Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.058374

ARTICLE

DecMamba: Mamba Utilizing Series Decomposition for Multivariate Time Series Forecasting

Jianxin Feng*, **Jianhao Zhang, Ge Cao, Zhiguo Liu and Yuanming Ding**

Communication and Network Key Laboratory, Dalian University, Dalian, 116622, China

*Corresponding Author: Jianxin Feng. Email: fengjianxin863@163.com

Received: 11 September 2024 Accepted: 22 October 2024 Published: 03 January 2025

ABSTRACT

Multivariate time series forecasting is widely used in traffic planning, weather forecasting, and energy consumption. Series decomposition algorithms can help models better understand the underlying patterns of the original series to improve the forecasting accuracy of multivariate time series. However, the decomposition kernel of previous decomposition-based models is fixed, and these models have not considered the differences in frequency fluctuations between components. These problems make it difficult to analyze the intricate temporal variations of real-world time series. In this paper, we propose a series decomposition-based Mamba model, DecMamba, to obtain the intricate temporal dependencies and the dependencies among different variables of multivariate time series. A variable-level adaptive kernel combination search module is designed to interact with information on different trends and periods between variables. Two backbone structures are proposed to emphasize the differences in frequency fluctuations of seasonal and trend components. Mamba with superior performance is used instead of a Transformer in backbone structures to capture the dependencies among different variables. A new embedding block is designed to capture the temporal features better, especially for the high-frequency seasonal component whose semantic information is difficult to acquire. A gating mechanism is introduced to the decoder in the seasonal backbone to improve the prediction accuracy. A comparison with ten state-of-the-art models on seven real-world datasets demonstrates that DecMamba can better model the temporal dependencies and the dependencies among different variables, guaranteeing better prediction performance for multivariate time series.

KEYWORDS

Data prediction; time series; Mamba; series decomposition

1 Introduction

Multivariate time series (MTS) forecasting is the task of predicting future information based on the historical information of multiple variables. MTS forecasting is widely used in the fields of finance [1,2], transportation [3,4], weather [5], and energy [6]. Recently, deep learning has achieved good prediction results in MTS forecasting tasks [7,8]. Cleveland et al. [9] denoted that separating long-term trends from cyclical variations is beneficial for analyzing complex time series. The various components after decomposition exhibit different underlying patterns. So, a better decomposition algorithm can help the model reveal the underlying patterns of the original series, thereby improving the accuracy of time series forecasting. Previous series decomposition algorithms [10–12] all use fixed kernels to decompose the original series into seasonal and trend components. However, for each variable of multivariate time series, the kernel of the series decomposition algorithm is fixed with the same value. Fixed kernels that only make fixed scale trends and periods interact with information make it difficult to analyze the complex trend correlations and periodic correlations between variables. This is detrimental to modeling the dependencies among different variables. The decomposed seasonal component exhibits stronger frequency fluctuations compared to the trend component. Previous decomposition-based forecasting models have overlooked the differences in sub-series frequency fluctuations between these two modes [10–12]. Transformer has become a mainstream model for MTS tasks, and previous Transformer-based [13] models [7,8,14–16] have achieved good results on MTS tasks. However, the disadvantage of its quadratic complexity in modeling the dependencies among different variables with variable dimensions that are too high leads to its high computation cost. To solve the above problems, we propose a Mamba model based on series decomposition, DecMamba. In DecMamba, we have designed a new decomposition module called Multivariate Adaptive Decomposition (MAD). MAD adopts a variable-level adaptive kernel search method to find the best set of kernel combinations that is most suitable for modeling the dependencies among different variables of seasonal and trend components in multivariate time series. DecMamba uses two different embedding methods to represent the trend and seasonal components separately. For the trend component, similar to iTransformer [17], we embed each variate of the trend component as a token and use Multilayer Perceptron (MLP) to model the temporal dependencies of the trend component easily. Considering its high-frequency fluctuations and semantically separable characteristics of the seasonal component, we have designed a new embedding block called Dimension Extension Embedding (DEE). DEE decomposes the seasonal component with a fixed time delay, and each seasonal sub-series is embedded into an independent token. Semantic information in seasonal components with high-frequency fluctuations is more easily obtained by univariate extending to multivariate. Furthermore, the fluctuation of the sub-series is smoother, and the semantic information is clearer than that of the original seasonal series. Due to the special embedding, the feature maps of seasonal components contain more complex information. Seasonal components using linear projection as a decoder [7,15] tend to make the decoder ignore important information. So, we introduce a gating mechanism to design a new Decoder, Gate Future Prediction (GFP). GFP uses an attention mechanism to score the final feature map of the seasonal component and scales it to the interval 0–1 as the weight matrix of representation. After the gating unit is completed, the important information in the feature map is compressed, and the noisy information is ignored. Finally, the prediction results of the seasonal component are obtained by a linear projection. Recently, the State Space Model (SSM), as a series modeling framework, has gradually become a popular tool in the field of time series analysis due to the virtue of its intrinsic mechanism of ordinary differential equations [18,19]. The ADSSM effectively translates PPG signals into ECG waveforms, showcasing the superior time series analysis capabilities of SSM [18]. Mamba [20], as an innovative extension of SSM, significantly improves the performance of traditional models by introducing a selection mechanism to filter useless information and dynamically adjust the state. Mamba also cleverly incorporates a hardware-aware design for efficient parallel training, making it a strong competitor to models like Transformer in tasks such as natural language understanding [21,22], audio waveform processing [23], computer vision [24–26], and point cloud learning [27]. Based on the linear complexity and high throughput of Mamba, we explore the application of Mamba in MTS by using it as a new master architecture to model the dependencies among different variables. In summary, our contributions are as follows:

1. We design a new decomposition algorithm, Multivariate Adaptive Decomposition (MAD), which can dynamically decompose each variable series in a multivariate time series. The optimal combination of kernels for decomposition is searched for multivariate time series at the variable level to facilitate better modeling the temporal dependencies and the dependencies among different variables.
2. Aiming at the characteristics of high-frequency fluctuation of seasonal components after series decomposition, a new Embedding module, Dimension Extension Embedding (DEE), is designed, which can decompose a single seasonal variable into multiple sub-seasonal variables. DEE can reduce the high-frequency fluctuation of the seasonal series, simplify the semantic information of the seasonal component, and capture the temporal dependency of the seasonal component in a better way with dimension expansion.
3. A new prediction module, Gate Future Prediction (GFP), is designed for seasonal components after series decomposition. This module enables the prediction layer to focus on the important information of seasonal series and ignore the noise information.
4. The linear complexity and high throughput of Mamba make it an alternative to replace the Transformer. Based on the superior performance of Mamba in other domains, we use it as a new master architecture for the dependencies among different variables modeling to explore the application of Mamba in the MTS domain.
5. Based on the above improvements, a new architecture, DecMamba, for multivariate time series forecasting is proposed. Its better forecasting performance is verified by comparing it with ten SOTA models on seven real-world datasets.

