Bi-Mamba4TS: Bidirectional Mamba for Time Series Forecasting

Aobo Liang[1] Xingguo Jiang[1] Yan Sun[1*] Chang Lu[1]

[1] Beijing University of Posts and Telecommunications, China
{liangaobo, sunyan, luchang}@bupt.edu.cn, jxgaugusto@gmail.com

Abstract

Long-term time series forecasting (LTSF) provides longer insights into future trends and patterns. In recent years, deep learning models especially Transformers have achieved advanced performance in LTSF tasks. However, the quadratic complexity of Transformers rises the challenge of balancing computaional efficiency and predicting performance. 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 can well capture long-term dependencies while maintaining linear computational complexity. Mamba has shown great ability for long sequence modeling and is a potential competitor to Transformer-based models in LTSF. In this paper, we propose Bi-Mamba4TS, a bidirectional Mamba for time series forecasting. To address the sparsity of time series semantics, we adopt the patching technique to enrich the local information while capturing the evolutionary patterns of time series in a finer granularity. To select more appropriate modeling method based on the characteristics of the dataset, our model unifies the channel-independent and channel-mixing tokenization strategies and uses a series-relation-aware decider to control the strategy choosing process. Extensive experiments on seven real-world datasets show that our model achieves more accurate predictions compared with state-of-the-art methods.

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. 2021b,a). 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. Furthermore, the semantic information density of time series data at time points is lower than other types of sequence data and 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 that the model needs to process, which also leads to lower computational complexity. A recent work iTransformer(Liu et al. 2023) uses a simple fully connected layer to map the whole input 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 evolutionary patterns inside the time series. Therefore, we are motivated to model the time series in a patching manner.

- **Channel-independent and channel-mixing strategies**. Time series data may have multiple variables which have complex correlations between each other. Due to various internal and external factors, the inter-series dependencies of multivariate time series (MTS) data over a period of time is dynamic. For example, the traffic flow of interconnected roads within a transportation network influence each other in different manners during different periods of time. However, modeling inter-series dependencies using channel-mixing strategies does not always produce SOTA performance. Works such as PatchTST and NHITS(Challu et al. 2023) process the MTS data in a channel-independent way, where each univariate series is input to the model independently to enhance the robustness of the model. A recent study TimeMachine(Ahamed and Cheng 2024) proposes a unified structure to handle channel-mixing and channel-independent situations. However, the boundary for the selection of the two strategies is ambiguous. Directly choosing strategies based on the relationship between the length of the historical observations and the number of variables in the dataset is lack of generality.

To solve the above challenges, we introduce Bi-Mamba4TS, a Bidirectional Mamba for Time Series forecasting. We aim to leverage Mamba's linear computational complexity and powerful long sequence modeling capabilities. We design a series-relation-aware (SRA) decider based on the Pearson coefficient correlation to automatically choose channel-independent or channel-mixing tokenization strategies for varying datasets. We divide the input sequence into patches and generate patch-wise tokens based on the chosen tokenization strategy. The patch-wise tokens contains richer semantic information and encourage the model to learn the evolutionary patterns of the time series in a finer granularity. Besides, we propose a bidirectional Mamba encoder to model the MTS data more comprehensively. With the above designs, our model can choose the optimal strategy to capture the intra-series dependencies or the inter-series dependencies based on the characteristics of the specific MTS data. The main contributions of this paper are summarized as follows:

- We propose Bi-Mamba4TS, a long-term multivariate time series forecasting model based on SSM architecture. our model unifies channel-independent and channel-mixing tokenization strategies. Besides, our model divides the time series into patches to capture long-term dependencies in a finer granularity.
- We design a SRA decider to automatically choose channel-independent or channel-mixing tokenization strategies. The SRA decider provides more objective results based on the Pearson correlation coefficient of different datasets.
- We conduct extensive experiments on seven widely used real-world datasets. The results show that Bi-Mamba4TS 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. 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

