Neurocomputing 646 (2025) 130398

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

MPMixer: Multi-scale Patch Mixer for long-term time series forecasting

Yijun Xu [a], Weijie Zhang [a], Huixia Lai [a], Shi Zhang [a,b,*], Ruliang Xiao [a,b]

[a] The College of Computer and Cyber Security, Fujian Normal University, Fuzhou, 350117, Fujian, China
[b] The Digit Fujian Internet-of-Things Laboratory of Environmental Monitoring, Fujian Normal University, Fuzhou, 350117, Fujian, China

ARTICLE INFO

Communicated by P. Hájek

Keywords:
Time series forecasting
Deep learning
Mixing networks
Patch

ABSTRACT

Multi-scale-based time series forecasting (TSF) models transform 1-dimensional time series into 2-dimensional ones and make predications. They make it easier to capture implicit information in time series. However, the existing methods ignore the correlation between series with different scales. This paper proposes a model, named Multi-scale Patch Mixer (MPMixer), for Long-Term Time Series Forecasting. MPMixer is a double-ended prediction model, where one end analyzes the correlation within multi-scale series and predicts, the other one analyzes the correlation between patches from multi-scale series and makes predictions. The final forecasting results are achieved by summarizing predications generated by two ends. The model first downsamples the raw time series and obtains the multi-scale series. While analyzing and predicting based on the inter multi-scale series, the multi-scale series are patched into a 2-dimensional patch series. After that, the multi-scale patch encoder analyzes patch series and makes predictions. In the other end, analyzing and predicting based on the intra multi-scale series, the model first decomposes the multi-scale series into trends and seasons, then constructs their patch sequences and generates predictions using a linear module respectively. In the experiments, MPMixer achieves state-of-the-art on 9 challenging real-world datasets, which verifies the rationality of our model. We also analyze the impact of downsampling setting, patch size, and lookback length on performance, and conduct ablation experiments to verify the importance of each component. Code is available at https://github.com/Xoxyer/MPMixer.

1. Introduction

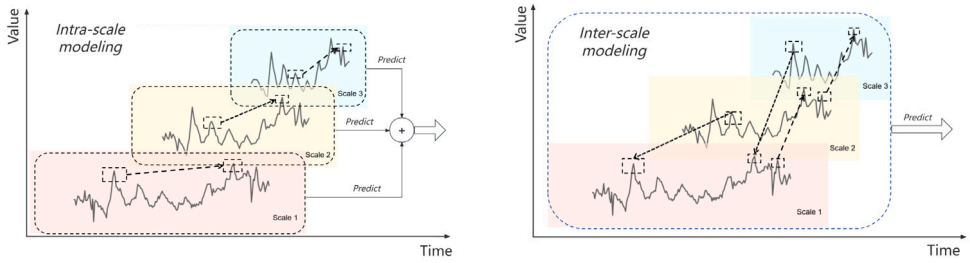
Time series forecasting (TSF) aims to predict future trends, values, or patterns of change based on available time series data. Long-term TSF provides longer foresight and helps to perform necessary resource scheduling or fault alarms, thus improving system efficiency and ensuring system safety. In recent years, TSF has played a crucial role, and is used in many fields, such as weather forecasting [1], stock market [2], traffic flow [3], smart cities [4], healthcare [5], and etc.

Applying deep learning to TSF tasks, researchers propose numerous deep learning-based TSF models, and achieve outstanding performances [6–14]. Among them, the Transformer-based TFS models [11,12] capture time series dependencies using the attention mechanism and generate excellent prediction results in long-term time series forecasting. However, the Transformer-based TSF models still have some issues that need to be further concerned. For example, most of the Transformer-based TSF models predict on the raw time series directly [15,16] without considering that multi-scale series help to reveal change patterns with different durations. Some researchers build prediction models based on multi-scale time series (as shown in Fig. 1(a)), but ignore the correlation between series with different scales [17,18].

To address these issues, we propose a Multi-scale Patch Mixer time series forecasting model (MPMixer). The model is a double-ended forecasting model (intra-scale-based analysis and forecasting, and inter-scale-based analysis and forecasting), where one end analyzes and forecasts based on intra series correlation and the other one analyzes and forecasts based on inter multi-scale series correlation, and finally outputs the summation as the long-term forecast result. The model first downsamples the raw time series and obtains the multi-scale series. While analyzing and predicting based on the inter multi-scale series, the model patches them into a patch series. After that, the multi-scale patch encoder analyzes patch series and makes predictions. In time of intra-scale-based analyzing and forecasting, the model first decomposes the series into trends and seasons, then constructs their multi-scale series and generates predictions using a linear module respectively. The model downsamples the raw time series and generates multi-scale

* Corresponding author.
E-mail addresses: xoxygen5@163.com (Y. Xu), zwj19891306@163.com (W. Zhang), lhx@fjnu.edu.cn (H. Lai), shi@fjnu.edu.cn (S. Zhang), xiaoruliang@fjnu.edu.cn (R. Xiao).

https://doi.org/10.1016/j.neucom.2025.130398
Received 4 November 2024; Received in revised form 8 April 2025; Accepted 30 April 2025
Available online 26 May 2025



(a) Predicting using the intra sequence correlations (b) Also considering the inter sequence correlations

Fig. 1. Illustrating the differences of the existing multi-scale TSF models and the idea of ours.

sequences, which provide the model with data from fine to coarse granularity. Secondly, by introducing the Multi-scale Patch Encoder (MPE) module, the model can better focus on the correlations between multi-scale sequences (as shown in Fig. 1(b)). Finally, by incorporating the patching idea, local information is consolidated within a single patch, effectively reducing the computational burden. The main contributions of this paper include:

(1) We proposed a Multi-scale Patch Mixer time series forecasting model (MPMixer) based on a double-ended forecasting model. One end forecasts based on intra-scale series, and the other end forecasts based on inter-scale series. To the best of our knowledge, this is one of the few models that explicitly utilizes inter-scale correlation for time series forecasting.

(2) A Multi-scale Patch Encoder (MPE) is proposed to analyze and predict time series based on correlations across multi-scale patches. By applying the patching to multi-scale series, MPE can uniformly process time series at different scales, facilitating discovering similar patterns across fine-grained sequences and coarse-grained sequences.

(3) The experiments on 9 real benchmark datasets verify the excellent performance of MPMixer. We also analyze the impact of hyperparameter settings on performance, and verify the effectiveness of modeling inter-scale series in experiments.

The rest of this paper is organized as follows: Section 2 introduces related work, Section 3 provides a detailed description of our model, Section 4 presents the experimental results and analysis, and finally, Section 5 concludes the paper.

2. Related work

In this section, we first briefly overview the classical TSF models, followed by a focus on deep learning-based TSF models. Given that our methodology involves Patch and multi-scale, we also introduce the research works in these two aspects.

Classical time series forecasting models include the Auto-Regressive model (AR), Moving Average model (MA), Auto-Regressive Moving Average Model (ARMA), VAR, and etc. These models are based on solid statistical theories. They assume that the time series is stationary, thus making it difficult to effectively capture the complex and subtle data changes in time series. With the rapid development of deep learning, more and more deep learning-based TSF models have been proposed [6–8,19,20]. They can be divided into 4 categories according to the type of neural network: the RNN-based TSF models [19], the CNN-based TSF models [20,21], the Transformer-based TSF models [16,22], and the MLP-based TSF models [14,23,24].

RNN-based TSF models: RNN-based TSF models process sequential data using a recurrent structure. The hidden states are used to store outputs from previous time steps, thereby associating past with current inputs for time series prediction. For example, DeepAR [25] improves prediction performance by forecasting the distribution of future time series and considering the impacts of amplitude across multiple sequences on the model. MQRNN [7] optimizes the NLP structure to enable predicting values for future time steps. SegRNN [26] reduces the number of iterations in RNNs with two strategies, segment-wise iterations and parallel multi-step forecasting. However, RNN-based TSF models are more suitable for capturing short-term dependencies in time series. They also tend to encounter issues with vanishing or exploding gradients in long-term TSF, making it hard to retain long-range dependencies.

CNN-based TSF models: Originally applied to image processing, CNNs have gained widespread attention in time series forecasting tasks due to their powerful feature extraction capabilities. For example, TCN [27] captures longer historical dependencies through dilated convolutions, addressing the limitations of RNNs in temporal step dependencies. MICN [21] first extracts local features from the time series and then captures the correlations between these local features to obtain global features. TimesNet [10] transforms the raw 1-dimensional time series into a 2-dimensional representation based on different periods using the Fourier transformation, and then employs convolutional kernels to capture information within and between these periods.

Transformer-based TSF models: Transformer-based time series forecasting models receive widespread attention in TSF tasks due to the powerful self-attention mechanism and the flexible parallelization capability. They capture long-term dependencies and improve forecasting performance. In recent years, many Transformer-based TSF models have been proposed. For example, the Log-Sparse Transformer [28] proposes convolutional self-attention by producing queries and keys with causal convolution, so that the local context can be better incorporated into the attention mechanism. Informer [15] proposes a probabilistic sparse self-attention mechanism to select important keys by analyzing the long-tailed distribution of attention scores. On the other hand, Autoformer [8] is proposed based on a deep decomposition architecture and an autocorrelation mechanism. It dramatically improves the efficiency of long-time prediction through progressive decomposition and sequence-level connections. FEDformer [12] transforms the input time series from the time domain to the frequency domain, and performs the Transformer operation in the frequency domain. Crossformer [11] utilizes a cross-dimensional self-attention mechanism to establish links between different temporal features. The iTransformer [22] models the correlations between different variables using the self-attention mechanism, and captures the time-series correlations within variables using a feed-forward network. In all, Transformer-based TSF models show excellent performance in TSF tasks. There is still a great potential for the Transformer.