2 Related Works

2.1 Multivariate Time Series Forecasting

MTS prediction models can be broadly classified into statistical models and deep learning models. The traditional statistical models are no longer sufficient to meet the current prediction needs of MTS. Due to the superior fitting capabilities, deep learning models are now commonly used as the main architecture for MTS. When modeling dependencies among different variables in multivariate series, deep models typically incorporate three approaches: local channel mixing, global channel mixing, and independent channels. Local channel mixing generally considers multidimensional information at a single time step independently and then captures the temporal dependencies of the series. For example, Recurrent Neural Networks (RNNs) are commonly used for multivariate time series (MTS) tasks. LSTM [28] and GRU [29] introduce gating mechanisms to mitigate the vanishing gradient issue in long sequences. LSTNet [30] improves multivariate time series forecasting by capturing both short-term and long-term patterns. DeepAR [31] enhances the robustness of the model by generating probabilistic forecasts through Monte Carlo sampling. Wen et al. [32] proposed a BiLSTM with temporal pattern attention for ignoring irrelevant information and amplifying the required information. TCN [33] in CNN introduces causal convolution to simulate the temporal relationships in the series with the method of local channel mixing. SCINet [34] in CNN repeatedly extracts and exchanges information at different time resolutions and learns effective representations to improve predictability. SCINet has shown significant advantages in traffic flow forecasting tasks [35]. Transformer models such as Informer [16], Autoformer [10], FEDformer [12], and Stationary [14] use point tokens to mix channels locally. Autoformer, FEDformer, and Stationary have achieved good results in Traffic, Weather, and finance tasks, respectively [36–38]. Crossformer [8] proposes a two-stage Routing Attention mechanism that extracts all dependencies among different variables by routing tokens to each variable. This approach also extends the scope of individual token information due to the patch embedding method. Crossformer has been applied in practical power load forecasting tasks [39]. However, the method of local channel mixing has limitations due to the different sampling frequencies of multiple variables and the semantic gap between them. PatchTST [15] model adopts an independent channel modeling approach, which enhances the robustness of the model by avoiding channel mixing. PatchTST has accurately predicted the air quality index [40]. TimesNet [17] in the CNN model and Dlinear [11] in the MLP model also use the independent channel approach for MTS prediction. TimesNet and Dlinear are widely used in finance and energy domains, respectively [41,42]. TIDE [43] completes the encoding and decoding work based on MLP. However, focusing solely on modeling temporal dependencies while neglecting the dependencies among different variables does not optimally analyze multivariate time series. iTransformer [7] treats individual variate as independent tokens and uses self-attention to model dependencies among different variables. iTransformer is a global channel-mixing approach.

2.2 Time Series Decomposition

Autoformer [10], PEDformer [12], and Dlinear [11], all separate long-term trends from cyclical variations to achieve series decomposition. Autoformer and PEDformer utilize the decomposition block as an internal operator and replace the self-attention mechanism with auto-correlation to model the temporal dependencies in the series. Dlinear first decomposes the original series into trend and seasonal components. Linear networks are used to predict the future information of each component, and the prediction results are summed up to obtain the final forecasting result. However, the aforementioned approaches neglect the different frequencies between the trend component and seasonal component. They also disregard the potential impact that different combinations of trend components and seasonal components among variables could have on forecast accuracy.

3 Methods

In MTS forecasting, our goal is to predict future values X_T based on multivariate historical data input X. Let $X \in R^{L \times C}$ represent an MTS with C variables and a history window L of length. Similarly, let $X_T \in R^{L \times C}$ represent a multivariate time series with C variables and a forecasting window T of length. To achieve higher prediction accuracy, we propose a model called DecMamba.

3.1 Model Structure

The overall architecture of DecMamba is shown in Fig. 1. DecMamba uses Multivariate Adaptive Decomposition (MAD) to perform adaptive decomposition operations on MTS. This process produces trend component $X_{trend} \in R^{L \times C}$, which represents the inconsistent trend changes among different variables, and seasonal component $X_{seasonal} \in R^{L \times C}$, which captures the inconsistent periodic changes among variables. Aligned to the right margin:

$$X_{trend}, X_{seasonal} = MAD(X) \tag{1}$$

Due to the differences in frequency fluctuations between the trend component and seasonal component, two distinct network structures were designed. The Seasonal Backbone (SB) is responsible for forecasting the future values of the seasonal component, while the Trend Backbone (TB) is dedicated to forecasting the future values of the trend component. The two distinct network structures allow DecMamba to effectively predict the two underlying patterns separately.

$$\hat{X}_{trend} = TB(X_{trend}) \tag{2}$$

$$\hat{X}_{seasonal} = SB(X_{seasonal}) \tag{3}$$

The final forecasting result output Y is obtained by summing.

$$Y = \hat{X}_{seasonal} + \hat{X}_{trend} \tag{4}$$

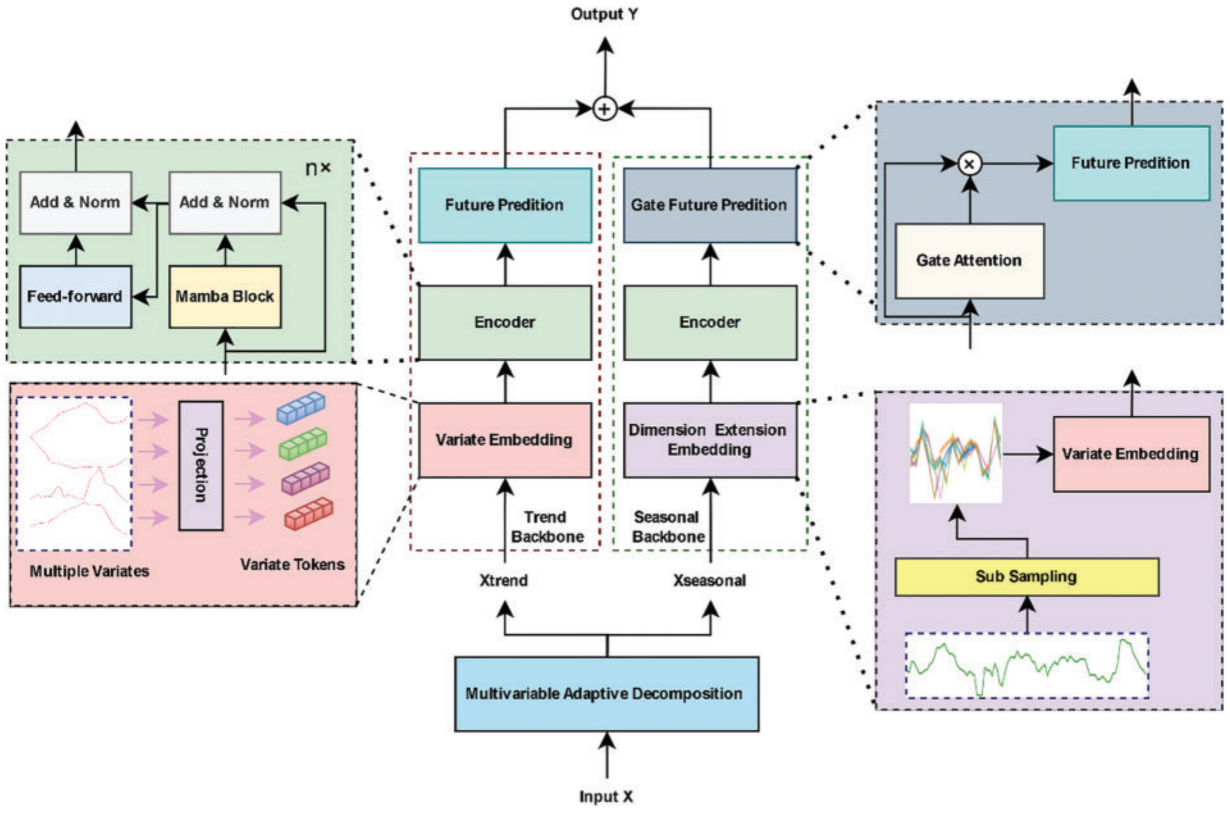


Figure 1: Overall architecture of DecMamba, which consists of Multivariate Adaptive Decomposition, Seasonal Backbone, and Trend Backbone for series decomposition, respectively

