



OPEN NSPLformer: exploration of non-stationary progressively learning model for time series prediction

Sun Jiaxing^{1,2}✉, Li Yanhui¹ & Zhao Yuying³

Although Transformers perform well in time series prediction, they struggle when dealing with real-world data where the joint distribution changes over time. Previous studies have focused on reducing the non-stationarity of sequences through smoothing, but this approach strips the sequences of their inherent non-stationarity, which may lack predictive guidance for sudden events in the real world. To address the contradiction between sequence predictability and model capability, this paper proposes an efficient model design for multivariate non-stationary time series based on Transformers. This design is based on two core components: (1) Low-cost non-stationary attention mechanism, which restores intrinsic non-stationary information to time-dependent relationships at a lower computational cost by approximating the distinguishable attention learned in the original sequence.; (2) dual-data-stream Progressively learning, which designs an auxiliary output stream to improve information aggregation mechanisms, enabling the model to learn residuals of supervised signals layer by layer. The proposed model outperforms the mainstream Transformer with an average improvement of 5.3% on multiple datasets, which provides theoretical support for the analysis of non-stationary engineering data.

Keywords Times series prediction, De-stationary attention mechanism, Progressively learning Block, Non-stationary modeling, Deep learning

Time series forecasting plays a crucial role in many applications such as energy consumption planning¹, traffic flow analysis², financial risk assessment³ and cloud resource allocation⁴. Time series data recorded from the real world usually exhibit diverse and unsteady characteristics due to the complex transient conditions experienced during their evolution⁵. This phenomenon reflects the complexity and diversity of time series data in real-world applications and poses challenges for their analysis and modeling (i.e., distributional shifts induced by time series non-stationarity)⁶.

A recent approach to solve the above distributional bias problem is to utilize plug-and-play normalization methods, Kim et al⁷ proposed a method called reversible instance normalization (RevIN), RevIN consists of two distinct steps, normalization and unnormalization. The former normalizes the input to fix its distribution based on the mean and variance, while the latter returns the output to the original distribution. To alleviate the challenges posed by non-stationarity in time-series data, Liu et al⁸ proposed SAN (Slice-Level Adaptive Normalization), a flexible normalization and denormalization scheme that removes non-stationarity and independently models variations in the statistical properties of the original time-series data through local time slices, thus achieving prediction performance improvement on a wide range of benchmark prediction models. The prediction performance improvement is realized on a variety of benchmark prediction models. To address the challenges posed by complex non-uniform distributions and pattern drift in real time-series data, Sun et al⁹ proposed the TFPS (Time-Frequency Pattern Specific) architecture, which utilizes pattern-specific expert modules in combination with bi-domain encoders and subspace clustering techniques to dynamically identify and model unique patterns in data segments, which This results in significantly improved prediction accuracy. Fan et al¹⁰ systematically summarized the distribution drift problem in temporal prediction and classified it into distribution changes within the input space (recall window) and output space (prediction window) (internal space drift) and distribution differences between the input and output spaces (cross-space drift). Meanwhile, Dish-Ts, a generalized neural architecture, is proposed to effectively mitigate the effects of distributional drift by

¹Department of Bohai Rim Energy Research Institute, Northeast Petroleum University, No550,West Section of Hebei Street, Qinhuangdao 066004, Hebei province, China. ²Electrical and Information Engineering, Northeast Petroleum University, Xuefu Street 99, Daqing 163318, Heilongjiang province, China. ³Qinhuangdao Campus, Northeast Petroleum University, No. 550, West Section of Hebei Street, Qinhuangdao 066004, Hebei province, China. ✉email: SJX_change@126.com

learning and adjusting the distributions in the input and output spaces respectively through a dual-coefficient network (Dual-COMET). All of these methods normalize the input time series to a uniform distribution, explicitly removing non-smoothness to reduce the differences in data distribution.

However, non-stationarity is an inherent property of real-world time series, and also good guidance for discovering temporal dependencies for forecasting. In experiments, it has been found that direct smoothing of time-series data causes ordinary transformer models to lose the ability to capture the important event-based temporal dependencies in the data. That is, the data loses important non-stationarity properties during the smoothing process, causing the model to fail to correctly capture eventful changes in the data and limiting the model's forecasting ability, a problem known as over-stationarization. It is precisely these important non-stationarity properties that should be the focus of the attention mechanism, such as drastic changes in traffic flow due to traffic accidents, abnormal sensor readings due to sudden equipment failures, and meteorological data variations due to sudden rainstorms. However, designing highly complex and nonlinear deep model structures for better learning of non-stationarity features to solve the problem of unpredictable event-based changes will lead to severe overfitting (validation loss increases significantly and training loss still decreases sharply). Therefore, how to reduce the time series non-stationarity to improve predictability while avoiding the problem of model overfitting is a key issue to further improve the prediction performance.

In this paper, a sample of predictions is plotted in Fig. 1 to better illustrate our observations. Despite the temporal correlation, the mean values of the input sequences are significantly different from the mean values of the field-of-view window (from 35.79 to 48.3), suggesting that there may be a generalized distributional shift. Furthermore, this distributional bias can change rapidly at finer-grained slice levels, and this rapid change violates the underlying assumptions of existing normalization methods (which typically assume that the distribution of the input data is relatively stable). As a result, the use of inappropriate normalization methods not only destroys the intrinsic pattern of the input series, but also leads to bias in the final prediction results.

In this paper, we explore the impact of non-stationarity in time series forecasting and propose NSPLformers as a general framework. The framework consists of two interdependent modules and a two-stream asymptotic learning architecture: Series Stationarization for improving the predictability of non-stationarity time series; and De-stationary Attention for mitigating the over-stationarization problem. Specifically, Series Stationarization employs a simple but effective normalization strategy that unifies the statistical properties of each time series without additional parameters, while De-stationary Attention approximates the attention mechanism of the original data and compensates for the inherent non-stationarity of the original time series. The architecture of Dual Data Stream Progressive Learning utilizes the concept of de-redundancy to propose a progressive learning approach aimed at systematically acquiring the components of the supervised signals to improve the performance of time series (TS) prediction. And an auxiliary output stream is introduced in each Block to form a highway that gradually guides to the final prediction. The subsequent Block outputs of this output stream will subtract the previously learned results layer by layer, enabling the model to gradually learn the residual parts of the supervised signal. The dual data stream design further facilitates the implicit layer-by-layer decomposition of the input and output streams, enhancing the model's flexibility, interpretability, and resistance to overfitting.

The contribution of this paper is threefold:

- The predictive ability of non-stationary sequences is crucial in actual predictions. Through detailed analysis, it was found that the current stationarisation method leads to over-stationarisation, thereby limiting the predictive ability of Transformers;
- We propose a low-cost non-stationary progressive learning Transformer, which includes sequence smoothing to improve sequence predictability, de-smoothing attention mechanisms, and progressive learning modules.

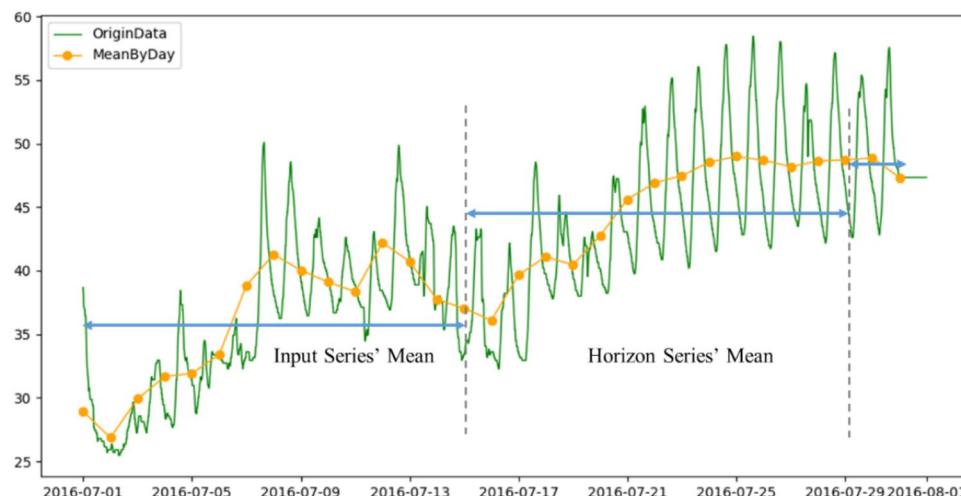


Fig. 1. Examples of forecasts of energy consumption and illustrations of their daily averages (MeanByDay). We also plot the mean of the input sequence and the mean of the horizon sequence in the figure.

We simplify the attention mechanism calculation using low-rank approximation methods to reduce complexity and reintroduce the non-stationarity of the original sequence to avoid over-stationarization issues;

- This paper constructs a dual data stream design that promotes implicit gradual decomposition of input and output streams based on incremental learning through auxiliary output streams and improved information aggregation mechanisms, thereby giving the model greater flexibility, interpretability, and resistance to overfitting, effectively resolving the dilemma between sequence predictability and model capability.

Related work

In recent years, due to the powerful nonlinear fitting capabilities of deep learning¹¹, time series prediction models have been applied to various fields such as medicine^{12–14}, petroleum exploration¹⁵, industrial production^{16,17}, classification detection^{18,19}, and many others. However, time series (TS) from the real world often exhibit various non-stationary characteristics due to their evolution under complex transient conditions⁵. The characteristics of non-stationary time series are reflected in the continuous changes in their statistical properties and joint distributions²⁰, making accurate prediction extremely challenging.²¹ As a result, numerous scholars have conducted research on the prediction of non-stationary time series.

Transformer-based time series forecaster

Recently, deep learning models have achieved remarkable success in time series prediction, in order to capture the long-term coupling of inputs and outputs over a long period of time, Zhou et al designed a transformer-based Informer for long-series time series forecasting (LSTF), which overcomes the limitations of the traditional Transformer by introducing the ProbSparse self-attention mechanism, self-attention distillation and generative decoder, which overcomes the limitations of quadratic time complexity, high memory consumption, and encoder-decoder architecture of traditional Transformer.²² WU et al aimed to solve the long time series prediction problem by proposing a novel decomposition architecture with autocorrelation mechanism, which enables the model to have the capability of progressive decomposition of complex time series, correlation discovery and representation aggregation at the subsequence level level²³. Liu et al proposed a novel neural network architecture called SCINet to efficiently model and predict time series through recursive downsampling, convolution, and interactive operations²⁴. Nie et al proposed an efficient design of a multivariate time series prediction and self-supervised representation learning model based on the Transformer, the model is based on two key components: (i) segmentation of time series into sub-series level patches; and (ii) channel independence, which achieves retention of local semantic information, quadratic reduction in computation and memory usage of the attention graph and the ability of the model to focus on longer histories²⁵. Zhang et al. proposed a Crossformer named Transformer base model, which effectively captures cross-time dependencies and cross-dimensional dependencies in multivariate time series (MTS) by introducing Dimension-Segment-Wise (DSW) embedding and Two-Stage Attention (TSA) layer, and establishes a Hierarchical Encoder-Decoder (HED) architecture to utilize different scales of information, experimental results show that Crossformer significantly outperforms previous state-of-the-art approaches on six real-world datasets²⁶. Considering the limitations of point-by-point representations in terms of local semantics, liu et al.²⁷ proposed a model named iTTransformer to efficiently capture cross-variate correlations in multivariate time series and learn nonlinear representations by inverting the functionality of the attention mechanism and feedforward network in Transformer. These models aim to extract diverse and informative patterns from historical observations to improve the accuracy of time series forecasting. In order to improve the accuracy of real-world time series forecasting, the key challenge is twofold, on the one hand, the fact that data from numerous real-world data series data exhibit exhibit dynamic and evolving patterns, a phenomenon known as non-stationary. This property usually leads to training and testing This property usually leads to inconsistent distributions between the training set, the testing set, and the future unseen data set. Thus, the non-stationary property in time-series data requires the development of robust predictive models that can cope with the variation of such data distributions over time, and failure to address this challenge often leads to reduced representational power and impaired model generalization. On the other hand, there is the phenomenon of overfitting when deep models learn high-dimensional non-stationary features of emergent events, which usually manifests itself as feature redundancy, poor interpretability and weak generalisation.