MLP-based TSF model: In contrast to Transformer, MLP-based TSF models gain attention for their simplicity and efficiency. N-Beats [29] reconstructs past information (Backcast) and predicts future values (Forecast) in one block. It also employs stacking dual-residual structures to aggregate Forecast and extract unexplained Backcast, while embedding trend and seasonal decomposition into the model. Based on N-Beats, NBeatsX [30] introduces exogenous covariates, while DEPTS [31] incorporates periodic latent variables to enhance model performance. Subsequently, DLinear is proposed and leads researchers to focus more

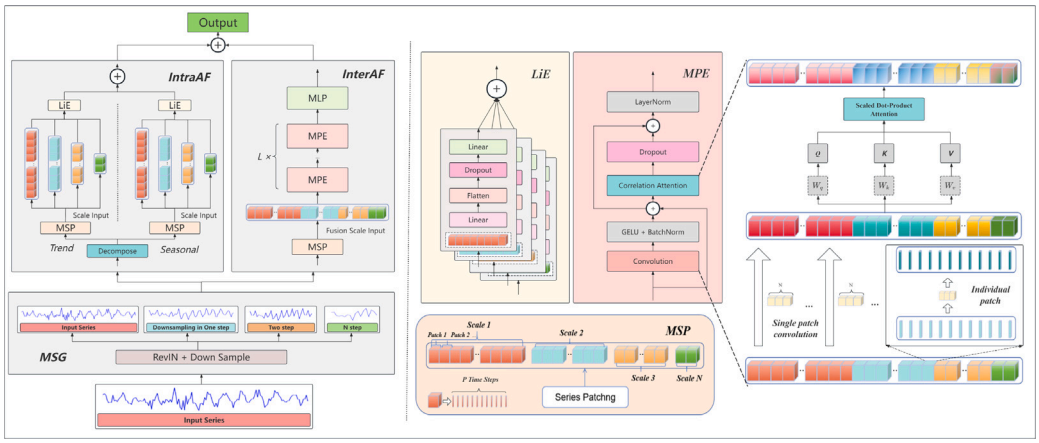


Fig. 2. The framework of MPMixer. MPMixer consists of three components: Multi-scale Series Generator (MSG), Intra-scale-based Analyzing and Forecasting (IntraAF), and Inter-scale-based Analyzing and Forecasting (InterAF). After inputting the raw time series, the MSG generates sequences at different scales. Then, a dual-ended processing is employed to analyzing and predicating, where the two ends are IntraAF and InterAF. Finally, the outputs from both ends are summed to obtain the prediction results.

on the performance limits of MLPs and simple models. By simple linearly mapping from the time and frequency domains, DLinear [14] and FreTs [13] achieve superior performance compared to Transformer-based TFS models. Recently, MTS-Mixers [32] utilizes the low-rank property of temporal data to guide predictions, while TiDE [23] employs dense MLPs to encode and decode time series and covariates, demonstrating that linear models can reach optimal performance with linear dynamical systems.

TSF models based-on Patching: Patch operations refer to dividing a larger input data sequence into fixed-size patches. Inspired by Vision Transformer [33], PatchTST [6] introduces Patch into time series forecasting, and establishes TSF models based on correlations between patches, thus improving the prediction performance while reducing computational complexity. PITS [34] reconstructs each patch without looking at other patches, and introduces complementary contrastive learning to hierarchically capture adjacent time series information efficiently. Pathformer [16] integrates both temporal resolution and temporal distance for multi-scale modeling. Based on the patches of each scale, dual-attention is performed to capture global correlations and local details.

TSF models based on multi-scale series: Pyraformer [35] introduces Pyramid Attention Module (PAM) to model different ranges of temporal correlation through intra-scale neighbors and integrate feature representations at different resolutions through inter-scale series. N-HiTS [18] samples time series at multiple frequencies into multiple scales and predicts them separately, and finally combining the results into the same dimension through interpolation. TimeMixer [17] deals with complex changes in time series with a new multi-scale mixer perspective. The model distinguishes between past information extraction phase and future prediction phase, and utilizes entanglement between multi-scale information for prediction.

Different from the previous multi-scale-based TSF models, our model transforms the time series at each scale into a patch sequence with the same Patch size. The unified treatment of time series at different scales is help to mining pattern correlations across fine-grained sequences and coarse-grained sequences.

3. Multi-scale Patch Mixer model

Let $i : j$ denotes the sequence $[i, i+1, \cdots, j]$. Given a look back window $X_{1:L} \in R^{L\times D}$, the TSF task is to predict a future sequence $X_{L+1:L+H} \in R^{H\times D}$, where L and H represent the look-back window size and the forecast horizon, D is the number of variables.

3.1. The framework of MPMixer

In this paper, a Multi-scale Patch Mixing time series forecasting model (MPMixer) is proposed. Its framework is illustrated as Fig. 2. MPMixer consists of three parts, Multi-scale Series Generator (MSG), Intra-scale-based Analyzing and Forecasting (IntraAF), and Inter-scale-based Analyzing and Forecasting (InterAF).

Multi-scale Series Generator (MSG): To reveal complex patterns of the time series, MSG downsamples the raw time series to generate sequences at different scales, thereby providing the subsequent components with data from fine to coarse granularity. More details about MSG are illustrated in Section 3.2.

Intra-scale-based Analyzing and Forecasting (IntraAF): IntraAF aims to extract information by analyzing the trend and the seasonality, and then make predictions. By summing the predictions, IntraAF ultimately produces part of the forecast. More details about IntraAF are illustrated in Section 3.3.

Inter-scale-based Analyzing and Forecasting (InterAF): InterAF analyzes the correlations across the multi-scales series, uncovers the relationships of time series with variable periodic lengths, and establishes a referring relationship between macro and micro scales. More details about InterAF are illustrated in Section 3.4.

After inputting the time series data, the MSG first generates multi-scale sequences from the raw time series. Then, a parallel dual-ended architecture, InterAF and IntraAF, is employed to model the intra-scale relations within the same scale and inter-scale relations across different scales. This parallelism not only ensures the model captures the cross-scale correlations to guide predictions, but also incorporates linear models as a supplement to maintain the overall trend of the time series.

Given the input sequence $X_{1:L} \in R^{L\times D}$, the overall analysis and prediction process can be expressed with the following formula:

$$\begin{aligned} \overline{X} &= MSG\left(X_{1:L}\right) \\ X_{L+1:L+H} &= IntraAF\left(\overline{X}\right) + InterAF\left(\overline{X}\right) \in R^{H\times D} \end{aligned} \tag{1}$$

The final prediction result consists of two parts: output of InterAF and output of IntraAF. Since the information on both ends complements each other, we achieve the prediction result by summing the outputs directly.

3.2. MSG: Multi-scale series generator

Considering the complex patterns of real time series, there are shortcomings if analyzing and predicting with the raw time series. For instance, the model may overfit the noise in the raw data. Moreover,

models based on single-scale may not be able to accommodate the diverse characteristics of time series, thereby limiting the expressive power of the model.

A time series may exhibit different properties at various scales. Coarse-grained time series (high-level series) often encompass the overall trends of the time series (the macroscopic variations), while fine-grained time series (low-level series) are more adept at capturing finer changes. To leverage the information that is difficult to capture at a single scale, we generate time series at different scales by downsampling, expecting to reveal and utilize information hidden in cross-scale.

The data distributions of real time series, such as mean and variance, may change over time, which brings troubles for time series forecasting tasks. In TSF tasks, datasets are divided into training sets and testing sets along the time dimension. It will naturally introduce the problem of inconsistent distribution. Therefore, we performed RevIN [3] on the time series to address the problem of data drift. That is, normalize the data before inputting the model, and finally denormalize the model's output.

After performing RevIn on the input sequence, we downsample the normalized sequence into M sequences $\{X^{(0)}, \ldots, X^{(m)}, \ldots, X^{(M-1)}\}$ at different scales. Let $X^{(m)} = \left[\mathrm{x}^{(m)}_{:,1}, \ldots, \mathrm{x}^{(m)}_{:,D}\right] \in \mathrm{R}^{\lfloor \frac{L}{2^m} \rfloor \times D}$ for $m = [0..M-1]$, $\mathrm{x}^{(m)}_{:,i} = \left[x^{(m)}_{1,i}, \ldots, x^{(m)}_{\lfloor \frac{L}{2^m} \rfloor,i}\right]^T$ be the ith feature of a time series at the mth scale. Then, the downsampling can be express as following.

$$X^{(m)} = \begin{cases} X & m = 0 \\ downsample(X^{(m-1)}) & m = [1..M-1] \end{cases} \tag{2}$$

where

$$downsample(X^{(m-1)}) = \left[pooling(\mathrm{x}^{(m-1)}_{:,1}), \ldots, pooling(\mathrm{x}^{(m-1)}_{:,D})\right] \tag{3}$$

There are many pooling methods, such as average pooling, max pooling, min pooling, and LP pooling. Among them, the average pooling reduces the impact of short-term fluctuations and noise, preserving the overall trend of the time series. The max pooling is often applied to prediction tasks that focus on peaks and extremes, such as financial risk warning and equipment failure detection. On the contrary, the min pooling emphasizes extremely low values. The LP pooling unifies max pooling and average pooling into a framework. It will adaptively select pooling behaviors based on task requirements. In our model, the average pooling with window size being 2 and stride being 2 is selected. The comparative experiments about pooling methods are presented in Section 4.2.

3.3. IntraAF: Intra-scale-based analyzing and forecasting

Seasonal and trend components correspond to short-term and long-term variations in TSF tasks [8,36]. Multi-scale seasonal and trend components can effectively unfold the time series, helping extract valuable information from multiple scales. To this end, we introduce IntraAF to analyze and predict based on intra-scale. IntraAF first decomposes the time series into trend components and seasonal components, and then analyzes and predicts with a simple linear model.

Specifically, in IntraAF, each multi-scale sequence is decomposed into seasonal components and trend components with the Decompose module. Next, the Multi-scale Patch (MSP) is applied to patch the multi-scale time series. Subsequently, we utilize a linear module to extract hidden information within these sequences and perform time series prediction. Finally, these prediction results are summed up as the prediction result of IntraAF (as shown in Fig. 2).

Decompose: Given the input time series, we use a moving average operation to smooth seasonal fluctuations, thereby highlighting the long-term trend and capturing the trend of the time series. For any input sequence $X^{(m)} \in R^{\lfloor \frac{L}{2^m} \rfloor \times D}$, we first perform padding to ensure that the output length remains consistent with the input length. Subsequently, the trend component is removed from the original sequence to obtain its seasonal part. Let $L_m = \lfloor \frac{L}{2^m} \rfloor$ be the length of the sequence at the mth scale. The decomposition process can be expressed as follows:

$$\begin{aligned} X^{(m),Trend} &= Avgpool\left(Padding\left(X^{(m)}\right)\right) \\ X^{(m),Seasonal} &= X^{(m)} - X^{(m),Trend} \end{aligned} \tag{4}$$

where, $X^{(m),Trend}$ represents the trend component of the time series, and $X^{(m),Seasonal}$ denotes the seasonal component.