3.2 Multivariate Adaptive Decomposition

Previous series decomposition algorithms have often used series decomposition with a fixed kernel. The process is as follows:

$$X_{trend} = AvgPool(Padding(X)_{kernel} \tag{5}$$

$$X_{seasonal} = X - X_{trend} \tag{6}$$

Unlike traditional fixed kernel decomposition algorithms [10–12], MAD finds the optimal combination of decomposition kernels for MTS during the training process with an adaptive search method. The method is as follows:

3.2.1 Initialization

m kernels $kernel_{all}$ with varying sizes, Cost matrix CM and Count Matrix $Count$ are initialized. $kernel_{all}$ is used to represent the search space of kernels. CM records the cost during the training process. $Count$ records the frequency of each kernel selected by the variables during the training process.

3.2.2 Training

$w * C$ kernels are randomly selected in $kernel_{all}$ and divided into w group kernel combinations. Each kernel combination contains C kernels, where $w << batchsize$. The same kernel can be repeatedly selected. Input X of the original batch is divided into w mini-batches that are decomposed by the w kernel combinations. The process is as follows:

$$X_{trend}^{i,j} = AvgPool(Padding(X^{i,j})_{kernel_{rand}^{j}} \tag{7}$$

$$X_{seasonal}^{i,j} = X^{i,j} - X_{trend}^{i,j} \tag{8}$$

where $X^{i,j}$, $X_{trend}^{i,j}$, and $X_{seasonal}^{i,j}$ represent the original series, trend component, and seasonal component for the ith sample in the jth combination, respectively. The kernels in the jth combination are denoted as $kernel_{rand}^{j}$. The value $lc = \{lc_1, \ldots, lc_j, \ldots, lc_w\}$ is used to estimate the effect of the kernel combinations on the prediction results.

$$lc_j = \frac{1}{\frac{bs}{w}} \sum\nolimits_{p=1}^{\frac{bs}{w}} \left| Y^{p,j} - X_T^{p,j} \right| \tag{9}$$

where $Y^{p,j}$, and $X_T^{p,j}$ represent the predicted and true results of the future information for the pth sample in the jth combination, respectively. CM is updated by lc, and the Mean-subtraction operation is used to eliminate the influence of different sample batches.

$$CM_k^i = CM_k^i + \left(lc_j - \frac{1}{w} \sum\nolimits_{q=1}^{w} lc_q \right) \tag{10}$$

CM_k^i represents the cost of the overall prediction outcome when the ith variable chooses the kth kernel as part of the decomposition combination.

3.2.3 Inference

The optimal combination of kernel $kernel_{best}$ is selected from $kernel_{all}$ according to CM. Because CM_k^i represents the cost of the prediction outcome for all variables rather than for individual variables. The ith variable uses k_{mean}th kernel to decompose, which can minimize the overall prediction outcome cost.

$$\frac{CM_{k_{mean}}^i}{Count_{k_{mean}}^i} = min \left(\frac{CM_1^i}{Count_1^i}, \ldots, \frac{CM_k^i}{Count_k^i}, \ldots, \frac{CM_w^i}{Count_w^i} \right) \tag{11}$$

$Count_k^i$ denotes the number of times the kth kernel is chosen by the ith variable. The original series will be decomposed by $kernel_{best}$ to obtain the combinations of trend and seasonal components that are most conducive to modeling the dependencies among different variables. During inference, the process is as follows:

$$X_{trend} = AvgPool(Padding(X)_{kernel_{best}} \tag{12}$$

$$X_{seasonal} = X - X_{trend} \tag{13}$$

3.3 Trend Backbone

Trend Backbone (TB) is composed of Embedding, Encoder (TE), and Future Prediction (FP). *TB* uses the instance normalization operations and embedding block as same as iTransformer [7]. *TB* embeds each time series of X_{trend} as variate tokens h_{trend}. For *TE*, given the embedded trend component hidden variables h_{trend}, the output is a feature map $\hat{h}_{trend}$ of the trend component.

$$\hat{h}_{trend} = TE(h_{trend}) \tag{14}$$

TE uses Mamba to model the dependencies among different variables. The internal structure of Mamba is shown in Fig. 2, where the left branch is a gated nonlinear layer, and the right branch serves as a linear time-invariant (LTI) system (SSM). The temporal dependencies are modeled using a feedforward network (FFN). For the lth layer, the input is h^{l-1}_{trend}, and the output is h^{l}_{trend}. The process of *TE* can be formalized as:

$$h^{l}_{trend} = FFN\left(SSM\left(Conv\left(Linear\left(h^{l-1}_{trend}\right)\right)\right) + \sigma\left(Linear\left(h^{l-1}_{trend}\right)\right)\right) \tag{15}$$

h_{trend} is the input of layer 1, and $\hat{h}_{trend}$ is the output of the last layer. *FP* directly outputs the prediction results of the trend component through a simple linear projection.

$$\hat{X}_{trend} = FP(\hat{h}_{trend}) \tag{16}$$

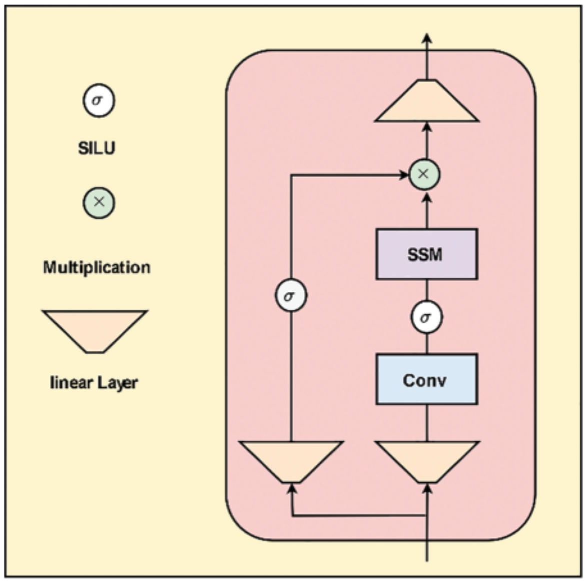


Figure 2: Mamba block

3.4 Seasonal Backbone

The Seasonal Backbone (SB) consists of the Dimension Extension Embedding (DEE), an Encoder (SE), and the Gate Future Prediction (GFP) module.

