

SST: Multi-Scale Hybrid Mamba-Transformer Experts for Long-Short Range Time Series Forecasting

Xiongxiao Xu¹, Canyu Chen¹, Yueqing Liang¹, Baixiang Huang¹, Guangji Bai²
Liang Zhao², Kai Shu²

¹Department of Computer Science, Illinois Institute of Technology

²Department of Computer Science, Emory University

{xxu85, cchen151, yliang40, bhuang15}@hawk.iit.edu, {gbai5, liang.zhao, kai.shu}@emory.edu,

Abstract

Despite significant progress in time series forecasting, existing forecasters often overlook the heterogeneity between long-range and short-range time series, leading to performance degradation in practical applications. In this work, we highlight the need of distinct objectives tailored to different ranges. We point out that time series can be decomposed into global patterns and local variations, which should be addressed separately in long- and short-range time series. To meet the objectives, we propose a multi-scale hybrid Mamba-Transformer experts model STATE SPACE TRANSFORMER (SST). SST leverages Mamba as an expert to extract global patterns in coarse-grained long-range time series, and Local Window Transformer (LWT), the other expert to focus on capturing local variations in fine-grained short-range time series. With an input-dependent mechanism, State Space Model (SSM)-based Mamba is able to selectively retain long-term patterns and filter out fluctuations, while LWT employs a local window to enhance locality-awareness capability, thus effectively capturing local variations. To adaptively integrate the global patterns and local variations, a long-short router dynamically adjusts contributions of the two experts. SST achieves superior performance with scaling linearly $O(L)$ on time series length L . The comprehensive experiments demonstrate the SST can achieve SOTA results in long-short range time series forecasting while maintaining low memory footprint and computational cost. The code of SST is available at <https://github.com/XiongxiaoXu/SST>.

1 Introduction

Time series forecasting is a crucial problem in a wide range of real-world scenarios, including weather forecasting (Abhishek et al. 2012), stock prediction (Sezer, Gudelek, and Ozbayoglu 2020), and scientific computing (Xu et al. 2023). In scientific computing, for instance, ML-based surrogate models are trained to accelerate simulations of supercomputers by predicting their high-performance computing (HPC) behaviors over various timescales (Cruz-Camacho et al. 2023). Despite its importance, the lack of distinction between long-range and short-range time series largely hinders the performance of existing forecasters. Figure 1 shows different statistics of supercomputers, including network traffic of a port on a router, execution time of a HPC application, and busy time of a port on the router, change over time. These time series can be decomposed into global patterns

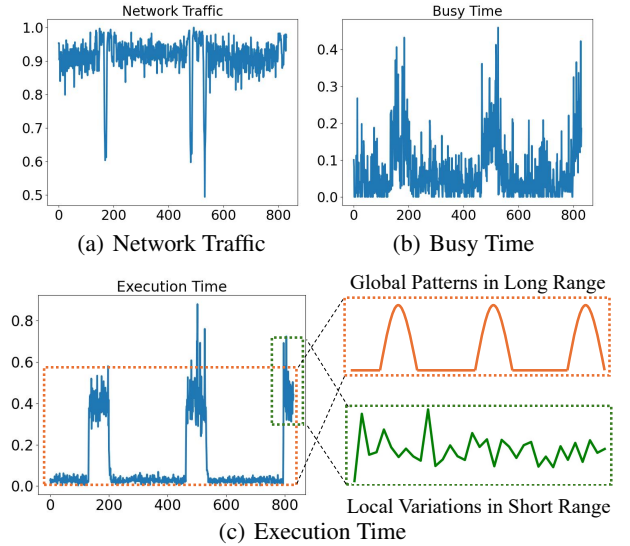


Figure 1: Three real datasets reflect (normalized) behaviors of supercomputers in the scientific computing field. They can be decomposed into global patterns and local variations by ranges. Take the execution time (figure (c)) for example. Global patterns (orange line) indicate repeated up-and-down trends because supercomputers intermittently execute applications, and local variations (green line) are extreme execution times caused by sudden network congestion.

in long range, such as repeated up-and-down trends, and local variations in short range, like extreme spike points. For long-range time series, it is essential to focus on global patterns, as local deviations, including outliers, can negatively impact forecasting accuracy. Conversely, short-range forecasting should emphasize local variations, because global patterns are less evident within limited time frames. Take the execution time for example. Extreme long execution times caused by sudden network congestion indicates an abnormal state in long-term trends but are pivotal for accurately forecasting next-term behaviors (Xu et al. 2024).

However, long-short range forecasting presents significant challenges. First, the distinction between long- and short-range time series remains ambiguous despite well-defined objectives: global patterns in long term, and local variations in short term. There is a pressing need for a strat-

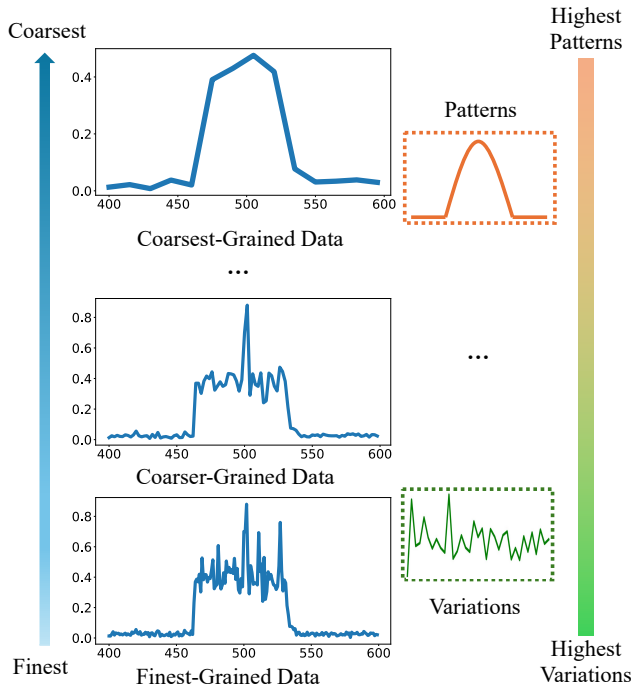


Figure 2: The execution time of HPC applications is organized from the finest to the coarsest resolution (from the bottom to the top). Global patterns are more apparent in low resolution while local variations emerge in high resolution.

egy to differentiate them and a new metric to measure the difference. Second, both long- and short-range time series intertwine patterns and variations. It is often unclear how to only capture long-term patterns and filter out variations in long range while accurately depicting local deviations in short range. Third, integrating long- and short-range dependencies is a non-trivial task. It requires a model to adaptively learn the relative importance between patterns and variations, ensuring enhanced performance.

To address the first challenge, we propose a multi-resolution framework by incorporating patches (Nie et al. 2023) to adjust resolutions of time series. The resolution plays an important role in time series forecasting (Nie et al. 2023; Liu et al. 2021; Zhang et al. 2023). For example, patching techniques (Nie et al. 2023) can aggregate time steps into subseries patches to enhance the receptive field, and become increasingly popular in time series analysis. However, existing forecasters often apply the same patching or equally the same resolution to both long- and short-range time series, resulting in suboptimal performance. Figure 2 plots the shape of execution time varies with different resolutions. It demonstrates patterns are more discernible at a coarser granularity, while variations emerge at a finer granularity. To exploit the observations, we leverage the patching to distinguish long- and short-range time series. In specific, we employ larger patches and longer stride for long range to obtain low-resolution patched time series (PTS); smaller patches and shorter stride for short range lead to high-resolution PTS. Moreover, despite early promising results of patching, existing literature lacks methods to quanti-

tatively assess the resolution of PTS. Therefore, we propose a novel metric to precisely quantify the resolution of PTS.

To solve the second and third challenges, we propose a novel hybrid Mamba-Transformer experts architecture, inspired by the idea of Mixture-of-Experts (MoEs) (Jacobs et al. 1991; Shazeer et al. 2017; Fedus, Zoph, and Shazeer 2022). Unlike traditional MoEs’ approaches where the roles of experts are ambiguous, we clearly delineates the responsibilities: Mamba serves as a global patterns expert to represent long-range dependencies, while Local Window Transformer (LWT) acts as a local variations expert to focus on short-range representations. As a representative State Space Model (SSM), Mamba is able to indefinitely retain relevant information in a manner dependent on the input with linear complexity (Gu and Dao 2023). This selective mechanism allows Mamba to preserve essential long-term patterns and disregard transient fluctuations. Meanwhile, LWT is specifically designed to focus on tokens within a local sliding window. The local window provides local inductive biases for vanilla Transformer, leading to enhanced local recognition capability, and reduces complexity from quadratic to linear. To seamlessly integrate the two specialized experts, a long-short router is proposed to adaptively learn their contributions. Remarkably, the hybrid architecture ensures a linear complexity $O(L)$ on time series length L , benefiting from the hardware-aware algorithm in Mamba and the local window mechanism in LWT.

