HIPPO-SSM: Channel correlation enhanced state space models for multivariate time series forecasting.

Kirill Yarosh Dazhao Du
Bauman Moscow State Technical University; Hangzhou Innovation Institute of Beihang University
k.yarosh.internet.2023@gmail.com
dudazhao16@gmail.com

Abstract

Recent advances in multivariate time series forecasting have been driven by linear-based, transformer-based, and convolution-based models. Transformer-based architectures have gained prominence due to their effectiveness in temporal and cross-channel mixing. Mamba, a state-space model, has emerged with robust sequence and feature mixing capabilities, but the suitability of its vanilla design for time series forecasting is an open question, particularly due to its inability to handle cross-channel dependencies. Capturing these dependencies is crucial for improving the performance of multivariate time series predictions. Recent findings suggest that self-attention outperforms simpler mechanisms like MLP in capturing these dependencies, which is counterintuitive, as MLP should theoretically be able to capture both correlations and dependencies. Relations and irrelevancies, potentially leading to neutral or improved performance, are explored. By diving into the self-attention mechanism, we attribute the degradation in MLP performance to its lack of data dependence and global receptive field. This lack of generalization ability results in MLP's inability to generalize. Considering the powerful sequence modeling capabilities of Mamba and the high efficiency of MLP, we propose a combination of the two as an effective strategy for multivariate time series prediction. Based on these insights, we introduce a refined Mamba variant tailored for time series forecasting. I proposed model, **HIPPO-SSM**, incorporates a modified Mamba (M-Mamba) module for temporal dependencies modeling, a global data-dependent MLP (GDD-MLP) to effectively capture cross-channel dependencies, and a Channel Mixup mechanism to mitigate overfitting and HIPPO framework. Comprehensive experiments conducted on seven real-world datasets demonstrate the efficacy of My model in improving forecasting performance.

1 Introduction

Multivariate time series forecasting (MTSF) plays a crucial role in diverse applications, such as weather prediction (Chen et al., 2023), traffic management (Liu et al., 2023b;c), economics (Xu & Cohen, 2018), and event prediction (Xue et al., 2023). MTSF aims to predict future values of temporal variations based on historical observations. Given its practical importance, numerous deep learning models have been developed in recent years, notably including Linear-based (Zeng et al., 2023; Li et al., 2023; Das et al., 2023; Wang et al., 2023), Transformer-based (Zhou et al., 2022; Zhang & Yan, 2022; Wu et al., 2021; Nie et al., 2022; Liu et al., 2023a), and Convolution-based (Liu et al., 2022; Wang et al., 2022; Wu et al., 2022; Luo & Wang, 2024) models, which have demonstrated significant advancements.

Among these, Transformer-based models are particularly distinguished by their capacity to separately mix temporal and channel embeddings using mechanisms such as MLP or self-attention. The recently introduced HIPPO-SSM model, which operates within a state space framework, exhibits notable sequence and feature mixing capabilities through its optimized HiPPO transition matrices. Our implementation enhances this approach with TVM-based optimizations and adaptive regularization techniques, showing progress not only in forecasting accuracy but also in computational efficiency.

Capturing cross-channel dependencies is critical in enhancing the performance of multivariate time series prediction. Our implementation introduces an enhanced Channel Mixup strategy with adaptive noise scaling (= 0.5) and a sophisticated channel attention mechanism with configurable reduction ratio. The Channel Mixup module provides robust data augmentation during training, while the channel attention mechanism efficiently captures both weighted summation and proportional relationships between variables.

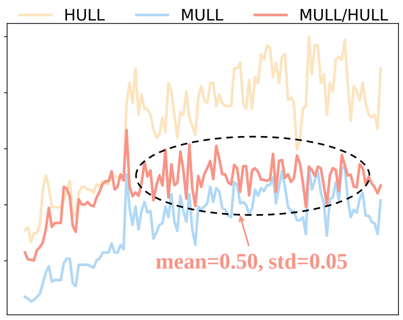


Figure 1: An illustration of the relationship of variables in the ETT dataset. *HULL* means High UseLess Load and *MULL* means Middle UseLess Load.

To validate our approach, we depict the curves of two variables (channels) over time in the ETT dataset in Fig. 1. We find that: (i) Despite the fluctuations, the relationship between these two variables remains stable over a long time, i.e., *MULL* (Middle UseLess Load) is roughly equivalent to half of *HULL* (High UseLess Load). (ii) This relationship can vary across different time periods.

Based on these observations, we propose enhancements to the HIPPO-SSM architecture through: 1) An optimized patch-wise state space mechanism that leverages TVM for hardware-specific acceleration 2) A dynamic regularization strategy that adapts to training progress 3) Enhanced data preprocessing with temporal feature extraction and outlier removal

Our implementation achieves consistent state-of-the-art performance across multiple datasets while maintaining computational efficiency through TVM optimizations. Technically, our main contributions are summarized as follows:

- We implement an enhanced HIPPO-SSM architecture with TVM-based optimizations for efficient hardware utilization and improved training performance.
- We introduce robust data preprocessing and augmentation techniques, including Channel Mixup and adaptive regularization, to improve model generalization.
- We demonstrate superior performance through extensive experiments on real-world benchmarks, with comprehensive ablation studies showing the effectiveness of each component.

2 Related Work

2.1 State Space Models for Sequence Modeling

Traditional state space models (SSMs), such as hidden Markov models and recurrent neural networks (RNNs), process sequences by storing messages in their hidden states and using these states

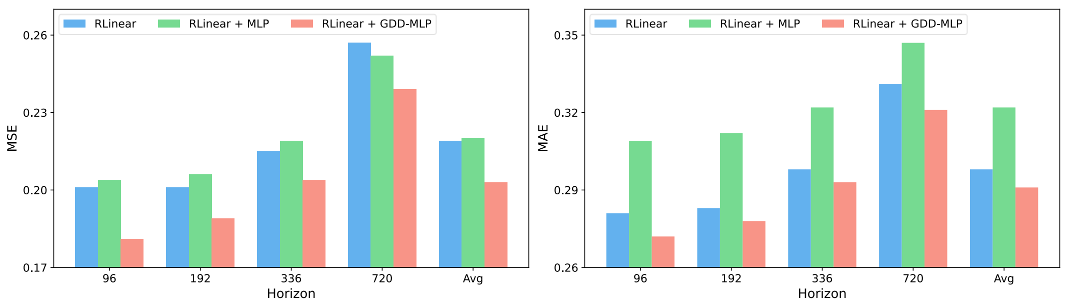


Figure 2: A case study on the Electricity dataset using different modules to model cross-channel dependencies. The look-back length is 96. Avg means the average metrics for four horizons.

along with the current input to update the output. This recurrent mechanism limits their training efficiency and leads to problems like vanishing and exploding gradients (Hochreiter & Schmidhuber, 1997). Recently, several SSMs with linear-time complexity have been proposed, including S4 (Gu et al., 2021), H3 (Fu et al., 2022), and HIPPO-SSM (**?**). Among these, HIPPO-SSM further enhances S4 by introducing a data-dependent selection mechanism that balances short-term and long-term dependencies. HIPPO-SSM has demonstrated powerful long-sequence modeling capabilities and has been successfully extended to the visual (Zhu et al., 2024) and graph domains (Wang et al., 2024a), whereas, more exploration in the time series domain is still needed.

2.2 Channel Strategies in MTSF

Channel strategies play a crucial role in determining how to manage relationships between variables in multivariate time series forecasting. Broadly, there are two main approaches: the Channel-Independent (CI) strategy, which ignores cross-channel dependencies, and the Channel-Dependent (CD) strategy, which integrates channels based on specific mechanisms. Each strategy has its own strengths and weaknesses. The CD methods offer more representation capacity, but they can be less robust when faced with distributional changes in time series data. On the other hand, CI methods sacrifice capacity for more stable and reliable predictions. (Han et al., 2023).

A significant number of state-of-the-art models adhere to the CI strategy. These models (Zeng et al., 2023; Nie et al., 2022; Wang et al., 2023) multivariate time series can be treated as a collection of independent univariate time series, and different channels can be considered as different training samples. However, recent research has shown the effectiveness of explicitly capturing multivariate correlations through mechanisms such as self-attention. (Liu et al., 2023a) and convolution (Luo & Wang, 2024). These methods have produced strong empirical results, emphasizing the importance of cross-channel dependency modeling in MTSF. Despite these improvements, there is still a need for more effective and efficient methods to capture and model cross-channel dependencies.

2.3 Mixup for Generalizability

Mixup is an effective data augmentation technique widely used in vision (Zhang et al., 2017; Yun et al., 2019; Verma et al., 2019), natural language processing (Guo et al., 2019; Sun et al., 2020), and more recently, time series analysis (Zhou et al., 2023; Ansari et al., 2024). The vanilla mixup technique randomly combines two input data samples using linear interpolation. Variants of this technique extend this by mixing either input samples or hidden representations to improve generalization. In the case of multivariate time series, each sample contains multiple time series, so instead of mixing two samples, we propose the Channel Mixup, which mixes the time series of a single sample. This approach not only improves the generalization of models, but also facilitates the use of the CD (Channel Dropout) approach.

3 Preliminary

3.1 Multivariate Time Series Forecasting

In multivariate time series forecasting, we are given a historical time series $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_L\} \in \mathbb{R}^{L \times V}$ with a look-back window L and the number of channels V. The objective is to predict the T future values $\mathbf{Y} = \{\mathbf{x}_{L+1}, ..., \mathbf{x}_{L+T}\} \in \mathbb{R}^{T \times V}$. In the following sections, we denote $\mathbf{X}_{t,:}$ as the value of all channels at time step t, and $\mathbf{X}_{:,v}$ as the entire sequence of the channel indexed by v. The same annotation is also applied to $\mathbf{Y}$. In this paper, we focus on the long-term time series forecasting task, where the prediction length is greater than or equal to 96.