3.4.1 Dimension Extension Embedding

DEE embeds the seasonal component by expanding its dimensions. To more effectively embed the future map of the decomposed seasonal component, *DEE* achieves semantic segmentation of seasonal components through dimension expansion. Given the inputs $X_{seasonal} \in R^{L \times C}$, the sampled components are obtained based on sub-sampling according to a fixed time delay. $X_{seasonal} \in R^{L \times C}$ is expanded into an extended time series $X_{sample} \in R^{Ls \times delay \times C}$ of length L_s and dimension $delay \times C$. This extended series $X_{sample} \in R^{Ls \times delay \times C}$ is then embedded as separate tokens $h_{seasonal}$ to represent the hidden variables of the seasonal component. Compared to trend embedding methods [7], *DEE* is better suited for embedding high-frequency series like seasonal series, making the semantic information represented by the embedded tokens clearer.

3.4.2 Seasonal Encoder

The encoder of the seasonal component *SE* is the same as *TE*.

$$\hat{h}_{seasonal} = SE(h_{seasonal}) \tag{17}$$

For the lth layer, the input is $h^{l-1}_{seasonal}$, and the output is $h^{l}_{seasonal}$. The process of SE can be formalized as follows:

$$h^{l}_{seasonal} = FFN\left(SSM\left(Conv\left(Linear\left(h^{l-1}_{seasonal}\right)\right)\right) + \sigma\left(Linear\left(h^{l-1}_{seasonal}\right)\right)\right) \tag{18}$$

$h_{seasonal}$ is the input of layer 1, and $\hat{h}_{seasonal}$ is the output of the last layer. *SB* then outputs the prediction results of the seasonal component through *GFP*, which acts as a decoder.

$$\hat{X}_{seasonal} = GFP(\hat{h}_{seasonal}) \tag{19}$$

where $\hat{X}_{seasonal} \in R^{T \times C}$ represents the prediction results of the Seasonal Backbone.

3.4.3 Gate Future Prediction

GFP introduces a gating mechanism to enhance the accuracy of the prediction results. This is done by taking the embedded seasonal component representation, performing a matrix projection through a linear layer to obtain the importance score matrix, and normalizing the score to a range between 0 and 1 using a sigmoid activation function. The score matrix is then multiplied with the seasonal component representation to obtain a more accurate representation. The final prediction of the seasonal component is obtained through a linear projection, similar to the *FP* process in the Trend Backbone.

$$\hat{X}_{seasonal} = Linear\left(sigmoid\left(Linear\left(\hat{h}_{seasonal}\right)\right) * \hat{h}_{seasonal}\right) \tag{20}$$

4 Experiments

4.1 Dataset

We conducted extensive experiments on seven real-world datasets [10], as shown in Table 1, including ETT datasets (including four subsets: ETTh1, ETTh2, ETTm1, ETTm2), Weather, Electricity, Traffic, and Exchange covering domains such as electricity, energy, transportation, weather, and finance.

Table 1: Detailed dataset descriptions

Datasets	Variate	Timesteps
ETTh1 & ETTh2	7	17420
ETTm1 & ETTm2	7	69680
Electricity	321	26304
Weather	21	52696
Traffic	862	17544

4.2 Baselines and Metrics

To demonstrate the effectiveness of DecMamba on the MTS task, ten popular SOTA models were selected for comparison. The baseline models include Autoformer [10], FEDformer [12], Stationary [14], Crossformer [8], PatchTST [15], DLinear [11], TiDE [43], SCINet [34], TimesNet [17], iTransformer [7]. Crossformer and DLinear are SOTA models in the domain of energy [39,42]. Autoformer and SCINet are SOTA models in the domain of transportation [35,36]. FEDformer and PatchTST are SOTA models in the domain of weather [37,40]. Stationary and TimesNet are SOTA models in the domain of finance [38,41]. To measure the prediction performance of multiple models on multiple datasets, Mean Squared Error (MSE) and Mean Absolute Error (MAE) are used as evaluation metrics.

4.3 Implementation Details

Experiments were performed using the Adam optimizer [44] on two NVIDIA T4 16 GB GPUs. The training was performed 10 times with early stopping using patience 3 to avoid overfitting.

4.4 Main Results

Comprehensive forecasting results are listed in Tables 2–9, with the best in bold and the second underlined. The lower MSE/MAE indicates the more accurate prediction result. Overall, DecMamba shows leading performance on most datasets, as well as on different prediction length settings, with 34 top-1 and 52 top-2 cases out of 56 in total. DecMamba performs even better on datasets with a large number of variables like Weather, Electricity, and Traffic. In particular, DecMamba produces all SOTA results on Electricity in MTS forecasting. The variable number of Exchange datasets in finance is less and the time series of financial domains usually include irregular seasonal components [38]. Under such extreme conditions, our model still achieves the 4 top-1 and 6 top-2 ranks across 8 evaluation metrics. The results on the Exchange dataset validate the model's ability to generalize for different types of time series. Compared to SOTA models in various domains, our model has performed well. In the Electricity dataset, our model outperforms Crossformer and DLinear on all metrics. In the Weather dataset, our model surpasses PatchTST on 87.5% of metrics. In the Traffic dataset, our model exceeds Autoformer and SCINet in all metrics, while in the Exchange dataset, it outperforms Stationary and TimesNet in all metrics. iTransformer is the previous best model on MTS forecasting. Our model outperforms iTransformer on 83.93% of the evaluation metrics with all prediction lengths in all datasets.

Table 2: Multivariate long-term series forecasting results on Electricity. All models employ a look-back window length of L = 96

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
Ours	**0.135**	**0.151**	**0.166**	**0.191**	**0.233**	**0.248**	**0.265**	**0.289**
iTransformer	<u>0.148</u>	<u>0.162</u>	<u>0.178</u>	0.225	<u>0.240</u>	<u>0.253</u>	<u>0.269</u>	<u>0.317</u>
PatchTST	0.181	0.188	0.204	0.246	0.270	0.274	0.293	0.324
Crossformer	0.219	0.231	0.246	0.280	0.314	0.322	0.337	0.363
TiDE	0.237	0.236	0.249	0.284	0.329	0.330	0.344	0.373
TimesNet	0.168	0.184	0.198	<u>0.220</u>	0.272	0.289	0.300	0.320
DLinear	0.197	0.196	0.209	0.245	0.282	0.285	0.301	0.333
SCINet	0.247	0.257	0.269	0.299	0.345	0.355	0.369	0.390
FEDformer	0.193	0.201	0.214	0.246	0.308	0.315	0.329	0.355
Stationary	0.169	0.182	0.200	0.222	0.273	0.286	0.304	0.321
Autoformer	0.201	0.222	0.231	0.254	0.317	0.334	0.338	0.361

Table 3: Multivariate long-term series forecasting results on Weather. All models employ a look-back window length of L = 96