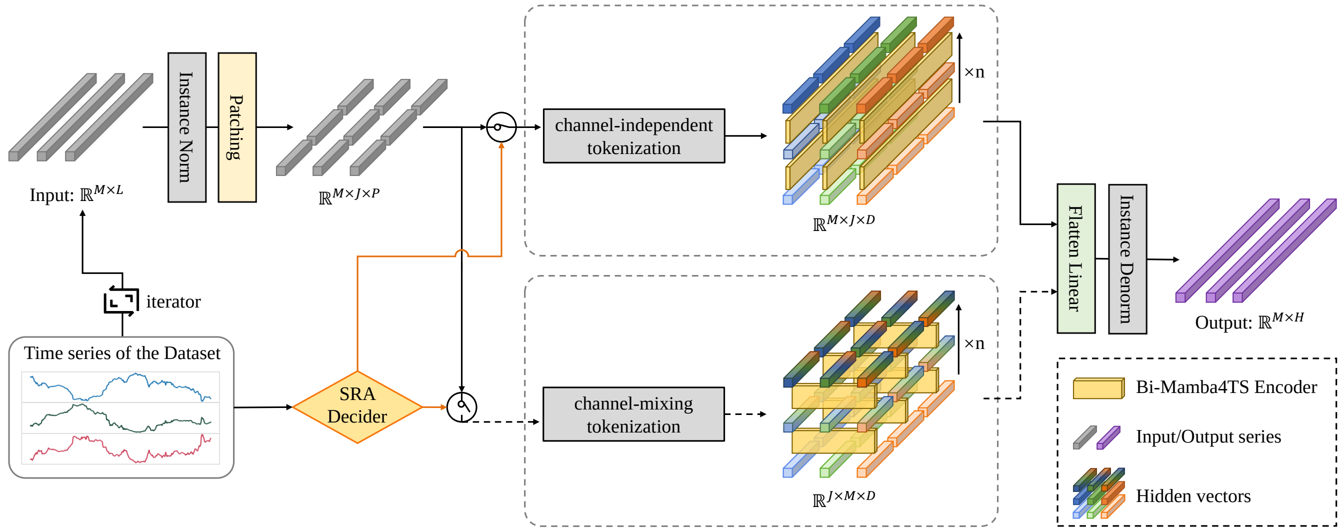


Figure 1: The architecture of Bi-Mamba4TS. The input time series is first divided into patches and then pass through a channel-independent or channel-mixing embedding layer based on the data characteristics of specific datasets. The embeddings are then fed into multiple Bi-Mamba Encoders and get the final output through an MLP projector.

structure is proposed, namely State Space Models (SSM). SSM(Gu et al. 2021b,a) 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. 2021a) 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/D-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. TimeMachine(Ahamed and Cheng 2024) proposes a multi-scale quadruple-Mamba architecture 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. This approach does not fully consider the characteristics of the MTS data.

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:

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

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:

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

The formula of discretized SSM can then be written as:

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

The discretized SSM(Gu et al. 2021b) 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. 2021a), 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-Mamba4TS 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 and calculate the Pearson 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 $\rho^{\lambda}_{\max}$ and $\rho^{0}_{\max}$ in the training set and calculate the relation ratio $r = \rho^{\lambda}_{\max}/\rho^{0}_{\max}$. We adopt channel-mixing strategy to generate sequence tokens for datasets with $r \geq 1 - \lambda$, otherwise, we adopt channel-independent strategy.

Tokenization Process. We generalize patch-wise tokens to emphasize capturing local evolutionary patterns of the time series. Specifically, each channel univariate sequence

Algorithm 1: SRA Decider

Input: Training set $T = \{t^1, t^2, ..., t^M\}$ of MTS data
Output: Tokenization strategy $ts \in \{0, 1\}$

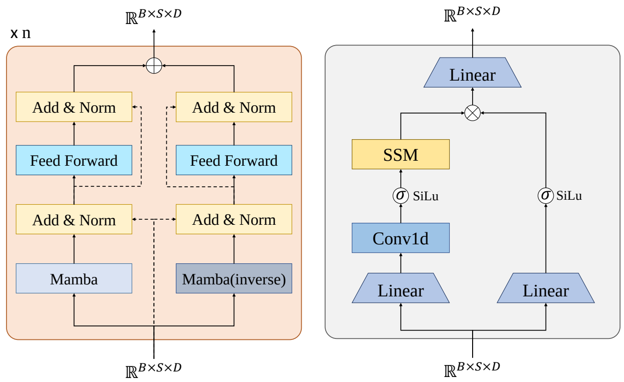
```
for i = 1 to M do
    ρ_{i,i} = 0
    for j = i + 1 to M do
        ρ_{i,j} = ρ_{j,i} = conv(t^i, t^j) / (σ_{t^i} · σ_{t^j})
    end for
end for
initialize list K^λ and K^0 of length M with all elements set to 0
for i = 1 to M do
    for j = 1 to M do
        if ρ_{i,j} ≥ λ then
            K^λ_i = K^λ_i + 1
        else if ρ_{i,j} > 0 then
            K^0_i = K^0_i + 1
        end if
    end for
end for
ρ^λ_max = argMax(K^λ), ρ^0_max = argMax(K^0)
r = ρ^λ_max / ρ^0_max
if r ≥ 1 − λ then
    return ts = 1 for channel-mixing tokenization
else
    return ts = 0 for channel-independent tokenization
end if
```

