2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

PatchMamba: Multivariate Time Series Forecasting Model Based on Patch Attention and Mamba

Haitao Xiong, Feng Shao, and Yuanyuan Cai (✉)

School of Computer and Artificial Intelligence, Beijing Technology and Business University, Beijing 100048, China
caiyuanyuan@btbu.edu.cn

Abstract. Multivariate time series forecasting (MTSF) has broad applications in real-life situations. However, multivariate time series data in different scenarios may exhibit different inter-sequence or intra-sequence dependencies. Existing models often struggle to capture these complex dependencies between time series and variables simultaneously, thus limiting forecasting accuracy. To address these issues, this paper proposes PatchMamba, a multivariate time series forecast model based on Patch Attention and Mamba. PatchMamba includes two novel Patch Attention mechanisms: CD-Patch Attention under the channel dependence strategy and CI-Patch Attention under the channel independence strategy. CD-Patch effectively captures the inter-variable dependencies. In contrast, CI-Patch Attention takes each variable individually to extract local features, avoiding cross-channel interference. Furthermore, we use bidirectional Mamba (Bi-Mamba) to capture long temporal dependency information. Experiments show that PatchMamba achieves higher forecast accuracy on multiple real-world datasets compared to current state-of-the-art (SOTA) models. In addition, this paper validates the role and robustness of the model components through ablation experiments and parameter sensitivity analysis.

Keywords: Multivariate Time series forecasting, Bi-Mamba, Patch Attention, PatchMamba.

1 Introduction

Time series data are sequences arranged in chronological order and are usually sampled at equal intervals of time frequency. A multivariate time series (MTS) consists of multiple variables recorded simultaneously at the same time nodes. As a key data mining technique, MTSF is widely used in many fields, including energy consumption[1],traffic flow estimation[2], climate forecasting[3], and economic trend forecasting[4]. Complex correlations between variables and long-term dependencies make MTSF more challenging. Its development has evolved from traditional statistical and machine learning models to the current SOTA deep learning models.

Statistical models include ARIMA, SARIMA, and GARCH[5-7]. They rely on strict mathematical derivation and probability assumptions to mine linear relationships in a single variable time series and have achieved some success in simple scenarios. To

handle complex situations of multivariate interaction, researchers have developed machine learning-based models. Models such as Support Vector Machines, neural networks, and Random Forests[8-10] can extract nonlinear relationships and dynamic changes in MTS, improving the ability to make predictions. However, these models are sensitive in parameter selection and have limited prediction ability.

In recent years, with the development of deep learning techniques in natural language processing (NLP) and computer vision (CV), researchers have increasingly applied these techniques to time series forecasting. Recurrent neural networks (RNNs) and their variants[11, 12] have been increasingly used in MTSF to achieve more advanced prediction performance. However, RNN-based models have two main[13] drawbacks: (1) error accumulation from dynamic decoding; (2) difficulty in extracting long-sequence features due to vanishing gradient and memory limits.

To handle these problems, the Transformer[14] model is introduced into MTSF. With its self-attention mechanism, Transformer can process sequence data in parallel, effectively capturing capture cross-time dependencies and improving the forecast performance. However, existing models have not fully considered the intra-sequence and inter-sequence correlations. (1) Intra-sequence correlation. It involves the extraction of long-term correlations and local features. Because time series data has a lower semantic information density at any given point in time compared to other sequence data types, capturing intra-series correlation is particularly challenging. (2) Inter-sequence correlation. It refers to exploring the relationships and dependencies between different variables. Multivariate time series data in different scenarios may exhibit different dependencies. For example, in weather forecasting, there are complex interactions between variables such as temperature, humidity, and barometric pressure; in power systems, different power load characteristics may have independent evolutionary laws.

In this paper, we present PatchMamba to overcome the challenges mentioned above. It divides data into patches to extract richer semantic information and combines channel independence (CI) and channel dependence (CD) processing strategies. This approach effectively captures complex intra-sequence and inter-sequence correlations in MTSF across different scenarios. Our contributions are as follows:

- We propose the PatchMamba model, which effectively combines the Patch Attention mechanism with the Mamba block. PatchMamba can capture local time series features through the Patch Attention mechanism and can extract long-term dependencies by Mamba.
- We propose a dual strategy attention mechanism, Patch Attention, which consists of CI-Patch Attention and CD-Patch Attention. It able to selectively capture local intra-sequence correlation and multivariate interactions based on the characteristics of multivariate time series data. And enables the model to be more flexible in different prediction tasks and significantly improves the prediction performance.
- We conducted extensive experiments on six real-world datasets, and PatchMamba significantly reduces the MSE and MAE metrics compared to the current SOTA, demonstrating its effectiveness in MTSF.