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
Ours	<u>0.166</u>	<u>0.215</u>	**0.272**	<u>0.346</u>	**0.213**	<u>0.255</u>	0.298	**0.347**
iTransformer	0.174	0.221	0.278	0.358	<u>0.214</u>	**0.254**	**0.296**	**0.347**
PatchTST	0.177	0.225	0.278	0.354	0.218	0.259	<u>0.297</u>	0.348
Crossformer	**0.158**	**0.206**	**0.272**	0.398	0.230	0.277	0.335	0.418
TiDE	0.202	0.242	0.287	0.351	0.261	0.298	0.335	0.386
TimesNet	0.172	0.219	0.280	0.365	0.220	0.261	0.306	0.359
DLinear	0.196	0.237	0.283	**0.345**	0.255	0.296	0.335	0.381
SCINet	0.221	0.261	0.309	0.377	0.306	0.340	0.378	0.427
FEDformer	0.217	0.276	0.339	0.403	0.296	0.336	0.380	0.428
Stationary	0.173	0.245	0.321	0.414	0.223	0.285	0.338	0.410
Autoformer	0.266	0.307	0.359	0.419	0.336	0.367	0.395	0.428

Table 4: Multivariate long-term series forecasting results on Traffic. All models employ a look-back window length of L = 96

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
Ours	<u>0.407</u>	**0.416**	<u>0.447</u>	<u>0.485</u>	**0.262**	**0.271**	**0.281**	**0.299**
iTransformer	**0.395**	<u>0.417</u>	**0.433**	**0.467**	<u>0.268</u>	<u>0.276</u>	<u>0.283</u>	<u>0.302</u>
PatchTST	0.462	0.466	0.482	0.514	0.295	0.296	0.304	0.322
Crossformer	0.522	0.530	0.558	0.589	0.290	0.293	0.305	0.328
TiDE	0.805	0.756	0.762	0.719	0.493	0.474	0.477	0.449
TimesNet	0.593	0.617	0.629	0.640	0.321	0.336	0.336	0.350
DLinear	0.650	0.598	0.605	0.645	0.396	0.370	0.373	0.394
SCINet	0.788	0.789	0.797	0.841	0.499	0.505	0.508	0.523
FEDformer	0.587	0.604	0.621	0.626	0.366	0.373	0.383	0.382
Stationary	0.612	0.613	0.618	0.653	0.338	0.340	0.328	0.355
Autoformer	0.613	0.616	0.622	0.660	0.388	0.382	0.337	0.408

Table 5: Multivariate long-term series forecasting results on ETTm1. All models employ a look-back window length of L = 96

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
Ours	**0.329**	<u>0.373</u>	**0.396**	<u>0.460</u>	**0.366**	0.390	**0.409**	<u>0.447</u>
iTransformer	0.334	0.377	0.426	0.491	0.368	0.391	0.420	0.459
PatchTST	**0.329**	**0.367**	<u>0.399</u>	**0.454**	<u>0.367</u>	**0.385**	<u>0.410</u>	**0.439**
Crossformer	0.404	0.450	0.532	0.666	0.426	0.451	0.515	0.589
TiDE	0.364	0.398	0.428	0.487	0.387	0.404	0.425	0.461
TimesNet	0.338	0.374	0.410	0.478	0.375	<u>0.387</u>	0.411	0.450
DLinear	0.345	0.380	0.413	0.474	0.372	0.389	0.413	0.453
SCINet	0.418	0.439	0.490	0.595	0.438	0.450	0.485	0.550
FEDformer	0.379	0.426	0.445	0.543	0.419	0.441	0.459	0.490
Stationary	0.386	0.459	0.495	0.585	0.398	0.444	0.464	0.516
Autoformer	0.505	0.553	0.621	0.671	0.475	0.496	0.537	0.561

Table 6: Multivariate long-term series forecasting results on ETTm2. All models employ a look-back window length of L = 96

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
Ours	<u>0.178</u>	<u>0.244</u>	**0.302**	**0.402**	<u>0.263</u>	<u>0.306</u>	**0.343**	**0.400**
iTransformer	0.180	0.250	0.311	0.412	0.264	0.309	0.348	0.407
PatchTST	**0.175**	**0.241**	<u>0.305</u>	**0.402**	**0.259**	**0.302**	**0.343**	**0.400**
Crossformer	0.287	0.414	0.597	1.730	0.366	0.492	0.542	1.042
TiDE	0.207	0.290	0.377	0.558	0.305	0.364	0.422	0.524
TimesNet	0.187	0.249	0.321	0.408	0.267	0.309	0.351	0.403
DLinear	0.193	0.284	0.369	0.554	0.292	0.362	0.427	0.522
SCINet	0.286	0.399	0.637	0.960	0.377	0.445	0.591	0.735
FEDformer	0.203	0.269	0.325	0.421	0.287	0.328	0.366	0.415
Stationary	0.192	0.280	0.334	0.417	0.274	0.339	0.361	0.413
Autoformer	0.255	0.281	0.339	0.433	0.339	0.340	0.372	0.432

Table 7: Multivariate long-term series forecasting results on ETTh1. All models employ a look-back window length of L = 96

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
Ours	<u>0.381</u>	<u>0.434</u>	<u>0.478</u>	**0.490**	**0.398**	**0.428**	**0.451**	**0.480**
iTransformer	0.386	0.441	0.487	0.503	0.405	0.436	<u>0.458</u>	0.491
PatchTST	0.414	0.460	0.501	<u>0.500</u>	0.419	0.445	0.466	<u>0.488</u>
Crossformer	0.423	0.471	0.570	0.653	0.448	0.474	0.546	0.621
TiDE	0.479	0.525	0.565	0.594	0.464	0.492	0.515	0.558
TimesNet	0.384	0.436	0.491	0.521	0.402	<u>0.429</u>	0.469	0.500
DLinear	0.386	0.437	0.481	0.519	<u>0.400</u>	0.432	0.459	0.516
SCINet	0.654	0.719	0.778	0.836	0.599	0.631	0.659	0.699
FEDformer	**0.376**	**0.420**	**0.459**	0.506	0.419	0.448	0.465	0.507
Stationary	0.513	0.534	0.588	0.643	0.491	0.504	0.535	0.616
Autoformer	0.449	0.500	0.521	0.514	0.459	0.482	0.496	0.512

Table 8: Multivariate long-term series forecasting results on ETTh2. All models employ a look-back window length of L = 96

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
Ours	**0.296**	0.381	**0.411**	0.432	**0.344**	**0.397**	**0.426**	0.449
iTransformer	0.297	**0.380**	0.428	**0.427**	0.349	0.400	0.432	**0.445**
PatchTST	0.302	0.388	0.426	0.431	0.348	0.400	0.433	0.446
Crossformer	0.745	0.877	1.043	1.104	0.584	0.656	0.731	0.763
TiDE	0.400	0.528	0.643	0.874	0.440	0.509	0.571	0.679
TimesNet	0.340	0.402	0.452	0.462	0.374	0.414	0.452	0.468
DLinear	0.333	0.477	0.594	0.831	0.387	0.476	0.541	0.657
SCINet	0.707	0.860	1.000	1.249	0.621	0.689	0.744	0.838
FEDformer	0.358	0.429	0.496	0.463	0.397	0.439	0.487	0.474
Stationary	0.476	0.512	0.552	0.562	0.458	0.493	0.551	0.560
Autoformer	0.346	0.456	0.482	0.515	0.388	0.452	0.486	0.511

Table 9: Multivariate long-term series forecasting results on Exchange. All models employ a look-back window length of L = 96