$x^i_{1:L}$ is first divided into patches $p^i \in \mathbb{R}^{J \times P}$ where P refers to the length of each patch and J is the number of patches. There are two strategies to process the input patches to get patch-wise tokens, namely channel-independent and channel-mixing strategies. For channel-independent strategy, each univariate channel is processed individually and finally concatenated to the embeddings $\mathbb{E}_{ind} \in \mathbb{R}^{M \times J \times D}$, 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. In this case, the obtained tokens can be represented as $\mathbb{E}_{mix} \in \mathbb{R}^{J \times M \times D}$.

Bidirectional Mamba Encoder

A original Mamba block is designed to process 1-D sequence on one direction. Considering the diverse representations of features within individual series in different directions, as well as the complexity of inter-series dependencies, we design a bidirectional Mamba structure to comprehensively capture the information of time series.

A Bi-Mamba encoder takes $\mathbb{E}^{(l)}_x \in \mathbb{R}^{B \times S \times D}$ as input, where l refers to the encoder layer, B and S corresponds to M or J depending on the tokenization strategy. Specifically, if $ts = 1$, $\mathbb{E}^{(l)}_x \in \mathbb{R}^{J \times M \times D}$ and $\mathbb{E}^{(0)}_x = \mathbb{E}_{mix}$, otherwise, $\mathbb{E}^{(l)}_x \in \mathbb{R}^{M \times J \times D}$ and $\mathbb{E}^{(0)}_x = \mathbb{E}_{ind}$. 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



(a) Bi-Mamba Encoder. (b) Mamba Block.

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

directions as $\mathbb{E}_{x,dir}^{(l)}$ where $dir \in \{forward, backward\}$. A Mamba block first passes the input $\mathbb{E}_{x,dir}^{(l)}$ through two linear layers to expand the hidden dimension to the size of E, as shown in Fig. 2(b). The result of one of the branches is passed through a 1-D convolutional layer, a SiLU activation function and a standard SSM block to get the candidate output $\tilde{\mathbb{E}}_{y,dir}^{(l)}$. The result of another branch is passed through a SiLU activation function to serve as a gate. It is multiplied with $\tilde{\mathbb{E}}_{y,dir}^{(l)}$ to get the final output $\mathbb{E}_{y,dir}^{(l)}$. 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.2.

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:

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

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

Experiments

Datasets

We choose 7 real-world datasets to evaluate our proposed model: Weather, Traffic, Electricity 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.

Algorithm 2: Implementation of Bi-Mamba Encoder

Input: Patch-wise token sequence $\mathbb{E}_{x}^{(l)} : (B, S, D)$
Output: Output features for the next encoder layer $\mathbb{E}_{x}^{(l+1)} : (B, S, D)$