We term the proposed multi-scale hybrid Mamba-Transformer experts model as STATE SPACE TRANSFORMER (SST). To the best of our knowledge, SST is a very early attempt to build a hybrid Mamba-Transformer architecture in time series. Our contributions are summarized:

- We propose to decompose time series into global patterns and local variations according to ranges. We identify that global patterns as the focus of long range and local variations should be captured in short range.
- To effectively capture long-term patterns and short-term variations, we leverage the patching to create coarser PTS in long range and finer PTS in short range. Moreover, we introduce a new metric to precisely quantify the resolution of PTS.
- We propose a novel hybrid Mamba-Transformer experts architecture SST, with Mamba as a global patterns expert in long range, and LWT as a local variations expert in short range. A long-short router is designed to adaptively integrate the global patterns and local variations. With Mamba and LWT, SST is highly scalable with linear complexity $O(L)$ on time series length L .

2 Related Work

Time Series Forecasting

Time series forecasting has been an crucial problem (Bon-tempi, Ben Taieb, and Le Borgne 2013; Xu 2023) for a long time. RNN (Cheng et al. 2018; Hewamalage, Bergmeir, and Bandara 2021) and LSTM (Yao et al. 2019; Xu et al. 2024) are two classical deep learning models for time series, but they fall short into gradient vanishing issues (Tetko, Liv-

ingstone, and Luik 1995) when dealing with long-range sequences. Inspired from the success of Transformer (Vaswani et al. 2017) in text data, a wide range of variants of Transformer (Wen et al. 2023; Li et al. 2019; Zhou et al. 2021; Liu et al. 2021; Wu et al. 2021; Zhou et al. 2022; Nie et al. 2023; Liu et al. 2024b) have proven effective in time series data. The latest iTransformer (Liu et al. 2024b) that simply applies the attention and feed-forward network on the inverted dimensions achieve SOTA performance. Recently, large language models (LLMs) are utilized for time series forecasting (Jin et al. 2023; Gruver et al. 2024; Tan et al. 2024) to explore potential of large foundation model. However, they ignore the heterogeneity of long- and short-range time series data and remain computationally challenging.

State Space Models and Mamba

State Space Models (SSMs) (Gu et al. 2021; Gu, Goel, and Re 2022; Gu and Dao 2023) emerge as a promising class of architectures for sequence modeling. With selective SSMs and a hardware-efficient algorithm, Mamba has achieved impressive performance across modalities such as languages (Gu and Dao 2023; Dao and Gu 2024), images (Zhu et al. 2024; Tang et al. 2024), medicine (Ma, Li, and Wang 2024; Xing et al. 2024), graph (Wang et al. 2024a; Behrouz and Hashemi 2024), recommendation (Liu et al. 2024a; Yang et al. 2024), and time series (Ahamed and Cheng 2024; Wang et al. 2024b; Patro and Agneeswaran 2024; Liang et al. 2024). A noteworthy line of research is to integrate Mamba and Transformer for the purpose of language modeling (Lieber et al. 2024; Park et al. 2024). A comparative study (Park et al. 2024) shows Mambaformer is effective in in-context learning tasks. Jamba (Lieber et al. 2024) is a production-grade attention-SSM hybrid model with 52B total available parameters for long context modeling. Different from the above hybrid models in text data, we first propose a hybrid Mamba-Transformer architecture SST in time series.

3 Preliminaries

Problem Statement

In the long-short range time series forecasting, historical time series with a look-back window $\mathcal{L} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L) \in \mathbb{R}^{L \times M}$ with length L are given, where each $\mathbf{x}_t \in \mathbb{R}^M$ at time step t is with M variates. Long-range time series \mathcal{L} denotes the full range of look-back window $\mathcal{L}[:,]$, and short-range time series $\mathcal{S} \in \mathbb{R}^{S \times M}$ denotes the partial latest range $\mathcal{L}[-S :,]$, $S < L$. We aim to forecast F future values $\mathcal{F} = (\mathbf{x}_{L+1}, \mathbf{x}_{L+2}, \dots, \mathbf{x}_{L+F}) \in \mathbb{R}^{F \times M}$ with length F .

State Space Models

The State Space Model (SSM) is a class of sequence modeling frameworks that are broadly related to RNNs, and CNNs, and classical state space models (Gu et al. 2021). They are inspired by a continuous system that maps an input function $x(t) \in \mathbb{R}$ to an output function $y(t) \in \mathbb{R}$ through an implicit latent state $h(t) \in \mathbb{R}^N$ as follows:

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

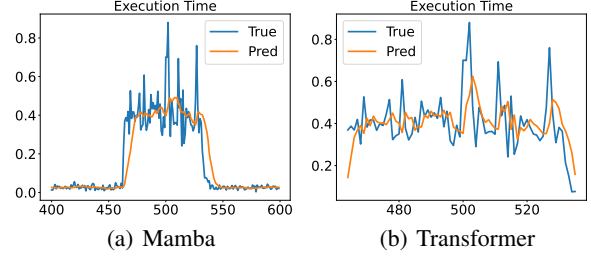


Figure 3: Preliminary results show that Mamba is able to extract long-term global patterns while Transformer has potential to capture short-term local variations.

where $\mathbf{A} \in \mathbb{R}^{N \times N}$, $\mathbf{B} \in \mathbb{R}^{N \times 1}$, and $\mathbf{C} \in \mathbb{R}^{1 \times N}$ are learnable matrices. SSM can be discretized from continuous signal into discrete sequences by a step size Δ as follows:

$$h_t = \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t, y = \mathbf{C}h_t \quad (2)$$

The discrete parameters $(\bar{\mathbf{A}}, \bar{\mathbf{B}})$ can be obtained from continuous parameters $(\Delta, \mathbf{A}, \mathbf{B})$ through a discretization rule, such as zero-order hold (ZOH) rule $\bar{\mathbf{A}} = \exp(\Delta\mathbf{A})$, $\bar{\mathbf{B}} = \exp(\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I}) \cdot \Delta\mathbf{B}$. After discretization, the model can be computed in two ways, either as a linear recurrence for inference as shown in Equation 2, or as a global convolution for training as the following Equation 3 where $\bar{\mathbf{K}}$ is a convolution kernel:

$$\bar{\mathbf{K}} = (\bar{\mathbf{C}}\bar{\mathbf{B}}, \bar{\mathbf{C}}\bar{\mathbf{A}}\bar{\mathbf{B}}, \dots, \bar{\mathbf{C}}\bar{\mathbf{A}}^{k-1}\bar{\mathbf{B}}, \dots), y = x * \bar{\mathbf{K}} \quad (3)$$

S4 is a structured SSM where the specialized Hippo (Gu et al. 2020) structure is imposed on the matrix \mathbf{A} to capture long-range dependency. Building upon S4, Mamba (Gu and Dao 2023) incorporates a selective SSM to propagate or forget information along the sequence, and a hardware-aware algorithm for efficient implementation.

Motivating Examples for Mamba and Transformer

Mamba and Transformer process data in different ways. **Mamba** parameterizes the SSM's parameters based on inputs, which enables it selectively focus on or ignore particular inputs. With a RNN-like hidden state, Mamba selectively stores global patterns into the hidden state and forgets irregular fluctuations. It motivates us to adopt Mamba to extract global patterns in time series. The preliminary result in Figure 3(a) demonstrates Mamba can retain repeated up-and-down patterns in long range. Furthermore, linear complexity allows Mamba to easily scale to long-range time series. **Transformer**, instead of maintaining a hidden state, adopts attention mechanism to directly access to past tokens and acquire dependencies. Such straightforward message-passing method facilitates depicting variations of data. The previous literature (Zeng et al. 2023) indicates Transformer easily adapts to sudden change noises, which aligns with our intuition. It motivates us to leverage Transformer to depict local variations. Figure 3(b) shows Transformer has potential to capture irregular fluctuations. However, two shortcomings exist for Transformer: (1) the missing local inductive bias hinders its ability to further capture local variations; (2) quadratic complexity of vanilla attention limits its scalability. We will later address the two issues with a local window.