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
Ours	**0.082**	**0.175**	0.326	0.851	**0.201**	**0.298**	0.412	0.694
iTransformer	0.086	0.177	0.331	0.847	0.206	0.299	0.417	**0.691**
PatchTST	0.088	0.176	**0.301**	0.901	0.205	0.299	**0.397**	0.714
Crossformer	0.256	0.470	1.268	1.767	0.367	0.509	0.883	1.068
TiDE	0.094	0.184	0.349	0.852	0.218	0.307	0.431	0.698
TimesNet	0.107	0.226	0.367	0.964	0.234	0.344	0.448	0.746
DLinear	0.088	0.176	0.313	**0.839**	0.218	0.315	0.427	0.695
SCINet	0.267	0.351	1.324	1.058	0.396	0.459	0.853	0.797
FEDformer	0.148	0.271	0.460	1.195	0.278	0.315	0.427	0.695
Stationary	0.111	0.219	0.421	1.092	0.237	0.335	0.476	0.769
Autoformer	0.197	0.300	0.509	1.447	0.323	0.369	0.524	0.941

4.5 Model Efficiency

We analyze the computational complexity of our models and baseline models. In Table 10, C represents the number of variables, L is the sequence length, P refers to the patch size, and L_{seg} denotes the segment length in Crossformer. Models like Autoformer, Stationary, FEDformer, and SCINet, which embed variables into tokens at each time step, have complexity dependent only on

L. However, excessively local receptive fields may reduce prediction accuracy. Channel-independent models like TiDE, DLinear, TimesNet, and PatchTST scale linearly with C. Crossformer also shows linear complexity in C, while iTransformer is quadratic. Due to the linear complexity of Mamba, the token of DecMamba aggregates the global representations but is only linearly with L.

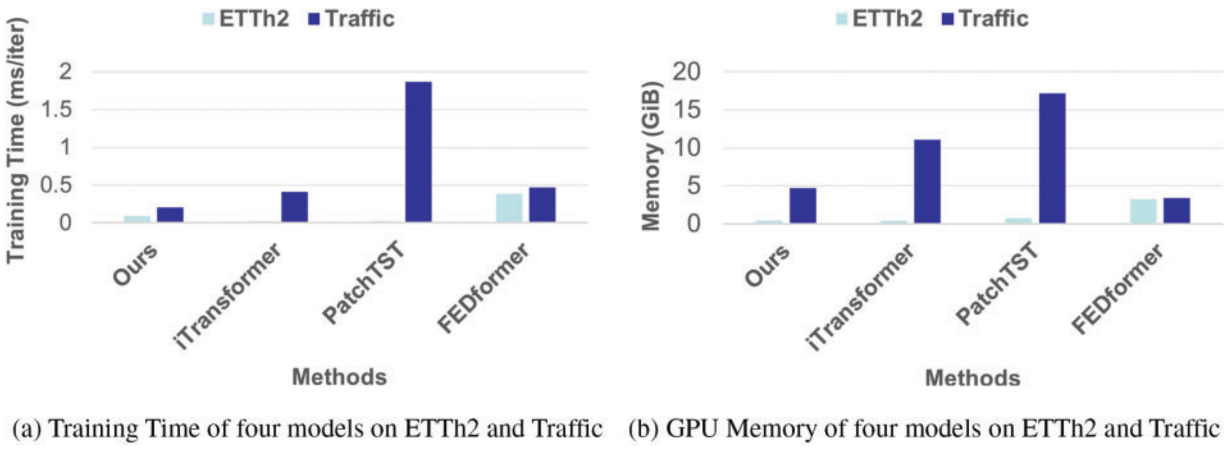
Table 10: Computational complexity analysis

Methods	Computational complexity
Ours	$O\left(C \times delay\right)$
iTransformer	$O\left(C^2\right)$
Crossformer	$O\left(\frac{C}{\left(L_{seg}\right)^2} \times \left(\frac{L}{P}\right)^2\right)$
PatchTST	$O\left(C \times \left(\frac{L}{P}\right)^2\right)$
TiDE	$O\left(C \times L\right)$
TimesNet	$O\left(C \times L\right)$
DLinear	$O\left(C \times L\right)$
SCINet	$O\left(L\right)$
FEDformer	$O\left(L\right)$
Stationary	$O\left(L^2\right)$
Autoformer	$O\left(LlogL\right)$

FEDformer, PatchTST, and iTransformer in Transformer are representative models of three modeling methods, respectively. We compare the running time and memory consumption of our model with these models on traffic and ETTh2 datasets. As shown in Fig. 3, on the ETTh2 dataset with fewer variables, FEDformer has the highest memory and training time. Our model's memory usage is similar to iTransformer and PatchTST, with a slightly longer training time. However, on the Traffic dataset (with more variables), iTransformer and PatchTST show significant increments in memory and time, while the complexity increment of our model is not evident compared to these models. These results show that Mamba is more efficient compared to Transformer, and our model has better scalability and applicability in real-world applications.

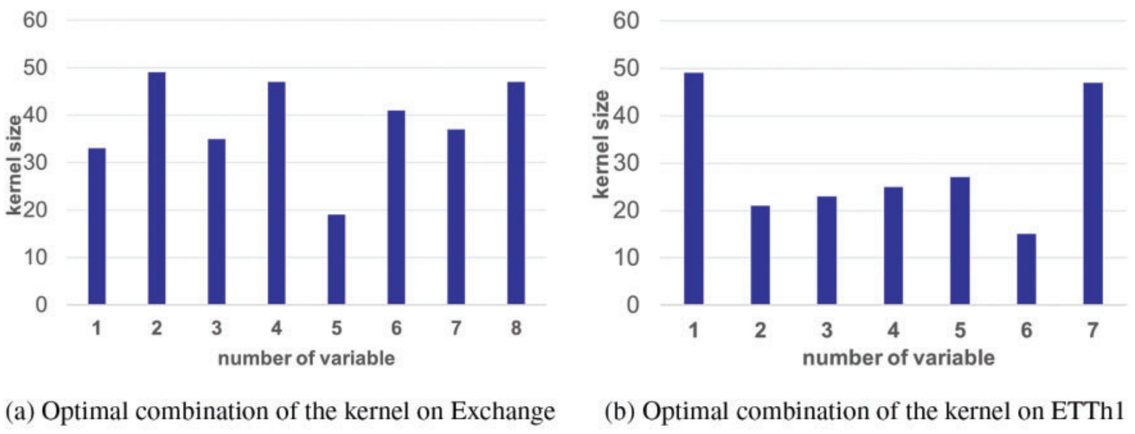
4.6 Visualize the Optimal Combination of Kernels

As shown in Fig. 4, the sizes of optimal decomposition kernel combinations are obtained by MAD on ETTh1 and Exchange datasets. The sizes on ETTh1 are different from Exchange. MAD tends to select larger sizes of decomposition kernel combinations to decompose original series on Exchange. Compared with ETTh1, Exchange exhibits more complex or irregular seasonality. The decomposition kernel combinations with larger sizes can make trend components with clearer trend information and seasonal components with more original information. So better prediction performance can be obtained. The kernel sizes assigned to each variable in ETTh1 and Exchange datasets differ. These kernel sizes are adjusted with the prediction results during the training process. MAD adaptively selects suitable decomposition kernel combinations according to the characteristics of datasets. Accurate prediction results can be obtained by optimizing decomposition with a better generalization ability for various types of multivariate time series.