for dir in $\{forward, backward\}$ **do**
  if $dir = backward$ **then**
    $\mathbb{E}_{x,dir}^{(l)} : (B, S, D) \leftarrow \mathrm{Flip}(\mathbb{E}_{x}^{(l)}, \mathrm{dim} = 1)$
  else
    $\mathbb{E}_{x,dir}^{(l)} : (B, S, D) \leftarrow \mathbb{E}_{x}^{(l)}$
  end if
  $\mathrm{x} : (B, S, E) \leftarrow \mathrm{Linear}^{\mathrm{x}}(\mathbb{E}_{x,dir}^{(l)})$
  $\mathrm{z} : (B, S, E) \leftarrow \mathrm{Linear}^{\mathrm{z}}(\mathbb{E}_{x,dir}^{(l)})$
  $\mathrm{x}' : (B, S, E) \leftarrow \mathrm{SiLU}(\mathrm{Conv1D}(\mathrm{x}))$
  $\mathbf{A} : (E, N) \leftarrow \mathrm{Parameter}^{\mathbf{A}}$
  $\mathbf{B} : (B, S, N) \leftarrow \mathrm{Linear}^{\mathbf{B}}(\mathrm{x}')$
  $\mathbf{C} : (B, S, N) \leftarrow \mathrm{Linear}^{\mathbf{C}}(\mathrm{x}')$
  $\Delta : (B, S, E) \leftarrow \log(1 + \exp(\mathrm{Linear}^{\Delta}(\mathrm{x}')) + \mathrm{Parameter}^{\Delta})$
  $\overline{\mathbf{A}}, \overline{\mathbf{B}} : (B, S, E, N) \leftarrow \mathrm{discretize}(\Delta, \mathbf{A}, \mathbf{B})$
  $\mathrm{y} : (B, S, E) \leftarrow \mathrm{SSM}(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C})(\mathrm{x}')$
  $\mathrm{y}' : (B, S, E) \leftarrow \mathrm{y} \otimes \mathrm{SiLU}(\mathrm{z})$
  $\tilde{\mathbb{E}}_{y,dir}^{(l)} : (B, S, D) \leftarrow \mathrm{Linear}^{\mathrm{y}}(\mathrm{y}')$
  $\tilde{\mathbb{E}}_{y,dir}^{(l)} : (B, S, D) \leftarrow \mathrm{Add\&Norm}(\tilde{\mathbb{E}}_{y,dir}^{(l)}, \mathbb{E}_{x,dir}^{(l)})$
  $\mathbb{E}_{y,dir}^{(l)} : (B, S, D) \leftarrow \mathrm{FeedForward}(\tilde{\mathbb{E}}_{y,dir}^{(l)})$
  $\mathbb{E}_{y,dir}^{(l)} : (B, S, D) \leftarrow \mathrm{Add\&Norm}(\mathbb{E}_{y,dir}^{(l)}, \tilde{\mathbb{E}}_{y,dir}^{(l)})$
end for
return $\mathbb{E}_{x}^{(l+1)} : (B, S, D) \leftarrow \mathbb{E}_{y,forward}^{(l)} + \mathbb{E}_{y,backward}^{(l)}$

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.
- **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.

Table 1: Statistics of 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

Table 2: The information of 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
CUDA Version	11.8
Python Version	3.10.14
Pytorch Version	2.1.1

- **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 intra-scale and inter-series dependencies.
- **D-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 $P = 24$ and $S = 12$ on all datasets. Considering that different models may have different sensitivity of granularity of local information, we use 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 Pearson coefficient correlation is larger than 0.6, it can be inferred that the two sequences have a strong correlation. More detailed evaluation of λ is shown in Fig. 3. As for the Mamba block, we set $d\_conv = 2$ and $expand = 1$ on all datasets. Considering that datasets with more variables may have more complex patterns, we set $d\_model = 64$, $d\_state = 32$ for Weather, Electricity and Traffic and $d\_model = 32$, $d\_state = 2$ for ETT datasets. We evaluate Bi-Mamba4TS with encoder layer $l \in \{1, 2, 3\}$ and conduct grid search for learning rate on [4e-5, 1e-4, 4e-4, 1e-3, 4e-3, 1e-2] to find the best results for all datasets. For the metrics to evaluate the performance, we utilize MSE and MAE, defined in Eq 5 and Eq 6. All models are optimized using the ADAM algorithm(Kingma and Ba 2014) and early stopping mechanism. The training process is limited to 60 epochs.

$$\mathrm{MSE}(Y, \hat{Y}) = \frac{1}{|Y|}\sum_{i=1}^{|Y|}|Y_i - \hat{Y}_i|^2 \tag{5}$$

$$\mathrm{MAE}(Y, \hat{Y}) = \frac{1}{|Y|}\sum_{i=1}^{|Y|}|Y_i - \hat{Y}_i| \tag{6}$$

Experimental Environment

We conduct all experiments on the same device. The detailed information about 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-Mamba4TS reduces the MSE, MAE errors by 4.92% and 2.16% on average. The improvement of MSE and MAE comes to 3.55% and 1.87% compared with D-Mamba, which is also an SSM-based model using Mamba. Note that Bi-Mamba4TS adopts channel-mixing strategy on Weather, Traffic and Electricity and channel-independent strategy on 4 ETT datasets. For channel-mixing strategy, the patching approach used by Bi-Mamba4TS helps the model capture inter-series dependencies at a finer granularity, therefore achieving more accurate predicting. 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. This also indicates that adopting different tokenization strategies for specific datasets help improve the model's performance.