2 Methodology

2.1 Problem Definition

The time series forecasting task is defined, given a historical observation window $X = [x_1, x_2, x_3 \ldots x_L] \epsilon R^{L \times N}$, where $x_n = [v_1, v_2, v_3 \ldots v_N]$, to forecast a lengthy window of future time series $Y = [x_{L+1}, x_{L+2}, \ldots, x_{L+T}] \epsilon R^{T \times N}$. In this context, L and T represent the historical look back window and the forecast length, respectively, while V is the variable, and N represents the number of variables.

2.2 Model Architecture

The overview architecture of PatchMamba is shown in **Fig. 1**. First, the input sequence X passes through the Sequence Relation Aware (SRA) decider to compute variable correlations and select a modeling strategy. Generate embedding vectors by Patch-Embedding and pass them to Patch Attention to extract local dependencies. Next, a Feed-forward Neural Network (FFN) boosts the model's expressive power to capture complex features and patterns. Then, Bi-Mamba captures long temporal features, followed by another FFN that extracts deeper into temporal information. Finally, a flatten linear layer generates the result Y. The details of modules are given in the following sections.

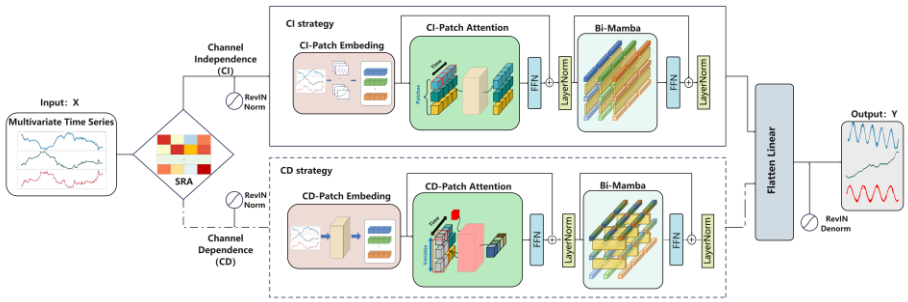


Fig. 1. The architecture of PatchMamba.

2.3 SRA decider

We employ the SRA decider[15] dynamically select different channel interactions based on the correlation of the multivariable time series: channel independence and channel dependence. The core of SRA decider is using the Spearman correlation coefficient to compute the correlation between variables. It provides a more effective strategy selection for multivariate time series modeling.

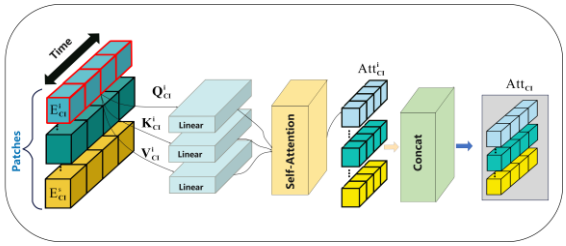
2.4 Patch Embedding

To address the heterogeneous characteristics of multivariate time series data, Patch-Mamba adopts a dual-stream adaptive embedding architecture: channel independence streaming where each univariate is processed individually, and channel dependence streaming that captures the system dependencies using cross-variate interactions. (1) CI-Patch Embedding: In the univariate channel sequences$X^i_{1:\ L}$, the data is initially divided into $P^i \epsilon R^{s\times m}$, where s denotes the number of patches and m signifies the length of each patch. Each univariate channel is concatenated to the embedding $E_{CI}\epsilon R^{s\times m\times d}$, where d is the hidden state dimension. (2) CD-Patch Embedding: For multivariate time series data $X^{l\times n}$ obtained embedding $E_{CD}\epsilon R^{l\times n\times d}$,where l is the sequence length, and n is the number of variables.