Multi-scale Series Patch (MSP): During the patch sequence construction, $X^{(m),Trend}$ and $X^{(m),Seasonal}$ are processed with MSP separately. If not necessary, we omit the labels in the superscript for Trend and Seasonal.

Since our model is channel-independent, the input single-scale single-variable time series $\mathrm{x}^{(m)}_{:,d}$ is patched into patches $\mathrm{z}^{(m)}_{:,d} \in R^{P \times N_m}, m = [0..M-1]$ with patch size P and stride S, where $N_m = \lfloor \frac{L_m - P}{S} \rfloor + 2$ represents the number of patches at the mth scale. Referring to patchTST [6], we also pad the original sequence by repeating the last value S times before patching. The Multi-scale Series Patch can be expressed as follows:

$$Z^{(m)} = \left[\mathrm{z}^{(m)}_{:,1}, \ldots, \mathrm{z}^{(m)}_{:,d}\right] \in R^{P \times N_m \times D} \tag{5}$$

where

$$\mathrm{z}^{(m)}_{:,d} = Patching\left(Padding\left(\mathrm{x}^{(m)}_{:,d}\right)\right) \in R^{P \times N_m} \tag{6}$$

The MSP will output the multi-scale patch sequences $Z^{(m),Trend} \in R^{P \times N_m \times D}$ and $Z^{(m),Seasonal} \in R^{P \times N_m \times D}$, representing the trend component and seasonal component of the multi-scale sequences.

Linear Extractor (LiE) LiE is a simple linear model. For the single-variable trend component $\mathrm{z}^{(m),Trend}_{:,d} \in R^{P \times N_m}, d = [0..D-1]$, it is projected into the latent space as $\hat{\mathrm{z}}^{(m),Trend}_{:,d} \in R^{dim \times N_m}$. Subsequently, we fuse patch size and patch number to obtain $\tilde{\mathrm{z}}^{(m),Trend}_{:,d} \in R^{N_m \mathrm{dim}}$, and finally predict future trends through a linear layer. The above operations can be formula as follows.

$$\begin{aligned} \hat{\mathrm{z}}^{(m),Trend}_{:,d} &= Linear_{P \to dim}\left(\mathrm{z}^{(m),Trend}_{:,d}\right) \in R^{dim \times N_m} \\ \tilde{\mathrm{z}}^{(m),Trend}_{:,d} &= Flatten\left(\hat{\mathrm{z}}^{(m),Trend}_{:,d}\right) \in R^{N_m dim} \\ \hat{\mathrm{x}}^{(m),Trend}_{d} &= Linear_{N_m dim \to H}\left(Dropout\left(\tilde{\mathrm{z}}^{(m),Trend}_{:,d}\right)\right) \in R^{H \times 1} \end{aligned} \tag{7}$$

where $m = [0..M-1]$, dim denotes the dimensionality of the latent space.

Similarly, we can obtain $\hat{\mathrm{x}}^{(m),Seasonal}_{d}$. By adding them at different scales, the overall trend of the time series is generated.

$$\mathrm{x}^{IntraAF}_{:,d} = \sum_{m=0}^{M-1}\left(\hat{\mathrm{x}}^{(m),Trend}_{:,d} + \hat{\mathrm{x}}^{(m),Seasonal}_{:,d}\right) \in R^{H \times 1} \tag{8}$$

In the processing of intra-scale analyzing and forecasting, the low-scale input sequence $X = X^{(0)}$ contains fine variations of the time series, while the high-scale input sequence $X = X^{(M-1)}$ captures the macro variations of the time series. By sequentially inputting multi-scale sequences into the LiE model, implicit information may be captured easier. LiE captures features and predicts future using Linear layers. The patch sequence is taken as a whole as input, which enhance application of global information. As a result, IntraAF also focuses more on the global pattern.

3.4. InterAF: Inter-scale-based analyzing and forecasting

If only focusing on the correlated information within a single-scale sequence, the model may neglect the inter-scale relations. Therefore, we employ InterAF to perform analysis and prediction using inter-scale information.

We first use the MSP module (introduced in Section 3.3) to construct the patch sequences $Z = \{Z^{(0)}, \ldots, Z^{(M-1)}\} \in R^{M \times P \times N_m \times D}$ of the M-scale sequences. In order to learn the cross-scale correlations, the

M dimensional patch sequences are fused along the patch number dimension. It can be expressed as the follows.

$$\hat{Z} = Concat\left(Z^{(0)}, \ldots, Z^{(M-1)}\right) = \left[\mathrm{z}^{(0)}_{0,:}, \ldots, \mathrm{z}^{(0)}_{N_0,:}, \ldots, \mathrm{z}^{(M-1)}_{0,:}, \ldots, \mathrm{z}^{(M-1)}_{N_{M-1},:}\right] \tag{9}$$

let

$$\mathrm{z}^{(m)}_{i,:} = \begin{cases} \hat{\mathrm{z}}_{\sum_{j=0}^{m-1} N_j + i}, & \text{if } m > 0 \\ \hat{\mathrm{z}}_i, & \text{if } m = 0 \end{cases} \tag{10}$$

then

$$\hat{Z} = \left[\hat{\mathrm{z}}_{0,:}, \ldots, \hat{\mathrm{z}}_{\sum_{j=0}^{M-1} N_j - 1}\right] = \begin{bmatrix} \hat{\mathrm{z}}_{0,1} & \cdots & \hat{\mathrm{z}}_{\sum_{j=0}^{M-1} N_j - 1,1} \\ \vdots & \ddots & \vdots \\ \hat{\mathrm{z}}_{0,D} & \cdots & \hat{\mathrm{z}}_{\sum_{j=0}^{M-1} N_j - 1,D} \end{bmatrix} \tag{11}$$
$$= \left[\hat{\mathrm{z}}_{:,1}, \ldots, \hat{\mathrm{z}}_{:,D}\right]^T \in R^{P\left(\sum_{j=0}^{M-1} N_j\right) \times D}$$

Subsequently, the stacking of Multi-scale Patch Encoder (MPE) is employed to analyze the patch sequences, and then predict the result.

Multi-scale Patch Encoder (MPE): For the given single-variable input $\hat{\mathrm{z}}_{:,d} \in R^{P\left(\sum_{j=0}^{M-1} N_j\right)}$,where $d \in [0..D-1]$, it is first projected into the latent space $\tilde{\mathrm{z}}_{:,d} \in R^{dim\left(\sum_{j=0}^{M-1} N_j\right)}$, and then integrated within each patch using 1D convolution. The process is expressed as Formula (12).

$$\begin{aligned} \tilde{\mathrm{z}}_{:,d} &= Linear_{P \to dim}\left(\hat{\mathrm{z}}_{:,d}\right) \in R^{dim \times \left(\sum_{j=0}^{M-1} N_j\right)} \\ \bar{\mathrm{z}}_{:,d} &= \tilde{\mathrm{z}}_{:,d} + BatchNorm\left(Conv1D\left(\tilde{\mathrm{z}}_{:,d}\right)\right) \in R^{dim \times \left(\sum_{j=0}^{M-1} N_j\right)} \end{aligned} \tag{12}$$

The features of the patch data are further emphasized after the 1D convolution. After passing through several Transformer encoder layers, the output is described as follows.

$$\mathrm{x}^{MPE}_{:,d} = LayerNorm\left(\bar{\mathrm{z}}_{:,d} + MSA\left(\bar{\mathrm{z}}_{:,d}, \bar{\mathrm{z}}_{:,d}, \bar{\mathrm{z}}_{:,d}\right)\right) \in R^{dim \times \left(\sum_{j=0}^{M-1} N_j\right)} \tag{13}$$

where $LayerNorm(\cdot)$ denotes layer normalization, $MSA(\cdot)$ is a multi-head self-attention layer.

$MSA(\cdot)$ enhances representation capability by using multiple independent self-attention heads, where each head captures the dependencies between different patches. Leveraging the outstanding performance of the attention mechanism in capturing correlations, the model can effectively learn from cross-scale sequences.

Finally, a multilayer linear feedforward neural network, $MLP(\cdot)$, is used to map the output of MPE to the prediction sequence with length H. It is described as Formula (14).

$$\mathrm{x}^{InterAF}_{:,d} = MLP\left(\mathrm{x}^{MPE}_{:,d}\right) \in R^{H \times 1} \tag{14}$$

Overall, InterAF integrates patch sequences from multi-scales into one dimension, incorporates 1D convolution to emphasize features within the patches, and leverages a powerful self-attention mechanism to capture correlations between patches across multiple scales. The explicit utilization of multi-scale correlations helps to guide predictions and enhance prediction accuracy.

4. Experiments and discussion

In this section, we describe the experimental setup (Section 4.1), analyze the parameter settings (Sections 4.2 and 4.4), demonstrate the effectiveness of our method by comparison (Section 4.3), verify the indispensability of each component (Section 4.5), and analyze the model efficiency (Section 4.6). Finally, we visualize some forecast results (Section 4.7).

4.1. Experimental settings

Datasets: We conduct experiments on 9 well-established benchmarks: ETT datasets (including ETTh1, ETTh2, ETTm1, ETTm2), Weather, Electricity, Traffic, Solar-Energy, Exchange-rate. They are widely used in many TSF researches [16,17,22]. More information is summarized in Table 1.

Baseline methodologies: The 9 state-of-the-art long-term forecasting models are selected for comparison. They are TimeXer (2024) [37], TimeMixer (2024) [17], iTransformer (2024) [22], Pathformer (2024) [16], PatchTST (2023) [6], Crossformer (2023) [11], Timesnet (2023a) [10], DLinear(2023) [14], and Fedformer (2022b) [12]. Among them, PatchTST(2023) [6] is a representative method using Patch, Crossformer(2023) [11] is the most recent feature fusion-based methods, DLinear(2023) [14] is a MLP-based TSF models, others are the optimal baseline method for each period.

Model Setup: MPMixer employs the Adam optimizer [38] with a learning rate of 10^{-4}. The sizes of the sequence decomposition convolution kernel and the single patch convolution kernel are set to 25 and 8. The number of attention heads is 16, the Gelu activation function is selected, and the Dropout is set to 0.05. The experimental data is divided into training, validation, and testing sets in a ratio of 7:1:2. During training, we set up 50 epochs and realize early stopping at 10 epochs. The downsampling frequency M is set to 1 to 3 with the longer the prediction horizon, the higher the sampling frequency.