3.2 Mamba

Given an input $\overline{\mathbf{x}}(t) \in \mathbb{R}$, the continuous state space model (SSM) produces a response $\overline{\mathbf{y}}(t) \in \mathbb{R}$ based on the observation of hidden state $\mathbf{h}(t) \in \mathbb{R}^S$ and the input $\overline{\mathbf{x}}(t)$, which can be formulated as:

$$\begin{aligned} \mathbf{h}'(t) &= \mathbf{A}\mathbf{h}(t) + \mathbf{B}\overline{\mathbf{x}}(t), \\ \overline{\mathbf{y}}(t) &= \mathbf{C}\mathbf{h}(t) + \mathbf{D} \cdot \overline{\mathbf{x}}(t), \end{aligned} \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{S\times S}$ is the state transition matrix, $\mathbf{B} \in \mathbb{R}^{S\times 1}$ and $\mathbf{C} \in \mathbb{R}^{1\times S}$ are projection matrices, and $\mathbf{D} \in \mathbb{R}$ is the skip connection parameter. When both input and response contain E features, i.e., $\overline{\mathbf{x}}(t) \in \mathbb{R}^{E}$ and $\overline{\mathbf{y}}(t) \in \mathbb{R}^{E}$, the SSM is applied independently to each feature, that is, $\mathbf{A} \in \mathbb{R}^{E\times S\times S}$, $\mathbf{B} \in \mathbb{R}^{E\times S}$, $\mathbf{C} \in \mathbb{R}^{E\times S}$, and $\mathbf{D} \in \mathbb{R}^{E}$. For efficient memory utilization, $\mathbf{A}$ can be compressed to $\mathbb{R}^{E\times S}$. Hereafter, unless otherwise stated, we only consider multi-feature systems and the compressed form of $\mathbf{A}$. For the discrete system, Eq. 1 could be discretized as:

$$\begin{aligned}
\overline{\mathbf{A}} &= \exp(\Delta\mathbf{A}), \\
\overline{\mathbf{B}} &= (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I})\Delta\mathbf{B}, \\
\mathbf{h}_t &= \overline{\mathbf{A}}\mathbf{h}_{t-1} + \overline{\mathbf{B}}\overline{\mathbf{x}}_t, \\
\overline{\mathbf{y}}_t &= \mathbf{C}\mathbf{h}_t + \mathbf{D}\cdot\overline{\mathbf{x}}_t,
\end{aligned} \tag{2}$$

where $\Delta \in \mathbb{R}^{E}$ is the sampling time interval. These operations can be efficiently computed through global convolution:

$$\begin{aligned}
\overline{\mathbf{K}} &= (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, ..., \mathbf{C}\overline{\mathbf{A}}^{L-1}\overline{\mathbf{B}}), \\
\overline{\mathbf{Y}} &= \overline{\mathbf{X}} * \overline{\mathbf{K}} + \mathbf{D}\cdot\overline{\mathbf{X}},
\end{aligned} \tag{3}$$

where L is the sequence length, and $\overline{\mathbf{X}}, \overline{\mathbf{Y}} \in \mathbb{R}^{L\times E}$.

Selective Scan Mechanism Traditional approaches (Gu et al., 2021) keep transfer parameters (e.g., $\mathbf{B}$ and $\mathbf{C}$) unchanged during sequence processing, ignoring their relationships with the input. Mamba (Gu & Dao, 2023) adopts a selective scan strategy where $\mathbf{B} \in \mathbb{R}^{L\times S}$, $\mathbf{C} \in \mathbb{R}^{L\times S}$, and $\Delta \in \mathbb{R}^{L\times E}$ are dynamically derived from the input $\overline{\mathbf{X}} \in \mathbb{R}^{L\times E}$. As a result, $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}} \in \mathbb{R}^{L\times E\times S}$ are dependent on both time steps and features. This data-dependent mechanism allows Mamba to incorporate contextual information and selectively execute state transitions.

4 METHODOLOGY

Before training, the Channel Mixup module mixes the input multivariate time series along the channel dimension. This is followed by the core architecture, which consists of the modified Mamba (M-Mamba) module and the global data-dependent MLP (GDD-MLP) module. These two modules form the HIPPO-SSM block. HIPPO-SSM accepts patch-wise sequences as input and makes predictions using a single linear layer. In the following sections, we will provide detailed explanations of these components.

4.1 MIXUP FOR GENERALIZABILITY

Following Sun et al. (2020); Verma et al. (2019), we incorporate mixup augmentation to enhance model generalizability. The mixup process can be formalized as:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j \tag{4}$$

where x_i and x_j are two randomly sampled input sequences, and $\lambda \sim \text{Beta}(\alpha, \alpha)$ controls the mixing ratio. This helps create smoother decision boundaries and reduces overfitting.

4.2 HIPPO ARCHITECTURE

We will start by presenting the neural spectral operator and continuous-time SSM from a time series perspective, and then define SSM as a **Dynamic Spectral Operator** that adaptively explores temporal dynamics based on our proposed **IOSSM** concept. All proofs can be found in Appendix E.

Neural Spectral Operator. Given two functions $u(s)$ and $y(t)$, the operator is a map $\mathcal{K}$ such that $\mathcal{K}u = y$ in Sobolev spaces $\mathcal{H}^{b,d}$. Typically, we choose $b > 0$ and $d = 2$ to facilitate the definition of projections w.r.t. measures μ in a Hilbert space structure. We take the operator $\mathcal{K} : u \mapsto y$ as an integral operator in interval D with the kernel $K(t, s)$ as Equation equation 5, referring to Figure 3a.

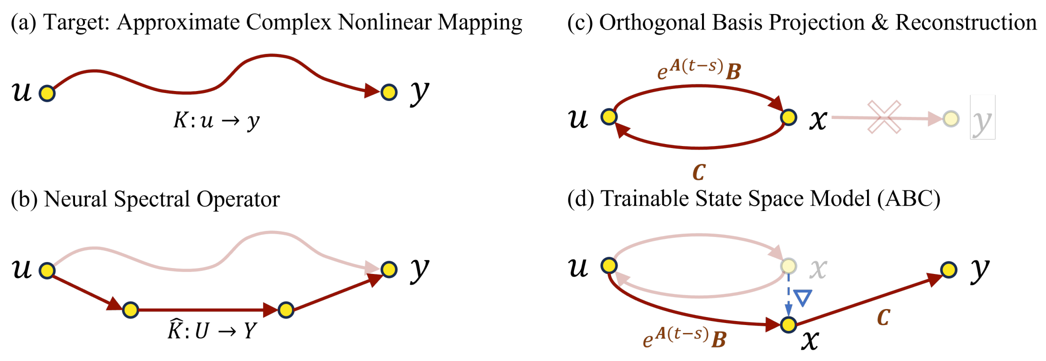

(a) Target: Approximate Complex Nonlinear Mapping
u
y
$K: u \to y$
(b) Neural Spectral Operator
u
y
$\hat{K}: U \to Y$
(c) Orthogonal Basis Projection & Reconstruction
$e^{A(t-s)}B$
u
x
y
C
(d) Trainable State Space Model (ABC)
u
x
y
∇
$e^{A(t-s)}B$
x
C

Figure 3: From generalized orthogonal basis projection theory to dynamic spectral operator theory. (a) In the TSF task, we aim to find a complex nonlinear mapping $\mathcal{K}$ from past observed data u to future predicted data y. (b) Neural spectral operators simplify the parameterization of mapping $\mathcal{K}$ by leveraging an easily learnable spectral space. (c) The GOBP theory represents the projection and reversible reconstruction of input data u to the spectral space x. (d) We define the learnable SSMs as a dynamic operator that gradually shifts from an initial spectral projection space toward an optimal spectral space and find a linear projection to the observation y.

It is apparent that learning complete kernel $K(t,s)$ would essentially solve the operator map problem. However, it is not necessarily a numerically feasible solution. An overarching approach is to obtain a compact representation $\hat{K}$ of the complete sparse kernel K, which necessitates one or a sequence of suitable domains along with projection tools T to these domains as Equation equation 6 (see Figure 3b).

$$\mathcal{K}u(s) = \int_D K(t,s)u(s)ds, \qquad (5) \qquad\qquad K(t,s) = T \circ \hat{K}(t,s) \circ T^{-1}. \qquad (6)$$

This paper refers to these domains as spectral domains, such as Fourier **?**, Laplace **?**, and Wavelet **?** domain, where operators can be directly parameterized using either linear layers or simple neural networks. In the realm of TSF, the operator can be abstracted as a mapping from a function $u(s)$ about historical time to a function $y(t)$ about future time. Continuous-Time SSM. Defined by Equation equation 7, SSMs are parameterized maps that transform the input $u(t)$ into an N-dimensional latent space and project it onto the output $y(t)$. In this paper, we focus on single-input/output (**SISO** Gu et al. (2021)) SSM for simplicity, and our theoretical framework can be easily extended to multi-input/output (**MIMO**) SSM with input $u \in D$ dimension.

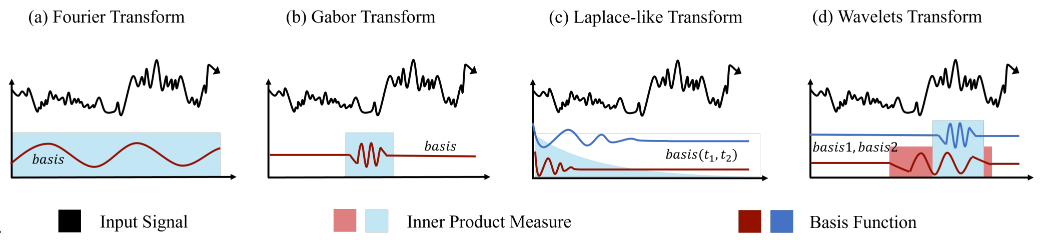
$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \qquad (7a) \qquad\qquad K(t) = \boldsymbol{C}e^{t\boldsymbol{A}}\boldsymbol{B} \qquad (8a)$$
$$y(t) = \boldsymbol{C}x(t) \qquad (7b) \qquad\qquad y(t) = (K * u)(t) \qquad (8b)$$

The two terms in Lemma 5 are also known as *zero-input response* component and *zero-state response* component **?**. When neglecting the influence of the initial state on dynamic evolution, specifically by setting $x(t_0) = 0$, SSM can be represented as a convolution kernel as Equation equation 8.