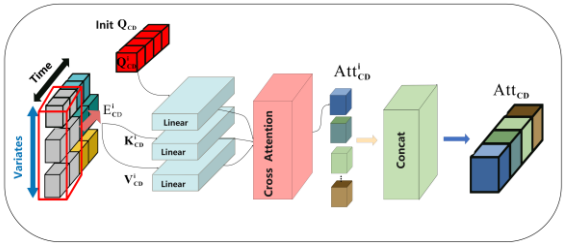
In addition, to address the distribution shift between the training set and the test set, the data normalization method RevIN[16] is employed to enhance the model's robustness.

2.5 Patch Attention

According to different handle strategies, we have designed CI-Patch Attention and CD-Patch Attention, as shown in **Fig. 2**.



(a) CI-Patch Attention



(b) CD-Patch Attention

Fig. 2. The architecture of Patch Attention mechanism.

CI-Patch Attention. CI-Patch Attention establishes relationships between time steps within each patch. For each channelized time series embedding $E_{CI}\epsilon R^{s\times m\times d}$, the i-th

Patch block $E_{CI}^{i} \epsilon R^{m \times d}$ is selected. A trainable linear transformation is applied to obtain the query, key, and value in the attention operation, denoted as $Q_{CI}^{i}, K_{CI}^{i}, V_{CI}^{i} \epsilon R^{m \times d_i}$. We then compute Attention $Att_{CI}^{i} \epsilon R^{m \times d}$, which involves interactions within the Patch. The CI-Patch Attention are calculated as follows:

$$Att_{CI}^{i} = Softmax\left(\frac{Q_{CI}^{i}\left(K_{CI}^{i}\right)^{T}}{\sqrt{d_i}}\right)V_{CI}^{i} \tag{1}$$

After CI-Patch Attention, each patch has its own input and output of constant length, which mixes the dynamic features of local regions in the time series and improves the capture of intra-sequence correlation. The attention results from all patches are combined to produce the output of CI-Patch Attention $\mathrm{Att}_{\mathrm{CI}} \epsilon R^{\mathrm{s} \times \mathrm{m} \times d}$, which represents local details from nearby steps in the time series. That process is as follows:

$$Att_{CI} = Concat\left(Att_{CI}^{1}, \ldots, Att_{CI}^{s}\right) \tag{2}$$

CD-Patch Attention. The CD-Patch Attention approach involves the partitioning of the input embedding $\mathrm{E}_{\mathrm{CD}} \epsilon R^{\mathrm{l} \times \mathrm{n} \times \mathrm{d}}$ into a Patch $\mathrm{E}_{\mathrm{CD}}^{i} \epsilon R^{\mathrm{m} \times \mathrm{n} \times \mathrm{d}}$, which contains m data points. Subsequently, the $\mathrm{E}_{\mathrm{CD}}^{i}$ is subjected to a trainable linear transformation to obtain the key and value in the attention operation, denoted as $K_{\mathrm{CD}}^{i}, \mathrm{V}_{\mathrm{CD}}^{i} \epsilon R^{\mathrm{m} \times d_m}$. A trainable query matrix, $Q_{CD}^{i} \epsilon R^{1 \times d_m}$ to pool the contexts within the Patch. Computing the cross-attention between $Q_{CD}^{i}, K_{\mathrm{CD}}^{i}, \mathrm{V}_{\mathrm{CD}}^{i}$ to capture the interactions between local variables within the i-th Patch. The CD-Patch Attention expressed as:

$$Att_{CD}^{i} = Softmax\left(\frac{Q_{CD}^{i}\left(K_{CD}^{i}\right)^{T}}{\sqrt{d_m}}\right)V_{CD}^{i} \tag{3}$$

Following CD-Patch Attention, each Patch output length becomes $\mathrm{Att}_{\mathrm{CD}}^{i} \epsilon R^{1 \times \mathrm{n} \times d_m}$. It enables the explicit capture of dependencies between different variables. The connection of attention results from all Patches produces the output $\mathrm{Att}_{\mathrm{CD}} \epsilon R^{\mathrm{s} \times \mathrm{m} \times d_m}$ of CD-Patch Attention, which represents dependencies between variables from local time steps. That process is as follows:

$$Att_{CD} = Concat(Att_{CD}^{1}, \ldots, Att_{CD}^{s}) \tag{4}$$

FFN. The input of FFN is the output of Patch Attention with the following expression:

$$FFN(Att) = Dropout\left(ReLU\left(Att_{CI/CD}W_1 + b_1\right)W_2 + b_2\right) \tag{5}$$

$$X_{out} = LayerNorm\left(FFN\left(Att_{CI/CD}\right) + x\right) \tag{6}$$

Where W_1 is the hidden layer weight and b_1 is the hidden layer bias, W_2 is the output layer weight and b_2 is the output layer bias. The output dimension of the FFN is $X_{out} \epsilon R^{B \times J \times D}$, with B and J corresponding to N or S, depending on the handling

strategy. Specifically, under a channel independence strategy, $X_{out} \epsilon R^{N \times S \times D}$; otherwise, $X_{out} \epsilon R^{S \times N \times D}$.