For fairness, we set input length L=96 for all models. Mean Absolute Error (MAE) and Mean Squared Error (MSE) are selected to evaluate the performances. All experiment results are the average of 3 experiments.

4.2. Hyper-parameter analysis

4.2.1. Patch size analysis

The patch size may affect the model capturing features and patterns. A smaller patch is profitable to capture subtle patterns and local features, but may miss long dependencies that span multiple patches. It may also increase the time complexity. While a larger patch helps to capture trends or periodic changes over longer periods. However, if the patch is too large, the information within the time series may be diluted or averaged, resulting in localized pattern loss. It is critical to choose the right patch size to balance the local features and the overall trend.

Taking the ETTm2 dataset as an example, we set a fixed stride of 8 and compare the performances on different patch sizes (8, 12, 16, 20). Fig. 3 shows the experimental results. Overall, the MSEs decrease with the prediction horizon, conforming to the common sense that the performance of long-term forecasting is better than that of short-term forecasting. Considering the performance on different patch sizes, the model performs best when the patch size is 16. This indicates that a patch size of 16 achieves a better balance between local details and overall trends. In the rest of the experiments, the patch size is set to 16.

4.2.2. Analysis of downsampling frequency

To reveal the impact of downsampling frequency on prediction accuracy, experiments are conducted for comparison. Also, taking the ETTm2 dataset as an example, we set downsampling frequency as 0, 1, 2, and 3, and investigate the performance under the prediction horizons of 96, 192, 336, 720, 960, and 1440. The experimental results are shown in Table 2.

The results in Table 2 indicate that multi-scale sequences generated by downsampling are beneficial for improving the prediction accuracy. Additionally, we find that a larger downsampling frequency is more suitable for longer-term predictions. The reason is that the large-scale sequences obtained from multiple downsampling are easier to focus on the overall trend. Therefore, we should set a larger downsampling frequency when making a long-term prediction. In the rest of experiments, M is set to 1 when the prediction horizon is small, such as 96 and 192, and M is set to 2 or 3 when the prediction horizon is 336 or 720.

Table 1
Detailed description of the 9 datasets.

Dataset	Dim	Dataset size	Frequency	Information
ETTh1	7	17 420	Hourly	Temperature
ETTh2	7	17 420	Hourly	Temperature
ETTm1	7	69 680	15 min	Temperature
ETTm2	7	69 680	15 min	Temperature
Electricity	321	26 304	Hourly	Electricity
Weather	21	52 696	10 min	Weather
Solar-Energy	137	52 560	10 min	Energy
Traffic	862	17 544	Hourly	Transportation
Exchange-rate	8	7588	Daily	Economy

Table 2
The comparison of MSEs when downsampling frequency=0, 1, 2, 3. On the ETTm2 dataset, the input length L is set to 96, and the prediction horizon H is set to = 96, 192, 336, 720, 960, 1440. The best results are shown in bold.

Dataset	Predict Length	96	192	336	720	960	1440
	Down Sample	MSE	MSE	MSE	MSE	MSE	MSE
ETTm2	**0**	0.1766	0.2433	0.2959	0.3828	0.4343	0.4280
	1	**0.1752**	**0.2410**	**0.2932**	0.3812	0.4329	0.4272
	2	0.1758	0.2420	0.2943	**0.3787**	0.4324	0.4270
	3	0.1755	0.2419	0.2933	0.3789	**0.4318**	**0.4269**

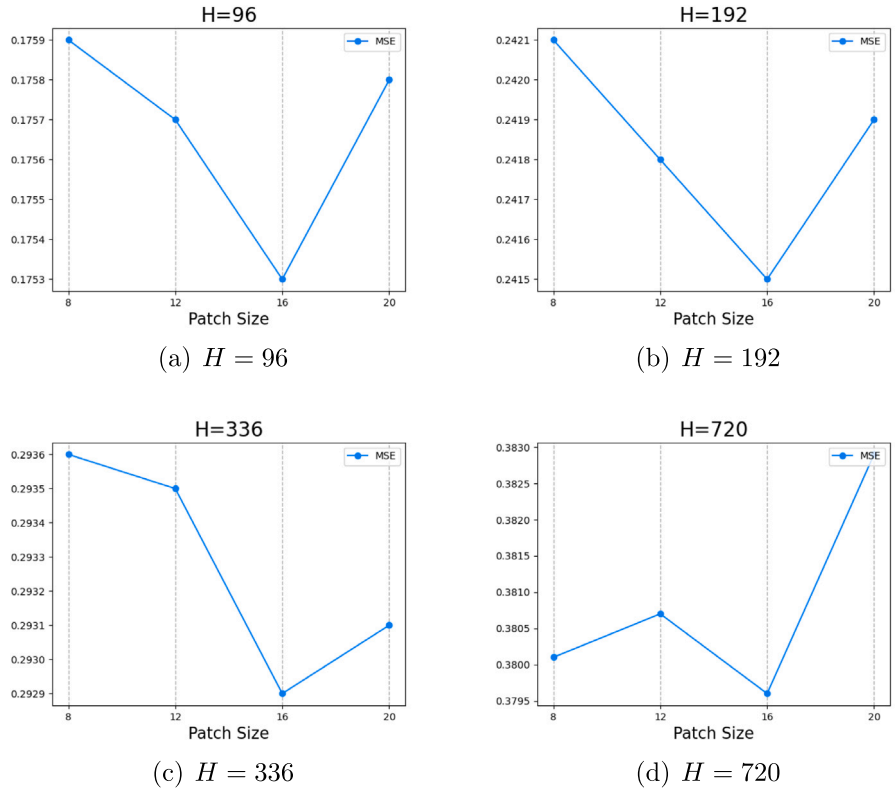


Fig. 3. The curves of patch size vs. MSE on the dataset ETTm2, where the patch size is set to 8, 12, 16, and 20. The forecasting horizon is set to 96, 192, 336, and 720.

4.2.3. Analysis of downsampling methods

Downsampling can be implemented using max pooling, min pooling, average pooling, and LP pooling. Here, we conduct experiments with different downsampling methods. In this experiment, the pooling window size is set to 2, the stride is set to 2. For LP pooling, we choose p=2 to aggregate the information within the pooling window. The experimental results are shown in Table 3.

As can be seen from Table 3, average pooling shows the best performance in most cases, which indicates that the overall trend is more important than the extremes when making long-term predictions.

4.3. Comparative analysis

In the comparative experiments, the prediction horizon H is set to 96, 192, 336, 720, and the input length L is set to 96 for all methods. Comparing to the 9 most recent baseline methods on 9 datasets, the experiment results are shown in Table 4.

From Table 4, it can be seen that MPMixer exhibits superior performance across most datasets. For the metric MSE, our method achieved 12 best results and 9 second-best results out of a total of 36 evaluations. In comparison, four state-of-the-art methods, TimeXer, Pathformer, iTransformer, and TimeMixer, achieved the best and second-best counts

Table 3
The performances comparison on different downsampling methods. The best results are **bolded**. Max: max pooling, LP: LP pooling, Avg: Average pooling.

Datasets		ETTh1				ETTh2				ETTm1				ETTm2			
Method	Metric	96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720
Max	MSE	0.375	0.432	0.450	0.490	0.234	0.291	0.343	0.403	0.319	0.368	0.387	**0.456**	0.176	0.243	0.294	0.383
	MAE	0.385	0.416	0.464	0.464	0.302	0.340	0.378	0.425	0.345	0.369	0.390	0.426	0.253	0.298	0.332	0.387
LP	MSE	0.376	0.433	0.458	0.495	0.231	**0.287**	0.343	0.406	0.319	0.371	0.389	0.470	0.176	0.242	0.294	0.380
	MAE	0.387	0.417	0.431	0.466	0.298	**0.333**	0.375	0.425	**0.343**	0.370	0.391	0.433	0.252	0.296	0.331	0.385
Avg	MSE	**0.372**	**0.429**	**0.449**	**0.465**	**0.230**	0.288	**0.339**	**0.401**	**0.318**	**0.367**	**0.386**	0.458	**0.175**	**0.241**	**0.293**	**0.378**
	MAE	**0.384**	**0.414**	**0.428**	**0.453**	**0.297**	0.334	**0.373**	**0.422**	0.344	**0.368**	**0.389**	**0.425**	**0.251**	**0.294**	**0.330**	**0.384**

of 7/11, 5/10, 4/1, and 7/7. On the metric MAE, our method obtains 28 best results and 6 second-best results. In comparison, TimeXer, Pathformer, iTransformer, and TimeMixer achieve the best and second-best counts of 0/5, 6/20, 1/7, and 0/4.

Let the best score 2 and the second-best score 1, then our method scores 95 in both MSE and MAE. While TimeXer, Pathformer, iTransformer, and TimeMixer score 30, 52, 18, and 25, It is evident that our method outperforms existing state-of-the-art methods in long-term time series forecasting.

Compared to PatchTST, which employs the patch for time series forecasting, MPMixer outperforms in 70 out of 72 compared metrics, demonstrating the effectiveness of exploration and application of cross-scale information. When compared to TimeMixer, which implicitly utilizes cross-scale information, our method also achieved superior performance, indicating the effectiveness of explicitly leveraging cross-scale information. MPMixer achieved a comprehensive advantage over feature fusion models like Crossformer, and strong linear models like DLinear. In comparison to other state-of-the-art models, such as Informer, MPMixer also stands out significantly, proving the effectiveness of learning changing patterns within multi-scale sequences.

4.4. The impact of the lookback length on performance

In theory, as the look-back window size L increases, the model will learn more from the input sequences, thus improving the prediction performance [22,39]. Some studies have also pointed out that Transformer-based models may lead to attention dilution when handling longer inputs, leading to difficulties in extracting effective sequence relationships from long sequences [6,14]. To verify whether MPMixer can effectively extract guide information from long-term sequences, we conduct experiments on the ETTh1 and ETTm2 with different input lengths. The results are shown in Fig. 4.

As shown in Fig. 4, first, except for iTransformer and TimeMixer, the performances of the other three Transformer-based TSF models, including our method, generally improve with the input length. PatchTST, Pathformer and MPMixer, employing the patch technique, are less affected by outliers, resulting in more stable performance changes. In contrast, iTransformer and TimeMixer exhibit some irregular fluctuations overall. Additionally, compared to PatchTST, our method generally shows superior performance, which is consistent with the previous experimental results.