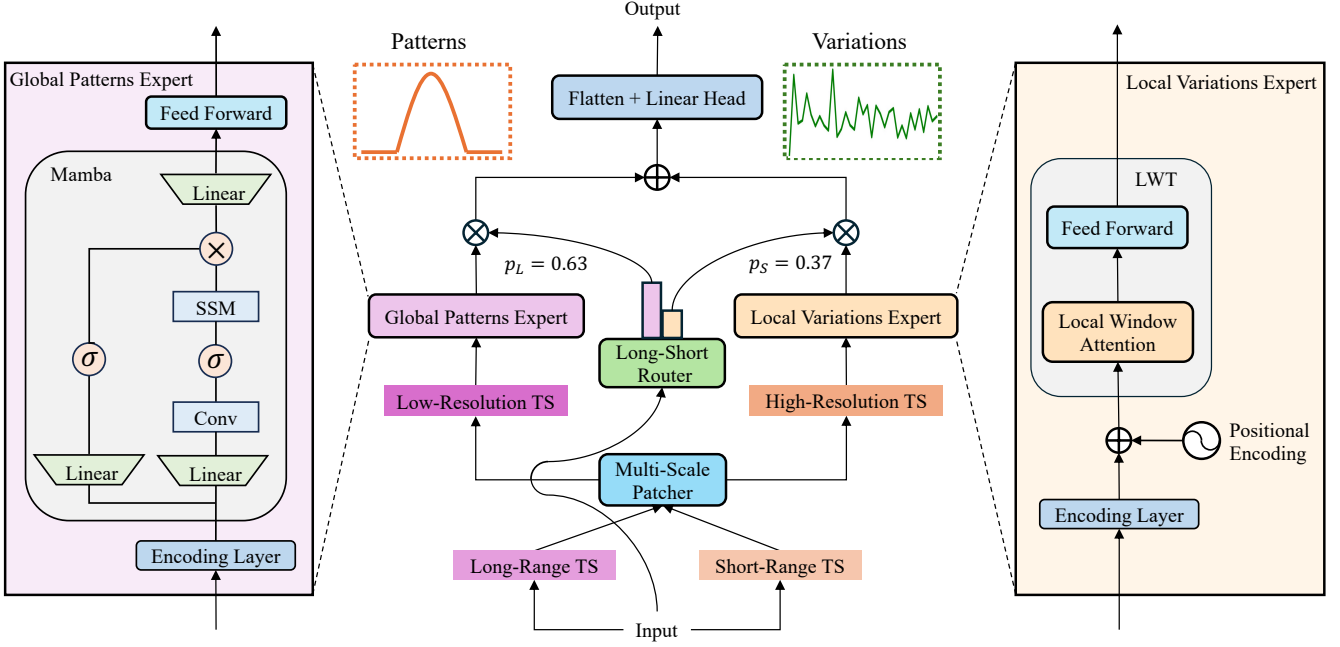


Figure 4: The overview of the SST (STATE SPACE TRANSFORMER). The multi-scale patcher transforms input TS in different resolutions according to ranges. The Mamba is dedicated for long-range TS in a low resolution as a global patterns expert. The LWT is responsible for short-range TS in a high resolution as a local variations expert. The long-short router adaptively learns the contributions of the two experts.

4 Methodology

As shown in Figure 8, SST (STATE SPACE TRANSFORMER) includes four modules: multi-scale patcher, global patterns expert, local variations expert, and long-short router. Multi-scale patcher transforms input time series (TS) into different resolutions according to ranges. Based on multi-scale TS, a global pattern expert is dedicated to finding long-term patterns in a low-resolution TS, and a local variations expert aims to capture short-term variations in a high-resolution TS. Finally, a long-short router dynamically learns contributions of the two experts.

Multi-Scale Resolutions

Patching. As shown in Figure 2, global patterns emerge when viewed at a broader-scale granularity, while local variations become clearer when examined at a finer-scale granularity. Consequently, we propose a multi-scale patcher to differentiate long- and short-range TS by providing them with distinct resolutions. i.e., low resolution for long-range TS and high resolution for short-range TS. In detail, the patcher modifies resolutions by aggregating a series of time steps into patches and independently operates patching for individual channels (Nie et al. 2023). Formally, input TS $\mathcal{L} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L) \in \mathbb{R}^{L \times M}$, $\mathbf{x}_t \in \mathbb{R}^{1 \times M}$ is divided into M univariate time series $\mathcal{L} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(M)}) \in \mathbb{R}^{L \times M}$, $\mathbf{x}^{(i)} \in \mathbb{R}^{L \times 1}$. The patching process involves two factors: the patch length P and the stride length Str (the interval between the end point of two consecutive patches). Accordingly, the number of patches is $N = \lfloor \frac{L-P}{Str} \rfloor + 1$, and the patcher for each variate $\mathbf{x}^{(i)}$ generates a sequence

of patches $\mathbf{x}_p^{(i)} \in \mathbb{R}^{N \times P}$, named patches time series (PTS). Note that for unpatched TS $\mathbf{x}^{(i)} \in \mathbb{R}^{L \times 1}$, L is the sequence dimension and 1 denotes the variate dimension. Correspondingly, for PTS $\mathbf{x}_p^{(i)} \in \mathbb{R}^{N \times P}$, N is the new sequence dimension and P is the new variate dimension.

Resolution. Intuitively, larger patches P , longer stride Str , and accordingly less number of patches N indicates a low resolution. We adopt such setting to generate a low-resolution TS for long range. It allows SST focus on modeling long-term global patterns and ignoring small fluctuation. By contrast, smaller patches and shorter stride imply a high resolution. It enables SST to depict short-term nuances. Although the patching is becoming popular in TS community, no existing work tries to quantify resolutions of PTS. To mitigate the gap, we define a new metric as follows:

Definition 1. *PTS (Patched Time Series) Resolution.* Let P , Str , and N denote patch length, stride length, and number of patches for a patched time series. The resolution of a PTS is defined as $R_{PTS} = \frac{\sqrt{P}}{Str}$.

The definition of PTS resolution takes two factors into account. First, PTS resolution aims to quantify the relative granularity of a PTS compared to its unpatched one. Second, PTS resolution describes the amount of temporal information of a PTS, in both the sequence dimension and variate dimension. Similar to the definition of the image resolution (Boellaard et al. 2004) equaling to $|width| \times |height|$, we multiply the two dimensions of PTS, but with a penalized root at variate dimension, i.e., $|sequence| \times \sqrt{|variate|}$ because we focus more on the sequence (temporal) dimension

of PTS. For example, after an univariate TS $\mathbf{x}^{(i)} \in \mathbb{R}^{L \times 1}$ is patched into a PTS $\mathbf{x}_p^{(i)} \in \mathbb{R}^{N \times P}$ where N is at the new sequence dimension, and P is at the new variate dimension, the PTS resolution R_{PTS} is defined as:

$$R_{PTS} = N\sqrt{P} = (\lfloor \frac{L-P}{Str} \rfloor + 1)\sqrt{P} \approx \frac{\sqrt{P}}{Str} \quad (4)$$

With the definition, an unpatched TS $\mathbf{x}^{(i)} \in \mathbb{R}^{1 \times L}$ can be regarded as a PTS with $R_{PTS} = 1$ (patch length $P = 1$ and stride length $Str = 1$). The multi-scale patcher processes long-range TS with a low R_{PTS}^L , and obtains $\mathbf{x}_{pL}^{(i)} \in \mathbb{R}^{N_L \times P_L}$. Conversely, the patcher handle short-range TS with a relatively high R_{PTS}^S , and outputs $\mathbf{x}_{pS}^{(i)} \in \mathbb{R}^{N_S \times P_S}$.

Hybrid Mamba-Transformer Experts

Inspired by Mixture-of-Experts (MoEs) (Jacobs et al. 1991; Shazeer et al. 2017; Fedus, Zoph, and Shazeer 2022), we introduce a hybrid Mamba-Transformer experts architecture. Unlike ambiguous roles in traditional MoEs models, we assign global patterns and local variations roles to two experts. **Global Patterns Expert.** Mamba achieves impressive performance on tasks requiring scaling to long-range sequences as it introduces a selective mechanism to remember relevant patterns and filter out irrelevant noises indefinitely. Consequently, we incorporate the Mamba block as an expert to extract long-term patterns and filter out small variations in long-range TS. As shown in Figure 8, the global patterns expert encodes long-range PTS $\mathbf{x}_{pL}^{(i)} \in \mathbb{R}^{N_L \times P_L}$ into high-dimension space $\mathbf{x}_L^{(i)} \in \mathbb{R}^{N_L \times D}$ in the encoding layer, and a Mamba block is followed. Mamba takes an input $\mathbf{x}_L^{(i)}$ and expands the dimension by two input linear projections. For one projection, Mamba processes the expanded embedding through a convolution and a SiLU activation before feeding into the SSM. The core discretized SSM module is able to select input-dependent patterns and filter out irrelevant variations. The other projection followed by SiLU activation, as a residual connection, is combined with output of the SSM module through a multiplicative gate. Finally, Mamba delivers output $\mathbf{z}_L^{(i)} \in \mathbb{R}^{N_L \times D}$ through an output linear projection. Note that we do not need a positional embedding typically existing in Transformer as Mamba’s recurrence mechanism (Gu et al. 2021) naturally encodes positions.