2.6 Bi-Mamba

Given the complex evolution patterns of time series in different directions and the intricate dependencies among sequences, we introduce a Bi-Mamba block to handle. The architecture of Bi-Mamba as shown in **Fig. 3**.

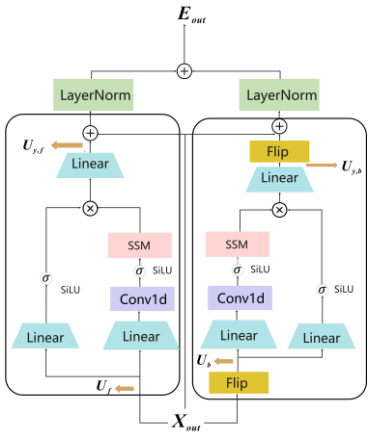


Fig. 3. The architecture of Bi-Mamba

Bi-Mamba's forward input $U_f = X_{out}$, backward input $U_b = Flip(X_{out})$. Each Mamba processing flow is as follows: firstly, the input $U_{dir}(dir \epsilon \{f, b\})$ is passed through two linear layers, which extend the hidden dimension to $\widetilde{U}$, as shown in **Fig. 3**. The result of one branch is passed through a 1-D convolutional layer, a SiLU activation function, and a standard SSM block to obtain the candidate output $U_{dir}^{'}$. The result of the other branch is passed through a SiLU activation function as a gate, which is multiplied with $U_{dir}^{'}$ to obtain the final output $U_{y,dir}$.The forward and backward outputs are summed by residual concatenation and layer normalization for bi-directional feature fusion, the formula of this part is demonstrated as follows:

$$E_{out} = LayerNorm\left(U_{y,f} + X_{out}\right) + LayerNorm\left(Flip(U_{y,b}) + X_{out}\right) \quad (7)$$

The fusion features are input to the FFN, which is computed as follows:

$$FFN(E_{out}) = Dropout(ReLU(E_{out}W_1 + b_1)W_2 + b_2) \quad (8)$$

$$M_{out} = LayerNorm(FFN(E_{out}) + X_{out}) \quad (9)$$

The output M_{out} is reconstructed into the target time series via flatten linear layer. Subsequently, through RevIn denormalization produces the final prediction $\hat{Y} \in R^{N \times H}$.

3 Experiments

3.1 Datasets and Baselines

Datasets. To comprehensively validate the efficacy of our proposed model. We employ several datasets of realistic scenarios, including Weather, ILI, and Electricity transformer temperature (ETTh1, ETTh2, ETTm1 and ETTm2). Weather data record 21 meteorological indicators. The ILI dataset contains weekly data on influenza-like illness patients, comprising seven indicators. ETT data contains 7 features, including oil temperature and power load features.
Baselines. We conducted a comparative analysis of PatchMamba with seven contemporary models, including one Mamba-based model (S-Mamba[17]), four Transformer-based models (iTransformer[18], PatchTST[19], Crossformer[20] and FEDformer[21]), one linear-based model (DLinear[22]), and one convolutional network-based model (TimesNet[23]).

3.2 Experimental Settings

PatchMamba is implemented based on Pytorch and trained on a 12 GB Nvidia RTX 3080Ti GPU. In the ILI dataset, all models were set to utilize the same historical lookback window size $L=36$, the forecast length designated as $T\epsilon\{24,36,48,60\}$. For all other datasets, the historical lookback window size was set to $L=96$, and the forecast length was set to $T\epsilon\{96,192,336,720\}$. Loss function is MSE, and optimization is performed by Adam. MSE and MAE are used as metrics for prediction evaluation. The training process employs early stopping mechanism and is limited to 60 epochs.