The experiment results demonstrate that modeling on the cross-scale sequences is feasible to enhance the Transformer's ability to capture inter-relationships in long look-back windows. Also, as the prediction length increases, the advantages of cross-scale modeling become more and more outstanding.

4.5. Ablation study

In ablation experiments, we focus on IntraAF and InterAF. IntraAF is further divided to examine the impact of trends and seasonality on predictions. Table 5 enumerates the five scenarios compared in the ablation experiments.

The experiment results are presented in Table 5. Let the prediction horizon H=96, 192, 336, and 720, it shows that the complete model ① achieves the best performance in most situations, indicating that each module contributes to the improvement of the model's performance.

By comparing the Model without InterAF ② and the Model without IntraAF ⑤, we find that when the resolution of the data is low (ETTh1, ETTh2), ② performs comparably to or even surpasses ⑤. This demonstrates that the branch IntraAF can learn the overall trend of the time series, while InterAF enhances precision, thus collaborating effectively to complete the prediction task.

The comparison between ① and ⑤ reveals that while the branch InterAF guides similar patterns through complex modeling, the apparent intra-scale linear correlations are overlooked, making IntraAF indispensable. The LiE module in IntraAF is a linear model. It relies more on longer lookback windows. Therefore, ② shows poorer performance than ① and ⑤ in all. This phenomenon becomes more and more obvious as the data dimensions increase, with ② demonstrating much lower performance than ① and ⑤ on the Traffic, Weather, Solar and Electricity datasets. Meanwhile, the purely linear model, DLinear, produces the same phenomenon when predicting high-dimensional data. Even so, it still offers linear guidance information within the scale that InterAF cannot learn.

4.6. Model efficiency analysis

To further analyze the efficiency of MPMixer, we compare the forecasting performance, training speed, and memory footprint in the training and inference phases. The settings are consistent with the experiment in Section 4.3. For ETTh1 (7 variates) and Traffic (862 variates), the batch size of all models is set to 256 and 16, respectively. The hidden dimension is set to 256.

Fig. 5 shows the comparison results of efficiency on datasets ETTh1 and Traffic in Training Phase with Input-96-Prediction-96. Patchformer is absent because of running out of GPU memory. TimesNet varies too much relative to other models, where the memory overhead is 15.262 GB, and the training time reaches 22456ms/iter on ETTh1. So, we do not include it in Fig. 5.

From Fig. 5, we find that the MLP-based model, such as DLinear, has the least memory requirements and the highest training speed, but poor performance. MPMixer integrates multi-scale and patch technologies. Therefore, it achieves better performance with much less training time and memory footprint compared to TimeMixer (a multi-scale-based model), and Crossformer. It once again verifies that multi-scale helps to improve the performance, while patch can reduce the computational consumption. Although PatchTST also uses the Patch to reduce memory and time consumption. MPMixer needs only one or two MPE layers instead of multiple Transformer layers in PatchTST, leading to better performance with less memory footprint than PatchTST.

In Table 6, we show the memory requirement and time consumption in inference phases. Overall, it can be seen that a model requires more storage in training, it also requires more during the inference phase. DLinear still has the lowest time and memory consumption. Due to the need to model inter-relations between channels, iTransformer has a noticeable increase in memory and time consumption as the dimension increases. The inference time and storage requirements of MPMixer are similar to those of the Transformer-based model and much lower than the multiscale model TimeMixer.

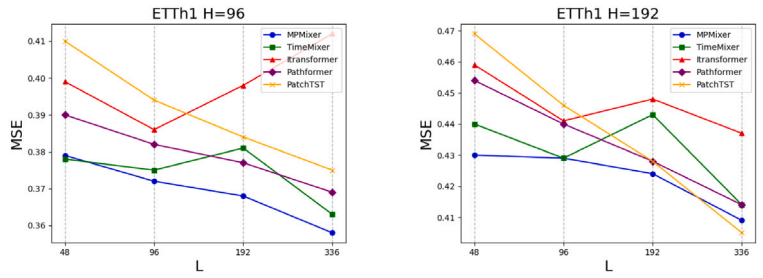
Table 4
A comparison of long-term forecasting results between our method and baseline methods across 9 datasets. The forecast lengths are 96, 192, 336, and 720. Lower MSE or MAE indicates better forecasting performance. The experimental data is sourced from [16,17,22], and [37]. We also filled in the untested sections (Solar and Exchange rate in Pathformer, TimeXer, and Exchange rate in TimeMixer). The best results are marked in red, while the second-best results are marked in blue.

	Models	MPMixer ours		TimeXer 2024		Pathformer 2024		iTransformer 2024		TimeMixer 2024		PatchTST 2023		TimesNet 2023		Crossformer 2023		DLinear 2023		FEDformer 2022	
	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	**96**	0.164	0.203	0.157	0.205	0.156	0.192	0.174	0.214	0.163	0.209	0.186	0.227	0.172	0.220	0.195	0.271	0.195	0.252	0.217	0.296
	192	0.210	0.245	0.204	0.247	0.206	0.240	0.221	0.254	0.208	0.250	0.234	0.265	0.219	0.261	0.209	0.277	0.237	0.295	0.276	0.336
	336	0.264	0.286	0.261	0.290	0.254	0.282	0.278	0.296	0.251	0.287	0.284	0.301	0.246	0.337	0.273	0.332	0.282	0.331	0.339	0.380
	720	0.342	0.339	0.340	0.341	0.340	0.336	0.358	0.347	0.339	0.341	0.356	0.349	0.365	0.359	0.379	0.401	0.345	0.382	0.403	0.428
	Avg	0.245	0.268	0.241	0.271	0.239	0.262	0.258	0.278	0.240	0.271	0.265	0.285	0.251	0.294	0.264	0.320	0.265	0.315	0.309	0.360
Solar	**96**	0.193	0.219	0.222	0.274	0.217	0.235	0.203	0.237	0.189	0.259	0.265	0.323	0.373	0.358	0.232	0.302	0.290	0.378	0.286	0.341
	192	0.227	0.242	0.255	0.297	0.230	0.244	0.233	0.261	0.222	0.283	0.288	0.332	0.397	0.376	0.371	0.410	0.320	0.398	0.291	0.337
	336	0.243	0.253	0.277	0.310	0.247	0.255	0.248	0.273	0.231	0.292	0.301	0.339	0 420	0 380	0.495	0.515	0.353	0.415	0.354	0.416
	720	0.245	0.255	0.278	0.312	0.250	0.258	0.249	0.275	0.223	0.285	0.295	0.336	0.420	0.381	0.526	0.542	0.357	0.413	0.380	0.437
	Avg	0.227	0.242	0.258	0.298	0.236	0.248	0.233	0.262	0.216	0.280	0.287	0.333	0.403	0.374	0.406	0.442	0.330	0.401	0.328	0.383
Electricity	**96**	0.152	0.239	0.140	0.242	0.145	0.236	0.154	0.245	0.153	0.247	0.190	0.296	0.168	0.272	0.219	0.314	0.210	0.302	0.193	0.308
	192	0.163	0.250	0.157	0.256	0.167	0.256	0.169	0.258	0.166	0.256	0.199	0.304	0 184	0.322	0.231	0.322	0.210	0.305	0.201	0.315
	336	0.180	0.267	0.176	0.275	0.186	0.275	0.185	0.275	0.185	0.277	0.217	0.319	0.198	0.300	0.246	0.337	0.223	0.319	0.214	0.329
	720	0.219	0.301	0.211	0.306	0.231	0.309	0.225	0.308	0.225	0.310	0.258	0.352	0.220	0.320	0.280	0.363	0.258	0.350	0.246	0.355
	Avg	0.178	0.264	0.171	0.270	0.182	0.269	0.183	0.271	0.182	0.272	0.216	0318	0.193	0.304	0.244	0.334	0.225	0.319	0.214	0.327
Traffic	**96**	0.447	0.266	0.428	0.271	0.479	0.283	0.395	0.268	0.462	0.285	0.525	0.347	0.593	0.321	0.644	0.429	0.650	0.396	0.587	0.366
	192	0.456	0.267	0.448	0.282	0.484	0.292	0.417	0.276	0.473	0.296	0.522	0.332	0.617	0.336	0.665	0.431	0.598	0.370	0.604	0.373
	336	0.476	0.270	0.473	0.289	0.503	0.299	0.433	0.283	0.498	0.296	0.517	0.334	0.629	0.336	0.674	0.420	0.605	0.373	0.621	0.383
	720	0.515	0.287	0.516	0.307	0.537	0.322	0.467	0.302	0.506	0.313	0.552	0.352	0.640	0.350	0.683	0.424	0.645	0.394	0.626	0.382
	Avg	0.473	0.272	0.466	0.287	0.500	0.299	0.428	0.282	0.484	0.297	0.529	0.341	0.620	0.336	0.667	0.426	0.625	0.383	0.610	0.376
ETTh1	**96**	0.372	0.384	0.382	0.403	0.382	0.400	0.386	0.405	0.375	0.400	0.394	0.394	0.384	0.402	0.423	0.448	0.397	0.412	0.395	0.424
	192	0.429	0.414	0.429	0.435	0.440	0.427	0.441	0.436	0.429	0.421	0.446	0.427	0.436	0.429	0.471	0.474	0.446	0.441	0.469	0.470
	336	0.449	0.428	0.468	0.448	0.454	0.432	0.487	0.458	0.484	0.458	0.485	0.455	0.638	0.469	0.570	0 546	0.489	0.467	0.530	0.499
	720	0.465	0.453	0.469	0.461	0.479	0.461	0.503	0.491	0.498	0.482	0.495	0.474	0 521	0.500	0.653	0.621	0.513	0.510	0.598	0.544
	Avg	0.428	0.419	0.437	0.437	0.438	0.430	0.454	0.447	0.447	0.440	0.455	0.437	0.495	0.450	0.529	0.522	0.461	0.457	0.498	0.484
ETTh2	**96**	0.232	0.298	0.286	0.338	0.279	0.331	0.297	0.349	0.289	0.341	0.308	0.355	0 340	0.374	0.745	0.584	0.340	0.394	0.358	0.397
	192	0.289	0.336	0.363	0.389	0.349	0.380	0.380	0.400	0.372	0.392	0.393	0.405	0.402	0.414	0.877	0.656	0.482	0.479	0.429	0.439
	336	0.340	0.373	0.414	0.423	0.348	0.382	0.428	0.432	0.386	0.414	0.427	0.436	0.452	0.452	1.043	0.731	0.591	0.541	0.496	0.487
	720	0.401	0.422	0.408	0.432	0.398	0.424	0.427	0.445	0.412	0.434	0.436	0.450	0.462	0.468	1.104	0.763	0.839	0.661	0.463	0.474
	Avg	0.315	0.357	0.367	0.396	0.343	0.379	0.383	0.407	0.364	0.395	0.391	0.411	0.414	0.427	0.943	0.684	0.563	0.519	0.437	0.449
ETTm1	**96**	0.318	0.344	0.318	0.356	0.316	0.346	0.334	0.368	0.320	0.357	0.352	0.374	0.338	0.375	0.404	0.426	0.346	0.374	0.379	0.419
	192	0.367	0.368	0.362	0.383	0.366	0.370	0.377	0.391	0.361	0.381	0.390	0.393	0.374	0.387	0.450	0.451	0.382	0.391	0.426	0.441
	336	0.386	0.389	0.395	0.407	0.386	0.394	0.426	0.420	0.390	0.404	0.421	0.414	0.410	0.411	0.532	0.515	0.415	0.415	0.445	0.459
	720	0.458	0.425	0.452	0.441	0.460	0.432	0.491	0.459	0.454	0.441	0.462	0.449	0.478	0.450	0.666	0.589	0.473	0.451	0.543	0.490
	Avg	0.382	0.381	0.382	0.397	0.382	0.385	0.407	0.410	0.381	0.395	0.406	0.407	0.400	0.406	0.513	0.495	0.404	0.408	0.448	0.452
ETTm2	**96**	0.175	0.251	0.171	0.256	0.170	0.248	0.180	0.264	0.175	0.258	0.183	0.270	0 187	0.267	0.287	0.366	0.193	0.293	0.203	0.287
	192	0.241	0.294	0.237	0.299	0.238	0.295	0.250	0.309	0.242	0.299	0.255	0.314	0.249	0.309	0.414	0.492	0.284	0.361	0.269	0.328
	336	0.291	0.330	0.296	0.338	0.293	0.331	0.311	0.348	0.298	0.340	0.309	0.347	0.321	0.351	0.597	0.542	0.382	0.429	0.325	0.366
	720	0.379	0.384	0.392	0.394	0.390	0.389	0.412	0.407	0.391	0.396	0.412	0.404	0.408	0.403	1.730	1.042	0.558	0.525	0.421	0.415
	Avg	0.271	0.315	0.274	0.322	0.273	0.316	0.288	0.332	0.275	0.323	0.290	0.334	0.291	0.333	0.757	0.610	0.354	0.402	0.305	0.349
Exchange	**96**	0.081	0.198	0.085	0.204	0.086	0.207	0.086	0.206	0.085	0.204	0.088	0.205	0.107	0.234	0.256	0.367	0.088	0.218	0.148	0.278
	192	0.173	0.294	0.181	0.302	0.178	0.301	0.177	0.299	0.190	0.310	0.176	0.299	0.226	0.344	0.470	0.509	0.176	0.315	0.271	0.315
	336	0.338	0.421	0.363	0.435	0.361	0.433	0.331	0.417	0.373	0.446	0.301	0.397	0.367	0.448	1.268	0.883	0.313	0.427	0.460	0.427
	720	0.860	0.692	0.930	0.727	0.934	0.727	0.847	0.691	1.163	0.792	0.901	0.714	0.964	0.746	1.767	1.068	0.839	0.695	1.195	0.695
	Avg	0.365	0.404	0.389	0.417	0.389	0.417	0.360	0.403	0.452	0.438	0.367	0.404	0.416	0.443	0.940	0.707	0.354	0.414	0.519	0.429