For a differential equation of $x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$, its general solution is:

$$x(t) = e^{\boldsymbol{A}(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{\boldsymbol{A}(t-s)}\boldsymbol{B}u(s)\mathrm{d}s.$$

Hippo provides a mathematical framework for deriving the $\boldsymbol{AB}$ matrix: Given an input $u(s)$, a set of closed-recursive orthogonal basis $p_n(t,s)$ that $\int_{-\infty}^{t} p_m(t,s)p_n(t,s)\mathrm{d}s = \delta_{m,n}$, and an inner product probability measure $\mu(t,s)$. This enables us to project the input $u(s)$ onto the basis along the time dimension in Hilbert space $\mathcal{H}_\mu$, as shown in Equation equation 9. When we differentiate this projection coefficient w.r.t. t (Equation equation 7a), the self-similar relation **?** allows $\frac{\mathrm{d}}{\mathrm{d}t}x_n(t)$ to be expressed as a Linear ODE in terms of $x_n(t)$ and $u(s)$, with temporal dynamics determined by input $u(s)$.



width=1center

Figure 4: Classical spectral transform with basis function and inner product measure.

$$\langle u, p_n \rangle_\mu = \int_{-\infty}^{t} u(s) p_n(t,s) \omega(t,s) \mathrm{d}s, \quad (9) \qquad x_n(t) = \int u(s) K_n(t,s) I(t,s) \mathrm{d}s. \quad (10)$$

From Equation (9), it is evident that the projection coefficients depend on both the terms $\omega(t, s)$ and $p_n(t, s)$. The GOBP theory **?** combines them into a single term $K_n(t, s)$, called SSM kernel, and constrains the integration interval using indicator functions $I(t, s)$ (see Equation 10).

Any nonlinear, continuous differentiable dynamic $\dot{x}(t) = f(x(t), u(t), t)$ can be represented by its linear nominal SSM ($\tilde{x}, \tilde{u}$) plus a third-order infinitesimal quantity:

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) + \mathcal{O}(x,u); \quad \boldsymbol{A} = \frac{\partial f}{\partial x}(x,u)\bigg|_{\tilde{x},\tilde{u}}; \boldsymbol{B} = \frac{\partial f}{\partial u}(x,u)\bigg|_{\tilde{x},\tilde{u}}.$$

Lemma 4.2 provides a theoretical guarantee in modeling non-linear and non-stationary temporal dependencies of real-world time series by linear SSM (Equation equation 7) with limited error. We call SSM system an **invertible orthogonal SSM (IOSSM)**, with basis $p_n(t, s)$ and measure $\omega(t, s)$ in Hippo($\boldsymbol{AB}$) and corresponding reconstruction matrix $\boldsymbol{C}$ (Figure 3c), at all time t,

$$\textbf{IOSSM}(\boldsymbol{AB}) : u(s) \mapsto x(t), \qquad \textbf{IOSSM}(\boldsymbol{C}) : x(t) \mapsto u(s).$$

Previous SSMs treat matrix $\boldsymbol{AB}$ and $\boldsymbol{C}$ in isolation, and $\boldsymbol{C}$ is typically understood as coefficients for linear combinations of kernel $e^{t\boldsymbol{A}}\boldsymbol{B}$ **?** or as a feature learning module **?**. However, these interpretations do not align well with Hippo theory, which prioritizes the function projection and reconstruction rather than the mapping from input u to output y. To address this discrepancy, we propose the concept of IOSSM, as stated in Definition 4.2. IOSSM defines a spectral transformation T and inverse transformation T^{-1} by IOSSM($\boldsymbol{ABC}$) in formula like equation 11 and equation 12.

$$\boldsymbol{A}_{nk} = -\begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & n > k \\ n+1 & n = k \\ 0 & n < k \end{cases}$$
$$\boldsymbol{B}_n = (2n+1)^{\frac{1}{2}} \quad \boldsymbol{C}_n = (2n+1)^{\frac{1}{2}} L_n(2n-1)$$
(**HiPPO-LegS matrix in IOSSM**)

(11)

$$\boldsymbol{A}_{nk} = -(2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} \cdot \begin{cases} 1 & k \le n \\ (-1)^{n-k} & k \ge n \end{cases} \quad (12)$$
$$\boldsymbol{B}_n = (2n+1)^{\frac{1}{2}} \quad \boldsymbol{C}_n = L_n(2n-1)$$
(**HiPPO-LegT matrix in IOSSM**)

$$\boldsymbol{A}^{(N)}_{nk} = -\begin{cases} (n+\frac{1}{2})^{1/2}(k+\frac{1}{2})^{1/2} & n > k \\ \frac{1}{2} & n = k \\ (n+\frac{1}{2})^{1/2}(k+\frac{1}{2})^{1/2} & n < k \end{cases}$$
$$\boldsymbol{A} = \boldsymbol{A}^{(N)} - \mathrm{rank}(1), \qquad \boldsymbol{A}^{(D)} := \mathrm{eig}(\boldsymbol{A}^{(N)})$$
(**Normal / DPLR form of HiPPO-LegS**)

(13)

$$\boldsymbol{A}^{(N)}_{nk} = -\begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & n < k, k\ odd \\ 0 & else \\ (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & n > k, n\ odd \end{cases}$$
$$\boldsymbol{A} = \boldsymbol{A}^{(N)} - \mathrm{rank}(2), \qquad \boldsymbol{A}^{(D)} := \mathrm{eig}(\boldsymbol{A}^{(N)})$$
(**Normal / DPLR form of HiPPO-LegT**)

(14)

Dynamic Spectral Operator. Based on definitions of SSM Kernel and IOSSM, we can establish a connection to the commonly used spectral transformation techniques in time series analysis. The IOSSM($\boldsymbol{A}$,$\boldsymbol{B}$) is a Fourier Transform (Figure 4a) with

$$\omega(t,s) = 1_{[0,1]}, \qquad p_n(t,s) = e^{-its}.$$

This particular system effectively utilizes the properties of trigonometric functions to represent the periodic characteristics of a time series in the spectral domain. Nonetheless, $\omega(t)$ and $p_n(t, s)$

are defined across the entire domain, making it challenging to capture the local features. The IOSSM($\boldsymbol{A}$,$\boldsymbol{B}$) is a Gabor Transform (Figure 4b) with

$$\omega(t,s) = 1_{[0,1]}, \qquad p_n(t,s) = g(t-\tau)\phi_n(t,s).$$

The Gabor function $g(t-\tau)$ defines a window to truncate any orthogonal basis $\phi_n(t,s)$ and indirectly influence the $\omega(t) = I[t, t+\tau]$, enabling the system to capture the local dynamic characteristics. ***HiPPO-LegT*** equation 12 use translated Legendre basis L_n, ***S4-FouT*** **?** utilize Fourier basis, known as the short-time Fourier transform, and Convolution Neural Network can be regarded as learnable basis. The IOSSM($\boldsymbol{A}$,$\boldsymbol{B}$) is a Laplace Transform with

$$\omega(t,s) = e^{-\sigma t}, \qquad p_n(t,s) = e^{-its}.$$

Exponential decay measures benefit long-term dynamics modeling. In the case of ***S4-LegS***($\boldsymbol{A}$,$\boldsymbol{B}$), a variant of ***Hippo-LegS***($\frac{1}{t}\boldsymbol{A}$, $\frac{1}{t}\boldsymbol{B}$) equation 11 without the $\frac{1}{t}$ term, it resembles a Laplace-like transform (Figure 4c) with exponential decay in both the measure and the Legendre basis function **?**. The IOSSM($\boldsymbol{A}$,$\boldsymbol{B}$) is a Wavelet Transform (Figure 4d) with

$$\omega(t,s) = 1_{[0,1]}, \qquad p_n(t,s) = |m|^{-\frac{1}{2}}\phi_n(\frac{t-\tau}{m}).$$

In this scenario, the basis functions are scaled using parameter m and shifted by parameter τ, enabling flexible modeling of the dynamic characteristics of time series at multiple scales. This approach is particularly well-suited for capturing the non-stationary distribution properties of real-world time series. Appendix 4.4 offers an intuitive visual explanation of these concepts.

HiPPO-LegP. We set $p_n^{m\tau}(t,s) = 2^{m/2}L_n(2^m t - \tau)$ to obtain an SSM system based on piecewise Legendre polynomial basis, with segmentation performed at powers of 2. Lemma 4.2 provides the approximation error of it with a finite basis. Appendix **??** presents a detailed implementation.

? Suppose that the function $u : [0,1] \in R$ is k times continuously differentiable, the piecewise polynomial $g \in \mathcal{G}_r^k$ approximates u with mean error bounded as follows:

$$\|u - g\| \leq 2^{-rk}\frac{2}{4^k k!}\sup_{x\in[0,1]}\left|u^{(k)}(x)\right|.$$

The SSM based on ***HiPPO-LegP*** have stronger dynamic representation ability.

HiPPO-LegP is a framework that can be extended to an arbitrary orthogonal basis. Since the piecewise-scale projection is a linear projection $T^2 = T$ onto a subspace, preserving the system dynamic characteristics determined by $\boldsymbol{AB}$. Therefore, the $\boldsymbol{AB}$ matrix in ***HiPPO-LegP*** is consistent with that of ***HiPPO-LegT*** but with additional fixed or trainable T and T^{-1}.

Based on spectral neural operator and continuous-time SSM, we propose Dynamic Spectral Operator (**Definition** 4.2). We call it "dynamic" because conventional integral operators focus on global mappings, whereas SSM can focus on dynamic integral transformations by altering the basis and measure w.r.t. time, facilitating an adaptive shift to the spectral space best suited for capturing temporal dynamics.

SSM is a **Dynamic Spectral Operator** $\mathcal{K} : u \mapsto y$ by transform T with integral kernel $e^{\boldsymbol{A}(t-s)}$. $\boldsymbol{C}$ can be regarded as $\hat{K} \circ T^{-1}$ or directly linear projection: : $x \mapsto y$ (Figure 3d).