3.3 Results and Analysis

As shown in **Table 1** and **Table 2**, among the 60 forecasting cases of 6 datasets, PatchMamba achieved the best results in 45 cases. Compared with the temporal Mamba model S-Mamba, PatchMamba achieves error reductions of 8.52% in MSE and 5.32% in MAE. Furthermore, compared to the prevailing Transformer-based model PatchTST, PatchMamba demonstrates average error reductions of 5.27% and 2.08% on the MSE and MAE metrics, respectively. In terms of predictive modeling, our models are differentiated according to the characteristics of the multivariate time series data. For the Weather and ILI datasets, a channel dependence strategy is adopted, and the CD-Patch Attention mechanism captures the dynamic interactions between multivariate variables at a fine-grained level. And combined with Bi-Mamba to enhance the information retention capability. For the ETT series dataset, a channel independence strategy is used, combining CI-Patch attention with Bi-Mamba. This effectively captures local fluctuation characteristics and global evolution patterns of each variable time series. This adaptive strategy can help improve model performance.

Table 1. Comparing PatchMamba with baseline.
The best results are in red and the second best are blue.

Models		PatchMamba (ours)		S-Mamba		ITransfomer		PatchTST		Crossformer	
metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.372	0.400	0.386	0.405	0.386	0.405	0.414	0.419	0.423	0.448
	192	0.422	0.428	0.443	0.437	0.441	0.436	0.460	0.445	0.471	0.474
	336	0.466	0.440	0.489	0.468	0.487	0.458	0.501	0.466	0.570	0.546
	720	0.471	0.459	0.502	0.489	0.503	0.491	0.500	0.488	0.653	0.621
	avg	0.433	0.432	0.455	0.450	0.454	0.448	0.469	0.455	0.529	0.522
ETTh2	96	0.294	0.346	0.296	0.348	0.297	0.349	0.302	0.348	0.745	0.584
	192	0.370	0.395	0.376	0.396	0.380	0.400	0.388	0.400	0.877	0.656
	336	0.380	0.411	0.424	0.431	0.428	0.432	0.426	0.433	1.043	0.731
	720	0.415	0.437	0.426	0.444	0.427	0.445	0.431	0.446	1.104	0.763
	avg	0.365	0.397	0.381	0.405	0.383	0.407	0.387	0.407	0.942	0.684
ETTm1	96	0.318	0.361	0.333	0.368	0.334	0.368	0.329	0.367	0.404	0.426
	192	0.362	0.382	0.376	0.390	0.377	0.391	0.367	0.385	0.450	0.451
	336	0.390	0.408	0.408	0.413	0.426	0.420	0.399	0.410	0.532	0.515
	720	0.451	0.442	0.475	0.448	0.491	0.459	0.454	0.439	0.666	0.589
	avg	0.380	0.398	0.398	0.405	0.407	0.410	0.387	0.400	0.513	0.495
ETTm2	96	0.178	0.264	0.179	0.263	0.180	0.264	0.175	0.259	0.287	0.366
	192	0.244	0.306	0.250	0.309	0.250	0.309	0.241	0.302	0.414	0.492
	336	0.305	0.346	0.312	0.349	0.311	0.348	0.305	0.343	0.597	0.542
	720	0.401	0.401	0.411	0.406	0.412	0.407	0.402	0.400	1.730	1.042
	avg	0.282	0.329	0.288	0.332	0.288	0.332	0.281	0.326	0.757	0.611
Weather	96	0.158	0.204	0.165	0.210	0.174	0.214	0.177	0.218	0.158	0.230
	192	0.207	0.250	0.214	0.252	0.221	0.254	0.225	0.259	0.206	0.277
	336	0.259	0.287	0.274	0.297	0.278	0.296	0.278	0.297	0.272	0.335
	720	0.342	0.345	0.350	0.345	0.358	0.347	0.354	0.348	0.398	0.418
	avg	0.242	0.272	0.251	0.276	0.258	0.278	0.259	0.281	0.259	0.315
ILI	24	2.264	0.922	2.866	1.123	2.472	0.994	2.149	0.887	4.484	1.458
	36	1.745	0.845	2.765	1.108	2.288	0.964	2.314	0.925	4.651	1.540
	48	1.845	0.865	2.760	1.110	2.227	0.951	2.231	0.921	4.901	1.567
	60	1.982	0.925	3.189	1.240	2.267	0.966	2.041	0.908	5.183	1.609
	avg	1.959	0.889	2.895	1.145	2.310	0.970	2.184	0.910	4.805	1.544
1st count		45		1		0		13		2	