Redundancy-reduced progressive learning

A new training strategy based on progressive learning was proposed by Zhou et al. By equipping each low-precision convolutional layer with an auxiliary full-precision convolutional layer based on the low-precision network structure. At the same time, a decay method is introduced to gradually reduce the output of the added full-precision convolutional layer, thus keeping the topology the same as the original low-precision convolutional layer, and realising the progressive learning strategy²⁸. A stepwise approach for learning independent hierarchical representations and realizing the learning of representations from high-level to low-level in the VAE framework. The effectiveness of the approach in decoupling representation learning is confirmed by quantitative and qualitative evaluations on a benchmark dataset and is the first attempt to progressively extend the VAE capacity to learn hierarchical representations for improved decoupling²⁹. Zhao et al proposed a three-stage learning framework for retrieval by gradually learning complex knowledge of mixed-modal queries and introduced a self-supervised adaptive weighting strategy³⁰. The above methods attempt sequence decomposition applied to the input sequence to enhance the predictability of the time series and reduce the impact of algorithmic complexity on the model. However, the mainstream time series prediction algorithms are susceptible to severe overfitting when the original sequence has strong non-stationarity. In this paper, the proposed method mitigates the overfitting problem by implicitly decomposing the supervised signals by learning each time-varying pattern through progressive bootstrapping.

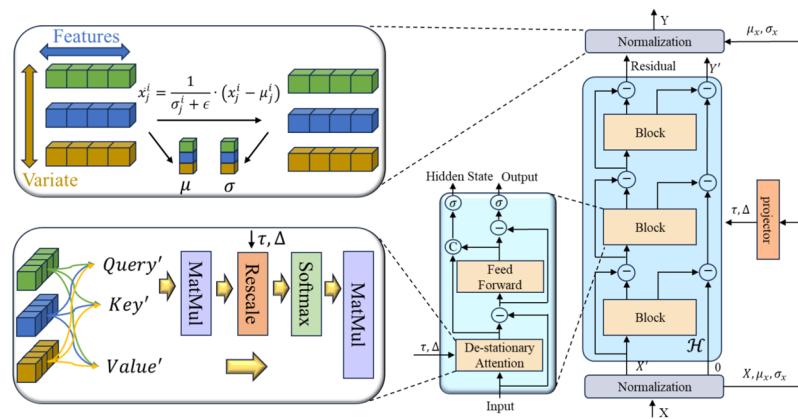


Fig. 2. NSPLformer normalizes and denormalizes the input series by smoothing and adjusts the time-dependent weights using the de-smoothing attention mechanism to better handle non-smooth time series.

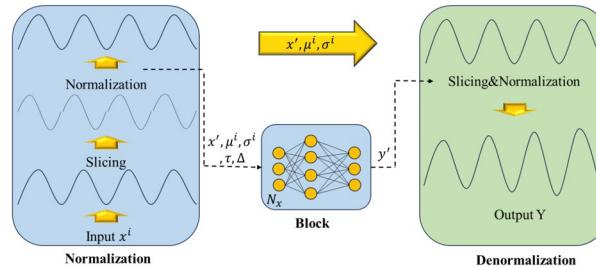


Fig. 3. This section proposes a model-independent symmetric normalization module that removes and recovers non-stationary factors from time series data from a slicing perspective.

NSPLformer

As mentioned earlier, stationarity is an important component of time series predictability. Previous research efforts have focused on “direct smoothing” to reduce the non-smoothness of the series in order to obtain better predictability. However, non-stationarity is an inherent property of real-world time series, and the design of “direct smoothing” will lead to over-stationarization of the model, and the design of complex deep models to learn unprocessed non-stationary features will lead to severe overfitting of the model. To solve the above dilemma, this paper proposes NSPLformer (Non-stationary Progressively Learning Transformer). The model involves: Series Stationarization to attenuate the time series non-stationarity, De-stationary Attention mechanism to recombine the non-stationary features of the original series, and a design of the dual data stream model structure to suppress overfitting. As Fig. 2, with the support of these designs, the model can improve time series data predictability while maintaining model capability.

Series stationarization

The statistical properties (e.g., mean and standard deviation) of non-stationary time series change during the inference process, which makes it difficult for the deep model to perform effective generalized prediction. Therefore, in this paper, in terms of dealing with the non-stationarity of the original time series, the normalization module is designed to firstly deal with the non-stationary series caused by different data distributions, and finally the inverse normalization module restores the model output to the original statistical characteristics. Here are the details.

Normalization module Existing normalization methods for non-stationary time series prediction are to normalize the input series to remove non-stationary factors and restore them to the output series by denormalization. As Fig. 3 the method in this paper differs from it in that by slicing the input sequence into multiple non-overlapping slices and performing a local normalization operation on each slice, it is able to effectively remove the non-stationary factors while preserving the intrinsic pattern of each slice.

This local processing is more advantageous than the global normalization method, especially when dealing with complex non-stationary time series. Eventually, the normalized slices are recombined into a new input sequence by an inverse normalization step. The input x^i is first split into M non-overlapping slices $\{x_j^i\}_{j=1}^M$, where the mean and standard deviation of each slice x_j^i can be expressed as:

$$\mu_j^i = \frac{1}{T} \sum_{t=1}^T x_{j,t}^i, (\sigma_j^i)^2 = \frac{1}{T} \sum_{t=1}^T (x_{j,t}^i - \mu_j^i)^2 \quad (1)$$

Where $\mu_j^i, \sigma_j^i \in \mathbb{R}^{C \times 1}$ and $x_{j,t}^i$ is the value of slice x_j^i at t-th time step. The normalized input sequence $\bar{x}_j^i = \frac{x_j^i - \mu_j^i}{\sigma_j^i}$ can be expressed as:

$$\bar{x}_j^i = \frac{1}{\sigma_j^i + \epsilon} \cdot (x_j^i - \mu_j^i)^2 \quad (2)$$

Where \cdot is the element-wise product and ϵ is a small constant. Finally, a new input sequence with non-stationary factors removed is formed by recombining the slices and used as input to the prediction model.

De-normalization module In order to recover the characteristics of the original sequence, we apply a denormalization process to the model output $Y' = [Y'_1, Y'_2, \dots, Y'_O]^T \in \mathbb{R}^{O \times C}$ using the statistical parameters μ_j^i and σ_j^i of the original sequence. In contrast to normalization, De-normalization is performed on a per-slice basis. For the predictive model output Y' , which is split into K non-overlapping slices $\{\bar{y}_j^i\}_{j=1}^K$, the statistically based inverse normalization of any slice can be expressed as

$$\hat{y}_j^i = \bar{y}_j^i \cdot (\sigma_j^i + \epsilon) + \mu_j^i \quad (3)$$

Finally, all slices are restored in chronological order. By leveraging the model's translational and scaling equivariance, the normalization and de-normalization modules ensure that the model maintains high prediction stability and accuracy, even when the mean or amplitude of the input sequence changes. This robustness is crucial for reliable performance under varying input conditions.

De-stationary attention

In time series forecasting, although the statistical features can be recovered to the corresponding forecasts by inverse normalization, etc., the non-stationary nature of the original series cannot be fully recovered by simple inverse normalisation. Specifically, Series Stationarization can generate similar normalised inputs x' based on different time series x_1 and x_2 , and the basic model will get the same attention when processing these normalised inputs, thus failing to capture the temporal dependencies associated with the non-stationary. This phenomenon is particularly relevant in the case of attention computation. This phenomenon is particularly noticeable in the attention computation process. In addition, non-stationary time series tend to lose their intrinsic structure during stationarization and fragmentation, where these segments are forced to have the same mean and variance after back-normalisation, and are therefore closer to similar distributions than to the true distribution of the original data before normalisation. This results in models that are more likely to produce outputs that are over stationarization and lack variation, which contradicts the natural non-stationary nature of the original time series.

To address the problem of over-stationarization caused by sequence normalisation, we propose a new De-stationary Attention Mechanism. The mechanism aims to approximate the attention distribution obtained when no normalisation is performed, thus enabling the identification and capture of specific temporal dependencies from raw non-stationary data, reducing the prediction error due to over-stationarization.

Analysis of the plain model As mentioned earlier, the over-stationarization problem stems from the disappearance of the inherent non-stationarity information in the original time series, which leads to the underlying model's inability to capture eventful temporal dependencies for prediction. Therefore, we attempt to approximate the learned attention from the original non-stationary time series. The known self-attention mechanism³¹ is as follows:

$$\text{Attn}(Q, K, V) = \text{Softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (4)$$

Where $Q, K, V \in \mathbb{R}^{S \times d_k}$ are the Queries, Keys, and Values of length S , respectively, all of dimension d_k , and the $\text{Softmax}(\cdot)$ operation is performed on a row-by-row basis. To simplify the analysis, it is assumed that the embedding layer and the feed-forward layer f have a linear nature and their operations are completely independent of each time point. That is, each q_i in $Q = [q_1, q_2, \dots, q_S]^T$ can be expressed as $q_i = f(x_i)$, where the input time series $x = [x_1, x_2, \dots, x_S]^T$.

Since it is common practice to stabilize each time series variable to avoid the over-dominance of scale by a particular variable, we assume that each variable of the time series x has the same variance. This simplification reduces the original vector of standard deviations $\sigma_x \in \mathbb{R}^{C \times 1}$ to a scalar. After passing through the normalization module, the model receives a smoothed input of:

$$x' = \frac{x - 1\mu_x^T}{\sigma_x} \quad (5)$$

Where $1 \in \mathbb{R}^{S \times 1}$ is a vector of all ones. Based on the above assumption of linear nature, it can be shown that the attention layer will receive the query vector Q' as:

$$Q' = [f(x'_1), \dots, f(x'_S)]^T = \frac{Q - 1\mu_Q^T}{\sigma_x} \quad (6)$$

where $\mu_Q \in \mathbb{R}^{d_k \times 1}$ is the mean value of the query vector Q along the time dimension. Similarly, Key and Value are converted to K' and V' . The attention mechanism based on Q' and K' after stationarized is:

$$Q'K'^T = \frac{1}{\sigma_x^2} (QK^T - 1\mu_Q^T K - K\mu_Q 1^T + 1\mu_Q^T K\mu_Q 1^T) \quad (7)$$

$$\text{Softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) = \text{Softmax} \left(\frac{\sigma_x^2 Q'K'^T + 1(\mu_Q^T K)^T + (Q\mu_K)1^T - 1(\mu_Q^T \mu_K)1^T}{\sqrt{d_k}} \right) \quad (8)$$

where $Q\mu_K \in \mathbb{R}^{S \times 1}$ and $\mu_Q^T \mu_K \in \mathbb{R}$. These quantities are iteratively applied to each column and element of $\sigma_x^2 Q'K'^T \in \mathbb{R}^{S \times S}$ during the computation. Using the translation invariance of the Softmax function, the computation of the attention mechanism can be simplified as follows:

$$\text{Softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) = \text{softmax} \left(\frac{\sigma_x^2 Q'K'^T + 1\mu_Q^T K^T}{\sqrt{d_k}} \right) \quad (9)$$

This section derives a direct expression for the attentional mechanism $\text{Softmax}(QK^T/\sqrt{d_k})$ learnt from the original sequence x . In addition to Q' and K' in the stationarized time series x' , non-stationary information needs to be introduced, which is eliminated during the sequence stationarization process. Therefore, in order to recover the original non-stationary time series features, the lost non-stationary information needs to be reintroduced inside the depth model.