Ablation Study

We verify the effectiveness of the SRA decider and the bidirectional Mamba settings. Before we conduct the ablation experiments, we first export the results of the SRA decider to view the strategy choosing for different datasets. For Pearson correlation coefficient ρ, we typically use $\lambda > 0.6$ to evaluate whether two variables have positive correlation strong enough to influence each other. We choose $\lambda = 0.6$ and calculate the strategy indicator ts for different datasets, the results 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 many variables, the model tends to adopt channel-mixing strategy to emphasis inter-series dependencies. We remove the SRA decider to form two specific Bi-Mamba4TS models: (a) Bi-Mamba4TS-I and (b) Bi-Mamba4TS-M that only use channel-independent or channel-mixing strategy respectively. For the bidirectional Mamba blocks, we remove the

Table 3: Experimental results of long-term multivariate time series forecasting task on 7 real-world datasets. The best results are in **bold** and the second best results are underlined.

Models		Bi-Mamba4TS		D-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	0.164	**0.212**	0.170	0.214	0.174	0.214	0.178	0.219	**0.162**	0.232	0.266	0.336	0.196	0.235	0.172	0.220	0.186	0.237	0.178	0.223
	192	0.214	0.256	0.220	0.257	0.221	0.254	0.224	0.259	**0.208**	**0.251**	0.307	0.367	0.241	0.271	0.219	0.261	0.233	0.273	0.223	0.261
	336	**0.269**	**0.296**	0.276	0.298	0.278	0.298	0.292	0.306	0.270	0.305	0.359	0.395	0.292	0.306	0.280	0.306	0.289	0.312	0.288	0.309
	720	**0.348**	0.349	0.356	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	**0.383**	**0.261**	0.388	0.267	0.395	0.268	0.457	0.295	0.534	0.280	0.613	0.388	0.647	0.384	0.593	0.321	0.676	0.407	1.037	0.441
	192	**0.398**	**0.269**	0.411	0.274	0.417	0.276	0.471	0.299	0.556	0.300	0.616	0.382	0.596	0.359	0.617	0.336	0.631	0.386	1.061	0.455
	336	**0.410**	**0.276**	0.423	0.280	0.433	0.283	0.482	0.304	0.578	0.315	0.616	0.382	0.601	0.361	0.629	0.336	0.640	0.387	1.095	0.470
	720	**0.447**	**0.298**	0.458	0.300	0.467	0.302	0.514	0.322	0.594	0.321	0.622	0.337	0.642	0.381	0.640	0.350	0.681	0.402	1.121	0.474
Electricity	96	**0.142**	**0.238**	**0.142**	**0.238**	0.148	0.240	0.174	0.259	0.152	0.249	0.201	0.317	0.206	0.288	0.168	0.272	0.217	0.304	0.237	0.335
	192	**0.157**	**0.253**	0.169	0.267	0.162	**0.253**	0.178	0.265	0.174	0.275	0.222	0.334	0.206	0.290	0.184	0.289	0.216	0.306	0.258	0.350
	336	**0.172**	0.271	0.178	0.275	0.178	**0.269**	0.196	0.282	0.197	0.294	0.231	0.338	0.220	0.305	0.198	0.300	0.232	0.321	0.273	0.362
	720	**0.200**	**0.297**	0.207	0.303	0.225	0.317	0.237	0.316	0.258	0.339	0.254	0.361	0.252	0.337	0.220	0.320	0.273	0.352	0.300	0.382
ETTh1	96	**0.376**	0.403	0.386	0.406	0.386	0.405	0.393	0.408	0.395	0.411	0.449	0.459	0.387	0.406	0.384	0.402	0.382	**0.398**	0.414	0.419
	192	**0.411**	**0.425**	0.439	0.437	0.441	0.436	0.445	0.434	0.460	0.447	0.500	0.482	0.439	0.435	0.436	0.429	0.427	**0.425**	0.464	0.448
	336	**0.455**	**0.445**	0.485	0.467	0.487	0.458	0.474	0.451	0.479	0.455	0.521	0.496	0.493	0.457	0.491	0.469	0.486	0.487	0.516	0.478
	720	**0.460**	**0.464**	0.495	0.490	0.503	0.491	0.480	0.471	0.480	0.469	0.514	0.512	0.490	0.478	0.521	0.500	0.472	0.468	0.538	0.509
ETTh2	96	**0.289**	**0.341**	0.298	0.349	0.297	0.349	0.302	0.348	0.311	0.372	0.346	0.388	0.305	0.352	0.340	0.374	0.302	0.349	0.325	0.364
	192	**0.367**	**0.389**	0.386	0.401	0.380	0.400	0.388	0.400	0.393	0.414	0.456	0.452	0.424	0.439	0.402	0.414	0.382	0.400	0.433	0.427
	336	**0.410**	**0.424**	0.412	0.425	0.428	0.432	0.426	0.433	0.437	0.449	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	0.424	0.445	0.428	**0.432**	0.426	0.433	0.437	0.449	0.482	0.486	0.476	0.493	0.462	0.468	0.437	0.458	0.499	0.480
ETTm1	96	**0.312**	**0.354**	0.329	0.364	0.334	0.368	0.329	0.367	0.327	0.363	0.505	0.475	0.353	0.374	0.338	0.375	0.340	0.374	0.375	0.402
	192	**0.358**	**0.383**	0.371	0.387	0.377	0.391	0.367	0.385	0.376	0.397	0.553	0.496	0.389	0.391	0.374	0.387	0.377	0.390	0.427	0.434
	336	**0.388**	**0.404**	0.403	0.409	0.426	0.420	0.399	0.419	0.424	0.425	0.621	0.537	0.421	0.413	0.410	0.411	0.401	0.407	0.455	0.452
	720	**0.444**	**0.433**	0.471	0.448	0.491	0.459	0.454	0.439	0.482	0.462	0.671	0.561	0.484	0.448	0.478	0.450	0.453	0.442	0.527	0.488
ETTm2	96	**0.174**	**0.259**	0.181	0.264	0.180	0.264	0.175	**0.259**	0.181	0.266	0.255	0.339	0.182	0.264	0.187	0.267	0.177	0.261	0.191	0.272
	192	**0.240**	0.304	0.251	0.311	0.250	0.309	0.241	0.302	0.244	0.307	0.281	0.340	0.257	0.315	0.249	0.309	**0.240**	**0.298**	0.261	0.316
	336	**0.303**	0.345	0.311	0.348	0.311	0.348	0.305	**0.343**	0.310	0.351	0.339	0.372	0.318	0.353	0.321	0.351	0.305	0.345	0.330	0.358
	720	**0.402**	0.407	0.410	0.404	0.412	0.407	**0.402**	**0.400**	0.406	0.404	0.433	0.432	0.426	0.419	0.408	0.403	0.403	**0.400**	0.450	0.427