Table 2. This table is a continuation of Table 1

Models		TimesNet		DLinear		FEDformer	
metric		MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.384	0.402	0.386	0.400	0.376	0.419
	192	0.436	0.429	0.437	0.432	0.420	0.448
	336	0.491	0.469	0.481	0.459	0.459	0.465
	720	0.521	0.500	0.519	0.516	0.506	0.507
	avg	0.458	0.450	0.456	0.452	0.440	0.460
ETTh2	96	0.340	0.374	0.333	0.387	0.358	0.397
	192	0.402	0.414	0.477	0.476	0.429	0.439
	336	0.452	0.452	0.594	0.541	0.496	0.487
	720	0.462	0.468	0.831	0.657	0.463	0.474
	avg	0.414	0.427	0.559	0.515	0.437	0.449
ETTm1	96	0.338	0.375	0.345	0.372	0.379	0.419
	192	0.374	0.387	0.380	0.389	0.426	0.441
	336	0.410	0.411	0.413	0.413	0.445	0.459
	720	0.478	0.450	0.474	0.453	0.543	0.490
	avg	0.400	0.406	0.403	0.407	0.448	0.452
ETTm2	96	0.187	0.267	0.193	0.292	0.203	0.287
	192	0.249	0.309	0.284	0.362	0.269	0.328
	336	0.321	0.351	0.369	0.427	0.325	0.366
	720	0.408	0.403	0.554	0.522	0.421	0.415
	avg	0.291	0.333	0.350	0.401	0.305	0.349
Weather	96	0.172	0.220	0.196	0.255	0.217	0.296
	192	0.219	0.261	0.237	0.296	0.276	0.336
	336	0.280	0.306	0.283	0.335	0.339	0.380
	720	0.365	0.359	0.345	0.381	0.403	0.428
	avg	0.259	0.287	0.265	0.317	0.309	0.360
ILI	24	2.317	0.934	4.794	1.658	2.882	1.179
	36	1.972	0.920	4.455	1.558	2.857	1.148
	48	2.238	1.982	4.120	1.476	2.703	1.113
	60	2.207	0.928	4.246	1.486	2.716	1.116
	avg	2.184	1.191	4.404	1.545	2.790	1.139
1^{st}count		0		1		2	

3.4 Ablation Study

Effect of Each Component. To evaluate the effectiveness of each component in Patch-Mamba, we constructed three different variants, each targeting a specific element. (1)w/o Patch Attention: This variant performs ablation studies by eliminating CI-Patch Attention under a channel independence strategy and CD-Patch Attention under a channel dependence strategy. (2)w/o Bi-Mamba: This variant eliminates the Bi-Mamba in the model and explores the role of the Mamba in the model. (3)w/o Residual: That removes two residual connection from the model. The results of the ablation experiments are shown in **Table 3**.

We observe the following key points: (1) PatchMamba achieves average error reductions of 3.53% in MSE and 1.71% in MAE compared to w/o Patch Attention. This indicates that our proposed Patch Attention mechanism has a better effect on local feature information extraction. (2) Removal of Bi-Mamba leads to significant performance degradation, with an average decrease of 9.36% in MSE and 7.63% in MAE. This highlights Bi-Mamba's critical role in capturing long-term information features in time series data. (3) w/o Residual removal of two residual connections in the model leads to performance degradation with an average decrease of 9.45% for MSE and 7.83% for MAE. It is demonstrated that residual connection in the model not only stabilizes the training process but also improves the performance of the model.

Table 3. Results of ablation study on ETTh2 and Weather dataset, the best results are in **bold**.

Models		PatchMamba		w/o Bi-Mamba		w/o Patch Attention		w/o Residual	
metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	96	**0.294**	**0.346**	0.315	0.361	0.296	**0.346**	0.324	0.368
	192	**0.370**	0.395	0.409	0.418	0.372	**0.394**	0.400	0.413
	336	**0.380**	**0.411**	0.407	0.429	0.400	0.420	0.425	0.439
	720	**0.415**	**0.437**	0.417	0.439	0.424	0.442	0.442	0.456
Weather	96	**0.158**	**0.204**	0.198	0.253	0.171	0.215	0.196	0.249
	192	**0.207**	**0.250**	0.243	0.287	0.217	0.255	0.235	0.279
	336	**0.259**	**0.287**	0.291	0.318	0.273	0.296	0.290	0.313
	720	**0.342**	**0.345**	0.362	0.362	0.351	0.347	0.362	0.363

Varying historical look back window Lengths. To verify the effectiveness of PatchMamba in extracting long-term time series information. We conducted experiments with different historical look-back window lengths on five datasets, with the forecast length set to *T=96.* As shown in **Fig. 4**, PatchMamba demonstrates a decrease in MSE and maintains stability as the historical lookback window increases. This indicates that PatchMamba can effectively learn correlation information from time series data with longer historical lookback windows, enhancing predictive capability.

