Bi-Mamba+: Bidirectional Mamba for Time Series Forecasting

Aobo Liang[1]   Xingguo Jiang[1]   Yan Sun[1*]   Xiaohou Shi[2]   Ke Li[3]

[1] Beijing University of Posts and Telecommunications, China
[2] China Telecom Research Institute, Beijing, China
[3] China Telecom Corporation Intelligent Cloud Network Operation Center, Beijing, China
{liangaobo, sunyan}@bupt.edu.cn, jxgaugusto@gmail.com,
{shixh6, lik24}@chinatelecom.cn

Abstract

Long-term time series forecasting (LTSF) provides longer insights into future trends and patterns. Over the past few years, deep learning models especially Transformers have achieved advanced performance in LTSF tasks. However, LTSF faces inherent challenges such as long-term dependencies capturing and sparse semantic characteristics. Recently, a new state space model (SSM) named Mamba is proposed. With the selective capability on input data and the hardware-aware parallel computing algorithm, Mamba has shown great potential in balancing predicting performance and computational efficiency compared to Transformers. To enhance Mamba's ability to preserve historical information in a longer range, we design a novel Mamba+ block by adding a forget gate inside Mamba to selectively combine the new features with the historical features in a complementary manner. Furthermore, we apply Mamba+ both forward and backward and propose Bi-Mamba+, aiming to promote the model's ability to capture interactions among time series elements. Additionally, multivariate time series data in different scenarios may exhibit varying emphasis on intra- or inter-series dependencies. Therefore, we propose a series-relation-aware decider that controls the utilization of channel-independent or channel-mixing tokenization strategy for specific datasets. Extensive experiments on 8 real-world datasets show that our model achieves better predictions compared with state-of-the-art methods. Our code is available at https://github.com/Leopold2333/Bi-Mamba+.

Introduction

Time series forecasting (TSF) is an indispensable part of many fields such as traffic flow prediction(Guo et al. 2019), energy management(Uremović et al. 2022), weather forecasting(Zhang et al. 2022a), finance(Sezer et al. 2020), etc. Among TSF tasks, the long-term time series forecasting (LTSF) task predicts the trend, periodicity and other key patterns of data with a longer future observations, enabling better long-term strategies development, resource allocation planning and risk management.

In the past few years, a large number of deep learning models have been developed for LTSF tasks. Among these efforts, Transformer-based models achieve great success. Transformers use self-attention mechanism to capture long-term dependencies of the time series. In addition, the vanilla Transformer implicitly models the inter-series dependencies through channel-mixing embeddings. However, the quadratic complexity of the self-attention mechanism consumes excessive computational resources, resulting in slow training and inference speeds. Although many works like Informer(Zhou et al. 2021) and Autoformer(Wu et al. 2021) try to propose some sparse attention to solve this issue in recent years, they face the challenges of balancing computational efficiency and predicting performance. Moreover, these models do not explicitly capture the inter-series dependencies, which may cause inadequate modeling of the relationships between different time series(Zhang et al. 2024).

Recently, state-space models (SSM) emerge as a promising architecture for sequence modeling(Gu et al. 2021a,b). Among them, a recent study, Mamba(Gu and Dao 2023), has achieved remarkable results in sequence processing tasks such as natural language processing(Lieber et al. 2024), DNA sequence learning, audio waveform processing(Gu and Dao 2023) and computer vision(Zhu et al. 2024). Benefiting from its design of selective scanning, Mamba shows superior performance for long sequence modeling, which makes it potentially suitable for the LTSF task. However, there are limited utilizations of SSM in LTSF currently, which may stem from the inherent challenges in time series analysis tasks as follows:

- **Long-term time series modeling**. LTSF deals with larger magnitudes of sequence elements and is often more affected by data non-stationarity, noise and outliers, making it more difficult to capture long-term dependencies(Liu et al. 2022). Furthermore, the semantic information density of time series data at time points is lower than other types of sequence data. Directly modeling point-wise tokens may introduce redundant noise(Cao et al. 2023). Works such as PatchTST(Nie et al. 2023) and Crossformer(Zhang and Yan 2023) emphasis capturing richer semantic information by dividing time series into patches for tokenization. Compared with point-wise input tokens, patch-wise tokens reduces the number of sequence elements and leads to lower computational complexity. A recent work iTransformer(Liu et al. 2023) uses a simple fully connected layer to map the whole sequence to hidden states. Although iTransformer gets state-of-the-art (SOTA) performance, it is coarse-grained and is not conducive to capturing fine-grained evolution-

ary patterns inside the time series. Therefore, we are motivated to model the time series in a patching manner.

- **Emphasis on intra- or inter-series dependencies**. Time series data may have multiple variables with complex correlations between each other. For example, the traffic flow of interconnected roads within a transportation network influence each other in different manners during different periods of time. However, capturing inter-series dependencies using channel-mixing strategies does not always produce SOTA performance. Works such as PatchTST and NHITS(Challu et al. 2023) get superior performance by capturing intra-series dependencies only. These models process the multivariate time series (MTS) data in a channel-independent way, where each univariate series is input to the model independently to enhance the robustness of the model. It can be inferred that different datasets shows different emphasis on intra- or inter-sequence dependencies. A recent study TimeMachine(Ahamed and Cheng 2024) proposes a unified structure for channel-independent and channel-mixing tokenization strategies. It aims to handle both intra-series-emphasis and inter-series-emphasis situations. However, the boundary for the selection of tokenization strategies is ambiguous and the statistical characteristics of datasets are overlooked.

To solve the above challenges, we design an improved Mamba block named Mamba+ through adding a forget gate in Mamba to selectively combine the new features with the historical features in a complementary manner, therefore preserving historical information in a longer range. Based on the Mamba+, we propose **Bi**directional **Mamba+** (Bi-Mamba+) to model the MTS data from both forward and backward, enhancing the model's robustness and ability to capture interactions between time series elements. In addition, to address the varying emphasis on intra-series evolutionary patterns and inter-series interactions, we design a **S**eries-**R**elation-**A**ware (SRA) decider inspired from Spearman coefficient correlation. The SRA decider measures the proportion of highly correlated series pairs in the MTS data to automatically choose channel-independent or channel-mixing tokenization strategies. Furthermore, we divide the input sequence into patches and generate patch-wise tokens based on the chosen tokenization strategy. The patch-wise tokens contain richer semantic information and encourage the model to learn the long-term dependencies of the time series in a finer granularity. The main contributions of this paper are summarized as follows:

- We propose Bi-Mamba+ for LTSF task. We design an improved Mamba+ block and model the MTS data from both forward and backward to provide more accurate predictions.
- We design an SRA decider based on the Spearman correlation coefficient to automatically choose channel-independent or channel-mixing tokenization strategies. Furthermore, we divides the time series into patches to capture long-term dependencies in a finer granularity.
- We conduct extensive experiments on 8 real-world datasets that are widely used. The results show that Bi-Mamba+ achieves superior performance with varying prediction lengths.

Related Work

Time Series Forecasting

TSF aims to predict future values based on historical observations(Lim and Zohren 2021). Recently, models based on deep neural network achieve great success in TSF. Among these deep learning models, Transformer-based models(Vaswani et al. 2017) get outstanding performance because of the self-attention mechanism. However, the quadratic complexity to the length of the sequence limits these models' application on long-term time series forecasting. Therefore, researches in recent years focus on balancing the computing efficiency and predicting performance of Transformers. Informer(Zhou et al. 2021) proposes a *ProbSparse* mechanism which selects top-k elements of the attention weight matrix to make distillation operation on self-attention. Autoformer(Wu et al. 2021) uses time series decomposition and proposes an *Auto-Correlation* mechanism inspired by the stochastic process theory. Pyraformer(Liu et al. 2021) introduces the pyramidal attention module to summarizes features at different resolutions and model the temporal dependencies of different ranges. FEDformer(Zhou et al. 2022) develops a frequency enhanced Transformer through frequency domain mapping. PatchTST(Nie et al. 2023) divides each univariate sequence into patches and uses patch-wise self-attention to model temporal dependencies. Crossformer(Zhang and Yan 2023) adopts a similar patching operation but additionally employs a Cross-Dimension attention to capture inter-series dependencies. While patching helps reduce the number of sequence elements to be processed and extract richer semantic information, the self-attention layers are only used on the simplified sequences. Therefore, the performance bottlenecks still occur when these models deal with longer sequences. To tackle this issue, iTransformer(Liu et al. 2023) inverts the attention layers to straightly model inter-series dependencies. However, the tokenization approach adopted by iTransformer is simply passing the whole sequence through a Multilayer Perceptron (MLP) layer, which overlooks the complex evolutionary patterns inside the time series. Overall, Transformer-based models still face the challenges in computational efficiency and predicting performance.