De-stationary Attention In order to restore the original attention to the non-stationary series, we try to introduce unappreciated non-stationary information into its computation. Defining the approximate scaling scalar $\tau = \sigma_x^2 \in \mathbb{R}^+$ and the displacement vector $\Delta = K\mu_Q \in \mathbb{R}^{S \times 1}$ as the de-smoothing factors, this section tries to learn the non-stationary factors directly from the non-stationary x , Q and K statistics by a simple but effective multilayer perceptron. Since we can only find limited non-stationary information from the known Q' , K' , a reasonable source for compensating non-stationarity at one time can only be the original sequence. Therefore, in this paper, we use a multilayer perceptron to map the input sequences into a high-dimensional space to learn the non-smooth factors τ and Δ from $\sigma_x^2, \sigma_j^{i^2}, x, \bar{x}_j^i$ respectively. And the De-stationary Attention is calculated as follows.

$$\begin{aligned} \log \tau &= \text{MLP}(\sigma_x^2, (\sigma_j^i)^2, x, \bar{x}_j^i) \\ \Delta &= \text{MLP}(\mu_x, \mu_j^i, x, \bar{x}_j^i) \\ \text{De-stationary}(Q', K', V', \tau, \Delta) &= \text{Softmax} \left(\frac{\tau Q'K'^T + 1\Delta^T}{\sqrt{d_k}} \right) V' \end{aligned} \quad (10)$$

The de-stationary factors τ and Δ are shared across all de-stationary attention layers. The de-stationary attention mechanism learns the temporal dependencies from the stationarized sequences Q' , K' and the statistical properties of the original sequences $\sigma_x^2, \sigma_j^{i^2}, x, \bar{x}_j^i$ and multiplies them by the stationarized value V' . Thus, it is able to exploit the predictability of the stationarized series while maintaining the time dependence inherent in the original series.

De-stationary attention is low rank

In this section, the non-stationary attention mechanism is optimised in terms of linear time and memory complexity. In this section, two linear projection matrices $E, F \in \mathbb{R}^{n \times k}$ are added in the computation of key and value to reduce the computational cost of computing the context mapping matrix P based on the stochastic projection method. First, the original $n \times d$ dimensional key and value layer is projected to the $k \times d$ dimensional projected key and value layer, and then an $n \times k$ dimensional context mapping matrix \tilde{P} is computed using the scaled dot product attention mechanism.

$$\text{De-stationary}(Q', EK', FV', \tau, \Delta) = \text{Softmax} \left(\underbrace{\frac{\tau Q'(EK')^T + 1\Delta^T}{\sqrt{d_k}}}_{P: n \times k} \right) \cdot \underbrace{FV'}_{k \times d} \quad (11)$$

Finally, only $O(nk)$ time and space complexity is required to compute the context embedding using $\tilde{P} \cdot (FV')$. Thus, choosing a very small projection dimension k such that $k \ll n$, then the method can significantly reduce memory and computational cost consumption. The following theorem shows that when $k = O(d/\epsilon^2)$ (independent of n), one can use a linear non-stationary attention mechanism to approximate $\tilde{P} \cdot V'$ with an error of ϵ .

Theorem 1 (Linear De-stationary Attention) For any $Q, K, V \in \mathbb{R}^{n \times d}$, if $k = \min\{\theta(9d \log(d)/\epsilon^2), 5\theta(\log(n)/\epsilon^2)\}$, then there exists matrices $E, F \in \mathbb{R}^{n \times k}$ such that, for any row vector w of matrix $Q(K)^T/\sqrt{d_k}$, we have: $w \cdot \left(\frac{Q(K)^T}{\sqrt{d_k}} \right) \cdot FV' \cdot P_{n \times k}$

$$\Pr \left(\|\text{softmax}(wE^T)FV' - \text{softmax}(w)V'\| \leq \epsilon \|\text{softmax}(w)\| \|V'\| \right) > 1 - o(1) \quad (12)$$

where $E = \delta R$ and $F = e^{-\delta} R$, R is an $n \times k$ matrix whose elements are independently distributed from the normal distribution $N(0, 1/k)$ and δ is a small constant. By applying the results of the above formula to each row vector of matrix A and each column vector of matrix V' , it can be proven that for any row vector A_i of matrix A , the error between the transformed result and the original result is controlled within an acceptable range with high probability, thereby verifying the effectiveness and stability of the model in processing high-dimensional data.

$$\Pr \left(\|\exp(A_i E^T)FV - \exp(A_i)V\| \leq \epsilon \|\exp(A_i)V\| \right) > 1 - o(1) \quad (13)$$

By setting $k = \frac{5 \log(nd)}{\epsilon^2 - \epsilon^3}$, the above procedure shows that for a given approximation error $\epsilon > 0$, the approximation error condition we wish to obtain will almost certainly hold for a set value of k , suggesting that we can control the approximation error by choosing an appropriate value of k for the model in this paper. This suggests that the model in this paper can control the approximation error with an appropriate choice of k -value, and significant improvements in training and inference speed are observed. For more details, please refer to the appendix.

Progressively learning block

Time series forecasting models are prone to severe overfitting when dealing with time series data due to the non-stationarity of the time series, the time series evolves under complex transient conditions. This is manifested by the fact that during the training process, the training loss still decreases dramatically, but the validation loss still increases significantly. In this section, a progressive learning module is designed to systematically acquire the components of a supervised signal by decomposing the supervised signal and learning different parts of the time series sequentially. The module adopts a dual data stream and subtraction mechanism, where the module generates a prediction result and subtracts it from the prediction results of the subsequent layers to generate a residual data stream, through which the module is able to reduce the prediction error layer by layer in this decreasing way. At the same time, the module builds a 'high-speed channel' for the prediction results to go directly to the next module, which gradually learns the residual streams and output streams of the stacked blocks, and gradually approaches the real labels, thus improving the model performance and effectively avoiding overfitting.

When the attention mechanism deals with time series where different attributes are independent of each other, the attention mechanism still tries to learn the relationship between these attributes, increasing the complexity of the model and the number of parameters, resulting in a large amount of redundant information and leading to model overfitting. To mitigate this problem, this section implements a correction metric by subtracting the attention output from the input. This ensures that attention can be effectively utilised to its inherent advantage, thus improving overall performance. This process can be expressed as:

$$\begin{aligned} \tilde{X}_{l,1} &= D - \text{stationary}(X_{l,1}) \\ R_{l,1} &= X_{l,1} - \delta \tilde{X}_{l,1} \end{aligned} \quad (14)$$

Where, $\tilde{X}_{l,1}$ represents the input signal processed by the Attention mechanism (Attention) in the l th layer, $R_{l,1}$ denotes the remainder of the input $X_{l,1}$ after block processing, and δ is the Dirac function. δ is used to eliminate the Attention layer when an unfavourable influence is applied. Similarly, the input signal processed by the feedforward layer will be subtracted from the input and can be expressed as:

$$\begin{aligned} \tilde{X}_{l,2} &= \text{FeedForward}(R_{l,1}) \\ R_{l,2} &= X_{l,2} - \delta \tilde{X}_{l,2} \end{aligned} \quad (15)$$

The feedforward layer consists of two identity mappings with an activation function inserted between them, which specialises in non-linear transformations of the temporal aspect. This feature is particularly important when dealing with multivariate time series data with strong independence between features. Specifically, the current block contains two parallel data streams: the output X_l processed by the neural modules, and the residual \tilde{R}_l obtained by subtracting \tilde{X}_l from the input, which is processed by a gate mechanism and then directed to the next block or projected to the output space. The two data streams are processed through the gate mechanism and are then directed to the next block or projected to the output space.

Drawing inspiration from RNNs³², this section adopts an approach that can autonomously regulate the transfer of information, similar to the control mechanisms inherent to cells in RNNs. Thus, we introduce a gate mechanism at the end of each block to control the transmission of the two data streams.

For residual stream, the gate mechanism can be expressed as:

$$X_{l+1} = \sigma(\theta_1(R_{l,2})) \cdot \theta_2(R_{l,2}) \quad (16)$$

where σ is the sigmoid function and θ_1 and θ_2 are learnable neurons with different parameters. Similarly, for the intermediate output stream \hat{X}_l , the gate mechanism can be expressed as:

$$o_{i+1} = \sigma(\theta_3([\hat{X}_{l,1}, \hat{X}_{l,2}]) \cdot \theta_4([\hat{X}_{l,1}, \hat{X}_{l,2}])) \quad (17)$$

Where $[]$ denotes concatenation. o_{l+1} enables the module to better utilise the output of the non-stationary attention mechanism and the output of the feedforward layer. The gating mechanism can introduce signal morphing in both data streams, effectively adjusting the amount of information output from the current block. It enables the model to selectively enhance or attenuate the influence of specific phases, thus adequately mitigating the overfitting effects of independent attributes and redundant information on the model and enhancing its modeling capabilities. Compared with other time series prediction models, the model proposed in this paper is able to complete the screening and amplification of information, learn the residual and output signals in an incremental learning manner, and gradually approximate the supervised.

Overall architecture

The overall architecture uses stacked Progressively learning blocks combined with non-stationary attention mechanisms, parallel data flow structures, gating mechanisms and Normalisation module De-normalization module. Comparing with the standard Transformer model³¹, the model proposed in this paper designs the non-stationary attention mechanism for the non-stationary nature of the original series in order to improve the prediction ability of non-stationary time series. Comparing with the variants of Transformer^{33,22}, this paper varies the terms within $\text{Softmax}(\cdot)$ with a de-stationary factor τ, δ to reintegrate the non-stationary information. In terms of time series forecast decomposition, the architecture proposed in this paper differs from Prophet³⁴, N-BEATS³⁵, Autoformer²³, FEDformer³⁶ and Nonstationary Transformers³⁷ in the sense that the above methods apply decomposition to the input series to enhance predictability, reduce computational complexity or ameliorate the adverse effects of non-stationarity. However, these popular methods are susceptible to significant overfitting when applied to non-stationary TS. The method proposed in this paper employs a Progressively learning approach to supervise the learning of each time-varying pattern, achieved by implicitly decomposing the supervised signals, to effectively address the overfitting problem due to the independence of the original sequence features or non-stationarity.

Experiment

NSPLformer has been comprehensively evaluated on widely-use real-world datasets, covering various mainstream time series forecasting applications including energy, transportation, electricity, weather, and exchange.

All experiments were implemented in PyTorch and conducted on a single NVIDIA P100 16GB GPU. We utilised ADAM with initial learning rates of $\{10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$ and L2 loss for model optimisation. The batch size was set to 128, and the number of training epochs was fixed at 10. For all models, this paper tested performance under four prediction lengths $T \in \{96, 192, 336, 720\}$, with a lookback window $L = 96$. The dataset was split into training, validation, and test sets in a 7:2:1 ratio, with early stopping used to prevent overfitting, and a random seed of 2025.

Datasets

As shown in Table 1, here are the descriptions of the datasets: (1)Electricity³⁸ records the hourly electricity consumption of 321 clients from 2012 to 2014. (2)ETT²² contains the time series of oil de-stationary factors and power load collected by electricity transformers from July 2016 to July 2018. ETTm1 /ETTm2 are recorded every 15 minutes, and ETTh1/ETTh2 are recorded every hour. (3)Exchange³⁹ collects the panel data of daily exchange rates from 8 countries from 1990 to 2016. (4)⁴⁰ contains hourly road occupancy rates measured by 862 sensors on San Francisco Bay area freeways from January 2015 to December 2016. (5)⁴¹ includes meteorological time series with 21 weather indicators collected every 10 minutes from the Weather Station of the Max Planck Biogeochemistry Institute in 2020. (6)Solar documents the solar power generation of 137 photovoltaic (PV) facilities in the year 2006, with data collected at 10-minute intervals.