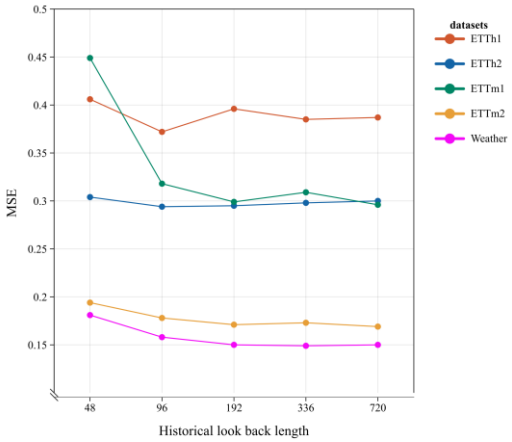
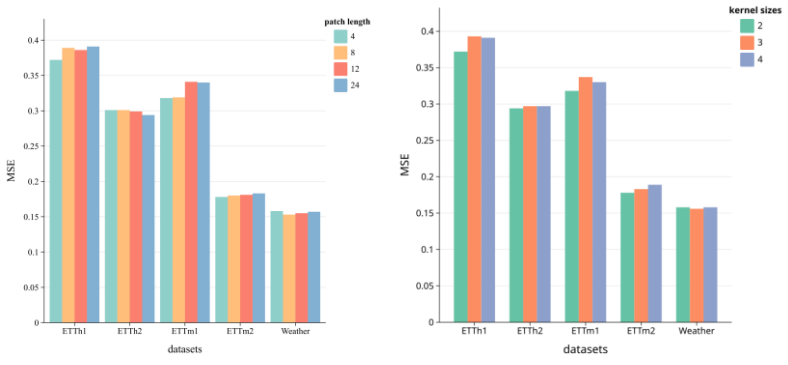


Fig. 4. MSE scores of PatchMamba with varying look-back window lengths on five datasets. The historical look back lengths are selected to be $\mathcal{L}= \{48,96,192,336,720\}$.

3.5 Sensitivity Analysis

To verify the sensitivity of PatchMamba to hyperparameters, we conducted experiments on patch length, and Mamba's Conv-1D kernel size parameters. We set patch length $\epsilon\{4,8,12,24\}$,and the experimental results are shown in **Fig. 5**(a). From the experimental results, we observe that different datasets respond differently to patch length. For example, in the ETTh1 dataset, due to its short-term dependencies of the transformer state, shorter segment lengths better capture its features.

The 1-D convolution in Mamba is used to capture the temporal correlation of the input sequence. We conducted experiments with kernel sizes of 2, 3, and 4. As shown in **Fig. 5**(b), the results indicate that the model achieves optimal performance with a kernel size of 2. Overall, our model demonstrates relative robustness to kernel size.



(a) Effect of patch length (b) Effect of Conv-1D kernel size

Fig. 5. Sensitivity experiments result

4 Conclusion

In this paper, we propose PatchMamba, a model comprising Patch Attention and Bi-Mamba. The model splits sequences into patches and dynamically selects a handling strategy based on multivariate time series data characteristics. Specifically, CI-Patch Attention focuses on the internal evolution of each univariate time series, while CD-Patch Attention captures inter-sequence dependencies. Bi-Mamba effectively extracts long-term dependencies. Experimental results demonstrate the superior performance of our model compared to SOTA models. We conducted comprehensive analytical experiments to investigate the impact of PatchMamba's components and their parameters on model performance. These findings confirm that PatchMamba enhances prediction accuracy in multivariate time series forecast tasks.