SSM-based models

Deep learning models for time series forecasting are initially based on Recurrent Neural Network (RNN) or Convolutional Neural Network (CNN). RNNs process the sequence elements step by step and maintain a hidden state which is updated with each input element. These models are relatively simple and have excellent inference speed. However, the calculated gradient must be passed through all cells one by one, which limits the training speed and leads to forgetting long-term information. CNNs use convolutional kernel to emphasis local information. These models benefit from parallel computing and have faster training speed.
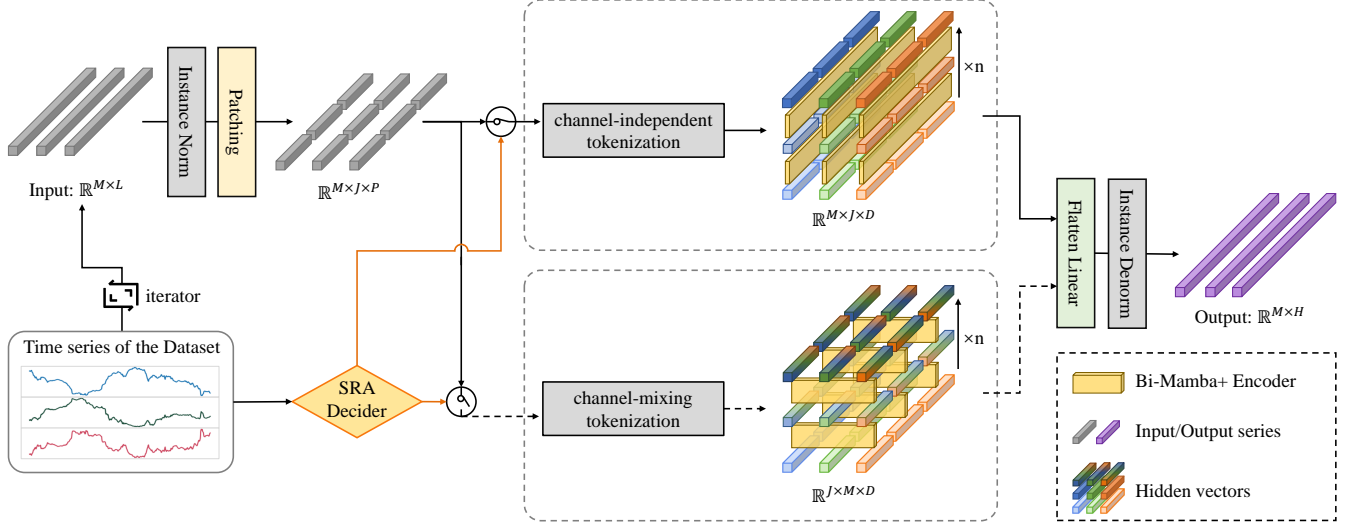
Figure 1: The architecture of Bi-Mamba+. The input time series is first divided into patches and then pass through a channel-independent or channel-mixing embedding layer based on the data characteristics of specific datasets. The embeddings are then fed into multiple Bi-Mamba Encoders and get the final output through an MLP projector.

However, the convolutional calculating process limits the inference speed and overlook the long-term global information. To solve the issues of these two models, a new model structure is proposed, namely State Space Models (SSM). SSM(Gu et al. 2021a,b) is inspired by the continious system. It is trained in parallel like CNN and inferences fastly like RNN. Some previous works have attempted to use SSM in TSF. SSDNet(Lin et al. 2021) combines the Transformer architecture with SSM to provide probabilistic and interpretable forecasts. SPACETIME(Gu et al. 2021b) proposes a new SSM parameterization based on the companion matrix to enhance the expressivity of the model and introduces a "closed-loop" variation of the companion SSM for long-term time series forecasting.

Recently, a new SSM-based model, Mamba(Gu and Dao 2023), is proposed. It introduces parameterized matrices and a hardware-aware parallel computing algorithm to SSM and achieves superior performance on language modeling, DNA sequence and audio waveform processing tasks. Several Mamba-based derivative models have been developed and used for computer vision tasks(Zhu et al. 2024; Ma et al. 2024) and time series tasks. S-Mamba(Wang et al. 2024) explores to use Mamba to capture inter-series dependencies of MTS data. It embeds each univariate time series like iTransformer and feeds the embeddings into Mamba blocks to model the relationships of different time series. However, the tokenization approach may overlook the complex evolutionary patterns inside the time series. MambaMixer(Behrouz et al. 2024) adjusts the Mamba block to bidirectional and uses two improved blocks to capture inter- and intra-series dependencies simultaneously. However, the gating branch is used to filter new features of both forward and backward directions, which may cause challenges for extracting new features. TimeMachine(Ahamed and Cheng 2024) proposes a multi-scale quadruple-Mamba architec-

ture to unify the handling of channel-mixing and channel-independence situations. However, the channel-mixing and channel-independent strategies are chosen simply based on the length of historical observations and variable number of different datasets. Therefore, the characteristics of the MTS data are not fully considered.

Methodology

Preliminaries

Long-term multivariate time series forecasting. We consider the Long-term multivariate time series forecasting task as follows: given a multivariate time series $\mathbf{X}_{in} = [x_1, x_2, ..., x_L] \in \mathbb{R}^{L \times M}$, we predict the future values $\mathbf{X}_{out} = [x_{L+1}, x_{L+2}, ..., x_{L+H}] \in \mathbb{R}^{H \times M}$, where L is the length of historical look-back window, M represents the feature dimension, and H is the prediction horizons.

State Space Models. SSM is inspired by the continious system, using first-order differential equations to map input function $x(t)$ to output function $y(t)$ through hidden state $h(t)$, defined as follows:

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t), \quad y(t) = \mathbf{C}h(t) \qquad (1)$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$, $\mathbf{B} \in \mathbb{R}^{D \times N}$ and $\mathbf{C} \in \mathbb{R}^{N \times D}$. The variable N and D refer to the state expansion factor and dimension factor respectively. The continuous parameters $\mathbf{A}, \mathbf{B}$ can be discretized to $\overline{\mathbf{A}}, \overline{\mathbf{B}}$ by zero-order holding and time sampling at intervals of Δ, defined as follows:

$$\begin{aligned} \overline{\mathbf{A}} &= \exp{(\Delta\mathbf{A})}, \\ \overline{\mathbf{B}} &= (\Delta\mathbf{A})^{-1}(\exp{(\Delta\mathbf{A})} - \mathbf{I}) \cdot \Delta\mathbf{B}. \end{aligned} \qquad (2)$$

The formula of discretized SSM can then be written as:

$$h_k = \overline{\mathbf{A}}h_{k-1} + \overline{\mathbf{B}}x_k, \quad y_k = \mathbf{C}h_k \qquad (3)$$

The discretized SSM(Gu et al. 2021a) can be trained in parallel in a convolutional operation way and make efficiently inference in a recurrent neural network manner. By introducing HIPPO Matrix(Gu et al. 2020) to the initialization of matrix $\mathbf{A}$, a variant of SSM, namely the structured state space model (S4)(Gu et al. 2021b), makes improvement on the ability to model long-term dependencies.

Mamba(Gu and Dao 2023) parameterizes the matrices $\mathbf{B}, \mathbf{C}$ and Δ in a data-driven manner, introducing a selection mechanism into S4 model. In addition, Mamba uses a novel hardware-aware parallel computing algorithm to ensure the efficient training of the model. With linear computational complexity and outstanding capabilities in modeling long-term dependencies, Mamba shows great potential in time series forecasting tasks and is expected to be a competitor to Transformer-based models.

Overview

The overall architecture of Bi-Mamba+ is shown in Fig. 1. We first calculate the tokenization strategy indicator through the SRA decider. we then divide the input series into patches and generate patch-wise tokens based on the tokenization strategy indicator. The obtained tokens are fed into multiple Bi-Mamba+ encoders. We adopt a flatten head and a linear projector to get the final output.

Instance Normalization

Time series in the real world usually conform to the characteristics of non-stationary sequences(Du et al. 2021; Long et al. 2022). The statistical properties of time series data usually change over time, resulting in a changing data distribution. We use RevIN (Kim et al. 2022) to eliminate the non-stationary statistics in the input sequence. The RevIn module normalizes the input batch data and denormalizs the output of the model. The instance normalization process corresponds to the **Instance Norm** and **Instance Denorm** modules in Fig. 1.

Token Generalization

SRA Decider. Recent studies(Nie et al. 2023; Zhou et al. 2022) show that both channel-independent and channel-mixing strategies can achieve SOTA accuracy in specific tasks. Typically, models using the channel-independent strategy can achieve better performance on datasets that have relatively few variables, while models using the channel-mixing strategy are more suitable for datasets with more variables. This can be viewed as a balance between the emphasis on intra-series dependencies and inter-series dependencies of MTS data.

Therefore, we design a SRA decider to automatically control the tokenization process of the model. The workflow of the dicision maker is shown in Alg.1. For a specific dataset, we extract the training set data $T = \{t^1, t^2, ..., t^M\}$ and calculate the Spearman correlation coefficients of different series t^i and t^j, denoted as $\rho_{i,j}$, where i and j are the indexes of the series ranging from 1 to M and $i \neq j$. We then use threshold λ and 0 to filter out series pairs with positive correlation. Finally, we count the maximum number of relevant series $\rho_{\max}^\lambda$ and $\rho_{\max}^0$ in the training set

---

Algorithm 1: SRA Decider

Input: Training set $T = \{t^1, t^2, ..., t^M\}$ of MTS data, each series has n observations
Output: Tokenization strategy $ts \in \{0, 1\}$
1: **for** $i = 1$ to M **do**
2:     $\rho_{i,i} = 0$
3:     **for** $j = i + 1$ to M **do**
4:         $\rho_{i,j} = \rho_{j,i} = 1 - \frac{6\sum_{k=0}^{n}\left(\mathrm{Rank}(t_k^i) - \mathrm{Rank}(t_k^j)\right)^2}{n(n^2 - 1)}$
5:     **end for**
6: **end for**
7: initialize list K^λ and K^0 of length M with all elements set to 0
8: **for** $i = 1$ to M **do**
9:     **for** $j = 1$ to M **do**
10:       **if** $\rho_{i,j} \geq \lambda$ **then**
11:         $K_i^\lambda = K_i^\lambda + 1$
12:       **else if** $\rho_{i,j} \geq 0$ **then**
13:         $K_i^0 = K_i^0 + 1$
14:       **end if**
15:     **end for**
16: **end for**
17: $\rho_{\max}^\lambda = \mathrm{argMax}(K^\lambda)$, $\rho_{\max}^0 = \mathrm{argMax}(K^0)$
18: $r = \rho_{\max}^\lambda / \rho_{\max}^0$
19: **if** $r \geq 1 - \lambda$ **then**
20:     **return** $ts = 1$ for channel-mixing tokenization
21: **else**
22:     **return** $ts = 0$ for channel-independent tokenization
23: **end if**

---

and calculate the relation ratio $r = \rho_{\max}^\lambda / \rho_{\max}^0$. We adopt channel-mixing strategy to generate sequence tokens for datasets with $r \geq 1 - \lambda$, otherwise, we adopt channel-independent strategy. Spearman coefficient is a nonparametric statistical indicator for evaluating the monotonic relationship between two sequences. It calculates the correlation based on the rank of the sequence elements, making it more suitable for non-linear relationships and more robust to outliers. Given two sequences t^i and t^j, it first sorts the sequence and then define the Spearman coefficient as $\rho_{i,j} = 1 - \frac{6\sum_{k=0}^{n}\left(\mathrm{Rank}(t_k^i) - \mathrm{Rank}(t_k^j)\right)^2}{n(n^2 - 1)}$, where n is the number of observation samples of each sequence and $\mathrm{Rank}(t_k^i)$ represents the rank level of the k-th element in the specific time series t^i.