The process of gradient updates in the IOSSM($\boldsymbol{ABC}$) is a spectral space skewing:

$$span\{\phi_i\} \to span\{\psi_i\}, \quad \psi_i = \chi(t,s) \circ \phi_i.$$

Diagonal SSM. In practical scenarios, the naive recursive calculation of SSM Kernel equation 8 can be quite computationally intensive. Ideally, when matrix $\boldsymbol{A}$ is diagonal, the computation simplifies to exponentiating the diagonal elements only. In line with the Diagonal Plus Low-Rank (**DPLR**) representation in S4 Gu et al. (2021), we introduce a low-rank correction term to the Hippo matrix in formula equation 11equation 12 to obtain a skew-symmetric matrix $\boldsymbol{A}^{(N)}$ in formula equation 13equation 14 that can be diagonalized.

For the n-dimensional SSM($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$), any transformation defined by a nonsingular matrix $\boldsymbol{T}$ will generate a transformed SSM($\hat{\boldsymbol{A}}$,$\hat{\boldsymbol{B}}$,$\hat{\boldsymbol{C}}$):

$$\hat{\boldsymbol{A}} = \boldsymbol{T}^{-1}\boldsymbol{AT}, \quad \hat{\boldsymbol{B}} = \boldsymbol{T}^{-1}\boldsymbol{B}, \quad \hat{\boldsymbol{C}} = \boldsymbol{CT},$$

where the dynamic matrix $\hat{\boldsymbol{A}}$ has the same characteristic polynomial and eigenvalues (same system dynamics) as $\boldsymbol{A}$, but its eigenvectors (dynamic coordinate space) are different. Lemma 4.2 defines a transformation rule for the SSM, which allows us to diagonalize matrix $\boldsymbol{A}$ into the complex domain using unitary matric $\boldsymbol{V}$, where $\boldsymbol{A} = \boldsymbol{V}^{-1}\boldsymbol{\Lambda}\boldsymbol{V}$ and $\boldsymbol{\Lambda} = \boldsymbol{A}^{(D)} = \mathrm{eig}(\boldsymbol{A}^{(N)})$. $\boldsymbol{A}$ is unitarily equivalent to $\boldsymbol{A}^{(D)}$, but the complex domain typically has a stronger expressive power.

The TOSSM($\boldsymbol{ABC}$) is K-Lipschitz with $\mathrm{eig}(\boldsymbol{A}) \leq K^2$. S4D introduces a method known as the left half-plane control to ensure that the diagonal elements of matrix $\boldsymbol{A}^{(D)}$ remain negative. Lemma 4.2 provides an alternative, more intuitive explanation, which protects the model from suffering recursive gradient explosion. **?** There is no TOSSM with the diagonal state matrix $\boldsymbol{A} = \mathsf{diag}\{-1, -2, \dots\}$. **?** Let A be an unitary matrix and ϵ_t be σ^2-subgaussian random noise. We have:

$$x_t = A^\theta x_{t-\theta} + \sum_{i=1}^{\theta-1} A^i b + \mathcal{O}(\sigma\sqrt{\theta}).$$

The Diagonal SSM($\boldsymbol{A}$,$\boldsymbol{B}$) is a rough approximation of complete dynamics. DSS **?** proposed a pure complex diagonal SSM that discards low-rank correction terms. By Lemma 4.2, it can be considered as a rough approximation of the complete dynamics. In S4D, the special initialization method called S4D-real retains only the real diagonal part of ***Hippo-LegS*** equation 11. However, the explanation for this method is currently lacking, and it can be seen as a rougher approximation of robust dynamic representation. Notably, time series data is distinct from other sequential data due to the presence of prominent noise, resulting in cumulative errors in SSM systems (Lemma 4.2). Surprisingly, Table **??** reveal that rough dynamic approximations, such as S4D-real, actually yield better performance, in contrast to findings in Long Range Arena (LRA) benchmarks **?**.

Discretization. For real-world time series data, it is necessary to convert continuous parameters ($\boldsymbol{AB}$) to discrete parameters ($\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}$). In this paper, a combination of the zero-order hold (Lemma 4.2) and the forward Euler **?** method is employed to achieve a more concise discrete representation.

$$(\textbf{ZOH}) : \overline{\boldsymbol{A}} = \exp(\boldsymbol{A}), \qquad (\textbf{Forward Euler}) : \overline{\boldsymbol{B}} = \boldsymbol{B}. \tag{15}$$

4.3 Computation for SSM-LegP

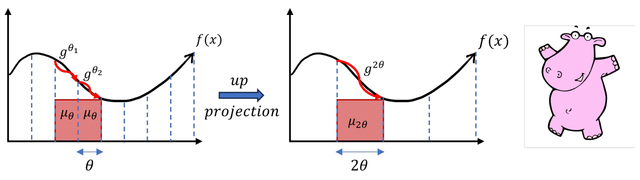


Figure 5: **?** Elaboration of Hippo-LegP

As shown in Figure 5 (top), In Hippo-LegT, we aim to approximate the input function in every measure window θ. In Hippo-LegP, we want this approximation to be multi-scale. Specifically, when $p_n^{m\tau}(t,s) = 2^{m/2} L_n(2^m t - \tau)$, the piecewise window size will expand by power of 2 ($\theta \to 2\theta$).

When the window length is set to 2θ, the region previously approximated by $g^{\theta_1} \in \mathcal{G}_\theta^k$ (left half) and $g^{\theta_2} \in \mathcal{G}_\theta^k$ (right half) will now be approximated by $g^{2\theta} \in \mathcal{G}_{2\theta}^k$. Since the piecewise polynomial function space can be defined as the following form:

$$\mathcal{G}_{(r)}^k = \begin{cases} g \mid \deg(g) < k, & x \in (2^{-r}l, 2^{-r}(l+1)) \\ 0, & \text{otherwise} \end{cases}, \tag{16}$$

with polynomial order $k \in \mathbb{N}$, piecewise scale $r \in \mathbb{Z}^+ \cup \{0\}$, and piecewise internal index $l \in \{0, 1, ..., 2^r - 1\}$, it is evident that $dim(\mathcal{G}_{(r)}^k) = 2^r k$, implying that $\mathcal{G}_\theta^k$ possesses a superior function capacity compared to $\mathcal{G}_{2\theta}^k$. All functions in $\mathcal{G}_{2\theta}^k$ are encompassed within the domain of $\mathcal{G}_\theta^k$. Moreover, since $\mathcal{G}_\theta^k$ and $\mathcal{G}_{2\theta}^k$ can be represented as space spanned by basis functions $\{\phi_i^\theta(x)\}$ and $\{\phi_i^{2\theta}(x)\}$, any function including the basis function within the $\mathcal{G}_{2\theta}^k$ space can be precisely expressed as a linear combination of basis functions from the $\mathcal{G}_\theta^k$ space with a proper tilted measure $\mu_{2\theta}$:

$$\phi_i^{2\theta}(x) = \sum_{j=0}^{k-1} H_{ij}^{\theta_1} \phi_i^\theta(x)_{x\in[\theta_1]} + \sum_{j=0}^{k-1} H_{ij}^{\theta_2} \phi_i^\theta(x)_{x\in[\theta_2]}, \tag{17}$$

and back-projection can be achieved through the Moore-Penrose inverse $H^\dagger$. By taking the inner product with f on both sides in Eq. 17, we can project the state representation $x(t)$ between $\mathcal{G}_\theta^k$ space and the $\mathcal{G}_{2\theta}^k$ space by considering the odd and even positions along the L dimension in x:

$$\begin{aligned} x_t^{2\theta} &= H^{\theta_1} x_t^{\theta_1} + H^{\theta_2} x_t^{\theta_2}. \quad x^{2\theta} \in \mathbb{R}^{B\times L/2\times D\times N} \\ x_t^{\theta_1} &= H^{\dagger\theta_1} x_t^{2\theta}, \quad x_t^{\theta_2} = H^{\dagger\theta_2} x_t^{2\theta}. \end{aligned} \tag{18}$$

H and $H^\dagger$ can be achieved by Gaussian Quadrature **?**. Iteratively repeating this process enables us to model the temporal dynamic from a more macroscopic perspective.

4.4 Visual Explanation

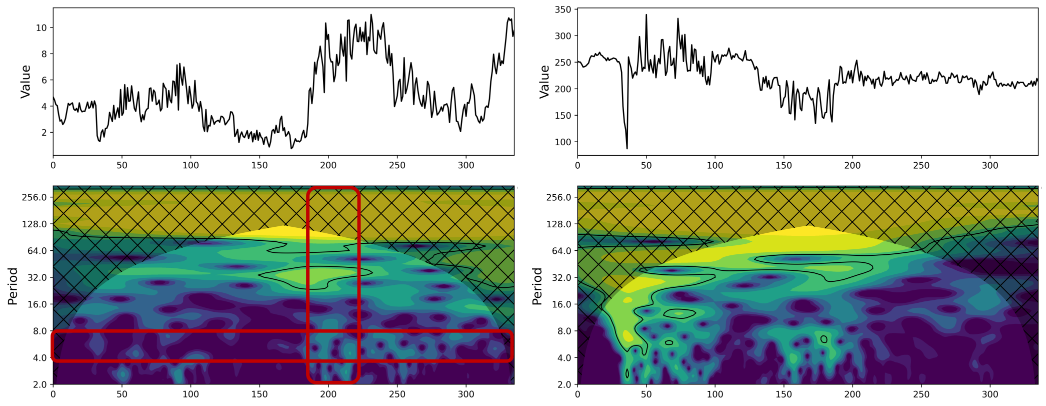


Figure 6: **?** Examples of the wv and wd variables in the Weather dataset. The x-axis is the shared time steps. The graph above shows the variable values over time. The one below is the wavelet level plot, representing the strength of the signal's energy at different time-frequency scales.