Table 4: Ablation studies of (a) Bi-Mamba4TS-I which use channel-independent strategy only, (b) Bi-Mamba4TS-M which use channel-mixing strategy only, (c) Mamba4TS which use forward direction Mamba block only and (d) D-Mamba and (d) PatchTST used for the benchmark models.

Models		Bi-Mamba4TS-I		Bi-Mamba4TS-M		Mamba4TS		D-Mamba		PatchTST	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	0.163	0.250	0.142	0.238	0.149	0.245	0.142	0.238	0.174	0.259
	192	0.171	0.259	0.157	0.253	0.163	0.258	0.169	0.267	0.178	0.265
	336	0.188	0.276	0.172	0.271	0.182	0.278	0.178	0.275	0.196	0.282
	720	0.229	0.311	0.200	0.297	0.208	0.302	0.207	0.303	0.237	0.316
ETTh1	96	0.376	0.403	0.394	0.410	0.383	0.403	0.386	0.406	0.393	0.408
	192	0.411	0.425	0.450	0.439	0.431	0.430	0.439	0.437	0.445	0.434
	336	0.455	0.445	0.492	0.454	0.474	0.450	0.485	0.467	0.474	0.451
	720	0.460	0.464	0.511	0.490	0.462	0.465	0.495	0.490	0.480	0.471

