmik, "Deep learning-based time series forecasting," in *Deep Learning Applications, Volume 3* (Advances in Intelligent Systems and Computing), vol. 1395, M. A. Wani, B. Raj, F. Luo, and D. Dou, Eds. Singapore: Springer, 2022, doi: 10.1007/978-981-16-3357-7_6.
- [14] J. Jiang, L. Wu, H. Zhao, H. Zhu, and W. Zhang, "Forecasting movements of stock time series based on hidden state guided deep learning approach," *Inf. Process. Manage.*, vol. 60, no. 3, May 2023, Art. no. 103328.
- [15] R. P. Masini, M. C. Medeiros, and E. F. Mendes, "Machine learning advances for time series forecasting," *J. Econ. Surv.*, vol. 37, no. 1, pp. 76–111, 2023.
- [16] R. Kumar, P. Kumar, and Y. Kumar, "Integrating big data driven sentiments polarity and ABC-optimized LSTM for time series forecasting," *Multimedia Tools Appl.*, vol. 81, no. 24, pp. 34595–34614, Oct. 2022.
- [17] G. Kumar, U. P. Singh, and S. Jain, "An adaptive particle swarm optimization-based hybrid long short-term memory model for stock price time series forecasting," *Soft Comput.*, vol. 26, no. 22, pp. 12115–12135, Nov. 2022.
- [18] J. F. Torres, F. Martínez-Álvarez, and A. Troncoso, "A deep LSTM network for the Spanish electricity consumption forecasting," *Neural Comput. Appl.*, vol. 34, no. 13, pp. 10533–10545, Jul. 2022.
- [19] 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.*, vol. 35, no. 12, May 2021, pp. 11106–11115.
- [20] K. Gajamannage, Y. Park, and D. I. Jayathilake, "Real-time forecasting of time series in financial markets using sequentially trained dual-LSTMs," *Expert Syst. Appl.*, vol. 223, Aug. 2023, Art. no. 119879.
- [21] H. V. Dudukcu, M. Taskiran, Z. G. Cam Taskiran, and T. Yildirim, "Temporal convolutional networks with RNN approach for chaotic time series prediction," *Appl. Soft Comput.*, vol. 133, Jan. 2023, Art. no. 109945.
- [22] A.-A. Semenoglou, E. Spiliotis, and V. Assimakopoulos, "Data augmentation for univariate time series forecasting with neural networks," *Pattern Recognit.*, vol. 134, Feb. 2023, Art. no. 109132.
- [23] N. N. Aung, J. Pang, M. C. H. Chua, and H. X. Tan, "A novel bidirectional LSTM deep learning approach for COVID-19 forecasting," *Sci. Rep.*, vol. 13, no. 1, p. 17953, Oct. 2023.
- [24] L. C. Ortega, L. D. Otero, M. Solomon, C. E. Otero, and A. Fabregas, "Deep learning models for visibility forecasting using climatological data," *Int. J. Forecasting*, vol. 39, no. 2, pp. 992–1004, Apr. 2023.
- [25] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf, "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8857–8868.
- [26] J. Shi, M. Jain, and G. Narasimhan, "Time series forecasting (TSF) using various deep learning models," 2022, *arXiv:2204.11115*.
- [27] G. Ban, Y. Chen, Z. Xiong, Y. Zhuo, and K. Huang, "The univariate model for long-term wind speed forecasting based on wavelet soft threshold denoising and improved autoformer," *Energy*, vol. 290, Mar. 2024, Art. no. 130225.
- [28] Y. Ensafi, S. H. Amin, G. Zhang, and B. Shah, "Time-series forecasting of seasonal items sales using machine learning—A comparative analysis," *Int. J. Inf. Manage. Data Insights*, vol. 2, no. 1, Apr. 2022, Art. no. 100058.
- [29] L. Ilias, E. Sarmas, V. Marinakis, D. Askounis, and H. Doukas, "Unsupervised domain adaptation methods for photovoltaic power forecasting," *Appl. Soft Comput.*, vol. 149, Dec. 2023, Art. no. 110979.
- [30] Z. Liu, Qianli Ma, Peitian Ma, and Linghao Wang, "Temporal-frequency co-training for time series semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, 2023, pp. 8923–8931.
- [31] P. Gaur, H. Gupta, A. Chowdhury, K. McCreddie, R. B. Pachori, and H. Wang, "A sliding window common spatial pattern for enhancing motor imagery classification in EEG-BCI," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–9, 2021.
- [32] W. Zha, Y. Jin, Y. Sun, and Y. Li, "A wind speed vector-wind power curve modeling method based on data denoising algorithm and the improved transformer," *Electr. Power Syst. Res.*, vol. 214, Jan. 2023, Art. no. 108838.
- [33] Y. Lu, D. Tang, D. Zhu, Q. Gao, D. Zhao, and J. Lyu, "Remaining useful life prediction for bearing based on coupled diffusion process and temporal attention," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–10, 2024.
- [34] P. Guo, Q. Liu, S. Yu, J. Xiong, X. Tan, and C. Guo, "A transformer with layer-cross decoding for remaining useful life prediction," *J. Supercomput.*, vol. 79, no. 10, pp. 11558–11584, Jul. 2023.



SIHAO WAN was born in Nanchang, Jiangxi, China. He is currently pursuing the B.S. degree in software engineering with the College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China. Since 2023, he has been an Intern with iFlyTek, Hefei, China. His research interests include time series forecasting with machine learning and vehicular communications with reinforcement learning.

• • •