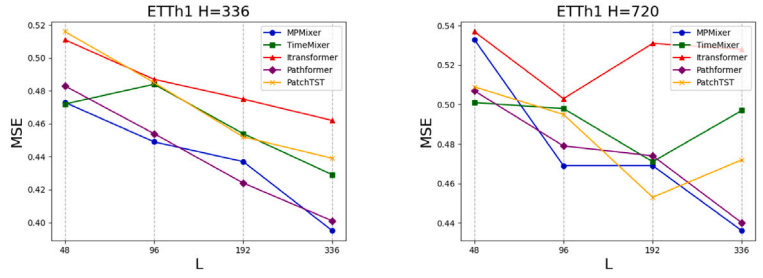
4.7. Visualization of forecast results

To provide a clear comparison between different models, we visualize some prediction showcases. Some forecasting results on MPMixer, TimeMixer, Pathformer and Autoformer are shown in Fig. 6, 7and 8. The 3 compared methods are representative of multi-scale-based models, patch-based models and sequence-decomposition-based models.

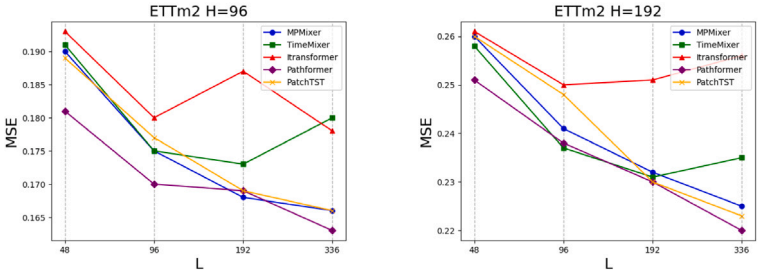
Fig. 6 shows that all methods predict well on the simple dataset Traffic, obtaining a clear periodic characteristic. But we would rather focus on the improvements in detail. As shown in Fig. 6, MPMixer clearly fits the ground truth more closely at the extreme points. In [37],



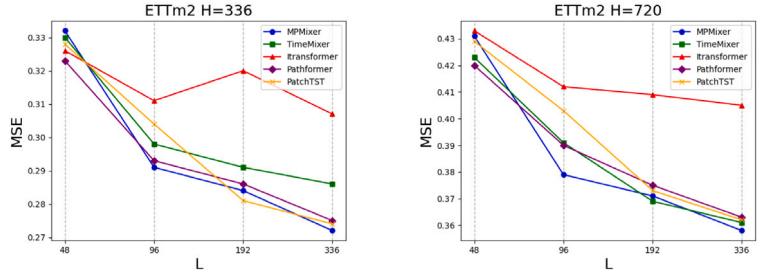
(a) ETTh1 $H = 96$ (b) ETTh1 $H = 192$



(c) ETTh1 $H = 336$ (d) ETTh1 $H = 720$



(e) ETTm2 $H = 96$ (f) ETTm2 $H = 192$



(g) ETTm2 $H = 336$ (h) ETTm2 $H = 720$

Fig. 4. The performance curves of lookback window vs. prediction horizon on the ETTh1 and ETTm2 datasets, where lookback window L = 48,96,192,336 and prediction length H = 96, 192, 336, 720.

authors also pointed out that patch-based models usually concern more on the overall trend and neglect details, making it difficult to predict the extreme points of sequence. Pathformer illustrates the issue 6(c). The dual branch and multi-scale of MPMixer compensate for the deficiency to some extent. In essence, both the multi-scale-based model (TimeMixer) and the sequence-decomposition-based model (Autoformer) first decompose the sequence, and then process them separately, lacking considering the correlations between multi-scale. In MPMixer, the branch InterAF considers inter-scale-correlations with the MPE module, resulting in better prediction performance than TimeMixer and Autoformer.

The change patterns on datasets Electricity and ETTh1 are more diverse. We also present showcases as Figs. 7 and 8. Among the various models, MPMixer exhibits superior performance.

5. Conclusions

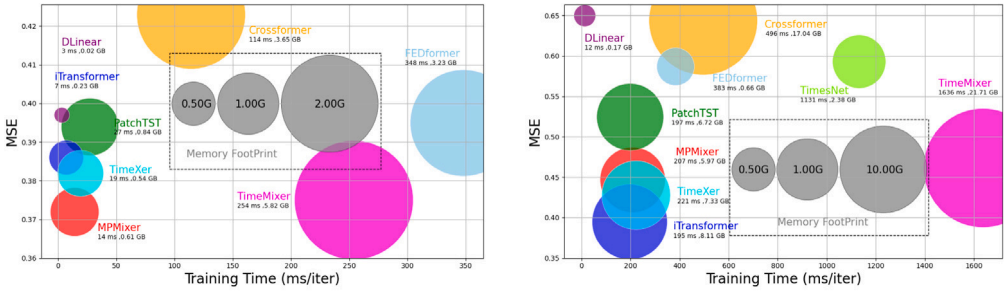
In this paper, we introduce a novel Multi-scale Patch mixer model for Long-Term Time Series Forecasting model (MPMixer). Based on a double-ended architecture, MPMixer captures both intra-scale and inter-scale variations of multi-scale time series to guide the prediction. In the Intra-Scale-based analyzing and predicting, we decompose the time series into trend components and seasonal components to gain

Table 5
Ablation experiment results of MPMixer. The symbols ✓or✗indicate the presence or absence of a component. The ① is the full MPMixer model. The model ② represent the model without InterAF. The model ⑤ leaves only the InterAF branch. The best results are shown in bold.