Tokenization Process. We generalize patch-wise tokens to emphasize capturing local evolutionary patterns of the time series. Specifically, each channel univariate sequence $x_{1:L}^i$ is first divided into patches $p^i \in \mathbb{R}^{J \times P}$ where J is the total number of patches and P is the length of each patch. We denote S as the stride among patches and $J = \lceil \frac{L-P}{S} + 1 \rceil$. For channel-independent strategy, each univariate channel is concatenated to the tokens $\mathbb{E}_{ind} \in \mathbb{R}^{M \times J \times D}$, where D is the hidden state dimension. For channel-mixing strategy, we first group patches with the same index of different series and pass each group through the tokenization layer. The obtained tokens can be represented as $\mathbb{E}_{mix} \in$

Algorithm 2: Implementation of Mamba+ block

Input: Patch-wise token sequence $\mathbb{E}_x : (B, W, D)$
Output: Output features $\mathbb{E}_y : (B, W, E)$
1: $\text{x} : (B, W, E) \leftarrow \text{Linear}^{\mathbf{x}}(\mathbb{E}_x)$
2: $\text{z} : (B, W, E) \leftarrow \text{Linear}^{\mathbf{z}}(\mathbb{E}_x)$
3: $\text{x}' : (B, W, E) \leftarrow \text{SiLU}(\text{Conv1D}(\text{x}))$
4: $\mathbf{A} : (E, N) \leftarrow \text{Parameter}^{\mathbf{A}}$
5: $\mathbf{B} : (B, W, N) \leftarrow \text{Linear}^{\mathbf{B}}(\text{x}')$
6: $\mathbf{C} : (B, W, N) \leftarrow \text{Linear}^{\mathbf{C}}(\text{x}')$
7: $\Delta : (B, W, E) \leftarrow \log(1 + \exp(\text{Linear}^{\Delta}(\text{x}')) + \text{Parameter}^{\Delta})$
8: $\overline{\mathbf{A}}, \overline{\mathbf{B}} : (B, W, E, N) \leftarrow \text{discretize}(\Delta, \mathbf{A}, \mathbf{B})$
9: $\text{y} : (B, W, E) \leftarrow \text{SSM}(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C})(\text{x}')$
10: $\text{y}' : (B, W, E) \leftarrow \text{y} \otimes \text{SiLU}(\text{z}) + \text{x}' \otimes (1 - \sigma(\text{z}))$
11: $\mathbb{E}_y : (B, W, D) \leftarrow \text{Linear}^{\text{y}'}(\text{y}')$
12: **return** $\mathbb{E}_y$

$\mathbb{R}^{J \times M \times D}$.

Mamba+ Block

An original Mamba block uses two branches to process the input features. One of the branches, denoted as b_1, passes the input features through a 1-D convolutional module and a SSM block. The other branch, denoted as b_2, feeds the input features into a SiLU activation function to serve as a gate. Although the HIPPO matrix(Gu et al. 2021b) embedded in the SSM block can retain a fairly long-term historical information, the obtained result is filtered directly through the gate of another branch, resulting in the tendency to prioritize proximal information(Wang et al. 2024). Therefore, we propose an improved Mamba+ block specifically designed for LTSF. Particularly, we add a forget gate calculated by $gate_f = 1 - gate_{b_2}$, where $gate_f$ represents the forget gate and $gate_{b_2}$ is the result of sigmoid function in b_2. The output of the 1-D convolutional module x' is multiplied with $gate_f$ and then added to the filtered result of SSM. $gate_f$ and $gate_{b_2}$ selectively combine the added new features with the forgotten historical features in a complementary manner. Compared to directly filtering the obtained new features, Mamba+ emphasis preserving historical information in a longer range. The detailed flow of Mamba+ is shown in Alg.2. The reprogrammed forward and backward propagation process of the hardware-aware parallel computing algorithm is shown in Appendix.

Bidirectional Mamba+ Encoder

An original Mamba block is designed to process 1-D sequence on one direction. Considering the rich evolutionary patterns of time series in different directions as well as the complexity of inter-series dependencies, we design a bidirectional Mamba+ structure to comprehensively model the MTS data.

A Bi-Mamba+ encoder takes $\mathbb{E}_x^{(l)} \in \mathbb{R}^{B \times W \times D}$ as input, where l refers to the encoder layer, B and W corresponds to M or J depending on the tokenization strategy. Specifically,

Algorithm 3: Implementation of Bi-Mamba+ Encoder

Input: Patch-wise token sequence $\mathbb{E}_x^{(l)} : (B, W, D)$
Output: Output features for the next encoder layer $\mathbb{E}_x^{(l+1)} : (B, W, D)$
1: **for** dir in $\{forward, backward\}$ **do**
2:      **if** $dir = backward$ **then**
3:          $\mathbb{E}_{x,dir}^{(l)} : (B, W, D) \leftarrow \text{Flip}(\mathbb{E}_x^{(l)}, \dim = 1)$
4:      **else**
5:          $\mathbb{E}_{x,dir}^{(l)} : (B, W, D) \leftarrow \mathbb{E}_x^{(l)}$
6:      **end if**
7:      $\tilde{\mathbb{E}}_{y,dir}^{(l)} : (B, W, D) \leftarrow \text{TSMamba}(\mathbb{E}_{x,dir}^{(l)})$
8:      $\tilde{\mathbb{E}}_{y,dir}^{(l)} : (B, W, D) \leftarrow \text{Add\&Norm}(\tilde{\mathbb{E}}_{y,dir}^{(l)}, \mathbb{E}_{x,dir}^{(l)})$
9: **end for**
10: $\tilde{\mathbb{E}}_y^{(l+1)} : (B, W, D) \leftarrow \mathbb{E}_{y,forward}^{(l)} + \mathbb{E}_{y,backward}^{(l)}$
11: $\tilde{\mathbb{E}}_x^{(l+1)} : (B, W, D) \leftarrow \text{FeedForward}(\tilde{\mathbb{E}}_y^{(l+1)})$
12: $\mathbb{E}_x^{(l+1)} : (B, W, D) \leftarrow \text{Add\&Norm}(\tilde{\mathbb{E}}_x^{(l+1)}, \tilde{\mathbb{E}}_y^{(l+1)})$
13: **return** $\mathbb{E}_x^{(l+1)}$

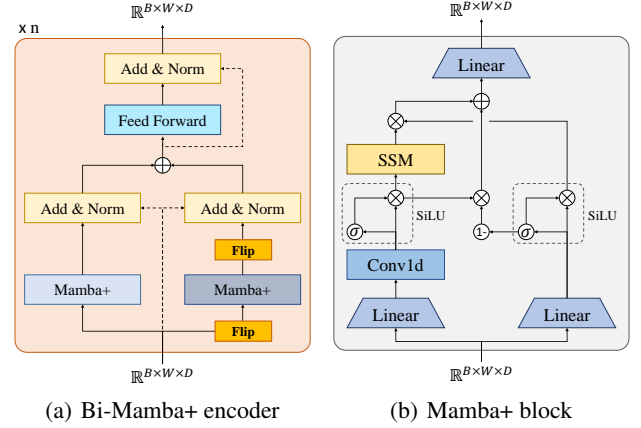(a) Bi-Mamba+ encoder   (b) Mamba+ block

Figure 2: The architecture of (a) Bi-Mamba+ encoder and (b) Mamba+ block.

if $ts = 1$, $\mathbb{E}_x^{(l)} \in \mathbb{R}^{J \times M \times D}$ and $\mathbb{E}_x^{(0)} = \mathbb{E}_{mix}$, otherwise, $\mathbb{E}_x^{(l)} \in \mathbb{R}^{M \times J \times D}$ and $\mathbb{E}_x^{(0)} = \mathbb{E}_{ind}$. There are two Mamba+ blocks in one Bi-Mamba+ encoder to model the input sequence from the forward and backward directions respectively, as shown in Fig 2(a). We denote the input of these two directions as $\mathbb{E}_{x,dir}^{(l)}$ where $dir \in \{forward, backward\}$. We get $\mathbb{E}_x^{(l+1)} = \sum_{dir}^{\{forward, backward\}} \mathcal{F}(\mathbb{E}_{y,dir}^{(l)}, \mathbb{E}_{x,dir}^{(l)})$ as the input of the next Bi-Mamba+ encoder layer. The function $\mathcal{F}$ here represents the combination of residual connection and feed forward neural network layers like those in a Transformer encoder. $\mathbb{E}_{y,forward}^{(l)}$ and $\mathbb{E}_{y,backward}^{(l)}$ are the outputs of the two Mamba+ blocks with two modeling directions respectively. The detailed implementation is shown in Alg.3.

Loss Function

We use Mean Squared Error (MSE) Loss, which measures the average squared difference between the predicted values and the ground truth. The definition of the Loss function is as follows:

$$\mathcal{L}(Y, \hat{Y}) = \frac{1}{|Y|} \sum_{i=1}^{|Y|} (y_{(i)} - \hat{y}_{(i)})^2 \qquad (4)$$

where $\hat{Y}$ is the predicted values and Y is the ground truth.

Experiments

Datasets

We choose 8 real-world datasets to evaluate our proposed model: Weather, Traffic, Electricity, Solar and 4 ETT datasets(ETTh1, ETTh2, ETTm1, ETTm2). These datasets are widely used, covering multiple fields including weather, transportation and energy management. The statistics of the datasets are shown in Table 1.

Table 1: Statistics of all datasets.