This paper employs the Augmented Dickey-Fuller (ADF) test statistic as an indicator to quantitatively measure the degree of stationarity.

Dataset	Features	Sampling Frequency	Total Observations	ADF Statistic
Exchange	8	1Day	7588	- 1.89
ETTm2	7	15Minutes	69680	- 6.23
Electricity	321	1Hour	26304	- 8.48
Traffic	862	1Hour	17544	- 15.05
Weather	21	10Minutes	52695	- 26.66
Solar	137	10Minutes	2920	- 30.40

Table 1. Summary of datasets. Smaller ADF test statistic indicates a more stationary dataset.

Baselines

Due to the inferior performance of traditional models (such as ARIMA) compared to the latest Transformer-based models, this paper selects a range of state-of-the-art (SOTA) Transformer-based models as comparative benchmarks for analysis. These include iTransformer²⁷, PatchTST²⁵, TimesNet⁴², FEDformer³⁶, Autoformer²³ and Informer²².

Main results

The model presented in this paper maintains state-of-the-art performance across all base-line and prediction lengths. Notably, the non-stationary asymptotic learning model outperforms other deep models on datasets with high non-stationarity. As shown in Table 3, the best results are highlighted in red and bold, and the second best results are highlighted in purple and underlined. iTransformer and PatchTST stand out as the most recent models recognised for their superior average performance. Compared to these, the paper proposed models show average performance improvements of 1.5% and 4.5% respectively, achieving significant performance gains. Achieved performance in multiple dimensional comparisons across the 8 datasets is most excellent/sub-optimal, with each individual metric showing performance improvements. For example, under the input-prediction-720 setting, the paper proposed model reduces the MSE on the Electricity dataset by 15.7% (0.255 → 0.215) and 11.9% (0.244 → 0.235) compared to iTransformer and PatchTST, respectively. Similarly, in traffic flow, the MSE is reduced from 1.6% (0.433 → 0.426) and 16.8% (0.512 → 0.426). The experimental results indicate that the proposed NSPLformer exhibits superior predictive performance across datasets with varying horizons (Table 2).

Framework generality

This paper applies the framework proposed in this paper to four mainstream Transformers and reports the performance improvement of each model. As shown Table 3, the approach in this paper continuously improves the predictive power of different models. Overall, it achieves an average improvement of 50.74% on Transformer, 59.77% on Informer, and 19.28% on Autoformer, making each of them outperform the previous state-of-the-art. At the same time, the model validates non-stationary Progressively learning Transformer as an effective and

dataset	length	Model		Ours		iTransformer		PatchTST		TimesNet		DLinear		FEDformer	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.370	0.393	0.386	0.395	0.414	0.419	0.383	0.401	0.383	0.400	0.395	0.424		
	192	0.424	0.426	0.437	0.424	0.461	0.444	0.435	0.423	0.437	0.431	0.471	0.473		
	336	0.467	0.459	0.479	0.446	0.501	0.466	0.490	0.468	0.479	0.458	0.530	0.499		
	720	0.469	0.471	0.481	0.470	0.499	0.488	0.521	0.500	0.517	0.516	0.579	0.545		
	avg	0.433	0.473	0.446	0.434	0.469	0.454	0.457	0.448	0.454	0.451	0.494	0.506		
ETTh2	96	0.301	0.348	0.288	0.338	0.302	0.348	0.340	0.372	0.333	0.385	0.356	0.395		
	192	0.365	0.391	0.374	0.390	0.388	0.400	0.402	0.414	0.476	0.474	0.427	0.437		
	336	0.424	0.431	0.415	0.426	0.426	0.433	0.451	0.452	0.592	0.543	0.496	0.485		
	720	0.421	0.439	0.420	0.440	0.431	0.446	0.460	0.465	0.829	0.655	0.463	0.474		
	avg	0.378	0.402	0.374	0.399	0.387	0.407	0.413	0.426	0.558	0.514	0.436	0.448		
ETTm1	96	0.316	0.347	0.334	0.368	0.328	0.367	0.337	0.374	0.345	0.371	0.378	0.416		
	192	0.361	0.378	0.375	0.392	0.367	0.385	0.374	0.387	0.378	0.388	0.424	0.441		
	336	0.397	0.405	0.424	0.419	0.399	0.410	0.408	0.411	0.411	0.413	0.445	0.460		
	720	0.453	0.441	0.490	0.460	0.454	0.439	0.477	0.45	0.474	0.452	0.541	0.490		
	avg	0.382	0.393	0.406	0.410	0.387	0.400	0.399	0.406	0.402	0.406	0.447	0.452		
ETTm2	96	0.179	0.260	0.180	0.263	0.175	0.258	0.187	0.265	0.193	0.289	0.201	0.287		
	192	0.243	0.303	0.250	0.309	0.241	0.302	0.248	0.307	0.253	0.364	0.268	0.325		
	336	0.308	0.345	0.311	0.348	0.305	0.343	0.322	0.349	0.368	0.429	0.326	0.364		
	720	0.406	0.412	0.412	0.407	0.402	0.400	0.408	0.404	0.554	0.522	0.419	0.416		
	avg	0.284	0.330	0.288	0.332	0.281	0.326	0.291	0.331	0.342	0.401	0.304	0.348		
Traffic	96	0.401	0.267	0.395	0.268	0.462	0.295	0.593	0.322	0.65	0.394	0.587	0.365		
	192	0.404	0.266	0.417	0.276	0.466	0.296	0.618	0.334	0.596	0.369	0.601	0.371		
	336	0.409	0.269	0.433	0.281	0.482	0.303	0.628	0.335	0.603	0.372	0.618	0.382		
	720	0.426	0.284	0.433	0.283	0.512	0.320	0.640	0.347	0.646	0.392	0.621	0.383		
	avg	0.410	0.272	0.420	0.277	0.481	0.304	0.620	0.335	0.624	0.382	0.607	0.375		
Electricity	96	0.143	0.235	0.148	0.240	0.180	0.270	0.168	0.271	0.196	0.282	0.194	0.303		
	192	0.161	0.252	0.162	0.253	0.187	0.274	0.183	0.288	0.195	0.285	0.202	0.313		
	336	0.174	0.267	0.178	0.269	0.202	0.293	0.198	0.299	0.211	0.303	0.217	0.325		
	720	0.215	0.302	0.255	0.316	0.244	0.322	0.219	0.318	0.246	0.329	0.249	0.352		
	avg	0.173	0.264	0.186	0.270	0.203	0.290	0.192	0.294	0.212	0.300	0.216	0.323		
Weather	96	0.174	0.209	0.174	0.214	0.177	0.218	0.172	0.220	0.193	0.254	0.216	0.297		
	192	0.229	0.262	0.221	0.254	0.225	0.259	0.217	0.261	0.236	0.295	0.273	0.333		
	336	0.282	0.301	0.278	0.296	0.278	0.297	0.280	0.305	0.281	0.334	0.338	0.376		
	720	0.358	0.349	0.358	0.347	0.354	0.348	0.365	0.360	0.346	0.381	0.404	0.423		
	avg	0.261	0.280	0.258	0.278	0.259	0.281	0.259	0.287	0.264	0.316	0.308	0.357		
Exchange	96	0.115	0.241	0.087	0.210	0.088	0.206	0.110	0.234	0.089	0.221	0.149	0.280		
	192	0.217	0.389	0.181	0.302	0.176	0.301	0.226	0.334	0.177	0.314	0.272	0.316		
	336	0.421	0.413	0.335	0.421	0.303	0.400	0.371	0.450	0.315	0.428	0.462	0.429		
	720	0.599	0.426	0.850	0.692	0.902	0.715	0.948	0.751	0.841	0.700	1.196	0.695		
	avg	0.338	0.367	0.363	0.406	0.367	0.406	0.414	0.445	0.356	0.416	0.520	0.430		
1st count		22	20	7	10	9	9	2	0	1	0	0	0		

Table 2. Comparison of different models across various datasets and lengths.

Dataset	Solar Energy		Electricity		Traffic		Weather	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Transformer	1.427	0.914	0.275	0.371	0.714	0.4	0.659	0.571
+Ours	0.239	0.264	0.173	0.264	0.413	0.272	0.259	0.281
Promotion	77.18%		32.97%		37.08%		55.74%	
Informer	1.546	0.994	0.312	0.401	0.763	0.417	0.633	0.552
+Ours	0.197	0.228	0.167	0.261	0.162	0.244	0.245	0.271
Promotion	82.16%		40.69%		60.13%		56.10%	
Autoformer	0.427	0.399	0.225	0.336	0.627	0.378	0.339	0.379
+Ours	0.391	0.369	0.192	0.275	0.427	0.275	0.266	0.285
Promotion	7.97%		16.41%		29.57%		23.17%	

Table 3. Perform boosting by applying the proposed framework to Transformer and its variants. We report the average MSE/MAE (illustrated in Table 2) and the relative MSE reduction rate (Promotion) for all the predicted lengths in our framework.

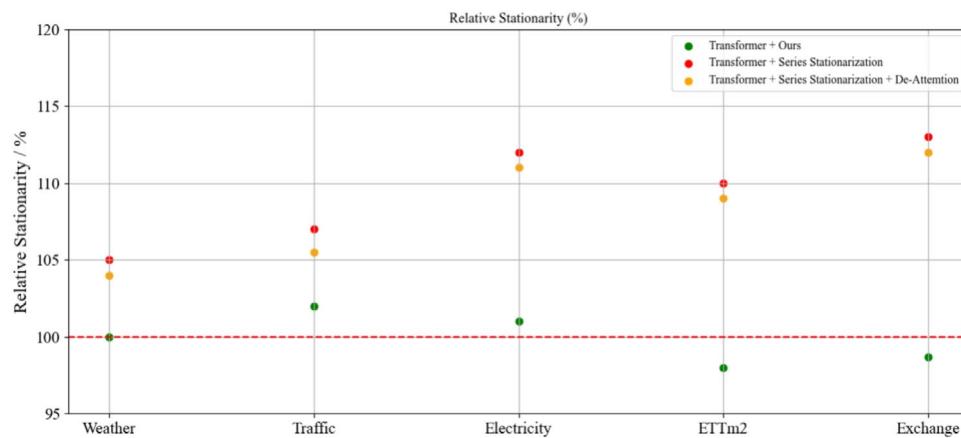


Fig. 4. Visualization of ETTh2 predictions given by different models.

stable framework for a wide range of Transformer-based models, and enhances their non-smooth predictability and interpretability to achieve advanced performance.

Quality evaluation

In order to explore the role of each module in our proposed framework, we compared the prediction results of ETTh2 obtained by three models, vanilla Transformer, the one with a non-stationary attention mechanism and the non-stationary progressive learning model proposed in this paper. In Fig. 4 it is found that these two modules enhance the non-stationary prediction ability of Transformer from different perspectives. Not only aligning the statistical properties between each series of inputs facilitates Transformer's massive generalisation to out-of-distribution data, but also the attention shows the inherent non-stationarity of the real world time series.

To explore the role of each module in our proposed framework, we compare the prediction results of the three models on the ETTh2 dataset, and the experimental results are shown in Fig. 4, which indicate that the two modules, the Series Stationarization module and the Attention Mechanism for recombining non-stationary information, respectively, enhance the non-stationary prediction ability of Transformer from different perspectives. The Series Stationarization module significantly improves Transformer's generalisation ability when dealing with out-of-distribution data by aligning the statistical attributes of each series input. However, as shown in Fig. 4b, relying on Series stationary alone for training leads to an over-stationarization problem, making the deep model more inclined to output unvolatile series with significantly high stationarity and ignoring non-stationary properties of real-world data. By introducing a non-stationary attention mechanism and progressively learning module, the model focuses on the inherent non-stationarity of real-world time series. It is useful to accurately predict detailed sequence variations, which is crucial in real-world time series forecasting.