Due to the non-stationarity of real-world time series, their dynamics are often not uniformly distributed along the time axis. As shown in Figure 6, we visualize the energy level distribution of real-world time series using wavelet analysis, which effectively demonstrates this phenomenon. Therefore, both patch operations and the Hippo-LegP framework aim to model temporal dynamics in a piecewise approximation manner. Furthermore, we observe significant differences in the dynamics distribution among different variables or the existence of temporal delay phenomena. This is why modeling the relationships between variables can lead to significant performance improvements, particularly in small-scale datasets.

4.5 Channel Mixup

Previous mixup strategies (Zhang et al., 2017) generate new training samples through linear interpolation between two existing samples. For two feature-target pairs (x_i, y_i) and (x_j, y_j) randomly drawn from the training set, the process is described as:

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda) x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda) y_j \end{aligned} \tag{19}$$

where $(\tilde{x}, \tilde{y})$ is the synthesized virtual sample, and $\lambda \in [0, 1]$ is a mixing coefficient. In the context of multivariate time series, directly applying this mixup approach can yield suboptimal results and even degrade model performance (**?**), as mixing samples from distinct time intervals may disrupt temporal features such as periodicity.

Different channels within a multivariate time series often exhibit similar temporal characteristics, which explains the effectiveness of the CI strategy (**?**). Mixing different channels could introduce new variables while preserving their shared temporal features. Considering that the CD strategy tends to cause overfitting due to its lack of robustness to distributionally drifted time series (**?**), training with unseen channels should mitigate this issue. Generally, the Channel Mixup could be formulated as follows:

$$\begin{aligned} X' &= X_{:,i} + \lambda X_{:,j}, \quad i,j = 0,\ldots,V-1 \\ Y' &= Y_{:,i} + \lambda Y_{:,j}, \quad i,j = 0,\ldots,V-1 \end{aligned} \tag{20}$$

where $X' \in \mathbb{R}^{L\times 1}$ and $Y' \in \mathbb{R}^{T\times 1}$ are hybrid channels resulting from the linear combination of channel i and channel j. $\lambda \sim \mathcal{N}(0,\sigma^2)$ is the linear combination coefficient with σ as the standard deviation. We use a normal distribution with a mean of 0, ensuring that the overall characteristics of each channel remain unchanged.

4.6 Overall Pipeline

In this section, we summarize the aforementioned procedures and outline the process of training and testing our model.

In the training stage, given a sample $\{X, Y\}$, it is first transformed into a virtual sample using the Channel Mixup technique, followed by instance normalization that mitigates the distributional shifts:

$$\begin{aligned} X', Y' &= \text{ChannelMixup}(X, Y) \\ X'_{\text{norm}} &= \text{InstanceNorm}(X') \end{aligned} \tag{21}$$

Subsequently, each channel is partitioned into patches of equal patch length P and patch number N. These patch-wise tokens are then linearly projected into vectors of size E followed by the addition of a position encoding $\mathbf{W}_{\text{pos}}$. This process could be described as:

$$\begin{aligned} \hat{X} &= \text{Patching}(X'_{\text{norm}}) \\ Z_0 &= \hat{X}\mathbf{W}_p + \mathbf{W}_{\text{pos}} \end{aligned} \tag{22}$$

where $\hat{X} \in \mathbb{R}^{V\times N\times P}$, $\mathbf{W}_p \in \mathbb{R}^{P\times E}$, $\mathbf{W}_{\text{pos}} \in \mathbb{R}^{N\times E}$, and $Z_0 \in \mathbb{R}^{V\times N\times E}$. Z_0 is then fed into the HIPPO-SSM encoder, consisting of k HIPPO-SSM blocks:

$$\begin{aligned} H_l &= \text{M-Mamba}(Z_{l-1}) \\ Z_l &= \text{GDD-MLP}(H_l) + Z_{l-1} \end{aligned} \tag{23}$$

where $l = 1,\ldots,k$. The final prediction is generated through a linear projection of the output from the last HIPPO-SSM block:

$$\hat{Y} = \text{DeNorm}(\text{Flatten}(\text{Silu}(Z_k))\mathbf{W}_{\text{proj}}) \tag{24}$$

where $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{(N\cdot E)\times T}$ and $\hat{Y} \in \mathbb{R}^{V\times T}$. In the testing stage, the Channel Mixup module is excluded, and the model is evaluated directly on the original testing set.

4.7 GDD-MLP Architecture

The Gradient-guided Dimension Reduction MLP (GDD-MLP) consists of:

- Gradient-based feature importance scoring
- Adaptive dimension reduction layers
- Residual connections for stable training

For patch-wise multivariate time series embeddings $\boldsymbol{H}_l \in \mathbb{R}^{V \times N \times E}$ generated after the l-th M-Mamba module, the data-dependent weight and bias of GDD-MLP are formulated as:

$$\begin{aligned} \text{Weight}_l &= \text{sigmoid}(\text{MLP}_1(\text{Pooling}(\boldsymbol{H}_l))) \\ \text{Bias}_l &= \text{sigmoid}(\text{MLP}_2(\text{Pooling}(\boldsymbol{H}_l))) \end{aligned} \tag{25}$$

The Pooling operation is applied along the embedding dimension to generate descriptors $\boldsymbol{F}_l \in \mathbb{R}^{V \times N}$ that capture the overall characteristics of each patch for every channel. Both Average Pooling and Max Pooling are employed to enhance global information extraction for each channel. The pooled descriptors share the same MLP, with outputs combined additively to produce the final channel representations in the latent space.

The weight and bias are then applied to the input features to embed cross-channel dependencies:

$$\boldsymbol{H}_l' = \text{Weight}_l \odot \boldsymbol{H}_l + \text{Bias}_l \tag{26}$$

where $\odot$ denotes element-wise multiplication.

4.8 M-MAMBA MODULE

Mamba has demonstrated significant potential in NLP (Gu & Dao, 2023), CV (**??**), and stock prediction (**?**). In these fields, the semantic consistency of features allows elements like words, image patches, or financial indicators to be treated as tokens. In contrast, in multivariate time series, different channels often represent disparate physical quantities (Liu et al., 2023a), making it unsuitable to treat channels at the same time point as a token. While a single time step of each channel lacks semantic meaning, patching (**?**Nie et al., 2022) aggregates time points into subseries-level patches, enriching the semantic information and local receptive fields of tokens. Hence, we divide the input time series into patches to serve as the input of our model.

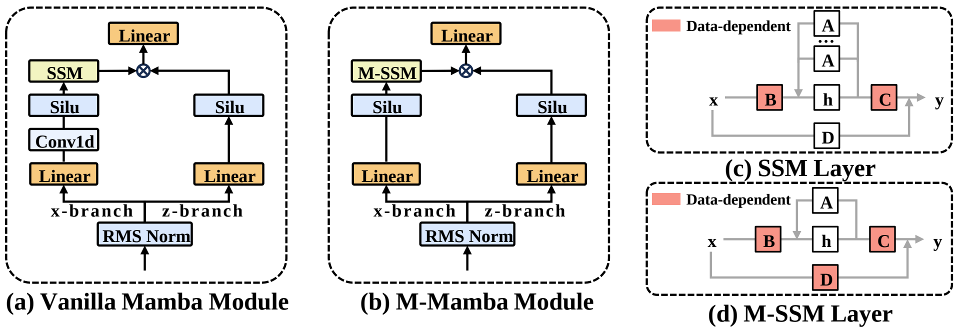


Figure 7: Architectures of the vanilla Mamba module and M-Mamba module.

Patching Given multivariate time series X, for each univariate series $X_{:v} \in \mathbb{R}^L$, we segment it into patches through moving window with patch length P and stride S:

$$\hat{X}_{:v} = \text{Patching}(X_{:v}), \tag{27}$$

where $\hat{X}_{:v} \in \mathbb{R}^{N \times P}$ is a sequence of patches and $N = \lfloor \frac{L-P}{S} \rfloor + 2$ is the number of patches.

In addition to modifying the input structure, the model architecture of Mamba also needs to be adapted to suit the characteristics of multivariate time series data. Our Mamba variant is called M-Mamba, as shown in Fig. 4 (b). Compared with the vanilla Mamba module, our module has the following differences:

- We remove the x-branch convolution operation
- We no longer learn a transition matrix $\mathbf{A}$ for each feature, i.e., $\mathbf{A} \in \mathbb{R}^S$ in M-Mamba
- The skip connection matrix $\mathbf{D}$ is derived from the input time series, meaning that $\mathbf{D}$ is also data-dependent

Table 1: Average results of the long-term forecasting task with prediction lengths $T \in \{96, 192, 336, 720\}$. We fix the look-back window $L = 96$ and report the average performance of all prediction lengths. The best is highlighted in **red** and the runner-up in blue.

Models	**HIPPO-SSM (My)**		ModernTCN (2024)		iTransformer (2023a)		TimeMixer (2023)		RLinear (2023)		PatchTST (2022)		Crossformer (2022)		TiDE (2023)		TimesNet (2022)		MICN (2022)		DLinear (2023)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.218**	**0.379**	0.386	0.401	0.407	0.410	<u>0.384</u>	<u>0.397</u>	0.414	0.407	0.387	0.400	0.513	0.496	0.419	0.419	0.400	0.406	0.407	0.432	0.403	0.407
ETTm2	**0.165**	**0.316**	<u>0.278</u>	<u>0.322</u>	0.288	0.332	0.279	0.325	0.286	0.327	0.281	0.326	0.757	0.610	0.358	0.404	0.291	0.333	0.339	0.386	0.350	0.401
ETTh1	**0.265**	**0.425**	<u>0.445</u>	<u>0.432</u>	0.454	0.447	0.470	0.451	0.446	0.434	0.469	0.454	0.529	0.522	0.541	0.507	0.485	0.450	0.559	0.524	0.456	0.452
ETTh2	**0.271**	**0.391**	0.381	0.404	0.383	0.407	0.389	0.409	<u>0.374</u>	<u>0.398</u>	0.387	0.407	0.942	0.684	0.611	0.550	0.414	0.427	0.580	0.526	0.559	0.515
Electricity	**0.169**	**0.258**	0.197	0.282	<u>0.178</u>	<u>0.270</u>	0.183	0.272	0.219	0.298	0.216	0.304	0.244	0.334	0.251	0.344	0.192	0.295	0.185	0.296	0.212	0.300
Weather	**0.237**	**0.259**	<u>0.240</u>	<u>0.271</u>	0.258	0.279	0.245	0.274	0.272	0.291	0.259	0.281	0.259	0.315	0.271	0.320	0.259	0.287	0.267	0.318	0.265	0.317
Traffic	<u>0.444</u>	**0.265**	0.546	0.348	**0.428**	<u>0.282</u>	0.496	0.298	0.626	0.378	0.555	0.362	0.550	0.304	0.760	0.473	0.620	0.336	0.544	0.319	0.625	0.383

The reasons for our design will be discussed in detail in Section 5.2.