Local Variations Expert. Transformer is potential to capture variations as discussed in the Section 3, but lack of locality inductive bias and quadratic complexity impacts its adoption. Therefore, we propose local window Transformer (LWT) with enhanced locality-awareness capabilities to capture local variations in short range. Figure 8 shows the local variations expert first projects short-range PTS $\mathbf{x}_{pS}^{(i)}$ and positional information into an embedding $\mathbf{x}_S^{(i)} \in \mathbb{R}^{N_S \times D}$. The embedding is fed into LWT where the local window attention forces each token only attends to surrounding tokens within the window. In a formal way, for a fixed window size w , each token only attends to $\frac{1}{2}w$ surrounding tokens on two sides. After input embedding $\mathbf{x}_S^{(i)}$ is projected into query Q ,

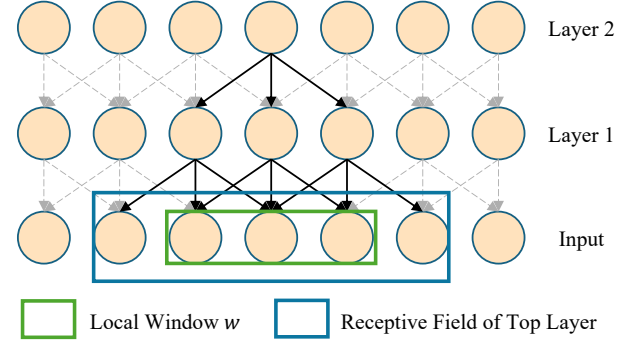


Figure 5: LWT employs a fixed-size w window to force each token only attend to local tokens within the window. By stacking multiple layers, the upper attention layer l can aggregate information from the lower layer and obtain a large receptive field $l \times w$.

key K , and value V in each head, LWT calculates attention scores between tokens within the w window size as follows:

$$Attention(Q_w, K_w, V_w) = softmax(\frac{Q_w K_w^T}{\sqrt{d_k}}) V_w \quad (5)$$

where Q_w , K_w , and V_w denote query, key, and value vector within the local window, $\frac{1}{\sqrt{d_k}}$ is used to avoid extremely small gradients (Vaswani et al. 2017) and d_k is the dimension of V_w . The output of multiple heads are concatenated and then projected back into the D -dimension space. We denote output of LWT as $\mathbf{z}_S^{(i)} \in \mathbb{R}^{N_S \times D}$. Figure 5 illustrates the mechanism of local window attention. Despite its strong local inductive bias, LWT maintains an extensive receptive field. By stacking multiple layers, the upper layers gain access to all input locations, enabling the construction of representations that integrate information across the entire input, similar to the capabilities of CNNs (Wu et al. 2019). The LWT is able to increase the receptive field by stacking multiple layers, and the respective field of layer l is $l \times w$. Moreover, the computation complexity of this pattern reduce from $O(S^2)$ to $O(w \times S)$ on the length of short-range TS (not considering the patching).

Mixture of Experts

Long-Short Router. In the MoE community, router (Fedus, Zoph, and Shazeer 2022) is a key module which primarily directs tokens to only a subset of experts for saving computational cost. Different from the path-assigning role of traditional MoE routers, the proposed long-short router is capable of learning the relative contributions of the two specialized experts, adaptively integrating long- and short-range TS representations. Formally, the router projects input TS $\mathcal{L} \in \mathbb{R}^{L \times M}$ into the D -dimension space, subsequently transformed by a flatten layer into a vector \mathbf{z}_R . A linear layer with a softmax function is followed to output two values $p_L, p_S \in (0, 1)$, indicating two weights of the global patterns and the local variations experts.

Forecasting Module. The forecasting module flattens long-range embedding $\mathbf{z}_L^{(i)}$ and short-range embedding $\mathbf{z}_S^{(i)}$ into a single-row vector and concatenates them with respective weights p_L and p_S . The resulting long-short range fu-

Table 1: Multivariate time series forecasting results on seven datasets. The forecast length $F \in \{96, 192, 336, 720\}$. The best results are in bold and the second best results are underlined.

Models	SST		S-Mamba		TimeMachine		iTransformer		RLinear		PatchTST		Crossformer		TimesNet		Dlinear		FEDformer		
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	
ETTh1	96	<u>0.381</u>	0.405	0.392	0.390	0.389	0.402	0.386	0.405	0.386	<u>0.395</u>	0.414	0.419	0.423	0.448	0.384	0.402	0.386	0.400	0.376	0.419
	192	0.430	0.434	0.449	0.439	0.435	0.440	0.441	0.436	0.437	0.424	0.460	0.445	0.450	0.471	0.474	<u>0.429</u>	0.437	0.432	0.420	0.448
	336	0.443	0.446	0.467	0.481	<u>0.450</u>	0.448	0.487	<u>0.458</u>	0.479	0.446	0.501	0.466	0.570	0.546	0.491	0.469	0.481	0.459	0.459	0.465
	720	0.502	0.501	0.475	<u>0.468</u>	<u>0.480</u>	0.465	0.503	0.491	0.481	0.470	0.500	0.488	0.653	0.621	0.521	0.500	0.519	0.516	0.506	0.507
ETTh2	96	<u>0.291</u>	<u>0.346</u>	0.292	0.357	0.230	0.349	0.297	0.349	0.288	0.338	0.302	0.348	0.745	0.584	0.340	0.374	0.333	0.387	0.358	0.397
	192	0.369	<u>0.397</u>	0.380	0.402	<u>0.371</u>	0.400	0.380	0.400	0.374	0.390	0.388	0.400	0.877	0.656	0.402	0.414	0.477	0.476	0.429	0.439
	336	0.374	0.414	<u>0.391</u>	<u>0.420</u>	0.402	0.449	0.428	0.432	0.415	0.426	0.426	0.433	1.043	0.731	0.452	0.452	0.594	0.541	0.496	0.487
	720	0.419	<u>0.447</u>	0.437	0.455	0.425	0.438	0.427	<u>0.445</u>	<u>0.420</u>	0.440	0.431	0.446	1.104	0.763	0.462	0.468	0.831	0.657	0.463	0.474
ETTm1	96	0.298	0.355	<u>0.311</u>	0.380	0.312	0.371	0.334	0.368	0.355	0.376	0.329	<u>0.367</u>	0.404	0.426	0.338	0.375	0.345	0.372	0.379	0.419
	192	0.347	0.381	0.389	0.419	<u>0.365</u>	0.409	0.377	0.391	0.391	0.392	0.367	<u>0.385</u>	0.450	0.451	0.374	0.387	0.380	0.389	0.426	0.441
	336	0.374	0.397	0.401	0.417	0.421	0.410	0.426	0.420	0.424	0.415	<u>0.399</u>	<u>0.410</u>	0.532	0.515	0.410	0.411	0.413	0.413	0.445	0.459
	720	0.429	0.428	0.488	0.476	0.496	<u>0.437</u>	0.491	0.459	0.487	0.450	<u>0.454</u>	<u>0.439</u>	0.666	0.589	0.478	0.450	0.474	0.453	0.543	0.490
ETTm2	96	<u>0.176</u>	<u>0.264</u>	0.191	0.301	0.185	0.290	0.180	<u>0.264</u>	0.182	0.265	0.175	0.259	0.287	0.366	0.187	0.267	0.193	0.292	0.203	0.287
	192	0.231	<u>0.303</u>	0.253	0.312	0.292	0.309	0.250	0.309	0.246	0.304	<u>0.241</u>	0.302	0.414	0.492	0.249	0.309	0.284	0.362	0.269	0.328
	336	0.290	0.339	<u>0.298</u>	<u>0.342</u>	0.321	0.367	0.311	0.348	0.307	<u>0.342</u>	0.305	0.343	0.597	0.542	0.321	0.351	0.369	0.427	0.325	0.366
	720	0.388	0.398	0.409	0.407	<u>0.401</u>	0.400	0.412	0.407	0.407	0.398	0.402	0.400	1.730	1.042	0.408	0.403	0.554	0.522	0.421	0.415
Weather	96	0.153	0.205	0.169	0.221	0.174	0.218	0.174	<u>0.214</u>	0.192	0.232	0.177	0.218	<u>0.158</u>	0.230	0.172	0.220	0.196	0.255	0.217	0.296
	192	0.196	0.244	0.205	<u>0.248</u>	<u>0.200</u>	0.258	0.221	0.254	0.240	0.271	0.225	0.259	0.206	0.277	0.219	0.261	0.237	0.296	0.276	0.336
	336	0.246	0.283	0.288	0.299	0.280	0.299	0.278	<u>0.296</u>	0.292	0.307	0.278	0.297	<u>0.272</u>	0.335	0.280	0.306	0.283	0.335	0.339	0.380
	720	0.314	0.334	<u>0.335</u>	0.369	0.352	0.359	0.358	<u>0.347</u>	0.364	0.353	0.354	0.348	0.398	0.418	0.365	0.359	0.345	0.381	0.403	0.428
ECL	96	0.141	0.239	0.157	0.255	0.156	0.240	<u>0.148</u>	<u>0.240</u>	0.201	0.281	0.181	0.270	0.219	0.314	0.168	0.272	0.197	0.282	0.193	0.308
	192	0.159	<u>0.255</u>	0.188	0.271	<u>0.161</u>	0.268	0.162	0.253	0.201	0.283	0.188	0.274	0.231	0.322	0.184	0.289	0.196	0.285	0.201	0.315
	336	0.171	0.268	0.192	0.275	0.195	0.272	<u>0.178</u>	<u>0.269</u>	0.215	0.298	0.204	0.293	0.246	0.337	0.198	0.300	0.209	0.301	0.214	0.329
	720	0.208	0.300	0.241	0.339	0.231	0.307	0.225	0.317	0.257	0.331	0.246	0.324	0.280	0.363	0.220	0.320	0.245	0.333	0.246	0.355
Traffic	96	0.367	0.257	0.401	0.259	0.398	0.274	<u>0.395</u>	0.268	0.649	0.389	0.462	0.295	0.522	0.290	0.593	0.321	0.650	0.396	0.587	0.366
	192	<u>0.385</u>	0.266	0.389	0.294	0.393	0.282	0.417	<u>0.276</u>	0.601	0.366	0.366	0.296	0.530	0.293	0.617	0.336	0.598	0.370	0.604	0.373
	336	0.401	0.275	0.427	0.296	0.443	0.368	<u>0.433</u>	<u>0.283</u>	0.609	0.369	0.482	0.304	0.558	0.305	0.629	0.336	0.605	0.373	0.621	0.383
	720	0.445	0.302	0.473	0.347	0.470	<u>0.309</u>	<u>0.467</u>	0.302	0.647	0.387	0.514	0.322	0.589	0.328	0.640	0.350	0.645	0.394	0.626	0.382