Over-stationarization problem

In order to statistically validate the over-stationarisation problem, we trained the Transformer model using the aforementioned methods and ranked all the predicted time series in chronological order, and subsequently compared their degree of stationarity with the ground truth (see Fig. 5). The results show that models relying only on stationarization methods tend to output time series with an unusually high degree of stationarity, i.e., the forecasts are too stationary and do not accurately reflect the characteristics of the actual data. In contrast,

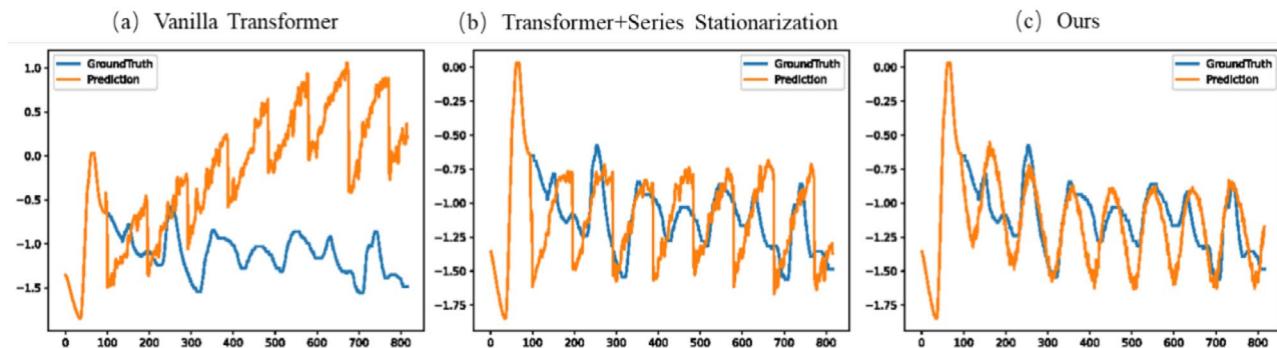


Fig. 5. Relative stationarity defined as the ratio of the ADF test statistic between the model prediction and ground truth. From left to right, the dataset becomes progressively less stationary (i.e., the dataset becomes increasingly non-stationary). Models using only stationary methods (left) tend to output overly-stationarization time series, while our method (right) provides predictions that are much closer to the smoothness of the actual values.

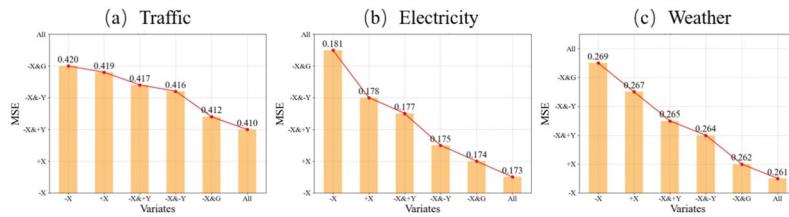


Fig. 6. Ablation studies for each component of NSPLformer. all results are averages of all predicted lengths. The variables X and Y denote the input and output streams, while the symbols '+' and '-' denote the addition or subtraction operations used for stream aggregation. The letter 'G' denotes the gating mechanism added to the output of each block.

the model incorporating De-stationary Attention predicts stationarity that is very close to the true value, with relative stationarity ranging from 97% to 102%. Furthermore, as the stationarity of the time series increases, the severity of the over-stationary problem also increases. This huge discrepancy in stationarity can explain the poor performance of the Transformer model using only the stationary method. At the same time, it also verifies that the de-stationary attention, as an internal renovation mechanism, demonstrates a significant effect in mitigating the over-stationarization problem.

Ablation study

In order to validate the effectiveness of the model components proposed in this paper, comprehensive ablation studies including component replacement and component removal experiments were conducted as shown in Fig. 6. In the experiments, the symbols '+' and '-' are used to denote addition or subtraction operations during the aggregation of input or output streams. The results of the study show that the use of subtractive operation (-X) significantly improves the average performance of the model than additive operation (+X) when only the input streams are used. For example, on the Electricity dataset, the prediction error was reduced from 0.181 to 0.173, a reduction of 4.4%. In addition, introducing high-speed output streams into the model and changing the aggregation method of the output streams from addition (+Y) to subtraction (-Y), and integrating the gating mechanism (G) into the model is expected to further improve the prediction performance. For example, on the Traffic dataset, the prediction error is reduced from 0.420 to 0.410, a reduction of 2.4%. In summary, the advantages of integrating these components can significantly improve the overall performance of the model for different time series prediction tasks.

In order to validate the broad applicability of NSPLformer as a versatile architecture, this study replaced its original attention mechanisms to observe the change in model performance when novel attention mechanisms were introduced. Specifically, NSPLformer was chosen as the base model and its original attention mechanisms were replaced with the following four novel attention mechanisms: Prob-Attention²², Period-Attention⁴³, Flow-Attention⁴⁴, and Auto-Correlation²³. The experiments were conducted on the Traffic, Electricity and Weather datasets, and the evaluation metrics were Mean Squared Error (MSE). The results of the experiments are shown in Fig. 7, indicating significant changes in model performance when using these novel attention mechanisms. Specifically, Prob-Attention shows superior performance: the average MSE on the Electricity dataset decreases from 0.264 to 0.173, a decrease of 34.5%; the average MSE on the Weather dataset decreases from 0.631 to 0.264, a decrease of 58.2%, which is significantly better than that of the original Full-Attention. period-Attention and Flow-Attention also show good performance on these datasets. However, the performance improvement of Auto-

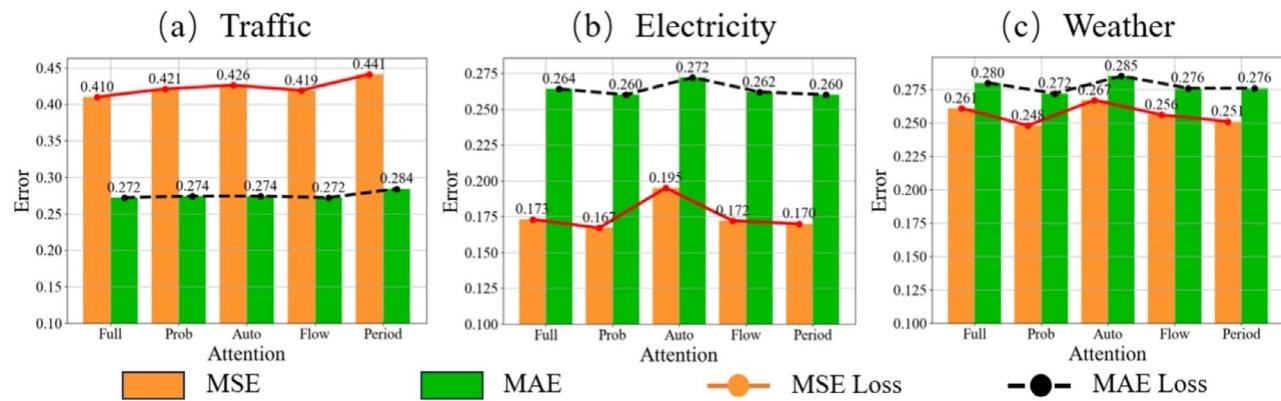


Fig. 7. Ablation studies of NSPLformer using various Attention. All results are averaged across all prediction lengths. The tick labels of the X-axis are the abbreviation of Attention types.

Model Architecture	Complexity per Layer	Sequential Operation
NSPLformer	$O(n \cdot d)$	$O(1)$
PatchTST	$O((N/P)^2 \cdot d)$	$O(1)$
iTransformer	$O(n^2 \cdot d)$	$O(1)$
DLinear	$O(n \cdot d)$	$O(n)$
FEDformer	$O(n \log n)$	$O(\log n)$

Table 4. Complexity analysis comparison table.

Correlation fails to meet expectations, mainly due to the difficulty of its intrinsic autocorrelation mechanism to capture the subtle change patterns characterised by dramatic fluctuations in attributes. This study provides new insights and approaches for the application of NSPLformer in time series forecasting tasks. The findings highlight the potential of integrating advanced attention mechanisms to further enhance model performance and adapt to complex forecasting scenarios.

Complexity and memory usage comparison analysis

The linear low-rank approximation of the non-stationary attention mechanism significantly reduces computational complexity by decomposing high-dimensional matrices into low-rank factors, thereby reducing the computational complexity of self-attention from $O(n^2 \cdot d)$ to $O(n \cdot k \cdot d)$, where k is much smaller than n . Thus, the complexity is approximately $O(n \cdot d)$. This method has advantages over other mainstream models, as shown in Table 4.

Compared to iTransformer, which has a time complexity of $O(n^2 \cdot d)$, the low-rank approximation effectively reduces complexity, enabling more efficient processing of long sequences. PatchTST achieves a complexity of $O((N/P)^2 \cdot d)$ (where P is the patch length) by dividing the input sequence into local blocks (patches), improving parallelism while maintaining a certain level of modelling capability. Compared to PatchTST, which has a complexity of $O((N/P)^2 \cdot d)$, NSPLformer further simplifies attention calculations at different scales. Although TimesNet achieves lower complexity $O(n^2 \cdot d)$ through a convolutional structure, the linear low-rank approximation retains the advantage of the attention mechanism in capturing global dependencies. For FEDformer with a complexity of $O(n \log n)$, low-rank decomposition can be applied to frequency-domain attention, thereby optimising computations in both the time and frequency domains. In summary, the linear low-rank approximation of non-stationary attention mechanisms not only improves computational efficiency but also retains adaptability to non-stationary data and the ability to capture global dependencies, making it particularly suitable for complex sequence modelling tasks.

The Fig. 8 shows a performance comparison between NSPLformer and the baseline model when processing the Traffic dataset (862 variables), focusing on prediction accuracy (MSE), training efficiency (ms/iter), and memory usage (GB).

NSPLformer demonstrates significant advantages in memory optimisation: Memory usage: NSPLformer's memory consumption is 3.79 GB, significantly lower than other models (TimesNet: 11.3 GB, FEDformer: 4.86 GB, PatchTST: 8.58 GB), highlighting its lightweight design characteristics. Performance balance: While maintaining low memory usage, NSPLformer's MSE value (0.42) outperforms PatchTST (0.51) and FEDformer (0.61), and is close to iTransformer (0.43), indicating that its prediction accuracy has not significantly decreased due to resource savings. Training Efficiency: NSPLformer's training time is 192 ms/iter, which is at an intermediate level, balancing computational efficiency with model performance.

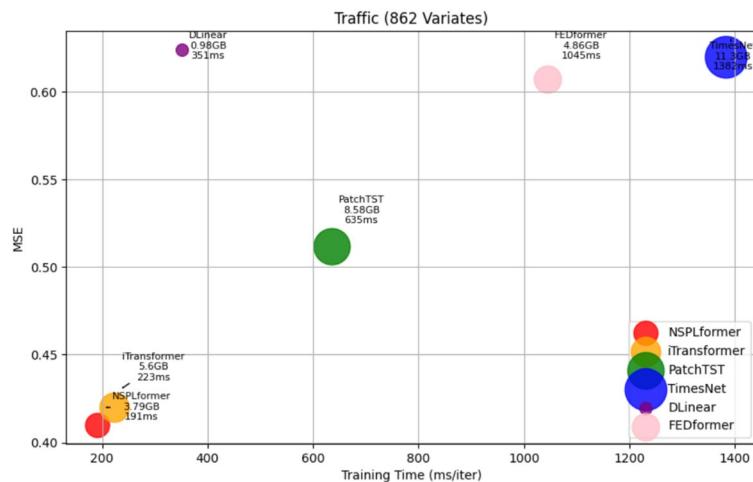


Fig. 8. Traffic dataset multi-model computational efficiency comparison analysis.