Acknowledgments. This research was supported by the National Natural Science Foundation of China (under Grant Nos. 72171004, 72301010), the Project of Cultivation for Young Top-notch Talents of Beijing Municipal Institutions (under Grant No. BPHR202203061).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Priesmann, J., Nolting, L., Kockel, C., and Praktiknjo, A.: Time series of useful energy consumption patterns for energy system modeling. Scientific Data. 8(1): p. 148 (2021)
2. Lippi, M., Bertini, M., and Frasconi, P.: Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. IEEE Transactions on Intelligent Transportation Systems. 14(2): p. 871-882 (2013)
3. Waqas, M., Humphries, U.W., and Hlaing, P.T.: Time series trend analysis and forecasting of climate variability using deep learning in Thailand. Results in Engineering. 24: p. 102997 (2024)
4. Elliott, G. and Timmermann, A.: Economic forecasting. Journal of Economic Literature. 46(1): p. 3-56 (2008)
5. Ariyo, A.A., Adewumi, A.O., and Ayo, C.K.: Stock price prediction using the ARIMA model. In 2014 UKSim-AMSS 16th international conference on computer modelling and simulation: IEEE (2014)
6. Fang, T. and Lahdelma, R.: Evaluation of a multiple linear regression model and SARIMA model in forecasting heat demand for district heating system. Applied energy. 179: p. 544-552 (2016)
7. Garcia, R.C., Contreras, J., Van Akkeren, M., and Garcia, J.B.C.: A GARCH forecasting model to predict day-ahead electricity prices. IEEE transactions on power systems. 20(2): p. 867-874 (2005)
8. Pai, P.-F., Lin, K.-P., Lin, C.-S., and Chang, P.-T.: Time series forecasting by a seasonal support vector regression model. Expert Systems with Applications. 37(6): p. 4261-4265 (2010)
9. Qiu, X., Zhang, L., Suganthan, P.N., and Amaratunga, G.A.: Oblique random forest ensemble via least square estimation for time series forecasting. Information Sciences. 420: p. 249-262 (2017)

10. Zhang, D., Xu, Y., Peng, Y., Du, C., Wang, N., Tang, M., Lu, L., and Liu, J.: An interpretable station delay prediction model based on graph community neural network and time-series fuzzy decision tree. IEEE Transactions on Fuzzy Systems. 31(2): p. 421-433 (2022)
11. Khaldi, R., El Afia, A., Chiheb, R., and Tabik, S.: What is the best RNN-cell structure to forecast each time series behavior? Expert Systems with Applications. 215: p. 119140 (2023)
12. Lai, G., Chang, W.-C., Yang, Y., and Liu, H.: Modeling long-and short-term temporal patterns with deep neural networks. In The 41st international ACM SIGIR conference on research & development in information retrieval (2018)
13. Sun, J. and Zhai, J.: MCNet: Multivariate long-term time series forecasting with local and global context modeling. Information Sciences. 676: p. 120864 (2024)
14. Su, L., Zuo, X., Li, R., Wang, X., Zhao, H., and Huang, B.: A systematic review for transformer-based long-term series forecasting. Artificial Intelligence Review. 58(3): p. 80 (2025)
15. Liang, A., Jiang, X., Sun, Y., and Lu, C.: Bi-Mamba4TS: Bidirectional Mamba for Time Series Forecasting. arXiv preprint arXiv:2404.15772 (2024)
16. Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J.: Reversible instance normalization for accurate time-series forecasting against distribution shift. In International Conference on Learning Representations (2021)
17. Wang, Z., Kong, F., Feng, S., Wang, M., Yang, X., Zhao, H., Wang, D., and Zhang, Y.: Is mamba effective for time series forecasting? Neurocomputing. 619: p. 129178 (2025)
18. Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M.: itransformer: Inverted transformers are effective for time series forecasting. arXiv preprint arXiv:2310.06625 (2023)
19. Nie, Y., H. Nguyen, N., Sinthong, P., and Kalagnanam, J.: A Time Series is Worth 64 Words: Long-term Forecasting with Transformers, in International Conference on Learning Representations (2023)
20. Zhang, Y. and Yan, J.: Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In The eleventh international conference on learning representations (2023)
21. Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R.: Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In International conference on machine learning: PMLR (2022)
22. Zeng, A., Chen, M., Zhang, L., and Xu, Q.: Are transformers effective for time series forecasting? In Proceedings of the AAAI conference on artificial intelligence (2023)
23. Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M.: TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis, in International Conference on Learning Representations (2023)