Dataset	Variables	Frequency	Length
Weather	21	10 min	52696
Electricity	321	1 hour	26304
Traffic	862	1 hour	17544
ETTh1	7	1 hour	17420
ETTh2	7	1 hour	17420
ETTm1	7	15 min	69680
ETTm2	7	15 min	69680
Solar	137	10min	52179

Table 2: The information of the experimental platform.

Component	Description
System	Ubuntu 20.04 LTS Linux
CPU model	Intel(R) Xeon(R) CPU E5-2686 v4
Memory Size	64GB
GPU model	NVIDIA Tesla V100 32GB ($\times 2$)
CUDA Version	11.8
Python Version	3.10.14
Pytorch Version	2.1.1

Baseline Models

We compare our model with 4 Transformer-based models, 1 MLP-based model, 1 CNN-based model, 1 RNN-based model, 1 GNN-based model and 1 SSM-based model. These models cover the mainstream design for time series forecasting tasks and achieve SOTA performance in their respective model categories. The detailed descriptions are as follows:

- **Autoformer** (Wu et al. 2021) uses a series decomposition technique with Auto-Correlation mechanism to capture cross-time dependency for long-term time series forecasting.

- **PatchTST** (Nie et al. 2023) adopts patching and channel independent techniques, making semantic extraction of time series from a single time step to multiple time steps.

- **Crossformer** (Zhang and Yan 2023) is aware of the fact that segmenting subsequences in LSTF is beneficial. It uses the same patching strategy as PatchTST. To additionally capture inter-series dependencies, it designs to use another attention layer working with a routing mechanism to reduce complexity.

- **iTransformer** (Liu et al. 2023) inverts the modeling method of Transformer by using self-attention to capture inter-series dependencies.

- **DLinear** (Zeng et al. 2023) decomposes time series into two different components, and generates a single Linear layer for each of them. Such a simple design has defeated all the complex transformer models proposed before it.

- **TimesNet** (Wu et al. 2022) extends the analysis of temporal variations into the 2-D space by transforming the 1-D time series into a set of 2-D tensors based on multiple periods.

- **WITRAN** (Jia et al. 2024) designs a novel RNN structure that process the univariate input sequence in the 2-D space with a fixed scale.

- **CrossGNN** (Huang et al. 2024) is a model mainly based on GNN. It views time series in a multi-scale way. It uses GNN to capture both cross-scale and cross-series dependencies.

- **S-Mamba** (Wang et al. 2024) generates embeddings for each time series through a simple MLP layer and uses Mamba to extract inter-series dependencies.

Experimental Settings

We set the same historical look-back window size of $L = 96$ for all models on all datasets. The predicting length is set to $H \in \{96, 192, 336, 720\}$. We set $S = \frac{1}{2}P$ and use patch length $P = \frac{1}{4}L$ by default. Considering that different models may have different sensitivity of granularity of local information, we maintain the patching settings consistent with the original papers for PatchTST and Crossformer. For the SRA decider, we set $\lambda = 0.6$ because generally when the Spearman coefficient correlation is larger than 0.6, it can be inferred that the two sequences have a strong positive correlation. More detailed evaluation of λ is shown in Fig. 3. Bi-Mamba+ splits a sequence into patches, which mainly retain local information but contains less global information compared to a whole sequence. Considering that datasets with more variables may contain more complex evolutionary patterns, for Bi-Mamba+, PatchTST and Crossformer that use patching technique, we set $D = 128$ for Weather, Traffic, Electricity, Solar and $D = 64$ for ETT datasets, while for S-Mamba and iTransformer that map the whole sequence to tokens, we set $D = 512$ for Weather, Traffic, Electricity, Solar and $D = 256$ for ETT datasets. As for parameters within Mamba+ block, we follow (Ahamed and Cheng 2024; Wang et al. 2024) and set convolutional kernel size $d\_conv = 2$ and hidden state expansion $expand = 1$ on all datasets. We set hidden dimension $d\_state = 16$ for Weather, Electricity

Table 3: Experimental results of long-term multivariate time series forecasting task on 8 real-world datasets. The best results are in **bold** and the second best results are underlined.

Models		Bi-Mamba+		S-Mamba		iTransformer		PatchTST		Crossformer		Autoformer		DLinear		TimesNet		CrossGNN		WITRAN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.159	0.205	0.166	0.210	0.174	0.214	0.178	0.219	0.158	0.230	0.266	0.336	0.196	0.235	0.172	0.220	0.186	0.237	0.178	0.223
	192	0.205	0.249	0.215	0.253	0.221	0.254	0.224	0.259	0.206	0.277	0.307	0.367	0.241	0.271	0.219	0.261	0.233	0.273	0.223	0.261
	336	0.264	0.291	0.276	0.298	0.278	0.298	0.292	0.306	0.272	0.335	0.359	0.395	0.292	0.306	0.280	0.306	0.289	0.312	0.288	0.309
	720	0.343	0.344	0.353	0.349	0.358	0.349	0.354	0.348	0.398	0.418	0.419	0.428	0.363	0.353	0.365	0.359	0.356	0.352	0.372	0.363
Traffic	96	0.375	0.258	0.381	0.261	0.395	0.268	0.457	0.295	0.522	0.290	0.613	0.388	0.647	0.384	0.593	0.321	0.676	0.407	1.037	0.441
	192	0.394	0.269	0.397	0.267	0.417	0.276	0.471	0.299	0.530	0.293	0.616	0.382	0.596	0.359	0.617	0.336	0.631	0.386	1.061	0.455
	336	0.406	0.274	0.423	0.276	0.433	0.283	0.482	0.304	0.558	0.305	0.616	0.382	0.601	0.361	0.629	0.336	0.640	0.387	1.095	0.470
	720	0.440	0.288	0.458	0.300	0.467	0.302	0.514	0.322	0.589	0.328	0.622	0.337	0.642	0.381	0.640	0.350	0.681	0.402	1.121	0.474
Electricity	96	0.140	0.238	0.142	0.238	0.148	0.240	0.174	0.259	0.219	0.314	0.201	0.317	0.206	0.288	0.168	0.272	0.217	0.304	0.237	0.335
	192	0.155	0.253	0.169	0.267	0.162	0.253	0.178	0.265	0.231	0.322	0.222	0.334	0.206	0.290	0.184	0.289	0.216	0.306	0.258	0.350
	336	0.170	0.269	0.178	0.275	0.178	0.269	0.196	0.282	0.246	0.337	0.231	0.338	0.220	0.305	0.198	0.300	0.232	0.321	0.273	0.362
	720	0.197	0.293	0.207	0.303	0.225	0.317	0.237	0.316	0.280	0.363	0.254	0.361	0.252	0.337	0.220	0.320	0.273	0.352	0.300	0.382
ETTh1	96	0.378	0.395	0.386	0.406	0.386	0.405	0.393	0.408	0.423	0.448	0.449	0.459	0.387	0.406	0.384	0.402	0.382	0.398	0.414	0.419
	192	0.427	0.428	0.448	0.444	0.441	0.436	0.445	0.434	0.471	0.474	0.500	0.482	0.439	0.435	0.436	0.429	0.427	0.425	0.464	0.448
	336	0.471	0.445	0.494	0.468	0.487	0.458	0.474	0.451	0.570	0.546	0.521	0.496	0.493	0.457	0.491	0.469	0.486	0.487	0.516	0.478
	720	0.470	0.457	0.493	0.488	0.503	0.491	0.480	0.471	0.653	0.621	0.514	0.512	0.490	0.478	0.521	0.500	0.482	0.477	0.538	0.509
ETTh2	96	0.291	0.342	0.298	0.349	0.297	0.349	0.302	0.348	0.745	0.584	0.346	0.388	0.305	0.352	0.340	0.374	0.302	0.349	0.325	0.364
	192	0.368	0.392	0.379	0.398	0.380	0.400	0.388	0.400	0.877	0.656	0.456	0.452	0.424	0.439	0.402	0.414	0.382	0.400	0.433	0.427
	336	0.407	0.424	0.417	0.432	0.428	0.432	0.426	0.433	1.043	0.731	0.482	0.486	0.456	0.473	0.452	0.452	0.421	0.439	0.471	0.457
	720	0.421	0.439	0.431	0.449	0.428	0.432	0.431	0.446	1.104	0.763	0.515	0.511	0.476	0.493	0.462	0.468	0.437	0.458	0.499	0.480
ETTm1	96	0.320	0.360	0.331	0.368	0.334	0.368	0.329	0.367	0.404	0.426	0.505	0.475	0.353	0.374	0.338	0.375	0.340	0.374	0.375	0.402
	192	0.361	0.383	0.371	0.387	0.377	0.391	0.367	0.385	0.450	0.451	0.553	0.496	0.389	0.391	0.374	0.387	0.377	0.390	0.427	0.434
	336	0.386	0.402	0.417	0.418	0.426	0.420	0.399	0.419	0.532	0.515	0.621	0.537	0.421	0.413	0.410	0.411	0.401	0.407	0.455	0.452
	720	0.445	0.437	0.471	0.448	0.491	0.459	0.454	0.439	0.666	0.589	0.671	0.561	0.484	0.448	0.478	0.450	0.453	0.442	0.527	0.488
ETTm2	96	0.176	0.263	0.179	0.263	0.180	0.264	0.175	0.259	0.287	0.366	0.255	0.339	0.182	0.264	0.187	0.267	0.177	0.261	0.191	0.272
	192	0.242	0.304	0.253	0.310	0.250	0.309	0.241	0.302	0.414	0.492	0.281	0.340	0.257	0.315	0.249	0.309	0.240	0.298	0.261	0.316
	336	0.304	0.344	0.312	0.348	0.311	0.348	0.305	0.343	0.597	0.542	0.339	0.372	0.318	0.353	0.321	0.351	0.305	0.345	0.330	0.358
	720	0.402	0.402	0.412	0.408	0.412	0.407	0.402	0.400	1.730	1.042	0.433	0.432	0.426	0.419	0.408	0.403	0.403	0.400	0.450	0.427
Solar	96	0.184	0.222	0.205	0.241	0.203	0.237	0.234	0.286	0.310	0.331	0.884	0.711	0.290	0.378	0.250	0.292	0.222	0.314	0.238	0.285
	192	0.225	0.254	0.237	0.270	0.233	0.261	0.267	0.310	0.734	0.725	0.834	0.692	0.320	0.398	0.296	0.318	0.240	0.318	0.282	0.302
	336	0.249	0.270	0.256	0.286	0.248	0.273	0.290	0.315	0.750	0.735	0.941	0.723	0.353	0.415	0.319	0.330	0.264	0.324	0.329	0.345
	720	0.250	0.273	0.260	0.288	0.249	0.275	0.289	0.317	0.769	0.765	0.882	0.717	0.356	0.413	0.338	0.337	0.260	0.321	0.326	0.348

and Traffic and $d\_state = 8$ for ETT datasets. We evaluate Bi-Mamba+ with encoder layer $l \in \{1, 2, 3\}$ and conduct grid search for learning rate on [5e-5, 1e-4, 2e-4, 5e-4, 1e-3, 2e-3, 5e-3] to find the best results. For the metrics to evaluate the performance, we utilize MSE and MAE. All models are optimized using the ADAM algorithm(Kingma and Ba 2014) and early stopping mechanism. The training process is limited to 40 epochs.