sion representation $\mathbf{z}_{LS}^i \in \mathbb{R}^{(N_S+N_L)LD}$ integrates long-term global patterns and short-term local variations information. Finally, a linear head is employed to forecast $\hat{\mathbf{x}}^{(i)} = \{\hat{x}_{L+1}, \hat{x}_{L+2}, \dots, \hat{x}_{L+F}\} \in \mathbb{R}^{F \times 1}$ for individual variate i .

Linear Complexity Analysis.

SST maintains a linear complexity $O(L)$ on input TS length L or $O(\frac{L}{N_L} + \frac{wS}{N_S})$. The high efficiency allows SST to avoid prohibitive computational complexity and memory usage, scaling to long time series. In detail, the complexity SST comes from two parts: Mamba and LWT. Mamba utilizes kernel fusion to reduce the amount of memory IOs and scan operation for parallel computation (Gu and Dao 2023) in the training, and adopts RNN-like mechanism in the inference, leading to a linear complexity $O(L)$ on long-range TS \mathcal{L} . Given self-attention operation occupies $O(w^2)$ complexity within a local window with fixed-size w each step, LWT consumes $O(w^2 * \frac{S}{w})$ complexity on short-range TS \mathcal{S} . Considering the use of multi-scale patcher, the total complexity of SST is $O(\frac{L}{N_L} + \frac{wS}{N_S})$. Note that w , N_S , and N_L are constant, and S is linear related to and smaller than L .

5 Experiments

Experimental Setup

Datasets. To evaluate the proposed SST (STATE SPACE TRANSFORMER), we adopt seven popular real-world datasets (Liu et al. 2024b), including ETTh1&ETTh2, ETTm1&ETTm2, Weather, ECL, and Traffic. The description of datasets are in the Appendix.

Baselines and Metrics. We compare SST with time series forecasting methods within three years, including S-Mamba (Wang et al. 2024b), TimeMachine (Ahmed and Cheng 2024), iTransformer (Liu et al. 2024b), RLinear (Li

et al. 2023), PatchTST (Nie et al. 2023), Crossformer (Zhang and Yan 2022), TimesNet (Wu et al. 2022), Dlinear (Zeng et al. 2023), FEDformerr (Zhou et al. 2022). Note that it is unfair to include LLMs approaches as they are pre-trained on a large corpus of data. To assess the performance of time series forecasters, we adopt two widely-used metrics MSE and MAE (Liu et al. 2024b). The lower MSE and MAE indicate more accurate forecasting results.

Experimental Setting. We use low-resolution $R_{PTS} = 0.43$ ($P_L = 48$ and $Str_L = 16$) for long-range time series and high-resolution $R_{PTS} = 0.5$ ($P_L = 16$ and $Str_L = 8$) for short range. We set the look-back window length $L = 2S = 672$, and the future values length $F \in \{96, 192, 336, 720\}$. The experimental setting of baselines follows the latest SOTA iTransformer (Liu et al. 2024b).

Time Series Forecasting Results. We conduct multivariate time series forecasting experiments and results are shown in Table 1. It shows that SST achieves superior performance compared to baselines, including Mamba-based and Transformer based methods, across all real-world datasets. For example, compared to the S-Mamba, Mamba-based forecaster, and iTransformer, Transformer-based forecaster, SST reduces MSE error by 13.75% and 14.45% for the longest forecast length $F = 720$ on the ETTm1 dataset. The impressive performance benefits from the reasonable design of SST. Based on different granularity, SST employs a Mamba-based global patterns expert and a LWT-based local variations expert for long- and short-range time series. A following long-short router adaptively fuses long- and short-range dependencies, thus facilitating time series forecasting.

Ablation Studies

We conduct ablation study by excluding key components of SST and comparing SST with Mambaformer family.

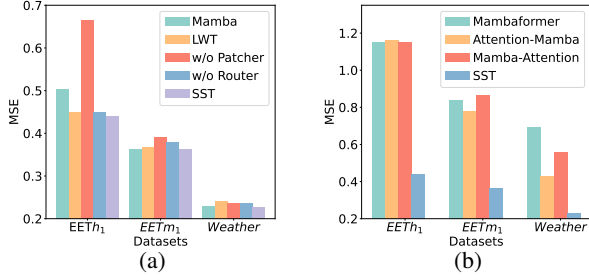


Figure 6: Ablation study on three datasets for (a) exclusion of key components; (b) comparison with Mambaformer family. The values are averaged for multiple forecasting lengths $F \in \{96, 192, 336, 720\}$.

Exclusion of Key Components. To verify the effectiveness of each key component of SST, including global patterns expert, local variations expert, multi-scale patcher, and long-short router, we design an ablation study on variants: *Mamba*, *LWT*, *w/o Patcher* (SST without the Patcher), *w/o Router* (SST without the Router) as shown in Figure 6(a). Accordingly, we have the following observations: (1) SST achieves impressive performance compared to *Mamba* and *LWT*. The reason is that SST can integrate strengths of Mamba and Transformer, thus effectively capturing global patterns in long range and local variations in short range. It demonstrates the effectiveness of hybrid Mamba-Transformer architecture in SST. (2) Compared to *w/o Patcher* and *w/o Router*, SST obtains superior performance. It is because the multi-scale patcher can patch long- and short-range TS in distinct resolutions, thus facilitating feature extraction of two experts; the long-short router can dynamically learn contributions of the two experts. It verifies reasonable design of multi-scale framework and MoE architecture in SST.