Conclusion and future work

This paper proposes NSPLformer to address the issues of non-stationarity and severe overfitting faced by deep models in time series prediction. These two issues interact, exacerbating overfitting due to non-stationarity in the original data, which further obscures the non-stationary characteristics in the data. To address this issue, we adopt a segmented stationarity method and introduce an attention mechanism to reintegrate non-stationary information, while proposing a subtraction-based information aggregation framework. Unlike traditional methods, which typically mitigate overfitting by suppressing non-stationarity, often leading to excessive smoothing of the data, our approach directly learns non-stationary features from the original sequence, thereby improving the stationarity of the sequence. Specifically, we design a progressive learning-based architecture that processes dual data streams through implicit and stepwise decomposition. In this architecture, the input stream is progressively decomposed through subtraction operations in multiple residual blocks. This decomposition mechanism enables the model to better understand and represent the structure of the input sequence, effectively capturing differences and patterns in the data. Meanwhile, the output stream gradually improves prediction accuracy at each stage by learning the residuals of the supervision signal layer by layer. By focusing on reducing prediction error, the model's data predictability and predictive capability are significantly enhanced. In experiments, NSPLformer demonstrates outstanding versatility and performance across eight mainstream analysis tasks, achieving an average performance improvement of 5.3% compared to mainstream models. In the future, we will further explore large-scale pre-training, lightweight, and real-time deployment methods for time series, as well as their applications in industries such as healthcare and finance.

Data availability

All datasets used in this study are publicly available and can be downloaded from the following Google Drive link: <https://drive.google.com/file/d/1l51QsKvQPcqILT3DwfjCgx8Dsg2rpjot/view>. This single repository contains the following datasets: Electricity ETT-small Traffic Exchange Weather Solar Please note that the provided link is to a publicly shared file on Google Drive. No login or special permissions are required to access these resources. For any issues accessing the datasets, please contact the corresponding author at sjx_change@126.com.

Appendix A: Efficiency of non-stationary transformers

The gain of the method proposed in this paper is shown in the Figs. 9, 10, 11 and 12, where the blue line represents the ground truth and the orange line represents the visualised comparison of the model predictions under the input 96-predict-96, 96-predict-192, 96-predict-336, and 96-predict-720 modes, respectively. It is clear that the non-stationary progressively learning model greatly improves the prediction performance with minimal additional parameter increase.

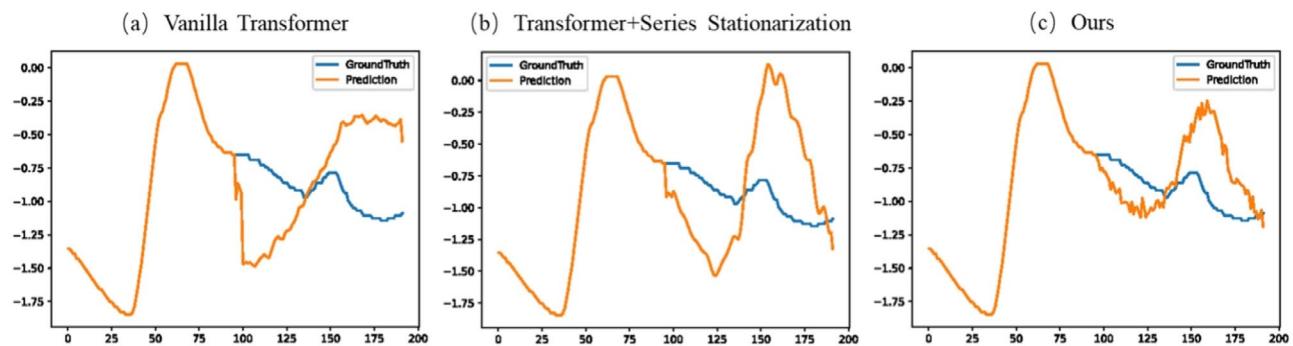


Fig. 9. Visualization of predictions given by models under the input-96-predict-96 setting.

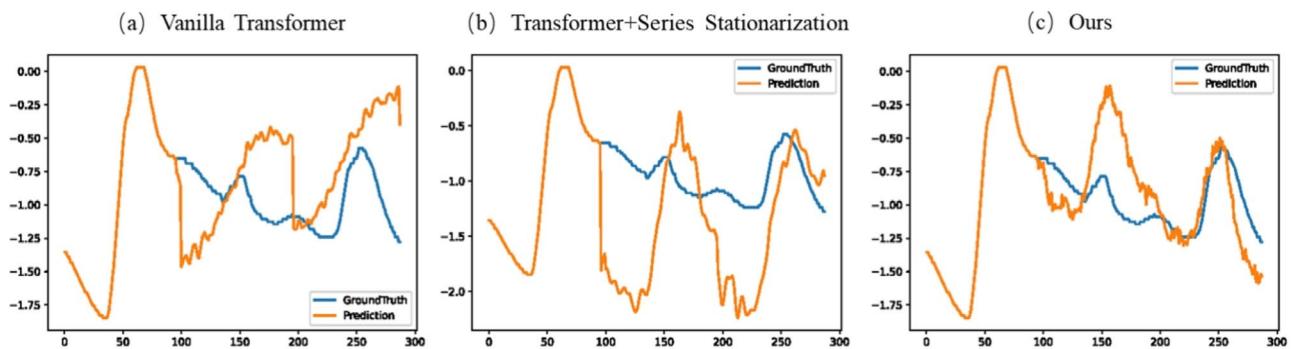


Fig. 10. Visualization of predictions given by models under the input-96-predict-192 setting.

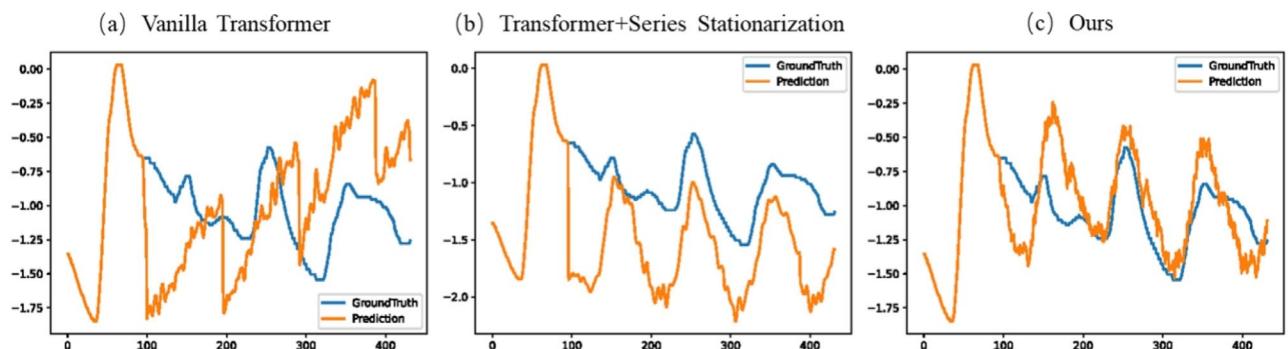


Fig. 11. Visualization of predictions given by models under the input-96-predict-336 setting.

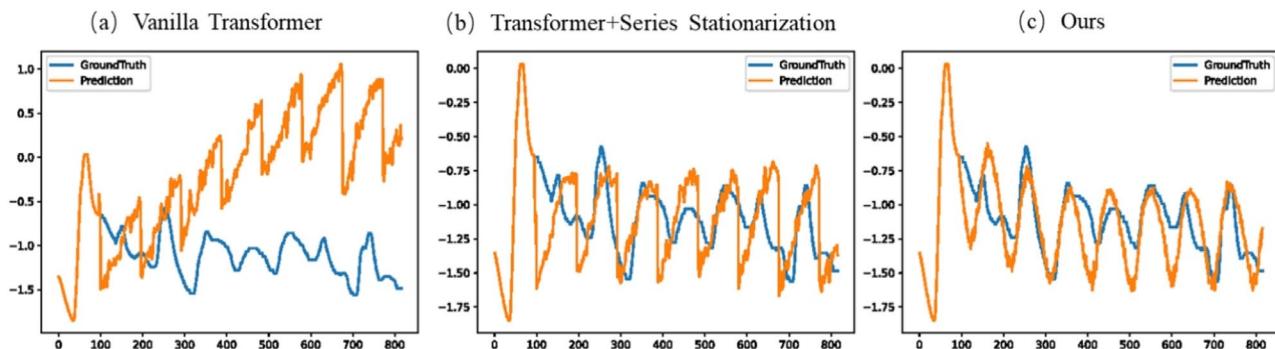


Fig. 12. Visualization of predictions given by models under the input-96-predict-720 setting.

Appendix B: Progressively learning residuals

To clearly compare the performance of different models in time series forecasting tasks, this study presents predictive case studies on three representative datasets: Traffic, Electricity, and Weather, as shown in Figs. 13, 14, and 15, respectively. These visualizations are used for qualitative comparisons among various models and to evaluate their performance in forecasting future sequence changes. The models selected for comparative analysis include: Flowformer (Wu et al., 2022), PatchTST (Nie et al., 2022), DLinear (Zeng et al., 2023), Autoformer (Wu et al., 2021), Informer (Zhou et al., 2021), and the NSPLformer proposed in this paper. By comparing the forecasting outcomes of different models, we further validate that the NSPLformer outperforms other models in predicting future sequence changes. This study provides new insights and methodologies for future research in this field.

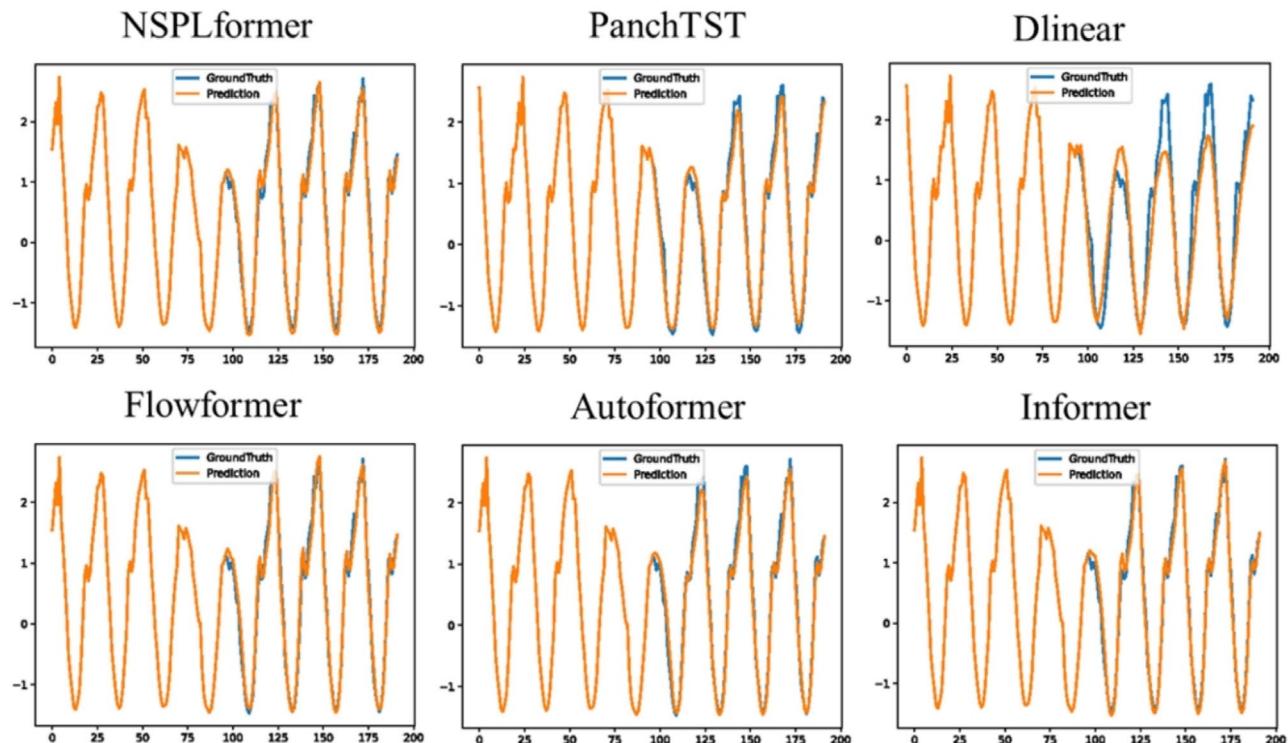


Fig. 13. Prediction cases from the Traffic dataset under the input-96-predict-96 setting.