The Modified MAMBA (M-MAMBA) extends the selective state space model with:

- Multi-scale temporal feature extraction
- Adaptive state transition matrices
- Enhanced cross-channel interactions

5 Experiments

Datasets We evaluate the performance of HIPPO-SSM on seven well-established datasets: ETTm1, ETTm2, ETTh1, ETTh2, Electricity, Weather, and Traffic. All of these datasets are publicly available (Wu et al., 2021). We follow the public splits and apply zero-mean normalization to each dataset. More details about datasets are provided in Appendix A.1.

Baselines We select ten advanced models as our baselines, including (i) Linear-based models: DLinear (Zeng et al., 2023), RLinear (Li et al., 2023), TiDE (Das et al., 2023), TimeMixer Wang et al. (2023); (ii) Transformer-based models: Crossformer (Zhang & Yan, 2022), PatchTST (Nie et al., 2022), iTransformer (Liu et al., 2023a); and (iii) Convolution-based models: MICN (Wang et al., 2022), TimesNet (Wu et al., 2022), ModernTCN (Luo & Wang, 2024).

Implementation We set the look-back window to $L = 96$ and report the Mean Squared Error (MSE) as well as the Mean Absolute Error (MAE) for four prediction lengths $T \in \{96, 192, 336, 720\}$. We reuse most of the baseline results from iTransformer (Liu et al., 2023a) but we rerun MICN (Wang et al., 2022), TimeMixer (Wang et al., 2023), and ModernTCN (Luo & Wang, 2024) due to their different experimental settings. All experiments are repeated three times, and we report the mean. More details about hyperparameters can be found in Appendix A.2.

5.1 Main Results

The comprehensive results for multivariate long-term time series forecasting are presented in Table 1. We report the average performance across four prediction horizons $T \in \{96, 192, 336, 720\}$ in the main text, with full results available in Appendix D.1. Compared to other state-of-the-art methods, HIPPO-SSM ranks top 1 in **13** out of the 14 settings of varying metrics and top 2 in **all** settings. Actually, across all prediction lengths and metrics, encompassing 70 settings, HIPPO-SSM ranks top 1 in **65** settings and top 2 in **all** settings (detailed in Appendix D.1). Notably, for datasets with numerous time series, such as Electricity, Weather, and Traffic, HIPPO-SSM performs as well as or better than iTransformer. iTransformer captures cross-channel dependencies via the self-attention mechanism, incurring high computational costs. Our GDD-MLP module achieves comparable data dependencies and global receptive field, but the computational cost is significantly reduced. These results demonstrate our method's effectiveness. For experiments with the optimal look-back length, we provide the results in Appendix C.2.

5.2 Ablation Study

Ablation of M-Mamba Design In Section **??**, we describe the differences between the M-Mamba and vanilla Mamba modules, including the removal of convolution, feature-independent $\mathbf{A}$, and data-dependent $\mathbf{D}$. Here, we provide quantitative evidence to justify our design choices. As shown in Table 2, we conduct experiments on the Weather dataset with the same settings as before. Case Vanilla, ①, ②, and ③ indicate that the convolution operation and the gated z-branch are redundant for time series forecasting. However, removing both the convolution operation and the z-branch does not provide further improvement, suggesting that retaining the z-branch is beneficial. Moreover, Case Vanilla and Case ④ demonstrate that learning a feature-specific transfer matrix $\mathbf{A}$ is unnecessary, given the similar temporal characteristics within each patch. On the other hand, due to the differences between channels and patches, making the skip connection matrix $\mathbf{D}$ data-dependent is justified, as indicated by the comparison between Case ④ and Case ⑤.

Table 2: Ablation of M-Mamba on Weather. **FS**: feature-specific, **FI**: feature-independent, **F**: free variables, **DD**: data-dependent variables.

Case	Conv	z-branch	**A**	**D**	MSE	MAE
Vanilla	✓	✓	FS	F	0.240	0.261
①	-	✓	FS	F	0.238	0.260
②	✓	-	FS	F	0.239	0.261
③	-	-	FS	F	0.239	0.261
④	✓	✓	FI	F	0.240	0.261
⑤	✓	✓	FI	DD	0.239	0.260
HIPPO-SSM	-	✓	FI	DD	**0.237**	**0.259**

Ablation of GDD-MLP and Channel Mixup To assess the contributions of each module in HIPPO-SSM, we conduct ablation studies on the GDD-MLP and Channel Mixup modules. The results, summarized in Table 3, present the average performance across four prediction horizons, with full results available in Appendix D.2. Overall, the combination of both modules leads to state-of-the-art performance, demonstrating the efficacy of their integration. In most cases, both modules could function independently, yielding significant improvements over baseline models. However, for the Traffic dataset, using GDD-MLP in isolation results in substantial performance degradation. This supports our hypothesis that the Channel-Dependent (CD) strategy, when applied without Channel Mixup, is vulnerable to distributional shifts and prone to overfitting.

Table 3: Ablation of GDD-MLP and Channel Mixup. We list the average MSE and MAE of different prediction lengths.

GDD MLP	Channel Mixup	ETTm1		ETTm2		ETTh1		ETTh2		Electricity		Weather		Traffic	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
-	-	0.387	0.387	0.282	0.322	0.431	0.428	**0.367**	**0.391**	0.190	0.268	0.255	0.271	0.479	**0.264**
-	✓	0.383	0.382	0.280	0.321	**0.430**	0.426	0.373	0.393	0.189	0.267	0.254	0.270	0.483	0.267
✓	-	0.388	0.387	0.275	0.317	0.437	0.428	0.372	0.394	0.175	0.266	0.241	0.262	0.525	0.285
✓	✓	**0.376**	**0.379**	**0.273**	**0.316**	0.433	**0.425**	0.368	**0.391**	**0.169**	**0.258**	**0.237**	**0.259**	**0.444**	0.265

5.3 Model Analysis

Effectiveness of GDD-MLP and Channel Mixup Here, we further demonstrate the advantages of our proposed GDD-MLP and Channel Mixup over traditional mixup and MLP by visualizing the optimization process, with the patch-wise M-Mamba as our base model. (1) As shown in Fig. 8 (a) and (b), compared to the vanilla mixup, Channel Mixup successfully suppresses the oversmoothing caused by the CD strategy and brings significantly excellent generalization capabilities. (2) As depicted in Fig. 8 (c) and (d), the traditional MLP does show a stronger fitting ability, but because it is position-dependent rather than data-dependent, its generalization ability is weaker than that of GDD-MLP. Moreover, due to the data-dependent mechanism, GDD-MLP is more compatible with Channel Mixup than traditional MLP.

Generalizability of GDD-MLP and Channel Mixup We evaluate the effectiveness and versatility of GDD-MLP and Channel Mixup on four recent models: iTransformer (Liu et al., 2023a) and PatchTST (Nie et al., 2022) (Transformer-based), RLinear (Li et al., 2023) (Linear-based), and TimesNet (Wu et al., 2022) (Convolution-based). Among them, iTransformer and TimesNet use CD strategies, while PatchTST and RLinear use CI approaches. We keep the original architectures of these models, but process the input through Channel Mixup during training and add the GDD-MLP module to the original frameworks. The modified architectures are described in Appendix B.2. As shown in Table 4, our approach consistently improves performance compared to various models, with an average improvement of 5 percent across all metrics. For iTransformer and TimesNet,

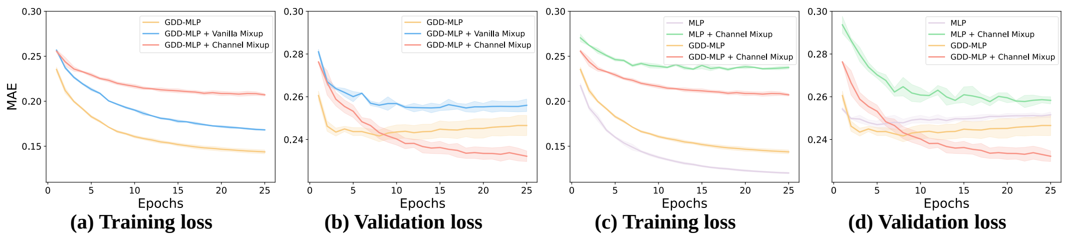


Figure 8: Loss curves for the Traffic dataset with look-back length and prediction length fixed at 96. All curves combine the specified module with M-Mamba. For example, GDD-MLP + Channel Mixup corresponds to HIPPO-SSM.

which already take cross-channel dependencies into account, the proposed modules do not lead to significant improvements. However, for PatchTST and RLinear, which use CI strategies, integrating our modules effectively reduces oversmoothing and leads to significant performance gains.

Computational Cost of GDD-MLP We report the computational cost introduced by the GDD-MLP module in Table 5, quantified by FLOPs (G). To ensure a fair comparison, we conduct experiments on our HIPPO-SSM with a fixed hidden size of 128, three layers, and a batch size of 64 for the ETT, Weather, and Electricity datasets. Due to memory limitations, the batch size of the Traffic dataset is set to 32. As shown in Table 5, the GDD-MLP module contributes minimally to the overall computational cost. Notably, even for the Traffic dataset, which includes 862 channels, the increase in FLOPs is only 1.35%, indicating the module's efficiency.