Experimental Environment

To exclude the influence of the model training environment, we conduct all experiments on the same device with 2 NVIDIA Tesla V100 (32G) GPUs. The detailed information of our experimental platform is shown in table 2.

Main Results

Table 3 shows the results of long-term multivariate forecasting. Our model achieves superior performance compared with other baselines on all datasets with different predicting lengths. Compared with the current SOTA Transformer model iTransformer, Bi-Mamba+ reduces the MSE, MAE errors by 4.72% and 2.60% on average. The improvement of MSE and MAE comes to 3.76% and 2.67% compared with S-Mamba, which is also an SSM-based model using Mamba. It is worth noting that Bi-Mamba+ adopts channel-mixing strategy on Weather, Traffic, Electricity and Solar and channel-independent strategy on 4 ETT datasets. For channel-mixing strategy, the patching approach used by Bi-Mamba+ helps the model capture inter-series dependencies at a finer granularity compared with S-Mamba and iTransformer that transform the whole sequence into a token. For channel-independent strategy, the patch-wise tokens fed into Mamba blocks contain more semantic information, enhancing the model's ability for capturing the inner evolutionary process of each univariate time series. Compared with PatchTST that adopt the same patching technique, our proposed Mamba+ block achieves more accurate predictions than self-attention.

Ablation Study

We verify the effectiveness of the SRA decider, the bidirectional module and the residual connection. Before we conduct the ablation experiments, we first export the results of the SRA decider to view the strategy choosing for different datasets. For Spearman correlation coefficient ρ, we typically use $\lambda > 0.6$ to evaluate whether two variables have positive correlation strong enough to influence each other. The decision results under varying λ are shown in Fig. 3. For ETT datasets that have few variables, the model tends to adopt channel-independent strategy to emphasis intra-series dependencies, while for Weather, Traffic and Electricity that have large amount of variables, the model tends to adopt channel-mixing strategy to emphasis inter-series dependencies. We remove the SRA decider to

Table 4: Ablation studies of (a) w/o SRA-I which use channel-independent strategy only, (b) w/o SRA-M which use channel-mixing strategy only, (c) w/o Bi which use forward direction Mamba block only, (d) w/o Residual that removes the residual connection, (e) S-Mamba and (f) PatchTST used for the benchmark models. The best results are in **bold**.

Models	w/o SRA-I		w/o SRA-M		w/o Bi		w/o Residual		w/o M & w/ A		S-Mamba		PatchTST	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	0.252	0.277	**0.243**	**0.272**	0.251	0.281	0.252	0.283	0.257	0.281	0.253	0.278	0.259	0.281
Traffic	0.453	0.287	**0.404**	**0.272**	0.435	0.289	0.439	0.306	0.417	0.281	0.415	0.276	0.481	0.305
Electricity	0.188	0.274	**0.166**	**0.263**	0.176	0.271	0.179	0.279	0.174	0.269	0.174	0.271	0.196	0.281
ETTh1	**0.437**	**0.431**	0.447	0.437	0.443	0.437	0.456	0.446	0.446	0.445	0.465	0.457	0.448	0.441
ETTh2	**0.372**	**0.399**	0.372	0.402	0.376	0.406	0.382	0.407	0.374	0.402	0.381	0.407	0.387	0.407
ETTm1	**0.378**	**0.396**	0.389	0.404	0.386	0.401	0.386	0.401	0.381	0.396	0.398	0.405	0.387	0.400
ETTm2	**0.281**	0.328	0.284	0.332	0.288	0.335	0.291	0.339	0.287	0.334	0.289	0.333	**0.281**	**0.326**
Solar	0.236	0.263	**0.227**	**0.255**	0.235	0.262	0.232	0.263	0.241	0.276	0.239	0.271	0.270	0.307

(a) Weather (b) Traffic (c) Electricity (d) Solar

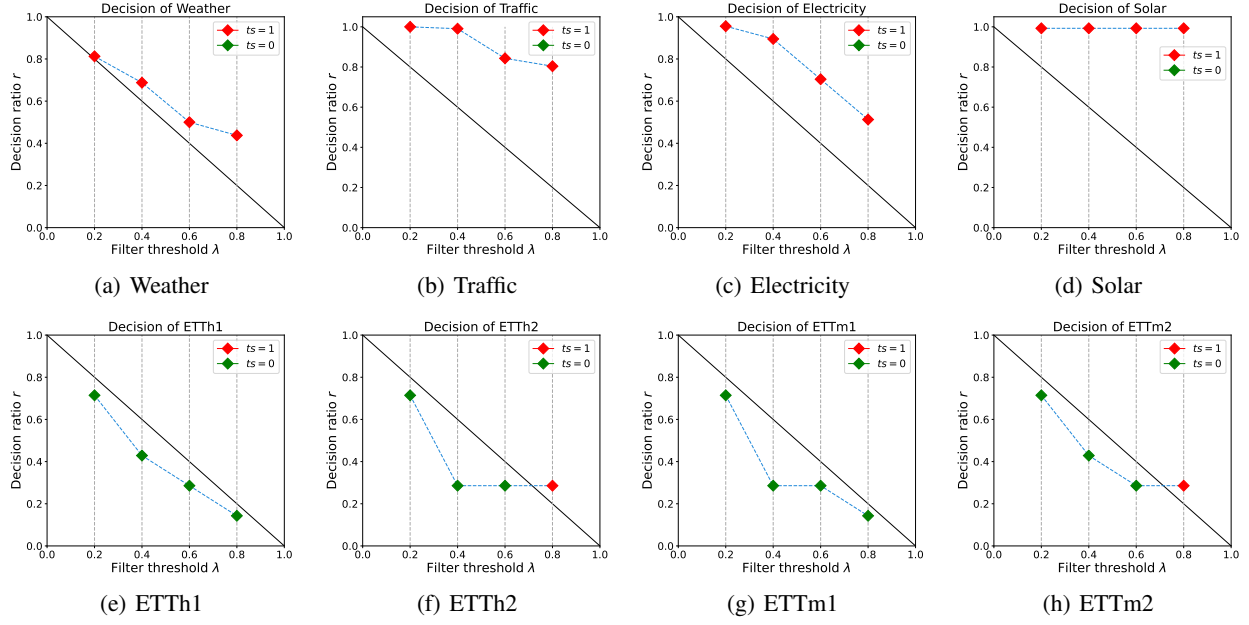(e) ETTh1 (f) ETTh2 (g) ETTm1 (h) ETTm2

Figure 3: The tokenization strategy indicator ts of different datasets with varying settings of filter threshold λ.

form two specific Bi-Mamba+ models: (a) w/o SRA-I which only uses channel-independent tokenization strategy and (b) w/o SRA-M which only uses channel-mixing tokenization strategy. For the bidirectional Mamba+ encoder, we remove the backward direction Mamba+ to evaluate the effectiveness of bidirectional structure. This setting forms the following model: (c) w/o Bi. Besides, we remove the **Add&Norm** layer to evaluate the effect of residual connection: (d) w/o Residual. We also replace Mamba+ blocks with the dot-product Attention: (e) w/o M & w/ A.

We calculate the average MSE and MAE scores as shown in table 4. Without the SRA decider, the model utilizes either channel-mixing or channel independent tokenization strategies. The MSE and MAE increase by 0.95% and 0.77% for w/o SRA-M and 3.56% and 1.45% for w/o SRA-I respectively. Therefore, the proposed SRA decider makes the optimal choice of channel-independent and channel-mixing

strategies and utilizing different tokenization strategies on different datasets produces better predicting performance.

When the backward Mamba block is removed, the predicting performance of MSE and MAE is deteriorated 3.25% and 2.44% respectively, which indicates that the design of bidirectional module enhances the model to get more comprehensive understanding of the MTS data.

The residual network enhances the expressive ability and feature utilization of the model by reusing early features. It can also reduce overfitting in the training process. The MSE and MAE of w/o Residual decrease by 4.37% and 4.05% respectively, indicating the effectiveness of the residual connection module. Full results of Ablation Study can be found in table 5 in Appendix.

It is worth noting that replacing Mamba+ with Attention causes performance degradation, indicating that Mamba may be a better choice for time series forecasting.
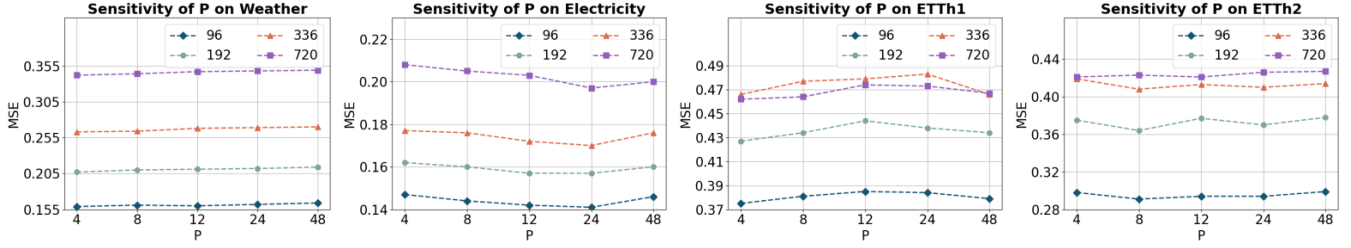
Figure 4: Results of varying patch length P on Weather, Electricity, ETTh1 and ETTh2.