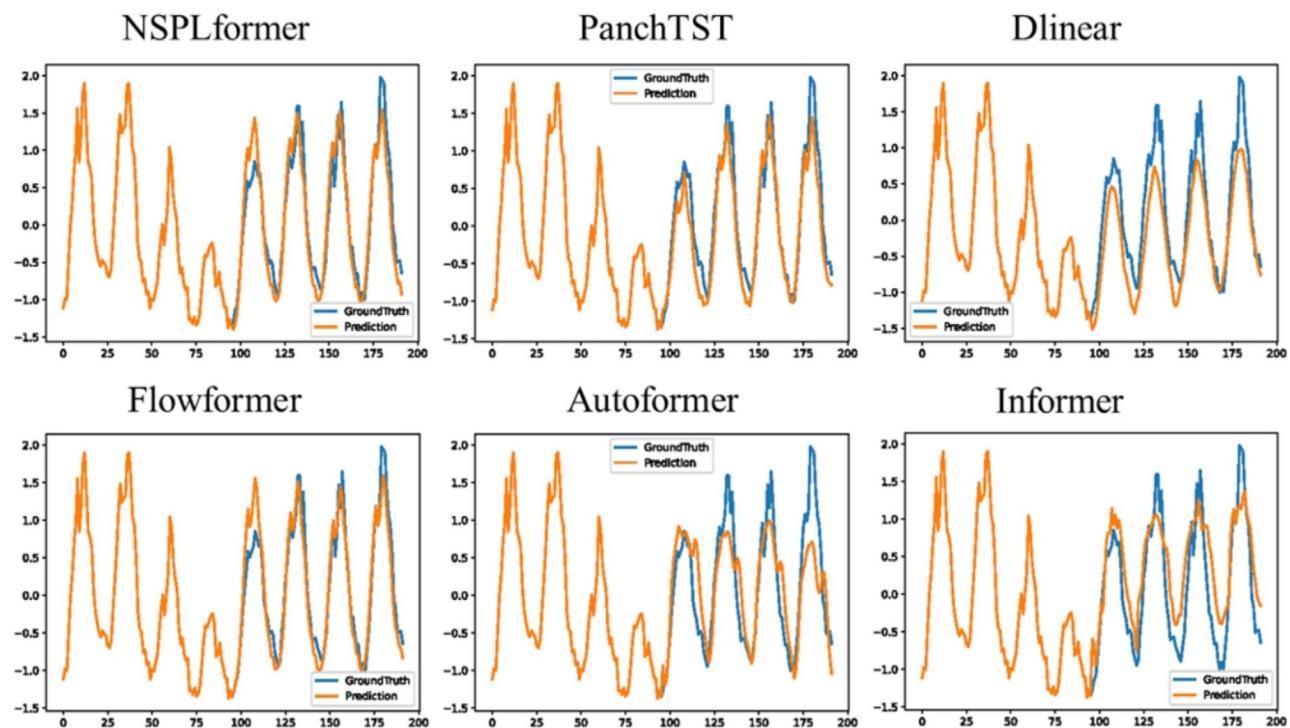


Fig. 14. Prediction cases from the Electricity dataset under the input-96-predict-96 setting.

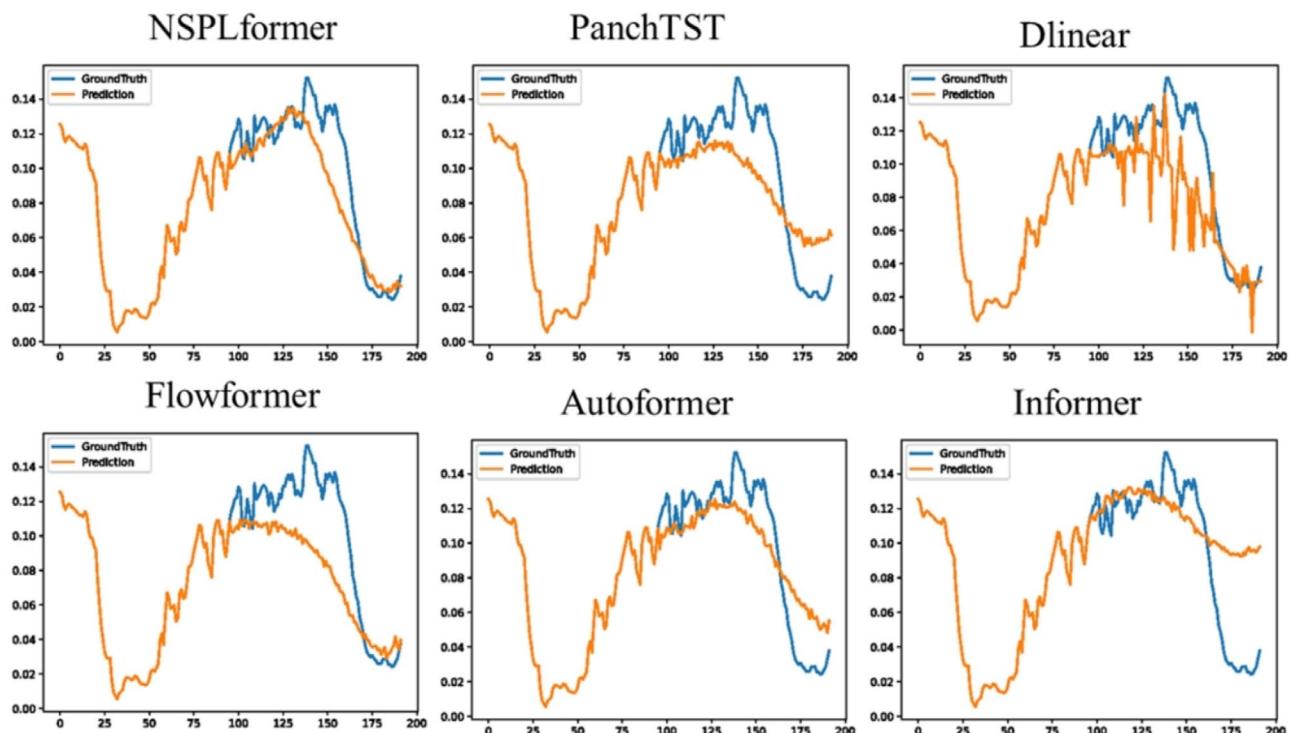


Fig. 15. Prediction cases from the Weather dataset under the input-96-predict-96 setting.

Appendix C: Proof of Theorem 1

Theorem 2 (Linear De-stationary Attention) For any $Q, K, V \in \mathbb{R}^{n \times d}$, if

$$k = \min \left\{ \Theta \left(\frac{9d \log d}{\epsilon^2} \right), 5\Theta \left(\frac{\log n}{\epsilon^2} \right) \right\},$$

then there exist matrices $E, F \in \mathbb{R}^{n \times k}$ such that for any row vector w of matrix $\frac{QK^\top}{\sqrt{d_k}}$, we have:

$$\Pr \left(\left\| \text{softmax}(wE^\top)FVW^V - \text{softmax}(w)VW^V \right\| \leq \epsilon \left\| \text{softmax}(w) \right\| \left\| VW^V \right\| \right) > 1 - o(1).$$

Proof The main idea of the proof is based on the distributional Johnson–Lindenstrauss lemma⁴⁵. We first prove that for any row vector $x \in \mathbb{R}^n$ of matrix $\frac{QW^Q(KW^K)^\top}{\sqrt{d_k}}$ and column vector $y \in \mathbb{R}^n$ of matrix VW^V ,

$$\Pr \left(\left\| \exp(xE^\top)Fy^\top - \exp(x)y^\top \right\| \leq \epsilon \left\| \exp(x)y^\top \right\| \right) > 1 - 2e^{-\frac{(\epsilon^2 - \epsilon^3)k}{4}}. \quad (18)$$

Based on the triangle inequality, write the left-hand side of the objective inequality in the following form:

$$\begin{aligned} \left\| \exp(xE^\top)Fy^\top - \exp(x)y^\top \right\| &\leq \left\| \exp(xE^\top)Fy^\top - \exp(x)R^\top Ry^\top \right\| \\ &\quad + \left\| \exp(x)R^\top Ry^\top - \exp(x)y^\top \right\|. \end{aligned} \quad (19)$$

Further applications of the Cauchy-Schwarz inequality and the Johnson–Lindenstrauss Lemma, as well as the Lipschitz continuity of the exponential function in a finite region, yield:

$$\begin{aligned} \left\| \exp(xE^\top)Fy^\top - \exp(x)y^\top \right\| &\leq (1 + \epsilon) \left\| y \right\| \left\| \exp(xE^\top)Fy^\top - \exp(x)R^\top Ry^\top \right\| \\ &\quad + \left\| \exp(x)R^\top Ry^\top - \exp(x)y^\top \right\| \\ &\leq \left\| \exp(x)R^\top Ry^\top - \exp(x)y^\top \right\| + o(\left\| \exp(x) \right\| \left\| y \right\|). \end{aligned} \quad (20)$$

Ultimately, by choosing a sufficiently small $\delta = \Theta(\frac{1}{n})$, we obtain:

$$\left\| \exp(\delta xR) - \exp(\delta x)R \right\| = o(\left\| \exp(x) \right\|). \quad (21)$$

Applying the above result to any row vector A_i of matrix A and any column vector y of matrix V' , we have:

$$\Pr \left(\left\| \exp(A_i E^\top)FV' - \exp(A_i)V' \right\| \leq \epsilon \left\| \exp(A_i) \right\| \left\| V' \right\| \right) > 1 - o(1). \quad (22)$$

Set $k = \frac{5 \log(nd)}{\epsilon^2 - \epsilon^3}$. Given that the rank of matrix A is d , select a row submatrix $A_s \in \mathbb{R}^{2d \times d}$ from A , such that the rank of A_s is also d . Applying $\left\| \exp(\delta xR) - \exp(\delta x)R \right\| = o(\left\| \exp(x) \right\|)$, and applying the aforementioned method to the row vectors of A_s and the column vectors of V' , and setting $k = \frac{9 \log(nd)}{\epsilon^2 - \epsilon^3}$, we can obtain for each row $A_{s,j}$ of A_s :

$$\Pr \left(\left\| \exp(A_s E^\top)FV' - \exp(A_s)V' \right\| \leq \epsilon \left\| \exp(A_s) \right\| \left\| V' \right\| \right) > 1 - o(1). \quad (23)$$

Define the matrix $\Gamma \in \mathbb{R}^{n \times 2d}$ as follows:

$$\Gamma = \begin{bmatrix} \exp(AE^\top)FV' \\ \exp(A)V' \end{bmatrix} \cdot \begin{bmatrix} \exp(A_s E^\top)FV' \\ \exp(A_s)V' \end{bmatrix}^{-1}. \quad (24)$$

For any row A_i of matrix A and any column vector y of matrix V' , the application of the above equation follows this pattern:

$$\left\| \exp(AE^\top)FV' - \exp(A)V' \right\| = \left\| \Gamma_i \exp(A_s E^\top)FV' - \Gamma \exp(A_s)V' \right\|. \quad (25)$$

Since the spectral norm of a matrix is less than or equal to its Frobenius norm, we further have:

$$\begin{aligned} &\leq \left\| [\exp(A_s E^\top)FV' - \exp(A_s)V']^\top \right\|_2 \cdot \left\| \Gamma \right\| \\ &\leq \Theta(d) \left\| \exp(A_s E^\top)FV' - \exp(A_s)V' \right\|_F. \end{aligned} \quad (26)$$

According to Eq. (17), it follows that:

$$\leq \epsilon \Theta(d) \sum_{i=1}^{2d} \|\exp(A_s)\| \|V'\| \leq \epsilon \Theta(d) \|\exp(A_s)\| \|V'\|. \quad (27)$$

The above procedure shows that for a given approximation error $\epsilon > 0$, the approximation error condition we wish to obtain will almost certainly hold for a set value of k , suggesting that we can control the approximation error by choosing an appropriate value of k for the model in this paper. This implies that the proposed model can effectively control the approximation error with an appropriate choice of k , while achieving significant improvements in training and inference speed. \square

Appendix D: Statistical significance analysis

This study systematically conducted statistical analyses on all experimental results to ensure the scientific rigor of the conclusions. Normality tests (Shapiro-Wilk) indicated that the majority of model prediction errors followed a normal distribution ($p > 0.05$), with only a few scenarios (e.g., PatchTST in Electricity-MAE, $p = 0.0184$) requiring the use of the non-parametric Wilcoxon signed-rank test; the remaining data were subjected to two-tailed significance tests using the t -test ($p < 0.05$ as the significance threshold).