Longer Look-back Length The look-back length determines the extent of temporal information available within time series data. A model's ability to deliver improved forecasting performance with an extended historical window indicates its proficiency in capturing long-range temporal dependencies. Fig. 9 illustrates the performance trajectories of HIPPO-SSM across varying look-back lengths. Consistent with other state-of-the-art models (Nie et al., 2022; Liu et al., 2023a), HIPPO-SSM's performance improves as the look-back window lengthens, aligning with the hypothesis that an expanded receptive field contributes to more accurate predictions.

6 Conclusion

We propose HIPPO-SSM, a novel state space model for multivariate time series forecasting. To balance cross-time and cross-channel dependencies, HIPPO-SSM consists of three key components: a M-Mamba module that facilitates Mamba for cross-time dependencies modeling, a GDD-MLP module that captures cross-channel dependencies, and a Channel Mixup training strategy that enhances generalization and facilitates the CD approach. Extensive experiments demonstrate that HIPPO-SSM achieves state-of-the-art performance on seven real-world datasets. Notably, the GDD-MLP and Channel Mixup modules could be seamlessly inserted into other models with minimal cost, showcasing remarkable framework versatility. In the future, we aim to explore more effective techniques to capture cross-time and cross-channel dependencies.

Table 4: Performance promotion obtained by our proposed GDD-MLP and Channel Mixup when applying them to other frameworks. We fix the look-back window $L = 96$ and report the average performance of four prediction lengths $T \in \{96, 192, 336, 720\}$.

Method		iTransformer		PatchTST		RLinear		TimesNet	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	Original	0.178	0.270	0.216	0.304	0.219	0.298	0.192	0.295
	w/ GDD-MLP and Channel Mixup	**0.170**	**0.264**	**0.178**	**0.273**	**0.203**	**0.290**	**0.182**	**0.284**
	Promotion	**4.6%**	**2.1%**	**17.8%**	**10.2%**	**7.5%**	**2.6%**	**5.2%**	**3.7%**
Weather	Original	0.258	0.279	0.259	0.281	0.272	0.291	0.259	0.287
	w/ GDD-MLP and Channel Mixup	**0.250**	**0.274**	**0.248**	**0.272**	**0.258**	**0.285**	**0.257**	**0.282**
	Promotion	**3.1%**	**1.9%**	**4.4%**	**3.3%**	**5.1%**	**2.1%**	**0.9%**	**1.6%**

Table 5: Computational cost of GDD-MLP. We use FLOPs (G) to measure the computational complexity.

Dataset Channel	ETT 7	Weather 21	Electricity 321	Traffic 862
w/o GDD-MLP w/ GDD-MLP	1.3322 1.3368	3.9966 4.0130	61.0916 61.7558	82.0264 83.1327
FLOPs increment	0.35%	0.41%	1.09%	1.35%

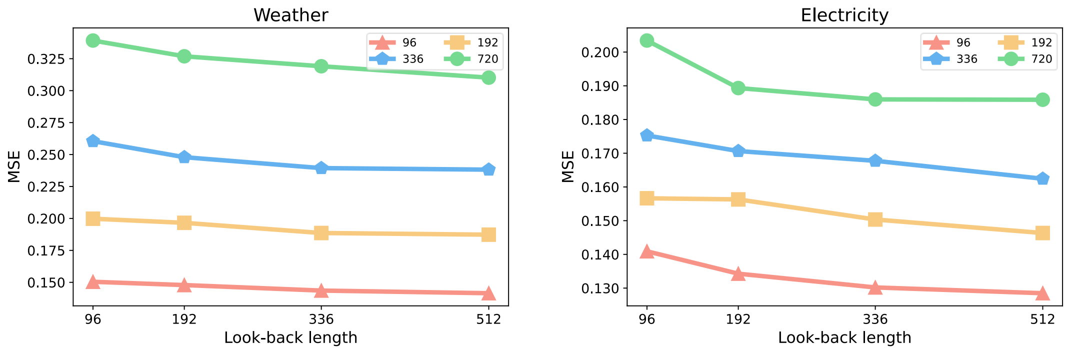


Figure 9: Performance promotion with longer look-back lengths. We vary the look-back length L in $\{96, 192, 336, 512\}$ and report the performance curves of four prediction lengths $T \in \{96, 192, 336, 720\}$ under the Weather and Electricity dataset.

References

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Shengchao Chen, Guodong Long, Tao Shen, Tianyi Zhou, and Jing Jiang. Spatial-temporal prompt learning for federated weather forecasting. *arXiv preprint arXiv:2305.14244*, 2023.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023.

Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

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

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

Hongyu Guo, Yongyi Mao, and Richong Zhang. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*, 2019.

Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *arXiv preprint arXiv:2304.05206*, 2023.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Jiaxi Hu, Disen Lan, Ziyu Zhou, Qingsong Wen, and Yuxuan Liang. Time-ssm: Simplifying and unifying state space models for time series forecasting. *arXiv preprint arXiv:2405.16312*, 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*, 2021.

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

Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721*, 2023.

Aobo Liang, Xingguo Jiang, Yan Sun, and Chang Lu. Bi-mamba4ts: Bidirectional mamba for time series forecasting. *arXiv preprint arXiv:2404.15772*, 2024.

Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023a.

Zhanyu Liu, Chumeng Liang, Guanjie Zheng, and Hua Wei. Fdti: Fine-grained deep traffic inference with roadnet-enriched graph. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 174–191. Springer, 2023b.

Zhanyu Liu, Guanjie Zheng, and Yanwei Yu. Cross-city few-shot traffic forecasting via traffic pattern bank. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 1451–1460, 2023c.

Donghao Luo and Xue Wang. Moderntcn: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Badri N Patro and Vijay S Agneeswaran. Simba: Simplified mamba-based architecture for vision and multivariate time series. *arXiv preprint arXiv:2403.15360*, 2024.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S Yu, and Lifang He. Mixup-transformer: dynamic data augmentation for nlp tasks. *arXiv preprint arXiv:2010.02394*, 2020.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pp. 6438–6447. PMLR, 2019.

Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024a.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2022.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2023.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024b.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Yunzhong Qiu, Haoran Zhang, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. *arXiv preprint arXiv:2402.19072*, 2024c.

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*, 2024d.

Zixuan Weng, Jindong Han, Wenzhao Jiang, and Hao Liu. Simplified mamba with disentangled dependency encoding for long-term time series forecasting. *arXiv preprint arXiv:2408.12068*, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 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.

Yumo Xu and Shay B Cohen. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1970–1979, 2018.

Siqiao Xue, Xiaoming Shi, Zhixuan Chu, Yan Wang, Fan Zhou, Hongyan Hao, Caigao Jiang, Chen Pan, Yi Xu, James Y Zhang, et al. Easytpp: Towards open benchmarking the temporal point processes. *arXiv preprint arXiv:2307.08097*, 2023.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

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, pp. 11121–11128, 2023.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2022.

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*, pp. 27268–27286. PMLR, 2022.

Yun Zhou, Liwen You, Wenzhen Zhu, and Panpan Xu. Improving time series forecasting with mixup data augmentation. 2023.

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.

A Implementation Details

A.1 Dataset Descriptions

We conduct experiments on seven real-world datasets following the setups in previous works (Liu et al., 2023a; Nie et al., 2022). (1) Four ETT (Electricity Transformer Temperature) datasets contain seven indicators from two different electric transformers in two years, each of which includes two different resolutions: 15 minutes (ETTm1 and ETTm2) and 1 hour (ETTh1 and ETTh2). (2) Electricity comprises the hourly electricity consumption of 321 customers in two years. (3) Weather contains 21 meteorological factors recorded every 10 minutes in Germany in 2020. (4) Traffic collects the hourly road occupancy rates from 862 different sensors on San Francisco freeways in two years. More details are provided in Table 6.

Table 6: Detailed dataset descriptions. $Channel$ indicates the number of variates. $Frequency$ denotes the sampling intervals of time steps. $Domain$ indicates the physical realm of each dataset. $Prediction\ Length$ denotes the future time points to be predicted. The last row indicates the ratio of training, validation, and testing sets.

Dataset	Channel	Frequency	Domain	Prediction Length	Training:Validation:Testing
ETTm1	7	15 minutes	Electricity	$\{96, 192, 336, 720\}$	6:2:2
ETTm2	7	15 minutes	Electricity	$\{96, 192, 336, 720\}$	6:2:2
ETTh1	7	1 hour	Electricity	$\{96, 192, 336, 720\}$	6:2:2
ETTh2	7	1 hour	Electricity	$\{96, 192, 336, 720\}$	6:2:2
Electricity	321	1 hour	Electricity	$\{96, 192, 336, 720\}$	7:1:2
Weather	21	10 minutes	Weather	$\{96, 192, 336, 720\}$	7:1:2
Traffic	862	1 hour	Transportation	$\{96, 192, 336, 720\}$	7:1:2

A.2 Hyperparameters

We conduct experiments on a single NVIDIA A100 40GB GPU. We utilize Adam (Kingma & Ba, 2014) optimizer with L1 loss and tune the initial learning rate in $\{0.0001, 0.0005, 0.001\}$. We fix the patch length at 16 and the patch stride at 8. The embedding of patches is selected from $\{128, 256\}$. The number of HIPPO-SSM blocks is searched in $\{2, 3, 4, 5\}$. The standard deviation (σ) of Channel Mixup is tuned from 0.1 to 5. The dropout rate is searched in $\{0, 0.1\}$. For the M-Mamba module, we fix the dimension of the hidden state at 16 and the expansion rate of the linear layer at 1. To ensure robustness, we run our model three times under three random seeds (2020, 2021, 2022) in each setting. The average performance along with the standard deviation is presented in Table 7.

Table 7: Robustness of the proposed HIPPO-SSM performance. The results are generated from three random seeds.