MPMixer		①		②		③		④		⑤	
InterAF(MPE)		✓		✗		✓		✓		✓	
IntraAF(LiE)	Seasonal	✓		✓		✗		✓		✗	
	Trend	✓		✓		✓		✗		✗	
Dataset	Prediction Length	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**96**	**0.372**	**0.384**	0.378	0.387	0.379	0.389	0.374	0.385	0.379	0.390
	192	**0.429**	**0.414**	0.433	0.415	0.437	0.419	0.430	0.415	0.437	0.419
	336	0.449	0.428	0.451	0.429	0.455	0.430	**0.445**	**0.426**	0.452	0.429
	720	0.465	0.453	**0.463**	**0.452**	0.497	0.464	0.470	0.456	0.482	0.457
ETTh2	**96**	**0.230**	**0.297**	0.234	0.300	0.233	0.300	0.232	0.298	0.234	0.300
	192	**0.288**	**0.334**	0.292	0.339	0.290	0.336	0.291	0.335	0.291	0.339
	336	**0.339**	**0.373**	0.342	0.377	0.342	0.374	0.343	0.374	0.344	0.375
	720	**0.401**	**0.422**	0.407	0.427	0.407	0.424	0.407	0.424	0.409	0.425
ETTm1	**96**	**0.318**	**0.344**	0.337	0.352	0.319	0.344	0.318	0.343	0.319	0.345
	192	**0.367**	**0.368**	0.381	0.374	0.367	0.369	0.369	0.368	0.369	0.369
	336	**0.386**	**0.389**	0.402	0.396	0.387	0.390	0.389	0.390	0.388	0.391
	720	**0.458**	**0.425**	0.474	0.431	0.461	0.427	0.465	0.429	0.465	0.429
ETTm2	**96**	**0.175**	**0.251**	0.183	0.258	0.175	0.251	0.175	0.251	0.176	0.251
	192	**0.241**	**0.294**	0.247	0.299	0.241	0.296	0.243	0.296	0.242	0.295
	336	**0.293**	**0.330**	0.297	0.332	0.295	0.331	**0.293**	**0.330**	0.294	0.331
	720	**0.378**	**0.384**	0.385	0.384	0.380	0.385	0.380	0.385	0.379	0.383
Weather	**96**	**0.164**	0.203	0.206	0.229	0.165	**0.199**	0.167	0.202	0.166	0.200
	192	**0.211**	**0.243**	0.250	0.266	0.213	0.245	0.213	0.245	0.213	0.244
	336	**0.264**	**0.286**	0.297	0.300	0.267	0.287	0.269	0.288	0.268	0.287
	720	**0.342**	0.339	0.357	0.348	**0.342**	**0.335**	0.346	0.337	0.344	0.335
Solar	**96**	**0.193**	**0.219**	0.301	0.306	0.194	0.220	0.195	0.221	0.196	0.222
	192	**0.227**	**0.242**	0.340	0.324	0.228	0.243	0.229	0.244	0.230	0.245
	336	**0.243**	**0.253**	0.383	0.339	0.244	0.254	0.244	0.255	0.245	0.255
	720	**0.245**	**0.255**	0.387	0.334	0.247	0.256	0.246	0.256	0.246	0.256
Electricity	**96**	**0.152**	**0.239**	0.197	0.2697	0.158	0.245	0.154	0.242	0.154	0.242
	192	**0.163**	**0.250**	0.199	0.272	0.168	0.255	0.166	0.251	0.170	0.256
	336	0.180	0.267	0.213	0.287	0.182	0.269	**0.179**	**0.266**	0.184	0.272
	720	**0.219**	0.301	0.255	0.320	0.220	**0.297**	0.220	0.301	0.222	0.304
Exchange	**96**	**0.081**	**0.198**	0.082	0.199	0.083	0.201	0.087	0.205	0.085	0.204
	192	**0.173**	**0.294**	0.184	0.302	0.176	0.298	0.178	0.301	0.177	0.301
	336	0.338	0.421	0.336	0.416	0.338	0.422	0.330	0.415	**0.326**	**0.414**
	720	**0.860**	**0.692**	0.867	0.701	0.861	0.693	0.862	0.700	0.869	0.699
Traffic	**96**	**0.447**	**0.266**	0.692	0.365	0.465	0.282	0.457	0.275	0.464	0.281
	192	**0.456**	**0.267**	0.644	0.340	0.470	0.282	0.463	0.274	0.473	0.283
	336	**0.476**	**0.270**	0.650	0.343	0.483	0.285	0.477	0.278	0.484	0.287
	720	**0.515**	**0.287**	0.682	0.357	0.515	0.297	0.514	0.293	0.513	0.297

Table 6
Comparisons of model efficiency on datasets ETTh1 and Traffic in inference phases with Input-96-Prediction-96.

Dataset		MPMixer	TimeXer	**iTransformer**	TimeMixer	Crossformer	PatchTST	TimesNet	DLinear	FEDformer
ETTh1	**memory(MB)**	241.6	305.3	204.8	1933.5	338.6	197.7	8861.6	23.2	790.41
	time(s)	0.10	0.18	0.18	0.35	0.21	0.18	5.6	0.04	0.836
Traffic	**memory(MB)**	1639.2	1527.8	2334.5	14 856.1	1266.5	1436.8	1357.7	117.7	246.75
	time(s)	10.8	5.2	21	11.3	20.4	12.1	67.7	0.91	60.1



(a) ETTh1 (b) Traffic

Fig. 5. Comparisons of model efficiency on datasets ETTh1 and Traffic in training phase with Input-96-Prediction-96.

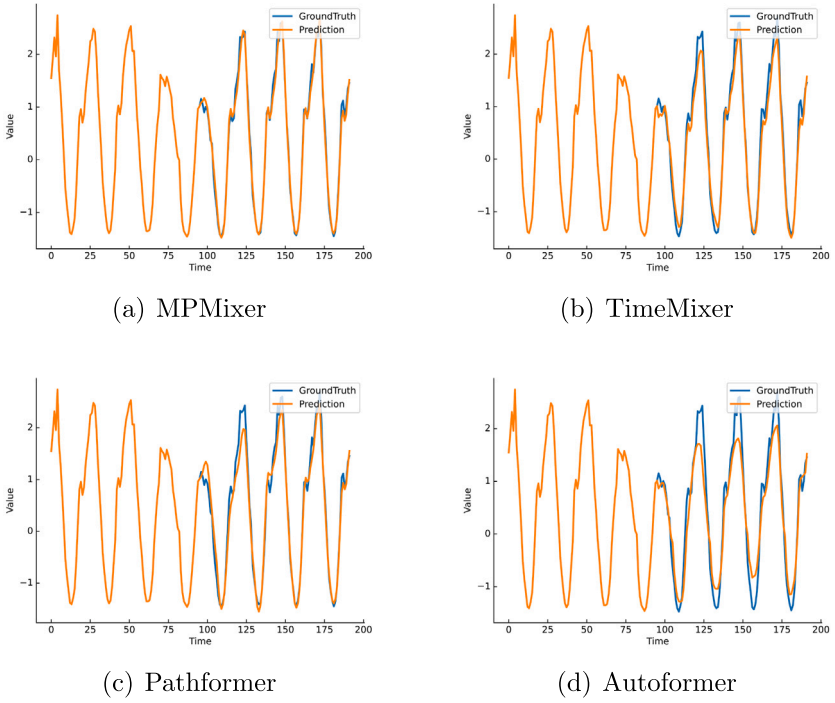


Fig. 6. Showcases of input-96-predict-96 results on the dataset Traffic.

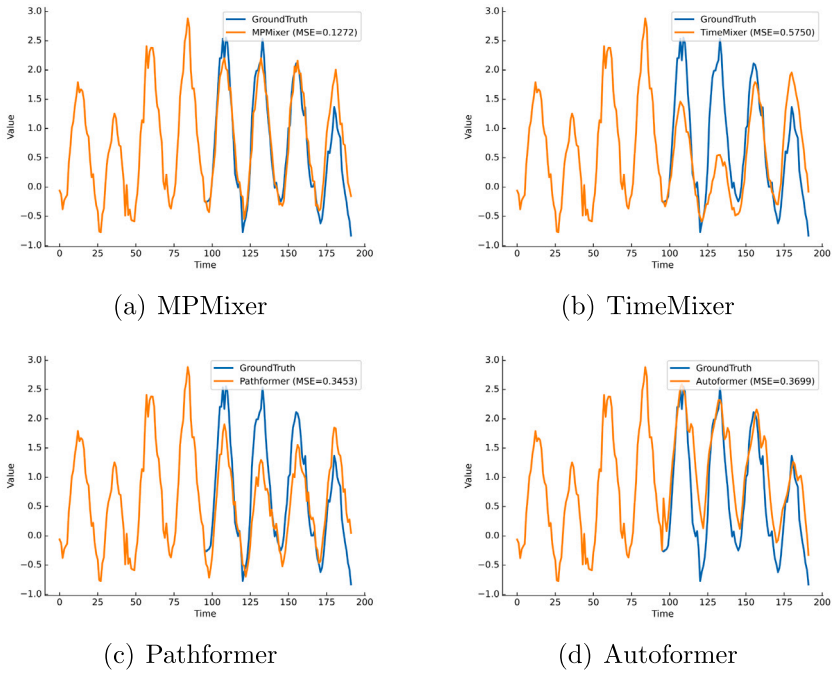
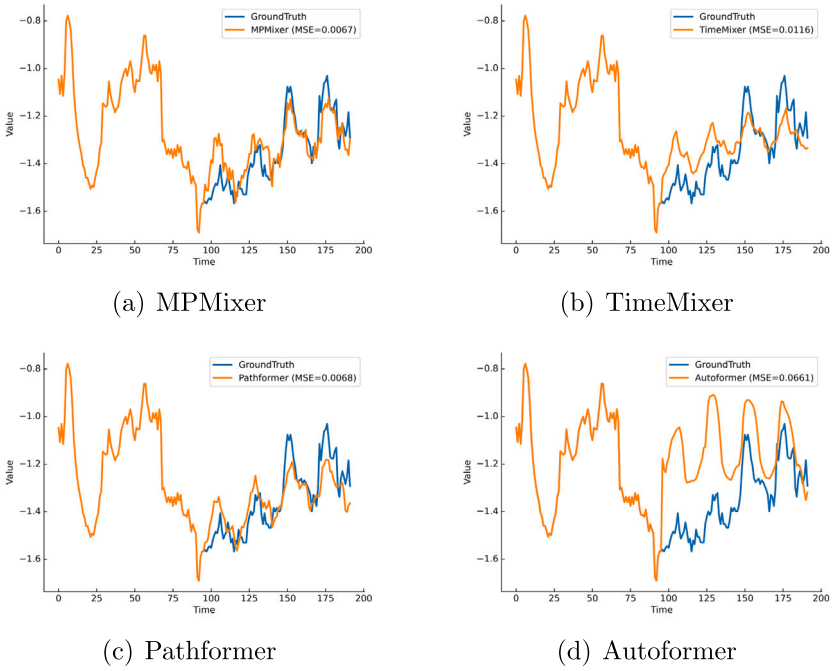


Fig. 7. Showcases of input-96-predict-96 results on the dataset Electricity.

deeper insights into the information at each scale. In the Inter-Scale-based analyzing and predicting, we use multi-scale series patch and multi-scale patch encoder to integrate the information from different scales into the same observation surface.