backward direction Mamba to evaluate the effectiveness of bidirectional design. This setting forms the following model: (c) Mamba4TS. We choose the results for Electricity and ETTh1, as shown in table 4. Without the decider, we mainly focus on the results of Bi-Mamba4TS-I on Electricity and Bi-Mamba4TS-M on ETTh1. The MSE and MAE increase by 10.65% and 3.35% on Electricity and 7.86% and 3.15% on ETTh1 respectively. Therefore, utilizing different tokenization strategies on different datasets brings better model performance. In addition, the proposed SRA decider makes the optimal choice of channel-independent and channel-mixing strategies. It is worth noting that Bi-Mamba4TS-I shows better performance on all cases than PatchTST that uses patching technique and channel-independent tokenization strategy. This indicates that Mamba demonstrates superior capabilities in long sequence modeling beyond self-attention. When the backward Mamba block is removed, the predicting performance is also deteriorated to some extent. This indicates that the bidirectional design helps the model get more comprehensive understanding of the MTS data.

Hyper-parameter Sensitivity Analysis

To verify whether Bi-Mamba4TS is sensitive to the hyper-parameters, we conduct experiments on (a) Pearson coefficient filter λ, (b) dropout, (c) Conv-1D kernel size of Mamba and (d) state dimension size. For each setting, we repeat the experiment 5 times with 60 epochs (with early-stop control) and 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.

Pearson 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

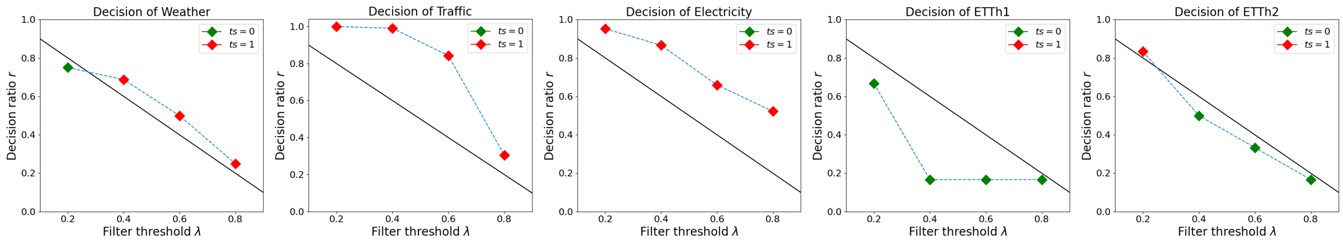


Figure 3: The tokenization strategy indicator ts of different datasets with varying settings of filter threshold λ.

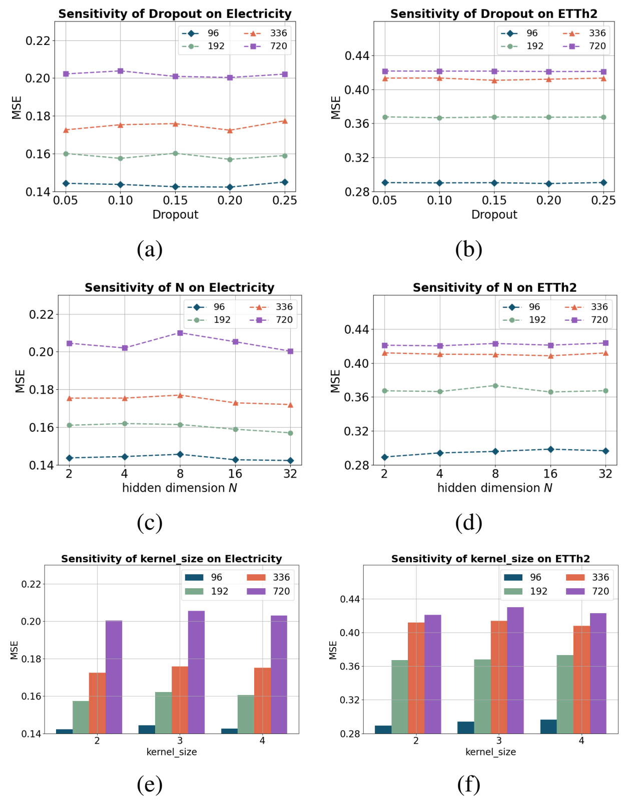


Figure 4: Results of varying dropout, hidden dimension N and kernel size values on Electricity and ETTh2.