Hyper-parameter Sensitivity Analysis

To verify whether Bi-Mamba+ is sensitive to the hyper-parameters, we conduct experiments on (a) Spearman coefficient filter λ, (b) length of patches, (c) dropout (d) Conv-1D kernel size of Mamba+ and (e) state dimension size. For each setting, we freeze other parameters that are not currently fine-tuned and repeat the experiment 5 times with 40 epochs (with early-stop control). We record the average MSE of $H \in \{96, 192, 336, 720\}$. Overall, different parameter settings do not cause significant changes in the results, which also shows the robustness of our proposed model.
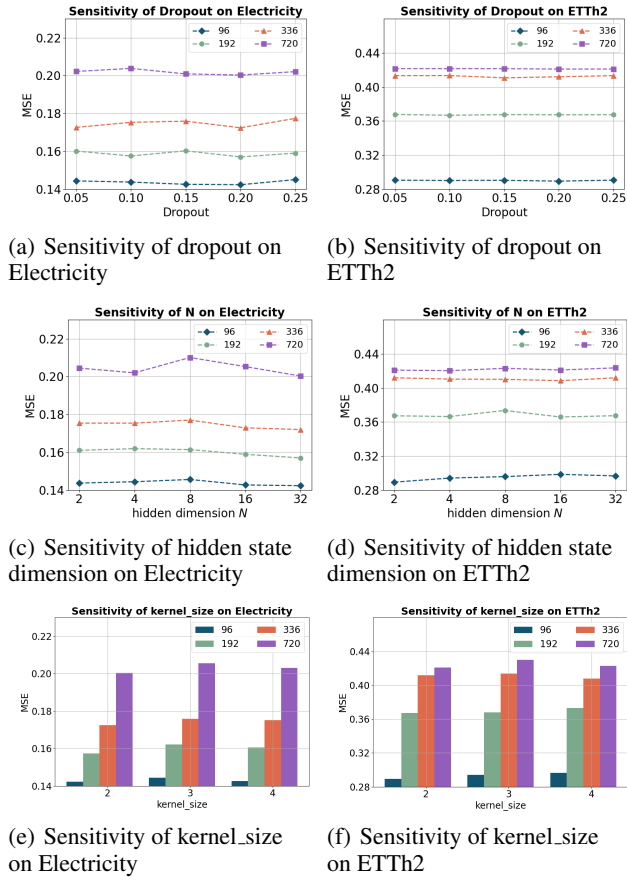

(a) Sensitivity of dropout on Electricity

(b) Sensitivity of dropout on ETTh2

(c) Sensitivity of hidden state dimension on Electricity

(d) Sensitivity of hidden state dimension on ETTh2

(e) Sensitivity of kernel_size on Electricity

(f) Sensitivity of kernel_size on ETTh2

Figure 5: Results of varying dropout, hidden dimension N and kernel size values on Electricity and ETTh2.

Spearman coefficient filter λ. The value λ is used for filter out variable pairs that have positive correlations. A larger λ corresponds to stronger positive correlations between two sequences. We set $\lambda \in \{0.2, 0.4, 0.6, 0.8\}$ and calculate the decision ratio r and decision indicator ts. The results are shown in Fig. 3. Overall, different λ does not frequently cause the change of tokenization strategy. For datasets that have large number of variables, such as Traffic and Electricity, the decision indicator ts prefers to be equal to 1, while for datasets with less variables such as ETTh1 and ETTh2, ts is more likely to be 0. Note that the ts may undergo a change from $\lambda = 0.6$ to $\lambda = 0.8$ on ETTh2 and ETTm2, resulting in different tokenization strategies. The results in table 4 indicate some correlations with the phenomenon. For ETTh2 and ETTm2, the tokenization strategy does not notably affect the predicting performance because of the existence of more highly correlated variables.

Patch length We use varying patch length of $P \in \{\frac{1}{2}L, \frac{1}{4}L, \frac{1}{8}L, \frac{1}{12}L, \frac{1}{24}L\}$. We conduct experiments on Weather, Electricity, ETTh1 and ETTh2. For datasets that use channel-independent or channel-mixing tokenization strategy, different patch lengths reflect the intra- or inter-series dependencies at different timescales. The results are shown in Fig. 4. For Weather and ETT datasets, the model gets optimal performance with $P = \frac{1}{24}L$ or $P = \frac{1}{12}L$. These datasets have relatively less variables and the data distribution exhibit stronger non-stationarity locally, hence suitable for more fine-grained modeling. The optimal patch length for Electricity is $P = \frac{1}{4}L$, indicating that coarse-grained modeling is beneficial for processing the data that is overall stable. In general, varying lengths of patch does not cause significant performance differences, which indicates that our model is robust in capturing intra- and inter-series dependencies at different time scales.

Dropout The Dropout layer randomly masks a portion of the input elements to zero to preventing overfitting. This helps to improve the generalization ability of the model. The dropout layer works in the Add&Norm layers in Fig 1. We set $dropout \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$. The results can be seen in Fig. 5(a) and Fig. 5(b). In most cases, the optimal result is obtained when $dropout = 0.2$, which indicates that a too low $dropout$ value does not work well to improve the generalization ability, while a too large $dropout$ value may cause performance degradation.
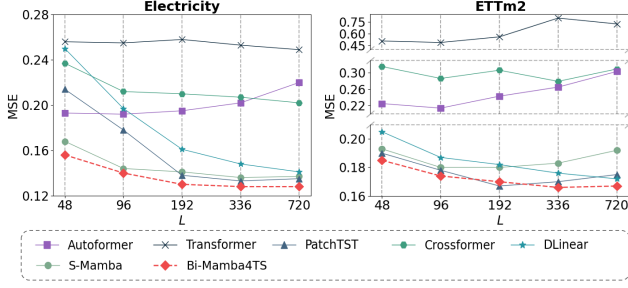
Figure 6: Results of varying look-back window size L on Electricity and ETTm2. The selected baseline models are Autoformer, Transformer, PatchTST, Crossformer, DLinear and S-Mamba. Overall, Bi-Mamba+ gets more accurate prediction with the increased receptive field and shows better predicting performance than other models.

Hidden state dimension of Mamba. A higher dimension N of the hidden state can help the model capture more complex system evolutionary patterns, but a too large N will reduce the efficiency of model training and may leads to overfitting. We set $N \in \{2, 4, 8, 16, 32\}$ and the results are shown in Fig. 5(c) and Fig. 5(d). For Electricity with more variables, a larger N produces better performance, this indicates that datasets with more complex inter-series dependencies prefer larger hidden state dimension for capturing the evolutionary patterns of time series. For ETTh2 with less variables, different values of N do not leads to varying predicting accuracy, which indicates that the temporal patterns in such datasets is less complex.

1-D convolutional kernel size of Mamba. The 1-D convolution module in Mamba is used for capturing temporal dependencies of the input series. We conduct experiments on different kernel size of 1, 2 and 3. The results in Fig. 5(e) and Fig. 5(f) show that Bi-Mamba+ gets optimal performance with the kernel size of 2. Overall, our model is not sensitive to the kernel size.

Long-term Dependencies Capturing

To evaluate the long-term dependencies capturing capability of Bi-Mamba+, we set $L \in \{48, 96, 192, 336, 720\}$ and record the average MSE scores of Bi-Mamba+, S-Mamba, Transformer, Autoformer, Crossformer and DLinear. We conduct experiments on Electricity and ETTm2. According to the SRA decider, these two datasets show preference to channel-mixing and channel-independent tokenization respectively, thus can more comprehensively reflect the adaptability of different models to the varying temporal evolutionary patterns.

Fig. 6 shows the results of the performance of models with varying look-back window size. Bi-Mamba+, S-Mamba, PatchTST and Crossformer get better predicting performance as the receptive field increases on Electricity. For previous channel-mixing models, Transformer does not show significant changes in predicting performance regardless of the size of the look-back window. Autoformer,
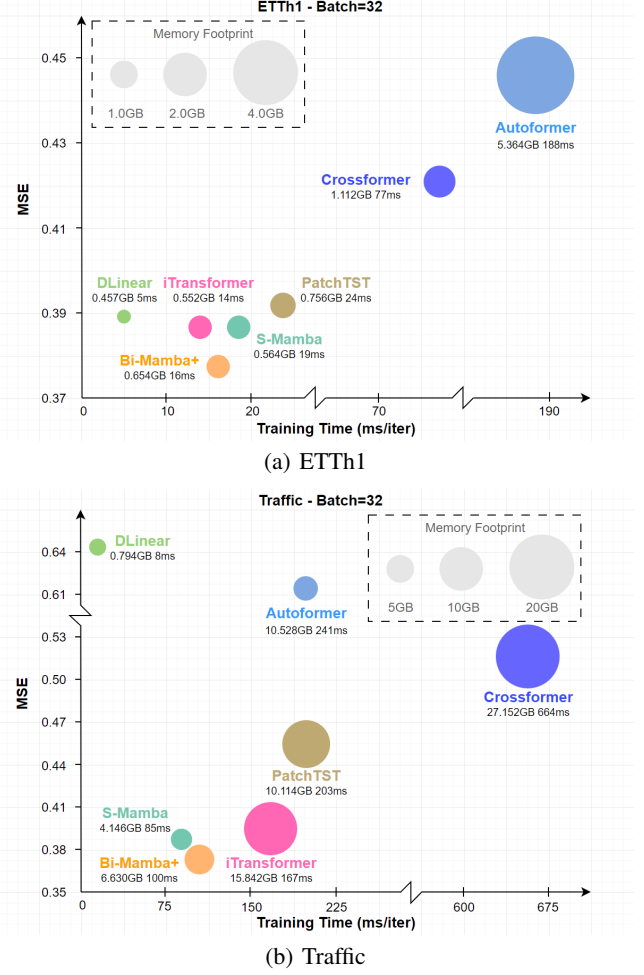

(a) ETTh1

(b) Traffic

Figure 7: Model efficiency comparison with $L = 96, H = 96$ on ETTh1 and Traffic. The batch size is set to 32.

however, even faces performance degradation at inflection point $L = 96$. These two models may encounter the challenge of overfitting caused by point-wise tokens. We note that S-Mamba, PatchTST and S-Mamba do not achieve performance improvements similar to those on Electricity as the receptive fields increase on ETTm2. Compared to Electricity, the periodicity of ETTm2 is less obvious, making it more difficult to model the evolutionary patterns within the series. Previous channel-mixing models such as Autoformer encounters performance degradation at inflection point $L = 96$, which may related to overfitting caused by point-wise tokens. Overall, Bi-Mamba+ can benefit from longer look-back window size and adapt to more complex temporal evolutionary patterns.