(a) Training Time of four models on ETTh2 and Traffic (b) GPU Memory of four models on ETTh2 and Traffic

Figure 3: Comparison of our model and three baselines on Training Time, and GPU Memory. All versions employ a look-back window length of L = 96 and a predicted length of T = 96 on Traffic and Etth2 datasets



(a) Optimal combination of the kernel on Exchange (b) Optimal combination of the kernel on ETTh1

Figure 4: Visualization of the optimal combination of kernels

4.7 Impact of the Number of Variables on Model Performance

In this section, we analyze how the number of variables affects the performance of our model. We conducted a controlled experiment using the Electricity dataset. The number of input variables is set to 20%, 60%, and 100% of the original variables. The number of output variables is uniformly set to 20% of the original variables. Table 11 shows that increasing the number of input variables improves prediction accuracy. Memory consumption increases with the increment of input variables, but training time remains stable. These results indicate that our model effectively captures the dependencies among additional and output variables. Our model demonstrates scalability and robustness in scenarios with varying numbers of variables.

Table 11: Impact of the number of variables on the model's performance. All versions employ a look-back window length of L = 96 and a predicted length of T = 96 on Electricity

Number of variables	MSE	MAE	Training time (ms/iter)	Memory (GiB)
100%	0.104	0.200	0.1245	2.000
60%	0.105	0.202	0.1181	1.400
20%	0.114	0.212	0.1110	0.7832

4.8 Ablation Studies and Analyses

4.8.1 Component Ablation

We removed the corresponding modules to perform ablation studies on the Electricity. W/O-MAD uses a traditional decomposition that is a fixed decomposition kernel instead of MAD. The embedding method of W/O-DEE is similar to itransformer [7] instead of DEE. W/O-GFP replaces GFP in the original decoder [7,15]. As shown in Table 12, the prediction performance of W/O-MAD and W/O-DEE is significantly lower than our original model. Especially in the long series tasks, performance degradation is very obvious. This experiment can demonstrate that MAD can decompose original series into the trend and seasonal component combinations, which are more suitable for modeling the dependencies among different variables. DEE is more conducive to the representation and prediction of seasonal components. GFP is more robust than the original decoder [7,15], resulting from focusing on the valid information of the high-frequency seasonal components by ignoring noise information.

Table 12: Ablation of different components on Electricity

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
Ours	**0.135**	**0.151**	**0.166**	**0.191**	0.233	**0.248**	0.265	**0.289**
W/O-MAD	**0.135**	0.154	0.167	0.198	**0.232**	0.250	0.265	0.291
W/O-DEE	0.141	0.158	0.173	0.208	0.238	0.253	0.270	0.302
W/O-GFP	0.137	0.153	0.167	0.196	0.233	0.250	**0.264**	0.293

4.8.2 Study on Hyperparameter Sensitivity

delay is an important parameter of DEE, and the length of *delay* directly affects the complexity of semantic information in a single seasonal component. We set the length of *delay* to 4, 6, and 8, respectively, to evaluate the performance of our models on Electricity. As shown in Table 13, with delay length 6, the best performance can be obtained.

Table 13: Results with different delay lengths on Electricity

Predicted length	MSE				MAE			
	96	192	336	720	96	192	336	720
delay = 4	0.136	0.152	0.175	0.194	0.233	0.248	0.275	0.292
delay = 6	0.135	0.151	**0.166**	**0.191**	0.233	0.248	**0.265**	**0.289**
delay = 8	**0.134**	**0.150**	0.167	0.201	**0.231**	**0.246**	0.266	0.297

5 Conclusion

We propose a model called DecMamba for multivariate time series (MTS) forecasting. DecMamba addresses the limitations of previous series decomposition algorithms with MAD searching for kernels adaptively. DEE, a new embedding method, overcomes the shortcomings of previous methods struggling to capture the semantic information in high-frequency seasonal components. A decoder called GFP with a gating mechanism is designed to minimize the impact of unnecessary information from the seasonal component. The experiment results show that MAD better resolves complex trend and period dependencies among variables, DEE enhances the ability to extract semantic information in seasonal components by improving prediction performance, and GFP improves robustness. Additionally, the main architectural structure with Mamba throughput is increased, and the complexity of excessive time is reduced. DecMamba excels in capturing the dependencies among different variables compared to seven real-world datasets. Varying in prediction lengths and metrics, DecMamba ranks first among the ten models in 34 out of 56 settings and achieves the top 2 in 52 of them.

Acknowledgement: We thank the members of the Communication and Network Key Laboratory for their contributions to this work.

Funding Statement: This work was supported in part by the Interdisciplinary Project of Dalian University (DLUXK-2023-ZD-001).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Jianxin Feng and Jianhao Zhang; data collection: Jianhao Zhang; analysis and interpretation of results: Jianxin Feng and Jianhao Zhang; draft manuscript preparation: Jianxin Feng, Jianhao Zhang, Ge Cao, Zhiguo Liu and Yuanming Ding. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All datasets are collected by Reference [10], and they are openly available at https://github.com/thuml/Autoformer (accessed on 10 September 2024).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] W. Yin, Z. L. Chen, X. X. Luo, and B. Kirkulak-Uludag, "Forecasting cryptocurrencies' price with the financial stress index: A graph neural network prediction strategy," *Appl. Econ. Lett.*, vol. 31, no. 7, pp. 630–639, 2024. doi: 10.1080/13504851.2022.2141436.

[2] J. Wang and J. Wang, "A new hybrid forecasting model based on SW-LSTM and wavelet packet decomposition: A case study of oil futures prices," *Comput. Intell. Neurosci.*, vol. 2021, no. 1, 2021, Art. no. 7653091. doi: 10.1155/2021/7653091.

[3] Q. Y. Luo, S. L. He, X. Han, Y. H. Wang, and H. F. Li, "LSTTN: A long-short term transformer-based spatiotemporal neural network for traffic flow forecasting," *Knowl.-Based Syst.*, vol. 293, no. 2, 2024, Art. no. 111637. doi: 10.1016/j.knosys.2024.111637.

[4] J. H. Chen, L. Yang, C. Qin, Y. Yang, L. Peng and X. T. Ge, "Heterogeneous graph traffic prediction considering spatial information around roads," *Int. J. Appl. Earth Obs. Geoinf.*, vol. 128, 2024, Art. no. 103709. doi: 10.1016/j.jag.2024.103709.

[5] H. Balti, A. B. Abbes, and I. R. Farah, "A Bi-GRU-based encoder-decoder framework for multivariate time series forecasting," *Soft Comput.*, vol. 28, no. 9–10, pp. 6775–6786, 2024. doi: 10.1007/s00500-023-09531-9.

[6] S. Smyl, G. Dudek, and P. Pelka, "Contextually enhanced ES-dRNN with dynamic attention for short-term load forecasting," *Neural Netw.*, vol. 169, no. 2, pp. 660–672, 2024. doi: 10.1016/j.neunet.2023.11.017.

[7] Y. Liu *et al.*, "iTransformer: Inverted transformers are effective for time series forecasting," in *Twelfth Int. Conf. Learn. Represent.*, Kigali, Rwanda, 2024.