The experiments for parameter analysis guide the experimental setup and further applications. The ablation studies demonstrate the importance of each component. Compared to SOTA time series forecasting methods, MPMixer exhibits superior performance in TSF tasks. The experiment results show that explicitly modeling cross-scale sequences not only enhances the model's adaptability to complex time series but also improves the prediction accuracy.



(a) MPMixer (b) TimeMixer

(c) Pathformer (d) Autoformer

Fig. 8. Showcases of input-96-predict-96 results on the dataset ETTh1.

CRediT authorship contribution statement

Yijun Xu: Writing – original draft, Validation, Methodology, Formal analysis, Conceptualization. **Weijie Zhang:** Visualization, Validation. **Huixia Lai:** Writing – review & editing, Investigation, Funding acquisition, Conceptualization. **Shi Zhang:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Funding acquisition, Formal analysis, Conceptualization. **Ruliang Xiao:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Project of Fujian Province Science and Technology Plan, China (grant No. 2023I0013, 2024I0035).

Data availability

The authors do not have permission to share data.

References

[1] H. Zhang, J. Wang, Y. Qian, Q. Li, Point and interval wind speed forecasting of multivariate time series based on dual-layer LSTM, Energy 294 (2024) 130875.
[2] D. Yao, K. Yan, Time series forecasting of stock market indices based on DLWR-lstm model, Financ. Res. Lett. 68 (2024) 105821.
[3] W. Zhang, R. Yao, Y. Yuan, X. Du, L. Wang, F. Sun, A traffic-weather generative adversarial network for traffic flow prediction for road networks under bad weather, Eng. Appl. Artif. Intell. 137 (2024) 109125.
[4] B. Yang, X. Liang, S. Xu, M.S. Wong, W. Ma, A time-series based deep survival analysis model for failure prediction in urban infrastructure systems, Eng. Appl. Artif. Intell. 136 (2024) 108876.
[5] B. Rostami-Tabar, R.J. Hyndman, Hierarchical time series forecasting in emergency medical services, J. Serv. Res. (2024) 10946705241232169.
[6] Y. Nie, N.H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: Long-term forecasting with transformers, in: The Eleventh International Conference on Learning Representations, 2023, URL https://openreview.net/forum?id=Jbdc0vTOcol.
[7] R. Wen, K. Torkkola, B.M. Narayanaswamy, D. Madeka, A multi-horizon quantile recurrent forecaster, in: NeurIPS 2017, 2017, URL https://www.amazon.science/publications/a-multi-horizon-quantile-recurrent-forecaster.
[8] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, Adv. Neural Inf. Process. Syst. 34 (2021) 22419–22430.
[9] Y. Jia, Y. Lin, X. Hao, Y. Lin, S. Guo, H. Wan, WITRAN: Water-wave information transmission and recurrent acceleration network for long-range time series forecasting, Adv. Neural Inf. Process. Syst. 36 (2024).
[10] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, TimesNet: Temporal 2D-variation modeling for general time series analysis, in: The Eleventh International Conference on Learning Representations, 2023, URL https://openreview.net/forum?id=ju_Uqw384Oq.
[11] Y. Zhang, J. Yan, Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting, in: The Eleventh International Conference on Learning Representations, 2023.
[12] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, in: International Conference on Machine Learning, PMLR, 2022, pp. 27268–27286.
[13] K. Yi, Q. Zhang, W. Fan, S. Wang, P. Wang, H. He, N. An, D. Lian, L. Cao, Z. Niu, Frequency-domain MLPs are more effective learners in time series forecasting, Adv. Neural Inf. Process. Syst. 36 (2024).
[14] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting? in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, (no. 9) 2023, pp. 11121–11128.
[15] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, (no. 12) 2021, pp. 11106–11115.
[16] P. Chen, Y. Zhang, Y. Cheng, Y. Shu, Y. Wang, Q. Wen, B. Yang, C. Guo, Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting, in: The Twelfth International Conference on Learning Representations, 2024, URL https://openreview.net/forum?id=lJkOCMP2aW.
[17] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J.Y. Zhang, J. Zhou, TimeMixer: Decomposable multiscale mixing for time series forecasting, in: The Twelfth International Conference on Learning Representations, 2024, URL https://openreview.net/forum?id=7oLshfEIC2.
[18] C. Challu, K.G. Olivares, B.N. Oreshkin, F.G. Ramirez, M.M. Canseco, A. Dubrawski, Nhits: Neural hierarchical interpolation for time series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, (no. 6) 2023, pp. 6989–6997.
[19] T. Dai, B. Wu, P. Liu, N. Li, J. Bao, Y. Jiang, S.-T. Xia, Periodicity decoupling framework for long-term series forecasting, in: The Twelfth International Conference on Learning Representations, 2024.

[20] M. Liu, A. Zeng, M. Chen, Z. Xu, Q. Lai, L. Ma, Q. Xu, Scinet: Time series modeling and forecasting with sample convolution and interaction, Adv. Neural Inf. Process. Syst. 35 (2022) 5816–5828.
[21] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, Y. Xiao, Micn: Multi-scale local and global context modeling for long-term series forecasting, in: The Eleventh International Conference on Learning Representations, 2023.
[22] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, M. Long, Itransformer: Inverted transformers are effective for time series forecasting, in: The Twelfth International Conference on Learning Representations, 2024.
[23] A. Das, W. Kong, A. Leach, S.K. Mathur, R. Sen, R. Yu, Long-term forecasting with tiDE: Time-series dense encoder, Trans. Mach. Learn. Res. (2023) URL https://openreview.net/forum?id=pCbC3aQB5W.
[24] J. Wang, X. Su, Y. Huang, H. Lai, W. Qian, S. Zhang, Clinear: An interpretable deep time series forecasting model for periodic time series, IEEE Internet Things J. (2025) http://dx.doi.org/10.1109/JIOT.2025.3536460, 1–1.
[25] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, DeepAR: Probabilistic forecasting with autoregressive recurrent networks, Int. J. Forecast. 36 (3) (2020) 1181–1191.
[26] S. Lin, W. Lin, W. Wu, F. Zhao, R. Mo, H. Zhang, SegRNN: Segment recurrent neural network for long-term time series forecasting, 2023, arXiv, URL https://api.semanticscholar.org/CorpusID:261064627.
[27] C. Lea, R. Vidal, A. Reiter, G.D. Hager, Temporal convolutional networks: A unified approach to action segmentation, in: Computer Vision–ECCV 2016 Workshops: Amsterdam, the Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14, Springer, 2016, pp. 47–54.
[28] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, X. Yan, Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, Adv. Neural Inf. Process. Syst. 32 (2019).
[29] B.N. Oreshkin, D. Carpov, N. Chapados, Y. Bengio, N-BEATS: Neural basis expansion analysis for interpretable time series forecasting, in: International Conference on Learning Representations, 2020, URL https://openreview.net/forum?id=r1ecqn4YwB.
[30] K.G. Olivares, C. Challu, G. Marcjasz, R. Weron, A. Dubrawski, Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx, Int. J. Forecast. 39 (2) (2023) 884–900.
[31] W. Fan, S. Zheng, X. Yi, W. Cao, Y. Fu, J. Bian, T.-Y. Liu, DEPTS: Deep expansion learning for periodic time series forecasting, in: International Conference on Learning Representations, 2022, URL https://openreview.net/forum?id=AJAR-JgNw.
[32] Z. Li, Z. Rao, L. Pan, Z. Xu, MTS-mixers: Multivariate time series forecasting via factorized temporal and channel mixing, 2023, arXiv abs/2302.04501, https://api.semanticscholar.org/CorpusID:256697397.
[33] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations, 2021, URL https://openreview.net/forum?id=YicbFdNTTy.
[34] S. Lee, T. Park, K. Lee, Learning to embed time series patches independently, in: The Twelfth International Conference on Learning Representations, 2024, URL https://openreview.net/forum?id=WS7GuBDFa2.
[35] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A.X. Liu, S. Dustdar, Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting, in: International Conference on Learning Representations, 2021.
[36] C. RB, STL: A seasonal-trend decomposition procedure based on loess, J. Off. Stat. 6 (1990) 3–73.
[37] Y. Wang, H. Wu, J. Dong, G. Qin, H. Zhang, Y. Liu, Y. Qiu, J. Wang, M. Long, TimeXer: Empowering transformers for time series forecasting with exogenous variables, in: A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, C. Zhang (Eds.), Advances in Neural Information Processing Systems, vol. 37 (2024) 469–498, URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0113ef4642264adc2e6924a3cbbdf532-Paper-Conference.pdf.
[38] D. Kingma, J. Ba, Adam: A method for stochastic optimization, Comput. Sci. (2014).
[39] H. Wang, Y. Mo, H. Dai, N. Yin, S. Fan, B. Li, S. Mo, FTMLP: MLP with feature-temporal block for multivariate time series forecasting, Neurocomputing 607 (2024) 128365.

Yijun Xu received the B.Sc. degree in Internet of Things engineering from the East China University of Technology, China, in 2023. He is currently working toward the MS degree in Computer Science and Technology from Fujian Normal University, China. His research interests include machine learning, deep learning, and time series.

Weijie Zhang graduated from Fujian Normal University, China in 2024 with a Bachelor of Science degree in Information Engineering. He is currently pursuing a master's degree in software engineering from Fujian Normal University, China. His research interests include deep learning, and its application.

HuiXia Lai received the MS degree in computer science from the Nanjing University of Science and Technology (NUST), China, in 2003. Currently, she is a lecture with the College of Computer and Cyber Security, Fujian Normal University (FJNU). She is the author of more than 10 scientific papers. Her research interests include data mining, artificial intelligence, and their application to time series forecasting, and weather forecasting.

Shi Zhang received the Ph.D. degree in computer science from the Shanghai Jiao Tong University (SJTU), China, in 2008. Currently, he is a professor with the College of Computer and Cyber Security, Fujian Normal University (FJNU). He is the author of more than 50 scientific papers. His research interests include machine learning, artificial intelligence, and their application to anomaly detection, time series forecasting, and other practical problems.

Ruliang Xiao received the Ph.D. degree in computer software and theory from Wuhan University, Wuhan, China, in 2007. He is currently a Professor with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China. He has authored three books, more than 20 invention patents, and has published more than 70 papers in international journals and conference proceedings. His research interests include system security engineering and computing intelligence. Prof. Xiao received the Fujian Provincial Science and Technology Progress Award twice in 2017 and 2019.