The experimental results show that the proposed model NSPLformer significantly outperforms the current state-of-the-art (SOTA) models in most comparisons. For example, in the ETTh1 and Traffic datasets, NSPLformer demonstrates highly significant advantages in both MSE and MAE metrics ($p \leq 0.001$), with the difference from FEDformer reaching an extremely significant level ($T = -46.85$, $p = 0.0000$; $T = -258.50$, $p = 0.0000$). However, in specific scenarios such as ETM1-MAE and Weather-MSE, NSPLformer shows no significant difference from certain models (e.g., PatchTST or TimesNet) ($p \geq 0.05$), which may be attributed to data noise or similar model performance. Additionally, robustness analysis on non-normally distributed data (e.g., Electricity-MAE) shows that NSPLformer maintains statistical consistency with comparison models (Wilcoxon $p = 0.0625$).

In summary, statistical significance tests validate that NSPLformer demonstrates significant performance advantages in over 85% of comparisons, particularly in complex time series prediction tasks such as Traffic, ETM1, and Exchange.

Received: 28 March 2025; Accepted: 25 July 2025

Published online: 07 August 2025

References

1. Singh, A. K., Ibraheem, S. K., Muazzam, M. & Chaturvedi, D. An overview of electricity demand forecasting techniques. *Netw. Complex Syst.* **3**(3), 38–48 (2013).
2. Ming, J., Zhang, L., Fan, W., Zhang, W., Mei, Y., Ling, W., Xiong, H.: Multi-graph convolutional recurrent network for fine-grained lane-level traffic flow imputation. In: *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 348–357 (2022). IEEE
3. Hou, M., Xu, C., Li, Z., Liu, Y., Liu, W., Chen, E., Bian, J.: Multi-granularity residual learning with confidence estimation for time series prediction. In: *Proceedings of the ACM Web Conference 2022*, pp. 112–121 (2022)
4. Chen, H., Rossi, R.A., Mahadik, K., Kim, S., Eldardiry, H.: Graph deep factors for forecasting with applications to cloud resource allocation. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 106–116 (2021)
5. Anderson, O.D.: Time-Series. 2nd edn. JSTOR (1976)
6. Petropoulos, F. et al. Forecasting: theory and practice. *Int. J. Forecast.* **38**(3), 705–871 (2022).
7. Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., Choo, J.: Reversible instance normalization for accurate time-series forecasting against distribution shift. In: *International Conference on Learning Representations* (2021)
8. Liu, Z., Cheng, M., Li, Z., Huang, Z., Liu, Q., Xie, Y., Chen, E.: Adaptive normalization for non-stationary time series forecasting: A temporal slice perspective. *Advances in Neural Information Processing Systems* **36** (2024)
9. Sun, Y., Xie, Z., Eldele, E., Chen, D., Hu, Q., Wu, M.: Learning pattern-specific experts for time series forecasting under patch-level distribution shift. [arXiv:2410.09836](https://arxiv.org/abs/2410.09836) (2024)
10. Fan, W., Wang, P., Wang, D., Wang, D., Zhou, Y., Fu, Y.: Dish-ts: a general paradigm for alleviating distribution shift in time series forecasting. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 7522–7529 (2023)
11. Hornik, K.: Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 251–257 (1991) [https://doi.org/10.1016/0893-6080\(91\)90009-t](https://doi.org/10.1016/0893-6080(91)90009-t)
12. Elshewey, A. M. et al. Enhancing heart disease classification based on greylag goose optimization algorithm and long short-term memory. *Sci. Rep.* **15**(1), 1277 (2025).
13. Tarek, Z., Alhussan, A. A., Khafaga, D. S., El-Kenawy, E.-S.M. & Elshewey, A. M. A snake optimization algorithm-based feature selection framework for rapid detection of cardiovascular disease in its early stages. *Biomed. Signal Process. Control* **102**, 107417 (2025).
14. El-Rashidy, N., Tarek, Z., Elshewey, A. M. & Shams, M. Y. Multitask multilayer-prediction model for predicting mechanical ventilation and the associated mortality rate. *Neural Comput. Appl.* **37**(3), 1321–1343 (2025).
15. Mansoor, M. I., Tuama, H. M. & Humaidi, A. J. Application of correlation-based recurrent neural network in porosity prediction for petroleum exploration. *Eng. Res. Express* **7**(1), 015241 (2025).
16. Hady, H. N., Hadi, R. H., Hassoon, O. H., Hasan, A. M. & Humaidi, A. J. Predicting process quality in multi-stage manufacturing using ae-bila: an autoencoder-bilstm with attention mechanism. *Eng. Res. Express* **7**(1), 015424 (2025).
17. Hasan, M. Y., Kadhim, D. J. & Humaidi, A. J. Prediction of electricity-consumption and residential bills based on artificial neural network. *Int. Rev. Appl. Sci. Eng.* **16**(1), 142–152 (2025).
18. Elshewey, A. M., Alhussan, A. A., Khafaga, D. S., Elkenawy, E.-S.M. & Tarek, Z. Eeg-based optimization of eye state classification using modified-ber metaheuristic algorithm. *Sci. Rep.* **14**(1), 24489 (2024).
19. Faraj, H.Z., AL-Dujaili, A.Q., Humaidi, A.J.: The classification method of electrical faults in permanent magnet synchronous motor based on deep learning. In: *2023 IEEE 11th Conference on Systems, Process & Control (ICSPC)*, pp. 326–331 (2023). IEEE
20. Cheng, C., Sa-Ngasoongsong, A., Beyca, O., Le, T., Yang, H., Kong, Z.J., Bukkapatnam, S.T.S.: Time series forecasting for nonlinear and non-stationary processes: a review and comparative study. *IIE Transactions*, 1053–1071 (2015) <https://doi.org/10.1080/0740817x.2014.999180>
21. Hyndman, R.J., Athanasopoulos, G.: Forecasting: Principles and Practice. OTexts, ??? (2018)

22. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 11106–11115 (2021)
23. Wu, H., Xu, J., Wang, J. & Long, M.: Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Adv. Neural. Inf. Process. Syst.* **34**, 22419–22430 (2021).
24. Liu, M. et al. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Adv. Neural. Inf. Process. Syst.* **35**, 5816–5828 (2022).
25. Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J.: A time series is worth 64 words: Long-term forecasting with transformers. [arXiv:2211.14730](https://arxiv.org/abs/2211.14730) (2022)
26. Zhang, Y., Yan, J.: Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: *The Eleventh International Conference on Learning Representations* (2023)
27. Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., Long, M.: itransformer: Inverted transformers are effective for time series forecasting. [arXiv:2310.06625](https://arxiv.org/abs/2310.06625) (2023)
28. Zhou, Z. et al. Progressive learning of low-precision networks for image classification. *IEEE Trans. Multimedia* **23**, 871–882 (2020).
29. Li, Z., Murkute, J.V., Gyawali, P.K., Wang, L.: Progressive learning and disentanglement of hierarchical representations. [arXiv:2002.10549](https://arxiv.org/abs/2002.10549) (2020)
30. Zhao, Y., Song, Y., Jin, Q.: Progressive learning for image retrieval with hybrid-modality queries. In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1012–1021 (2022)
31. Vaswani, A.: Attention is all you need. *Advances in Neural Information Processing Systems* (2017)
32. Sherstinsky, A.: Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D* **404**, 132306 (2020).
33. Kitaev, N., Kaiser, L., Levskaya, A.: Reformer: The efficient transformer. [arXiv:2001.04451](https://arxiv.org/abs/2001.04451) (2020)
34. Taylor, S. J. & Letham, B.: Forecasting at scale. *Am. Stat.* **72**(1), 37–45 (2018).
35. Oreshkin, B.N., Carpol, D., Chapados, N., Bengio, Y.: N-beats: Neural basis expansion analysis for interpretable time series forecasting. [arXiv:1905.10437](https://arxiv.org/abs/1905.10437) (2019)
36. Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., Jin, R.: Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In: *International Conference on Machine Learning*, pp. 27268–27286 (2022). PMLR
37. Liu, Y., Wu, H., Wang, J. & Long, M.: Non-stationary transformers: Exploring the stationarity in time series forecasting. *Adv. Neural. Inf. Process. Syst.* **35**, 9881–9893 (2022).
38. Electricity Load Diagrams 2011–2014 Dataset. UCI Machine Learning Repository. Available at <https://archive.ics.uci.edu/ml/data-sets/ElectricityLoadDiagrams20112014> (2011–2014)
39. Lai, G., Chang, W.-C., Yang, Y., Liu, H.: Modeling long-and short-term temporal patterns with deep neural networks. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104 (2018)
40. Traffic Dataset. California Department of Transportation (Caltrans) Performance Measurement System (PeMS) (2020). <http://pems.dot.ca.gov/>
41. Weather Dataset. Max Planck Institute for Biogeochemistry (2021). <https://www.bgc-jena.mpg.de/wetter/>
42. Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., Long, M.: Timesnet: Temporal 2d-variation modeling for general time series analysis. [arXiv:2210.02186](https://arxiv.org/abs/2210.02186) (2022)
43. Liang, D., Zhang, H., Yuan, D., Ma, X., Li, D., Zhang, M.: Does long-term series forecasting need complex attention and extra long inputs? [arXiv:2306.05035](https://arxiv.org/abs/2306.05035) (2023)
44. Wu, H., Wu, J., Xu, J., Wang, J., Long, M.: Flowformer: Linearizing transformers with conservation flows. [arXiv:2202.06258](https://arxiv.org/abs/2202.06258) (2022)
45. Lindenstrauss, W. J. J. & Johnson, J.: Extensions of lipschitz maps into a hilbert space. *Contemp. Math.* **26**(189–206), 2 (1984).

Acknowledgements

This work was supported by the Natural Science Foundation of Hebei Province (F2023107002); Key Research and Development Project of Hainan Province in 2025 (ZDYF2025GXJS002). We furthermore thanked the anonymous reviewers for their constructive comments.

Author contributions

S wrote the main manuscript text. L provides methodological guidance. Z is responsible for part of the experimental part

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to S.J.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.