[8] Y. H. Zhang and J. C. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *Eleventh Int. Conf. Learn. Represent.*, Kigali, Rwanda, 2023.

[9] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A seasonal-trend decomposition procedure based on loess," *J. Off. Stat.*, vol. 6, no. 3, pp. 3–73, 1990.

[10] H. X. Wu, J. H. Xu, J. M. Wang, and M. S. Long, "Autoformer: Decomposition transformers with autocorrelation for long-term series forecasting," in *Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2021, vol. 34, pp. 22419–22430.

[11] A. L. Zeng, M. X. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?," *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 9, pp. 11121–11128, 2023. doi: 10.1609/aaai.v37i9.26317.

[12] T. Zhou, Z. Q. Ma, Q. S. Wen, X. Wang, L. Sun and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proc. 39th Int. Conf. Mach. Learn.*, Baltimore, MD, USA, 2022, vol. 162, pp. 27268–27286.

[13] Vaswani *et al.*, "Attention is all you need," in *Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, vol. 30.

[14] Y. Liu, H. X. Wu, J. M. Wang, and M. S. Long, "Non-stationary transformers: Exploring the stationarity in time series forecasting," in *Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2022, vol. 35, pp. 9881–9893.

[15] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *Eleventh Int. Conf. Learn. Represent.*, Kigali, Rwanda, 2023.

[16] H. Y. Zhou *et al.*, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *Proceedings AAAI Conf. Artifi. Intell.*, vol. 35, no. 12, pp. 11106–11115, 2021. doi: 10.1609/aaai.v35i12.17325.

[17] H. X. Wu, T. G. Hu, Y. Liu, H. Zhou, J. M. Wang and M. S. Long, "TimesNet: Temporal 2D-variation modeling for general time series analysis," in *Eleventh Int. Conf. Learn. Represent.*, Kigali, Rwanda, 2023.

[18] K. Vo, M. El-Khamy, and Y. Choi, "PPG to ECG signal translation for continuous atrial fibrillation detection via attention-based deep state-space modeling," 2023, *arXiv:2309.15375*.

[19] S. Moontaha, B. Arnrich, and A. Galka, "State space modeling of event count time series," *Entropy*, vol. 25, no. 10, 2023, Art. no. 1372. doi: 10.3390/e25101372.

[20] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," 2023, *arXiv:2312.00752*.

[21] O. Lieber *et al.*, "Jamba: A hybrid Transformer-Mamba language model," 2024, *arXiv:2403.19887*.

[22] L. L. Ren, Y. Liu, Y. D. Lu, Y. L. Shen, C. Liang and W. Z. Chen, "Samba: Simple hybrid state space models for efficient unlimited context language modeling," 2024, *arXiv:2406.07522*.

[23] M. H. Erol, A. Senocak, J. Feng, and J. S. Chung, "Audio Mamba: Bidirectional state space model for audio representation learning," 2024, *arXiv:2406.03344*.

[24] J. T. Zhang, K. Bian, P. Cheng, W. B. An, J. N. Liu and J. Zhou, "Vim-F: Visual state space model benefiting from learning in the frequency domain," 2024, *arXiv:2405.18679*.

[25] G. Y. M. Fu, F. C. Xiong, J. F. Lu, and J. Zhou, "SSUMamba: Spatial-spectral selective state space model for hyperspectral image denoising," 2024, *arXiv:2405.01726*.

[26] Z. Y. Zhang, A. Liu, I. Reid, R. Hartley, B. Zhuang and H. Tang, "Motion Mamba: Efficient and long sequence motion generation with hierarchical and bidirectional selective SSM," 2024, *arXiv:2403.07487*.

[27] T. Zhang, X. T. Li, H. B. Yuan, S. P. Ji, and S. C. Yan, "Point Could Mamba: Point cloud learning via state space model," 2024, *arXiv:2403.00762*.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[29] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[30] G. K. Lai, W. Chang, Y. M. Yang, and H. X. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, 2018, pp. 95–104.

[31] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *Int. J. Forecast*, vol. 36, no. 3, pp. 1181–1191, 2020. doi: 10.1016/j.ijforecast.2019.07.001.

[32] J. h. Wen and Z. J. Wang, "Short-Term power load forecasting with hybrid TPA-BiLSTM prediction model based on, CSSA," *Comput. Model. Eng. Sci.*, vol. 136, no. 1, pp. 749–765, 2023. doi: 10.32604/cmes.2023.023865.

[33] S. J. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.

[34] M. H. Liu *et al.*, "SCINet: Time series modeling and forecasting with sample convolution and interaction," in *Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2022, vol. 35, pp. 5816–5828.

[35] K. Gong *et al.*, "TrafficSCINet: An adaptive spatial-temporal graph convolutional network for traffic flow forecasting," in *Adv. Intell. Comput. Technol. Appl.*, Singapore, 2023, pp. 628–639.

[36] S. Luo and Q. Li, "Application of Autoformer to short-term traffic flow prediction," *Int. J. Scientif. Adv.*, vol. 5, no. 1, pp. 85–87, 2024. doi: 10.51542/ijscia.v5i1.15.

[37] Y. N. Cao, Q. Zhou, J. L. Tang, and Z. H. Liu, "Research on haze prediction method of Xianyang city based on STL decomposition and FEDformer," in *Third Int. Conf. Algorithms Microchips Netw. Appl.*, Xi'an, China, 2024, vol. 13171. doi: 10.1117/12.3031964.

[38] C. Y. Gou, R. Zhao, and Y. H. Guo, "Stock price prediction based on non-stationary transformers model," in *2023 9th Int. Conf. Comput. Commun.*, 2023, pp. 2227–2232. doi: 10.1109/ICCC59590.2023.10507459.

[39] S. T. Li and H. F. Cai, "Short-term power load forecasting using a VMD-Crossformer model," *Energies*, vol. 17, no. 11, 2024, Art. no. 2773. doi: 10.3390/en17112773.

[40] W. Y. Cao, R. F. Zhang, and W. X. Cao, "Multi-site air quality index forecasting based on spatiotemporal distribution and PacthTST-enhanced: Evidence from Hebei province in China," *IEEE Access*, vol. 12, no. 83, pp. 132038–132055, 2024. doi: 10.1109/ACCESS.2024.3460187.

[41] Y. L. Huang, C. J. Zhou, K. Cui, and X. P. Lu, "A multi-agent reinforcement learning framework for optimizing financial trading strategies based on TimesNet," *Expert. Syst. Appl.*, vol. 237, no. 2, 2024, Art. no. 121502. doi: 10.1016/j.eswa.2023.121502.

[42] Y. J. Zhao, S. F. Cen, J. G. Hur, and C. Lim, "Energy demand and renewable energy generation forecasting for optimizing dispatching strategies of virtual power plants using time decomposition-based Dlinear," in *Adv. Syst. Eng.*, Wroclaw, Poland, 2023, pp. 3–11.

[43] A. Das, W. Kong, A. Leach, S. Mathur, R. Sen and R. Yu, “Long-term forecasting with TIDE: Time-series dense encoder,” 2023, *arXiv:2304.08424*.

[44] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd Int. Conf. Learn. Represent.*, Kigali, Rwanda, 2015.