Model Efficiency

We conduct the following experiments to comprehensively evaluate the model efficiency from (a) predicting accuracy, (b) memory usage and (c) training speed. We set $L = 96, H = 96$ as the forecasting task and use $Batch = 32$

(a) Results of Bi-Mamba+ and iTransformer on Electricity

(b) Results of Autoformer, DLinear and PatchTST on Electricity

(c) Results of Bi-Mamba+ and iTransformer on Traffic

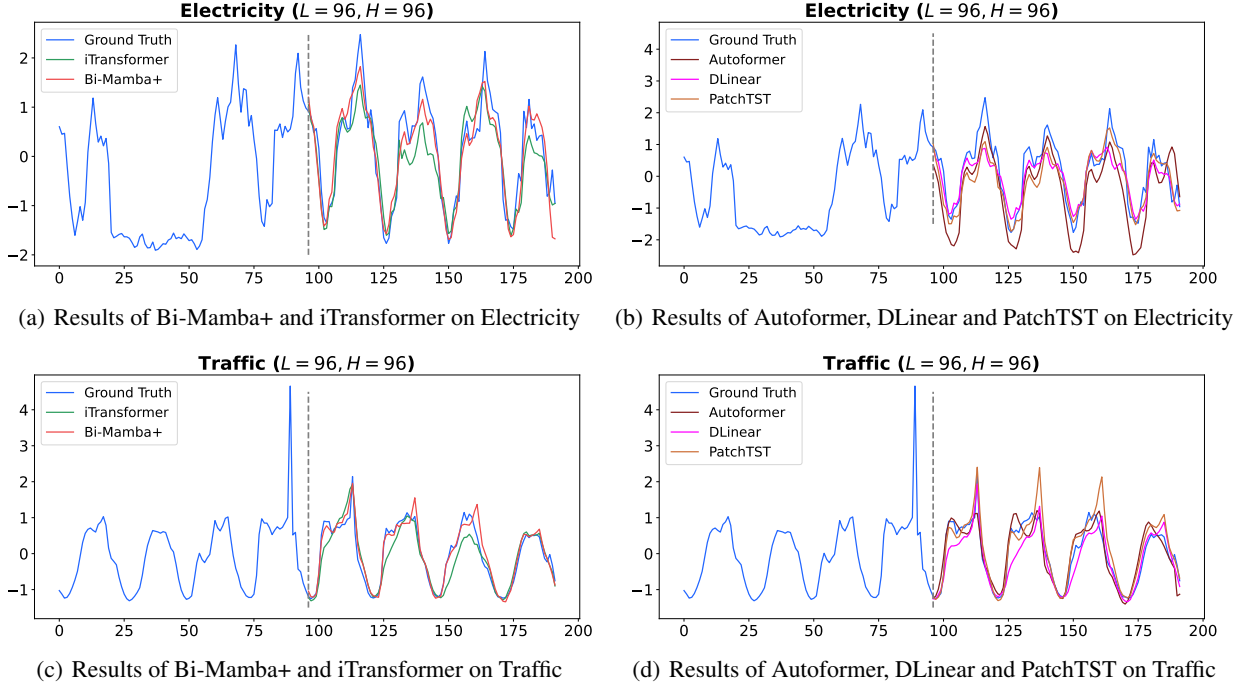(d) Results of Autoformer, DLinear and PatchTST on Traffic

Figure 8: Model predicting visualization of Bi-Mamba+, iTransformer, Autoformer, DLinear and PatchTST on Electricity and Traffic. The look-back window size is set to $L = 96$ and the predicting length is set to $H = 96$.

for ETTh1 and Traffic. The results are shown in Fig. 7(a) and Fig. 7(b). We freeze the patch length to $P = \frac{1}{4}L$ to provide consistent modeling process. We do not adopt **Effcient Training**(Liu et al. 2023) mechanism for Bi-Mamba+, S-Mamba and iTransformer on Traffic, in order to provide an intuitive comparison of the self-attention mechanism with Mamba.

For small datasets like ETTh1, Bi-Mamba+, S-Mamba, iTransformer, PatchTST and DLinear cost similar memory. For large datasets like Traffic, when the number of variables grows numerous, due to the the quadratic complexity of self-attention, the memory cost by iTransformer increases significantly, while Bi-Mamba+ and S-Mamba retain linear growth of resource usage, benefiting from the linear complexity of SSM. Therefore, Bi-Mamba+ strikes a good balance among predicting performance, training speed and memory usage. Overall, the computational cost of Mamba+ is less than self-attention mechanism in Transformer with the same embedding size D, which is reflected in the training speed and GPU memory usage. With the better predicting performance, faster training speed and less memory costs, Mamba-like models show great potential in LTSF.

Visualization

We visualize the long-term forecasting results of Bi-Mamba+, PatchTST, DLinear, and Autoformer in Fig. 8. We predict 96 steps on Traffic and Electricity. Bi-Mamba+ provides stable prediction results which is the closest to the ground truth.

Conclusion

In this paper, we propose Bi-Mamba+, a novel model that adaptively captures intra- or inter-series dependencies of MTS data. We propose Mamba+ specifically designed for LTSF. Mamba+ adds a forget gate in Mamba to selectively combine the added new features with the forgotten historical features in a complementary manner, therefore preserving historical information in a longer range. We design a bidirectional structure to model the MTS data more comprehensively. We divide the time series into patches to encourage the model to be focused on the inner evolutionary process or inter-series dependencies at a finer granularity. We explore the validity of Mamba-like structures and find that Mamba has great potential in time series forecasting tasks. Extensive experiments show the superior performance and high efficiency of our model compared to the SOTA methods. Future research will explore more diverse and complex scenarios such as network flow forecasting.

References

Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pages 922–929, 2019.

Niko Uremović, Marko Bizjak, Primož Sukič, Gorazd Štumberger, Borut Žalik, and Niko Lukač. A new framework for multivariate time series forecasting in energy management system. IEEE Transactions on Smart Grid, 2022.

Gang Zhang, Dazhi Yang, George Galanis, and Emmanouil Androulakis. Solar forecasting with hourly updated numerical weather prediction. Renewable and Sustainable Energy Reviews, 154:111768, 2022a.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied soft computing, 90:106181, 2020.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In The Thirty-Fifth AAAI Conference on Artificial Intelligence, volume 35, pages 11106–11115, 2021.

Yang Lin, Irena Koprinska, and Mashud Rana. Ssdnet: State space decomposition neural network for time series forecasting. In 2021 IEEE International Conference on Data Mining (ICDM), pages 370–378. IEEE, 2021.

Michael Zhang, Khaled Kamal Saab, Michael Poli, Tri Dao, Karan Goel, and Christopher Re. Effectively modeling time series with simple discrete state spaces. In The Eleventh International Conference on Learning Representations, 2022b.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. arXiv preprint arXiv:2401.09417, 2024.

Jun Ma, Feifei Li, and Bo Wang. U-mamba: Enhancing long-range dependency for biomedical image segmentation. arXiv preprint arXiv:2401.04722, 2024.

Md Atik Ahamed and Qiang Cheng. Timemachine: A time series is worth 4 mambas for long-term forecasting. arXiv preprint arXiv:2403.09898, 2024.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. Advances in Neural Information Processing Systems, 35:9881–9893, 2022.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. Advances in neural information processing systems, 33:1474–1487, 2020.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. Advances in neural information processing systems, 34:572–585, 2021a.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. arXiv preprint arXiv:2111.00396, 2021b.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752, 2023.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In Proceedings of the AAAI conference on artificial intelligence, volume 37, pages 11121–11128, 2023.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023.

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pages 6989–6997, 2023.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023.

Qihe Huang, Lei Shen, Ruixin Zhang, Shouhong Ding, Binwu Wang, Zhengyang Zhou, and Yang Wang. Crossgnn: Confronting noisy multivariate time series via cross interaction refinement. Advances in Neural Information Processing Systems, 36, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In Advances in Neural Information Processing Systems, 2021.

Zhenwei Zhang, Linghang Meng, and Yuantao Gu. Sageformer: Series-aware framework for long-term multivariate time series forecasting. IEEE Internet of Things Journal, 2024.

Yuxin Jia, Youfang Lin, Xinyan Hao, Yan Lin, Shengnan Guo, and Huaiyu Wan. Witran: Water-wave information transmission and recurrent acceleration network for long-range time series forecasting. Advances in Neural Information Processing Systems, 36, 2024.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In The eleventh international conference on learning representations, 2022.

Zihan Wang, Fanheng Kong, Shi Feng, Ming Wang, Han Zhao, Daling Wang, and Yifei Zhang. Is mamba effective for time series forecasting? arXiv preprint arXiv:2403.11144, 2024.

Ali Behrouz, Michele Santacatterina, and Ramin Zabih. Mambamixer: Efficient selective state space models with dual token and channel selection. arXiv preprint arXiv:2403.19888, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. Philosophical Transactions of the Royal Society A, 379(2194):20200209, 2021.

Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In International conference on learning representations, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In International conference on machine learning, pages 27268–27286. PMLR, 2022.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In The Twelfth International Conference on Learning Representations, 2023.

Haizhou Cao, Zhenhao Huang, Tiechui Yao, Jue Wang, Hui He, and Yangang Wang. Inparformer: evolutionary decomposition transformers with interactive parallel attention for long-term time series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pages 6906–6915, 2023.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. arXiv preprint arXiv:2403.19887, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. Adarnn: Adaptive learning and forecasting of time series. In Proceedings of the 30th ACM international conference on information & knowledge management, pages 402–411, 2021.

Lifan Long, Qian Liu, Hong Peng, Jun Wang, and Qian Yang. Multivariate time series forecasting method based on nonlinear spiking neural p systems and non-subsampled shearlet transform. Neural Networks, 152:300–310, 2022.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In International Conference on Learning Representations, 2022.

Appendix

Full Results of Ablation Study

Table 5 shows the detailed MSE and MAE results of (a) w/o SRA-I which use channel-independent strategy only, (b) w/o SRA-M which use channel-mixing strategy only, (c) w/o Bi which use forward direction Mamba block only, (d) w/o Residual that removes the residual connection, (e) S-Mamba and (f) PatchTST. The look-back window size is set to $L = 96$, we conduct experiments on varying predicting length $H \in \{96, 192, 336, 720\}$.

Reprogrammed Back Propagation

For the original Mamba block, the forward propagation process passes the input sequence through two branches, denoted as b_1 and b_2. Mamba calculates the result by the formula $\text{output} = \text{Linear}^{y'}(\text{y} \otimes \text{SiLU(z)})$, where y is the output of SSM in b_1 and z is the linear layer output in b_2. Our proposed Mamba+ block adjust the calculation process to $\text{output} = \text{Linear}^{y'}(\text{y} \otimes \text{SiLU(z)} + \text{x}' \otimes (1 - \sigma(\text{z})))$, where x' is the output of 1-D convolutional layer in b_1. The hardware-aware parallel computing algorithm of Mamba is implemented through CUDA programming. We modify the corresponding *.cuh* files programmed with **C++** and implement the back propagation logic in Alg. 4. We obtain the *selective_scan_cuda* program by enforcing the use of source code compilation, so that the training process of the model is completed by Mamba+.

Algorithm 4: The back propagation process adjustment of Mamba+ block

---

Input: Loss gradient $\frac{\delta\mathcal{L}}{\delta\text{output}}$ of linear layer $\text{Linear}^{y'}$

Output: Gradient $\frac{\delta\mathcal{L}}{\delta\text{y}}$ passed to SSM module in b_1, gradient $\frac{\delta\mathcal{L}}{\delta\text{x}'}$ passed to 1-D convolutional layer in b_1 and gradient $\frac{\delta\mathcal{L}}{\text{z}}$ passed to z in b_2.

1: $\frac{\delta\text{output}}{\delta\text{y}} = \text{SiLU(z)} = \frac{\text{z}}{1+\exp(-\text{z})}$

2: Store: $\frac{\delta\mathcal{L}}{\delta\text{y}} = \frac{\delta\mathcal{L}}{\delta\text{output}} \cdot \frac{\delta\text{output}}{\delta\text{y}}$

3: $\frac{\delta\text{output}}{\delta\text{x}'} = 1 - \sigma(\text{z}) = \frac{\exp(-\text{z})}{1+\exp(-\text{z})}$

4: Store: $\frac{\delta\mathcal{L}}{\delta\text{x}'} = \frac{\delta\mathcal{L}}{\delta\text{output}} \cdot \frac{\delta\text{output}}{\delta\text{x}'}$

5: $\frac{\delta\text{output}}{\delta\text{z}} = \text{y} \cdot \frac{\delta\text{SiLU(z)}}{\delta\text{z}} + \text{x}' \cdot \frac{\delta(1-\sigma(\text{z}))}{\delta\text{z}}$

6: $\frac{\delta\text{output}}{\delta\text{z}} = \text{y} \cdot \frac{1}{1+\exp(-\text{z})} \left(1 + \text{z}\left(\frac{\exp(-\text{z})}{1+\exp(-\text{z})}\right)\right) - \text{x}' \cdot \frac{1}{1+\exp(-\text{z})} \left(\frac{\exp(-\text{z})}{1+\exp(-\text{z})}\right)$

7: Store: $\frac{\delta\mathcal{L}}{\delta\text{z}} = \frac{\delta\mathcal{L}}{\delta\text{output}} \cdot \frac{\delta\text{output}}{\delta\text{z}}$

8: **return** $\frac{\delta\mathcal{L}}{\delta\text{y}}, \frac{\delta\mathcal{L}}{\delta\text{x}'}, \frac{\delta\mathcal{L}}{\delta\text{z}}$

Table 5: Full results of ablation studies of (a) w/o SRA-I which use channel-independent strategy only, (b) w/o SRA-M which use channel-mixing strategy only, (c) w/o Bi which use forward direction Mamba block only, (d) w/o Residual that removes the residual connection, (e) S-Mamba and (f) PatchTST.

Models		w/o SRA-I		w/o SRA-M		w/o Bi		w/o Residual		w/o M & w/ A		S-Mamba		PatchTST	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.164	0.211	0.159	0.205	0.169	0.215	0.166	0.216	0.174	0.215	0.166	0.211	0.178	0.219
	192	0.219	0.255	0.205	0.249	0.212	0.257	0.214	0.260	0.222	0.260	0.216	0.253	0.224	0.259
	336	0.274	0.295	0.264	0.291	0.273	0.300	0.274	0.301	0.279	0.300	0.275	0.298	0.278	0.298
	720	0.352	0.347	0.343	0.344	0.349	0.350	0.352	0.353	0.352	0.347	0.353	0.349	0.354	0.348
Traffic	96	0.429	0.276	0.375	0.258	0.406	0.279	0.409	0.291	0.390	0.266	0.381	0.261	0.457	0.295
	192	0.440	0.280	0.394	0.269	0.422	0.283	0.420	0.295	0.411	0.280	0.397	0.267	0.471	0.299
	336	0.453	0.286	0.406	0.274	0.439	0.289	0.449	0.315	0.421	0.282	0.423	0.276	0.482	0.304
	720	0.488	0.305	0.440	0.288	0.472	0.306	0.478	0.321	0.447	0.297	0.460	0.298	0.514	0.322
Electricity	96	0.163	0.250	0.140	0.238	0.149	0.245	0.148	0.250	0.148	0.243	0.142	0.238	0.174	0.259
	192	0.171	0.259	0.155	0.253	0.163	0.258	0.167	0.266	0.162	0.256	0.163	0.261	0.178	0.265
	336	0.188	0.276	0.170	0.269	0.182	0.278	0.183	0.284	0.177	0.273	0.178	0.275	0.196	0.282
	720	0.229	0.311	0.197	0.293	0.208	0.302	0.219	0.316	0.210	0.303	0.207	0.303	0.237	0.316
ETTh1	96	0.378	0.395	0.387	0.407	0.383	0.403	0.386	0.406	0.394	0.408	0.386	0.406	0.393	0.408
	192	0.427	0.428	0.438	0.428	0.431	0.430	0.442	0.434	0.448	0.439	0.448	0.444	0.445	0.434
	336	0.471	0.445	0.486	0.446	0.474	0.450	0.483	0.457	0.458	0.449	0.494	0.468	0.474	0.451
	720	0.470	0.457	0.478	0.466	0.482	0.465	0.511	0.488	0.485	0.484	0.493	0.488	0.480	0.471
ETTh2	96	0.291	0.342	0.291	0.344	0.299	0.353	0.298	0.349	0.296	0.346	0.298	0.349	0.302	0.348
	192	0.368	0.392	0.371	0.393	0.374	0.400	0.382	0.400	0.377	0.393	0.379	0.398	0.388	0.400
	336	0.407	0.424	0.409	0.429	0.410	0.428	0.420	0.431	0.401	0.424	0.417	0.432	0.426	0.433
	720	0.421	0.439	0.415	0.442	0.421	0.442	0.427	0.446	0.421	0.444	0.431	0.449	0.431	0.446
ETTm1	96	0.320	0.360	0.320	0.363	0.323	0.363	0.326	0.365	0.317	0.357	0.331	0.368	0.329	0.367
	192	0.361	0.383	0.370	0.389	0.369	0.389	0.366	0.388	0.363	0.383	0.371	0.387	0.367	0.385
	336	0.386	0.402	0.402	0.413	0.396	0.408	0.400	0.409	0.392	0.404	0.417	0.418	0.399	0.410
	720	0.445	0.437	0.463	0.451	0.457	0.443	0.452	0.443	0.451	0.438	0.471	0.448	0.454	0.439
ETTm2	96	0.176	0.263	0.176	0.265	0.177	0.264	0.181	0.271	0.180	0.265	0.179	0.263	0.175	0.259
	192	0.242	0.304	0.244	0.308	0.247	0.310	0.249	0.315	0.249	0.310	0.253	0.310	0.241	0.302
	336	0.304	0.344	0.309	0.348	0.314	0.355	0.314	0.354	0.310	0.349	0.312	0.349	0.305	0.343
	720	0.402	0.402	0.405	0.406	0.412	0.411	0.418	0.415	0.410	0.409	0.412	0.408	0.402	0.400
Solar	96	0.204	0.243	0.184	0.222	0.197	0.226	0.195	0.231	0.193	0.242	0.205	0.241	0.234	0.286
	192	0.236	0.264	0.225	0.254	0.230	0.254	0.227	0.259	0.247	0.279	0.237	0.270	0.267	0.310
	336	0.252	0.270	0.249	0.270	0.260	0.288	0.246	0.276	0.268	0.299	0.256	0.287	0.290	0.315
	720	0.251	0.274	0.250	0.273	0.251	0.278	0.261	0.286	0.257	0.283	0.259	0.285	0.289	0.317