Comparison with Mambaformer Family. Given SST is the first hybrid Mamba-Transformer model in time series, we are interested in if the method to integrate Mamba and Transformer in SST is the optimal. To answer the question, we adapt the Mambaformer family (Park et al. 2024), which is originally developed for language modeling, to time series. Mambaformer family attempts to combine strengths of Mamba and Transformer by directly interleaving Mamba and attention layers. In particular, the Mambaformer family consists of: *Mambaformer* where a pre-processing Mamba block is followed by interleaved Attention-Mamba layers, *Attention-Mamba* where interleaved Attention-Mamba layers are utilized, *Mamba-Attention* where interleaved Mamba-Attention layers are employed. More details are shown in Appendix. The results are displayed in Figure 6(b). It shows that Mambaformer family cannot reach desirable performance compared to SST. It implies that directly interleaving Mamba and attention cannot fully unleash potential of hybrid architectures. The more careful and specific design for time series like SST is necessary.

Memory and Speed Analysis

To check the efficiency of SST in practice, we conduct memory and speed analysis. As shown in Figure 7, we depict figures where consumed memory and the elapsed time

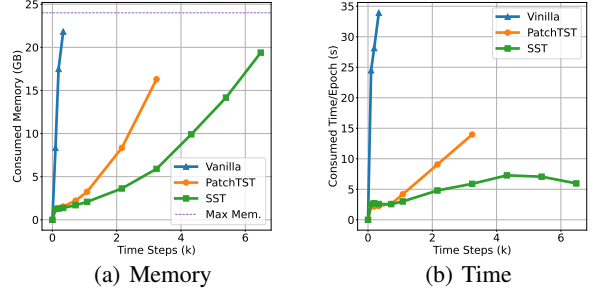


Figure 7: Comparison of consumed memory and the elapsed time of each epoch for vanilla Transformer, PatchTST, and SST. The OOM (out-of-memory) issues emerge for vanilla transformer, PatchTST, and SST when the length of input is 336, 3240, and 6480 time steps.

each epoch vary with the length of input TS L . We compare SST with PatchTST (Nie et al. 2023) and vanilla Transformer (Vaswani et al. 2017). Note that we do not compare iTransformer because its attention operates on variate dimension instead of sequence dimension. All the computations were performed on 24 GB NVIDIA RTX A5000 GPU at Ubuntu 20.04.6 LTS. From the figure, we observe SST is efficient and has promising scalability on time steps. SST is able to scale linearly to 6k time steps. The impressive scalability stems from the fact that Mamba implement a hardware-aware algorithm, and LWT leverages a fixed-size sliding window to operate attentions. In contrast, vanilla Transformer struggles in quadratic complexity, significantly preventing it from attending to long time series. In the experiments, vanilla Transformer can only attend to maximum 336 time steps. It cannot continue increasing input length due to the OOM (Out-Of-Memory) issue. PatchTST uses patch techniques to reduce complexity by a factor of stride, thus alleviating the issue to a degree. However, it still fall short of scalability and can only scale to 3k time steps, much lower than 6k time steps of SST. The memory and speed analysis demonstrate efficiency of SST in practice.

6 Conclusion and Future Work

In this paper, we point out that time series can be decomposed into global patterns and local variations based on ranges. Global patterns should be extracted from long range, while local variations are more effectively captured in short range. To this end, we introduce a multi-scale hybrid Mamba-Transformer framework SST. Specifically, Mamba, serving as a global patterns expert, focuses on extracting long-term patterns in low resolution. Conversely, LWT, as a local variations expert, addresses subtle nuances in short-range time series in high resolution. To adaptively integrate the two experts, a long-short router dynamically learn contributions of Mamba and LWT. SST exhibits high efficiency with scaling linearly $O(L)$ with time series length L . Comprehensive experiments across seven real-world datasets demonstrate SST achieves superior forecasting performance while maintaining low memory usage and high computation speed. The future work includes exploring hybrid Mamba-Transformer architectures in other time series analysis, such as classification and anomaly detection.

References

- Abhishek, K.; Singh, M. P.; Ghosh, S.; and Anand, A. 2012. Weather forecasting model using artificial neural network. *Procedia Technology*, 4: 311–318.
- Ahamed, M. A.; and Cheng, Q. 2024. TimeMachine: A Time Series is Worth 4 Mambas for Long-term Forecasting. *arXiv preprint arXiv:2403.09898*.
- Behrouz, A.; and Hashemi, F. 2024. Graph Mamba: Towards Learning on Graphs with State Space Models. *arXiv preprint arXiv:2402.08678*.
- Boellaard, R.; Krak, N. C.; Hoekstra, O. S.; and Lam-mertsma, A. A. 2004. Effects of noise, image resolution, and ROI definition on the accuracy of standard uptake values: a simulation study. *Journal of Nuclear Medicine*, 45(9): 1519–1527.
- Bontempi, G.; Ben Taieb, S.; and Le Borgne, Y.-A. 2013. Machine learning strategies for time series forecasting. *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures 2*, 62–77.
- Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.
- Chang, C.; Peng, W.-C.; and Chen, T.-F. 2023. Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms. *arXiv preprint arXiv:2308.08469*.
- Cheng, X.; Zhang, R.; Zhou, J.; and Xu, W. 2018. Deep-transport: Learning spatial-temporal dependency for traffic condition forecasting. In *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.
- Cruz-Camacho, E.; Brown, K. A.; Wang, X.; Xu, X.; Shu, K.; Lan, Z.; Ross, R. B.; and Carothers, C. D. 2023. Hybrid PDES Simulation of HPC Networks Using Zombie Packets. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 128–132.
- Dao, T.; and Gu, A. 2024. Transformers are SSMS: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.
- Fathi, M.; Pilault, J.; Bacon, P.-L.; Pal, C.; Firat, O.; and Goroshin, R. 2023. Block-state transformer. *arXiv preprint arXiv:2306.09539*.
- Fedus, W.; Zoph, B.; and Shazeer, N. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120): 1–39.
- Gruver, N.; Finzi, M.; Qiu, S.; and Wilson, A. G. 2024. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36.
- Gu, A.; and Dao, T. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.
- Gu, A.; Dao, T.; Ermon, S.; Rudra, A.; and Ré, C. 2020. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487.
- Gu, A.; Goel, K.; and Re, C. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*.
- Gu, A.; Johnson, I.; Goel, K.; Saab, K.; Dao, T.; Rudra, A.; and Ré, C. 2021. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34: 572–585.
- Hewamalage, H.; Bergmeir, C.; and Bandara, K. 2021. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1): 388–427.
- Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1): 79–87.
- Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J. Y.; Shi, X.; Chen, P.-Y.; Liang, Y.; Li, Y.-F.; Pan, S.; et al. 2023. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*.
- Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32.
- Li, Z.; Qi, S.; Li, Y.; and Xu, Z. 2023. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721*.
- Liang, A.; Jiang, X.; Sun, Y.; and Lu, C. 2024. Bi-Mamba4TS: Bidirectional Mamba for Time Series Forecasting. *arXiv preprint arXiv:2404.15772*.
- Lieber, O.; Lenz, B.; Bata, H.; Cohen, G.; Osin, J.; Dalmedigos, I.; Safahi, E.; Meirum, S.; Belinkov, Y.; Shalev-Shwartz, S.; Abend, O.; Alon, R.; Asida, T.; Bergman, A.; Glozman, R.; Gokhman, M.; Manevich, A.; Ratner, N.; Rozen, N.; Shwartz, E.; Zusman, M.; and Shoham, Y. 2024. Jamba: A Hybrid Transformer-Mamba Language Model. *arXiv:2403.19887*.
- Liu, C.; Lin, J.; Wang, J.; Liu, H.; and Caverlee, J. 2024a. Mamba4Rec: Towards Efficient Sequential Recommendation with Selective State Space Models. *arXiv preprint arXiv:2403.03900*.
- Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.
- Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024b. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.
- Ma, J.; Li, F.; and Wang, B. 2024. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *arXiv preprint arXiv:2401.04722*.

- Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*.
- Park, J.; Park, J.; Xiong, Z.; Lee, N.; Cho, J.; Oymak, S.; Lee, K.; and Papailiopoulos, D. 2024. Can Mamba Learn How to Learn? A Comparative Study on In-Context Learning Tasks. *arXiv preprint arXiv:2402.04248*.
- Patro, B. N.; and Agneeswaran, V. S. 2024. SiMBA: Simplified Mamba-Based Architecture for Vision and Multivariate Time series. *arXiv preprint arXiv:2403.15360*.
- Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training.
- Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.
- Sezer, O. B.; Gudelek, M. U.; and Ozbayoglu, A. M. 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90: 106181.
- Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.
- Tan, M.; Merrill, M. A.; Gupta, V.; Althoff, T.; and Hartvigsen, T. 2024. Are Language Models Actually Useful for Time Series Forecasting? *arXiv preprint arXiv:2406.16964*.
- Tang, Y.; Dong, P.; Tang, Z.; Chu, X.; and Liang, J. 2024. VMERN: Integrating Vision Mamba and LSTM for Efficient and Accurate Spatiotemporal Forecasting. *arXiv:2403.16536*.
- Tetko, I. V.; Livingstone, D. J.; and Luik, A. I. 1995. Neural network studies. 1. Comparison of overfitting and overtraining. *Journal of chemical information and computer sciences*, 35(5): 826–833.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Wang, C.; Tsepa, O.; Ma, J.; and Wang, B. 2024a. Graphmamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*.
- Wang, Z.; Kong, F.; Feng, S.; Wang, M.; Zhao, H.; Wang, D.; and Zhang, Y. 2024b. Is Mamba Effective for Time Series Forecasting? *arXiv preprint arXiv:2403.11144*.
- Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; and Sun, L. 2023. Transformers in Time Series: A Survey. In Elkind, E., ed., *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 6778–6786. International Joint Conferences on Artificial Intelligence Organization. Survey Track.
- Wu, F.; Fan, A.; Baeviski, A.; Dauphin, Y. N.; and Auli, M. 2019. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.
- Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The eleventh international conference on learning representations*.
- Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34: 22419–22430.
- Xing, Z.; Ye, T.; Yang, Y.; Liu, G.; and Zhu, L. 2024. Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation. *arXiv preprint arXiv:2401.13560*.
- Xu, X. 2023. Exploring Machine Learning Models with Spatial-Temporal Information for Interconnect Network Traffic Forecasting. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 56–57.
- Xu, X.; A. Brown, K.; Mallick, T.; Wang, X.; Cruz-Camacho, E.; B. Ross, R.; D. Carothers, C.; Lan, Z.; and Shu, K. 2024. Surrogate Modeling for HPC Application Iteration Times Forecasting with Network Features. In *Proceedings of the 38th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 93–97.
- Xu, X.; Wang, X.; Cruz-Camacho, E.; D. Carothers, C.; A. Brown, K.; B. Ross, R.; Lan, Z.; and Shu, K. 2023. Machine Learning for Interconnect Network Traffic Forecasting: Investigation and Exploitation. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 133–137.
- Yang, J.; Li, Y.; Zhao, J.; Wang, H.; Ma, M.; Ma, J.; Ren, Z.; Zhang, M.; Xin, X.; Chen, Z.; et al. 2024. Uncovering Selective State Space Model’s Capabilities in Lifelong Sequential Recommendation. *arXiv preprint arXiv:2403.16371*.
- Yao, H.; Tang, X.; Wei, H.; Zheng, G.; and Li, Z. 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 5668–5675.
- Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11121–11128.
- Zhang, J.; Zheng, S.; Cao, W.; Bian, J.; and Li, J. 2023. Warpformer: A Multi-scale Modeling Approach for Irregular Clinical Time Series. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3273–3285.
- Zhang, Y.; and Yan, J. 2022. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*.
- Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, 27268–27286. PMLR.

Zhu, L.; Liao, B.; Zhang, Q.; Wang, X.; Liu, W.; and Wang, X. 2024. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*.

7 Appendix

Datasets

The seven widely-used real-world datasets (Liu et al. 2024b) in the experiments, consisting of, ETTh1&ETTh2, ETTm1&ETTM2, Weather, ECL, and Traffic, are summarized in Table 2. The detailed description of the seven real-world datasets are as follows:

- ETT dataset (Zhou et al. 2021) includes data on seven factors related to electricity transformers, spanning from July 2016 to July 2018. It consists of four subsets: ETTh1 and ETTh2, recorded hourly, and ETTm1 and ETTm2, recorded every 15 minutes.
- Weather dataset (Wu et al. 2021) captures 21 meteorological factors, recorded every 10 minutes by the Weather Station at the Max Planck Institute for Biogeochemistry throughout 2020.
- ECL dataset (Wu et al. 2021) offers hourly electricity consumption data from 321 clients.
- Traffic dataset (Wu et al. 2021) contains hourly road occupancy rates collected by 862 sensors across the San Francisco Bay Area freeways, covering the period from January 2015 to December 2016.

Table 2: The statistics of seven time series datasets.

Datasets	Variates	Timestamps
ETTh1	7	17,420
ETTh2	7	17,420
ETTM1	7	69,680
ETTM2	7	69,680
Weather	21	52,696
ECL	321	26,304
Traffic	862	17,544

Implementation Details

All experiments are performed on 24 GB NVIDIA RTX A5000 GPU at Ubuntu 20.04.6 LTS. For SST, we set low resolution $R_{PTS} = 0.43$ ($P_L = 48$ and $Str_L = 16$) for long-range time series and high-resolution $R_{PTS} = 0.5$ ($P_L = 16$ and $Str_L = 8$) for short range. We set the length of long-range time series L as twice as the length of short-range time series S , i.e., $L = 2 * S$, to obtain global patterns in longer time series. For the length of short-range time series, we set $S = 336$ to be consistent with PatchTST (Nie et al. 2023) to inherit the merits of the patching. Following most previous work (Liu et al. 2024b; Nie et al. 2023; Zeng et al. 2023), we fix the forecasting length as $F \in \{96, 192, 336, 720\}$. For the length of local window w in LWT, we set $w = 7$ for ETTh1, ETTm1, ETTm2, and Weather, $w = 21$ for ETTh2, $w = 31$ for ECL and Traffic due to the hyperparameter tuning. The experimental setting of baselines either follows the latest SOTA iTransformer (Liu et al. 2024b) (if applicable) or based on configurations of the original paper. For the Mambaformer family in the ablation studies, we set input length as 336 like SST, and re-

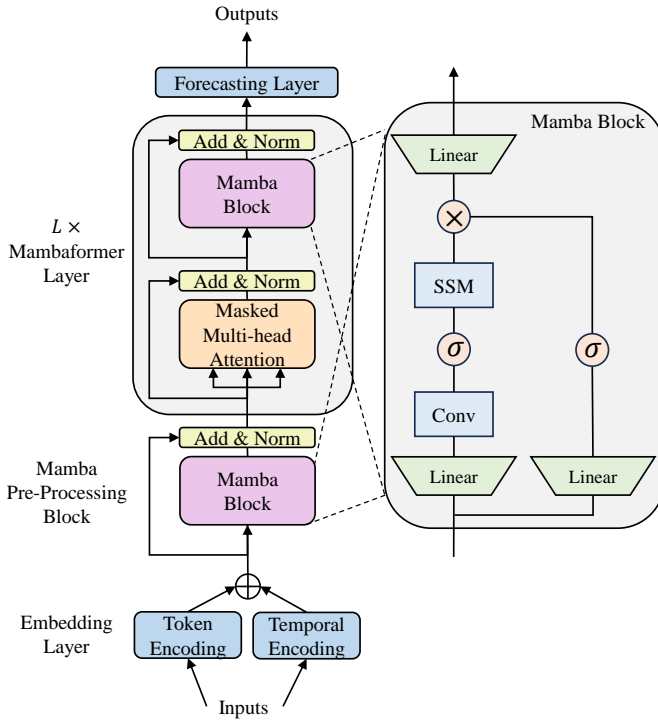


Figure 8: The overview of the Mambaformer.

port the average value of results across forecasting lengths $F \in \{96, 192, 336, 720\}$.

Mambaformer in the Ablation Studies

Recent findings show that SSMs and Transformers are complementary for language modeling (Lieber et al. 2024; Fathi et al. 2023; Park et al. 2024). Mambaformer (Park et al. 2024) is early work to combine Mamba and Transformer in text data where Mambaformer attempts to combine the advantages of Mamba and Transformer by directly interleaving Mamba and attention layer. Although Mambaformer has been proved effective for in-context learning tasks of text data, we are interested in if the observation is consistent in time series data given such directly stacking design of Mambaformer. Therefore, we adapt Mambaformer to time series data and compare Mambaformer family with SST in the ablation studies.

Here we introduce the details of Mambaformer. Following (Park et al. 2024), Mambaformer adopts a decoder-only style as GPT (Radford et al. 2018, 2019; Brown et al. 2020) family. As shown in Figure 8, Mambaformer includes an embedding layer to encode token and temporal information, a Mamba pre-processing layer to incorporate positional encoding, a core Mambaformer layer to capture long- and short-range time series dependencies, and a forecasting layer to output forecasts.