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 $\lambda = 0.2$ to $\lambda = 0.4$ on Weather and ETTh2, resulting in different tokenization strategies. However, a Pearson coefficient closed to 0.2 can only indicate a weak relationship between the two sequences, therefore, this threshold is not a general option. We mainly focus on the decision performance of $\lambda \geq 0.4$ and the results show that the strategy decision is stable.

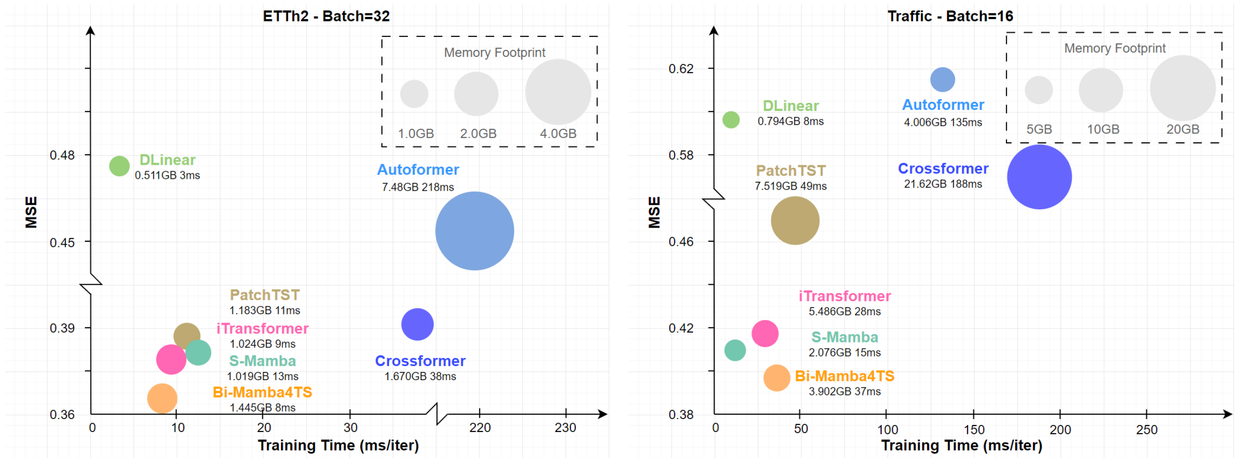
Dropouts. The Dropout layer randomly masks a portion of the input elements to zero to preventing overfitting. This helps to improve the generalization ability of the model. The dropout layer works in the Add&Norm layers in Fig 1. We set $dropout \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$. The results can be seen in Fig. 4(a) and Fig. 4(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.

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. 4(c) and Fig. 4(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. 4(e) and Fig. 4(f) show that Bi-Mamba4TS gets optimal performance with the kernel size of 2. Overall, our model is not sensitive to the kernel size.

Model Efficiency

We conduct the following experiments to comprehensively evaluate the model efficiency from (a) forecasting performance, (b) memory footprint and (c) training speed. We set $L = 96, H = 192$ as the forecasting task and use $Batch = 32$ for ETTh2 and $Batch = 16$ for Traffic. We train all the models according to their optimal settings that achieve the best performance. The results are shown in Fig. 5(a) and Fig. 5(b). Our proposed Bi-Mamba4TS strikes a good balance among predicting performance, training speed and memory usage. Overall, Mamba's computational cost is comparable to the self-attention mechanism in Transformer, which is reflected in the training speed. However, it is worth noting that with the same number of model layers, a model using Mamba costs less memory than that using a Transformer-based model. With the better predicting performance, faster training speed and less memory costs, Mamba shows the great potential in LTSF.



(a) ETTh1

(b) Traffic

Figure 5: Model efficiency comparison with $L = 96, H = 192$ on ETTh1 and Traffic. The batch size is 32 for ETTh1 and 16 for Traffic.

Conclusion

In this paper, we propose Bi-Mamba4TS, a novel model that uses bidirectional Mamba blocks to adaptively capture intra-series dependencies or inter-series dependencies of MTS data. We divide the time series into patches to encourage the model to be focused on the inner evolutionary process of each univariate time series or understand inter-series dependencies at a finer granularity. We explore the validity of Mamba 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

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

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.

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.

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.

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

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

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, 2021b.

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.

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.

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.

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.

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

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.

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

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.

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.

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.

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

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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

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.

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.

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.