Embedding Layer

We utilize an embedding layer to map the low-dimension time series data into a high-dimensional space, including to token embedding and temporal embedding.

Token Embedding. To convert raw time series data into high-dimensional tokens, we utilize a one-dimensional convolutional layer as a token embedding module because it can retain local semantic information within the time series data (Chang, Peng, and Chen 2023).

Temporal Embedding. Besides numerical value itself in the sequence, temporal context information also provides informative clues, such as hierarchical timestamps (week, month and year) and agnostic timestamps (holidays and events) (Zeng et al. 2023). We employ a linear layer to embed temporal context information.

Formally, let $\mathbf{X} \in \mathbb{R}^{B \times L \times M}$ denote input sequences with batch size B , input length L , and input variate dimension M . $\mathbf{C} \in \mathbb{R}^{B \times L \times C}$ denotes the associated temporal context information, e.g. day-of-the-week and hour-of-the-day, with dimension C . The embedding layer can be expressed as follows:

$$\mathbf{E} = E_{token}(\mathbf{X}) + E_{tem}(\mathbf{C}) \quad (6)$$

where $\mathbf{E} \in \mathbb{R}^{B \times L \times D}$ is output embedding, D is the dimension of the embedding, E_{token} and E_{tem} denote token embedding layer and temporal embedding layer, respectively.

We do not need a positional embedding typically existing in Transformer model. Instead, a Mamba pre-processing block introduced in the next subsection is leveraged to internally incorporate positional information to the embedding.

Mamba Pre-Processing Layer

To endow the embedding with positional information, we pre-process the sequence by a Mamba block to internally embed order information of input tokens. Mamba can be regarded as a RNN where the hidden state h_t at current time t is updated by the hidden state h_{t-1} at previous time $t-1$ as shown in the Equation 2. Such recurrence mechanism to process tokens enables Mamba naturally consider order information of sequences. Therefore, unlike positional encoding being an essential component in Transformer, Mambaformer replace positional encoding by a Mamba pre-processing block. In a formal way, the Mamba pre-processing block can be expressed as follows:

$$\mathbf{H}_1 = Mamba(\mathbf{E}) \quad (7)$$

where $\mathbf{H}_1 \in \mathbb{R}^{B \times L \times D}$ denotes a mixing representation including token embedding, temporal embedding, and positional information.

Mambaformer Layer

The core Mambaformer layer interleaves Mamba layer and self-attention layer to attempt to integrate advantages of Mamba and Transformer to facilitate long-short range time series forecasting.

Attention Layer. To inherit impressive performance of depicting short-range time series dependencies in the transformer, we leverage masked multi-head attention layer to obtain correlations between tokens. In particular, each head $i = 1, 2, \dots, h$ in the attention layer transforms the embedding \mathbf{H}_1 into queries $\mathbf{Q}_i = \mathbf{H}_1 \mathbf{W}_i^Q$, keys $\mathbf{K}_i = \mathbf{H}_1 \mathbf{W}_i^K$, and values $\mathbf{V}_i = \mathbf{H}_1 \mathbf{W}_i^V$, where $\mathbf{W}_i^Q \in \mathbb{R}^{D \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{D \times d_k}$, and $\mathbf{W}_i^V \in \mathbb{R}^{D \times d_v}$ are learnable matrices. After-

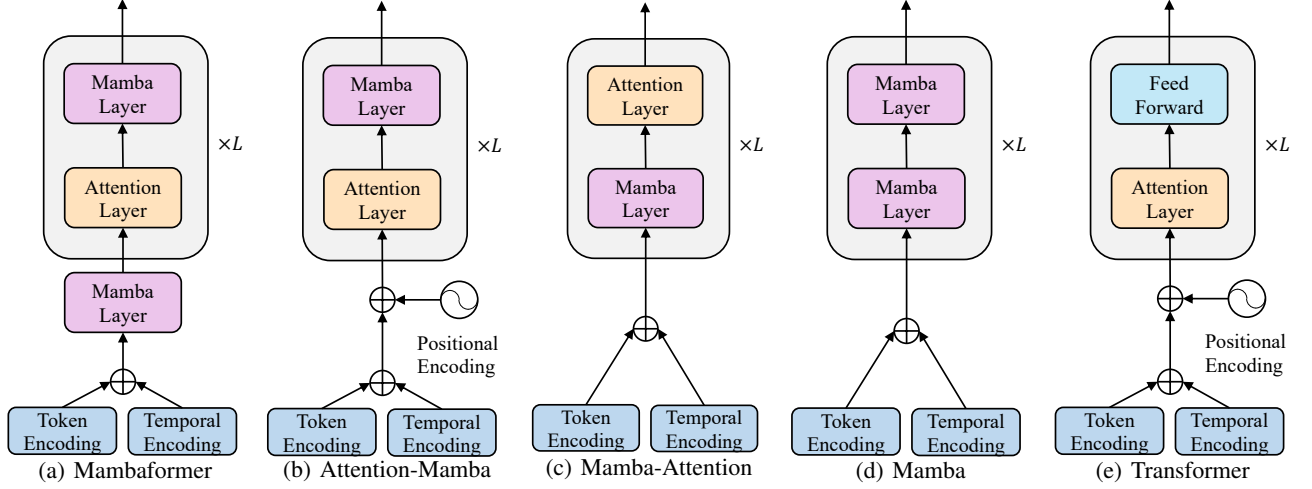


Figure 9: The architectures of Mambaformer family (including figure (a), figure (b), and figure (c)) and Mamba (figure (d)) and Transformer (figure (e)).

wards, a scaled dot-product attention is utilized:

$$\mathbf{O}_i = \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^T}{\sqrt{d_k}}\right) \mathbf{V}_i \quad (8)$$

where the outputs \mathbf{O}_i of each head are concatenated into a output vector \mathbf{O} with the embedding dimension hd_v . Following a learnable projection matrix $\mathbf{W}^O \in \mathbb{R}^{hd_v \times D}$, the output of attention layer $\mathbf{H}_2 = \mathbf{O} \mathbf{W}^O \in \mathbb{R}^{B \times L \times D}$. We adopt the masking mechanism to prevent positions from attending to subsequent positions, and set $d_k = d_v = D/h$ following vanilla Transformer setting (Vaswani et al. 2017). **Mamba Layer.** To overcome computational challenges of the Transformer and be beyond the performance of Transformer, we incorporate the Mamba layer into the model to enhance the capability of capturing long-range time series dependency. As shown in Figure 8, Mamba block is a sequence-sequence module with the same dimension of input and output. In particularly, Mamba takes an input \mathbf{H}_2 and expand the dimension by two input linear projection. For one projection, Mamba processes the expanded embedding through a convolution and SiLU activation before feeding into the SSM. The core discretized SSM module is able to select input-dependent knowledge and filter out irrelevant information. The other projection followed by SiLU activation, as a residual connection, is combined with the output of the SSM module through a multiplicative gate. Finally, Mamba delivers output $\mathbf{H}_3 \in \mathbb{R}^{B \times L \times D}$ through an output linear projection.

Forecasting Layer

At this layer, we obtain forecasting resulting by a linear layer to convert high-dimension embedding space into original dimension of time series data as follows:

$$\hat{\mathbf{X}} = \text{Linear}(\mathbf{H}_3) \quad (9)$$

where $\hat{\mathbf{X}} \in \mathbb{R}^{B \times L \times M}$ denotes forecasting results.

Mambaformer Family in the Ablation Studies

We introduce Mambaformer family by exploring more possible hybrid architectures of Mamba and Transformer. As shown in Figure 9, we interleave Mamba layer and attention layer in different orders and compare them with Mamba and vanilla Transformer. The architectures of Mambaformer family is as follows:

- **Mambaformer** utilizes a pre-processing Mamba block and Mambaformer layer without a positional encoding.
- **Attention-Mamba** leverages a Attention-Mamba layer where an attention layer is followed by a Mamba layer with a positional encoding.
- **Mamba-Attention** adopts a Mamba-Attention layer where a Mamba block layer is followed by an attention layer without a positional encoding.

For convenience of compression, we also depicts Mamba and vanilla Transformer in Figure 9.

- **Mamba** leverages two Mamba block as a layer.
- **Transformer** denotes a decoder-only vanilla Transformer architecture.

Positional encoding is optional in the above architectures because Mamba layer internally consider positional information while Transformer does not. For Mambaformer family, if a Mamba layer is before an attention layer, the model does not need a positional encoding; if not, the model needs a positional encoding.