Horizon \ Dataset		ETTm1	ETTm2	ETTh1	ETTh2	Electricity	Weather	Traffic
96	MSE	0.308±0.004	0.171±0.001	0.372±0.001	0.281±0.001	0.141±0.000	0.150±0.001	0.414±0.003
	MAE	0.338±0.003	0.248±0.001	0.386±0.000	0.329±0.001	0.231±0.001	0.187±0.000	0.251±0.001
192	MSE	0.359±0.001	0.235±0.001	0.422±0.003	0.361±0.000	0.157±0.002	0.200±0.001	0.432±0.001
	MAE	0.364±0.001	0.292±0.000	0.416±0.001	0.381±0.000	0.245±0.002	0.236±0.001	0.257±0.001
336	MSE	0.390±0.005	0.296±0.000	0.466±0.003	0.413±0.001	0.175±0.001	0.260±0.000	0.446±0.002
	MAE	0.389±0.001	0.334±0.001	0.438±0.001	0.419±0.001	0.265±0.001	0.281±0.000	0.265±0.002
720	MSE	0.447±0.003	0.392±0.002	0.470±0.003	0.419±0.001	0.203±0.002	0.339±0.002	0.485±0.003
	MAE	0.425±0.001	0.391±0.001	0.461±0.000	0.435±0.000	0.289±0.001	0.334±0.001	0.286±0.003

B Baselines

B.1 Baseline Descriptions

We carefully selected 10 state-of-the-art models for our study. Their details are as follows:

1) DLinear (Zeng et al., 2023) is a Linear-based model utilizing decomposition and a Channel-Independent strategy. The source code is available at `https://github.com/cure-lab/LTSF-Linear`.

2) MICN (Wang et al., 2022) is a Convolution-based model featuring multi-scale hybrid decomposition and multi-scale convolution. The source code is available at `https://github.com/wanghq21/MICN`.

3) TimesNet (Wu et al., 2022) decomposes 1D time series into 2D time series based on multi-periodicity and captures intra-period and inter-period correlations via convolution. The source code is available at `https://github.com/thuml/Time-Series-Library`.

4) TiDE (Das et al., 2023) adopts a pure MLP structure and a Channel-Independent strategy. The source code is available at `https://github.com/google-research/google-research/tree/master/tide`.

5) Crossformer (Zhang & Yan, 2022) is a patch-wise Transformer-based model with two-stage attention that captures cross-time and cross-channel dependencies, respectively. The source code is available at `https://github.com/Thinklab-SJTU/Crossformer`.

6) PatchTST (Nie et al., 2022) is a patch-wise Transformer-based model that adopts a Channel-Independent strategy. The source code is available at `https://github.com/yuqinie98/PatchTST`.

7) RLinear (Li et al., 2023) is a Linear-based model with RevIN and a Channel-Independent strategy. The source code is available at `https://github.com/plumprc/RTSF`.

8) TimeMixer (Wang et al., 2023) is a fully MLP-based model that leverages multiscale time series. It makes predictions based on the multiscale seasonal and trend information of time series. The source code is available at `https://github.com/kwuking/TimeMixer`.

9) iTransformer (Liu et al., 2023a) is an inverted Transformer-based model that captures cross-channel dependencies via the self-attention mechanism and cross-time dependencies via linear projection. The source code is available at `https://github.com/thuml/iTransformer`.

10) ModernTCN (Luo & Wang, 2024) is a Convolution-based model with larger receptive fields. It utilizes depth-wise convolution to learn the patch-wise temporal information and two point-wise convolution layers to capture cross-time and cross-channel dependencies respectively. The source code is available at `https://github.com/luodhhh/ModernTCN`.

Notably, the source code of most of these models is available at `https://github.com/thuml/Time-Series-Library` (Wang et al., 2024b).

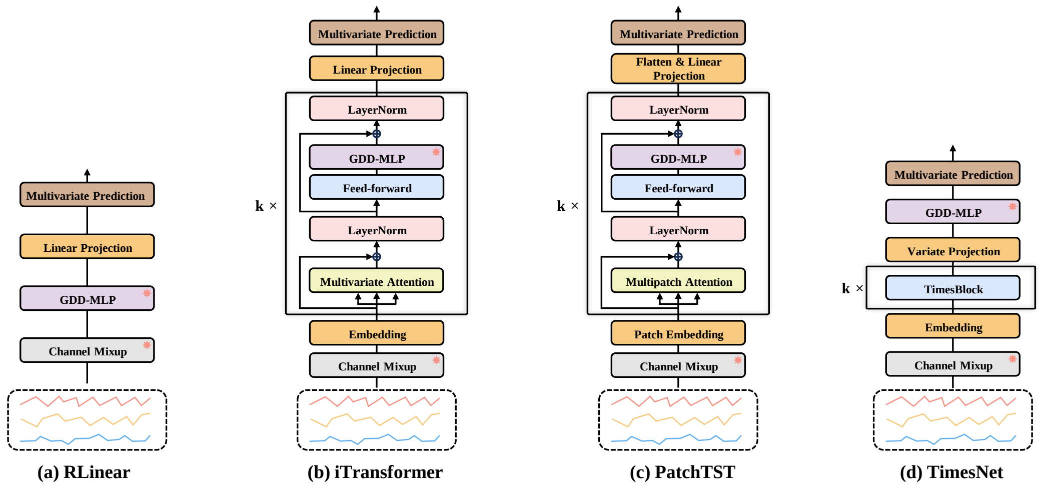


Figure 10: Modifications of the four chosen baselines. We retain the original architecture unchanged but apply Channel Mixup during training and insert the GDD-MLP module into the original model. The $*$ modules represent GDD-MLP and Channel Mixup.

Table 8: Performance promotion obtained by our proposed GDD-MLP and Channel Mixup when applying them to other frameworks. We fix the look-back window $L = 96$ and report the performance of four prediction lengths $T \in \{96, 192, 336, 720\}$. Avg means the average metrics for four prediction lengths. $\uparrow$ indicates improved performance and $\downarrow$ denotes decreasing performance.

Method			iTransformer		PatchTST		RLinear		TimesNet	
Metric			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	Original	96	0.148	0.240	0.195	0.285	0.201	0.281	0.168	0.272
		192	0.162	0.253	0.199	0.289	0.201	0.283	0.184	0.289
		336	0.178	0.269	0.215	0.305	0.215	0.298	0.198	0.300
		720	0.225	0.317	0.256	0.337	0.257	0.331	0.220	0.320
		Avg	0.178	0.270	0.216	0.304	0.219	0.298	0.192	0.295
	w/ GDD-MLP and Channel Mixup	96	0.142$\uparrow$	0.238$\uparrow$	0.151$\uparrow$	0.250$\uparrow$	0.182$\uparrow$	0.274$\uparrow$	0.163$\uparrow$	0.268$\uparrow$
		192	0.161$\uparrow$	0.255$\downarrow$	0.165$\uparrow$	0.259$\uparrow$	0.187$\uparrow$	0.276$\uparrow$	0.172$\uparrow$	0.274$\uparrow$
		336	0.179$\downarrow$	0.274$\downarrow$	0.178$\uparrow$	0.276$\uparrow$	0.202$\uparrow$	0.291$\uparrow$	0.183$\uparrow$	0.286$\uparrow$
		720	0.196$\uparrow$	0.290$\uparrow$	0.217$\uparrow$	0.306$\uparrow$	0.239$\uparrow$	0.321$\uparrow$	0.211$\uparrow$	0.308$\uparrow$
		Avg	0.170$\uparrow$	0.264$\uparrow$	0.178$\uparrow$	0.273$\uparrow$	0.203$\uparrow$	0.290$\uparrow$	0.182$\uparrow$	0.284$\uparrow$
	Promotion Count (Promotion / Total)	-	4 / 5	3 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5
Weather	Original	96	0.174	0.214	0.177	0.218	0.192	0.232	0.172	0.220
		192	0.221	0.254	0.225	0.256	0.240	0.271	0.219	0.261
		336	0.278	0.296	0.278	0.297	0.292	0.307	0.280	0.306
		720	0.358	0.349	0.354	0.348	0.364	0.353	0.365	0.359
		Avg	0.258	0.279	0.259	0.281	0.272	0.291	0.259	0.287
	w/ GDD-MLP and Channel Mixup	96	0.164$\uparrow$	0.207$\uparrow$	0.164$\uparrow$	0.207$\uparrow$	0.181$\uparrow$	0.228$\uparrow$	0.166$\uparrow$	0.213$\uparrow$
		192	0.214$\uparrow$	0.250$\uparrow$	0.211$\uparrow$	0.249$\uparrow$	0.223$\uparrow$	0.264$\uparrow$	0.221$\downarrow$	0.259$\uparrow$
		336	0.272$\uparrow$	0.292$\uparrow$	0.269$\uparrow$	0.289$\uparrow$	0.277$\uparrow$	0.301$\uparrow$	0.282$\downarrow$	0.305$\uparrow$
		720	0.350$\uparrow$	0.345$\uparrow$	0.346$\uparrow$	0.342$\uparrow$	0.352$\uparrow$	0.347$\uparrow$	0.357$\uparrow$	0.353$\uparrow$
		Avg	0.250$\uparrow$	0.274$\uparrow$	0.248$\uparrow$	0.272$\uparrow$	0.258$\uparrow$	0.285$\uparrow$	0.257$\uparrow$	0.282$\uparrow$
	Promotion Count (Promotion / Total)	-	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	3 / 5	5 / 5

B.2 Baseline Modification

In Section 5.3, we evaluate the effects of GDD-MLP and Channel Mixup modules on four state-of-the-art models: iTransformer (Liu et al., 2023a) and PatchTST (Nie et al., 2022) (Transformer-based), RLinear (Li et al., 2023) (Linear-based), and TimesNet (Wu et al., 2022) (Convolution-based). During experiments, we retain the original architecture unchanged but process the input via Channel Mixup during training and insert the GDD-MLP module into the original model. The modified frameworks of these models are shown in Fig. 10. All models adopt instance norm or RevIN (Kim et al., 2021) based on their original settings. We only tune the standard deviation σ, and learning rate lr for optimal performance. The full results are shown in Table 8.

C More Evaluation

C.1 Hyperparameter Sensitivity

We conduct a thorough evaluation of the hyperparameters influencing HIPPO-SSM's performance, focusing on two critical factors: the standard deviation (σ) in the Channel Mixup mechanism and the expansion rate (r) for the GDD-MLP. Experiments are performed on the ETTh1 and ETTm1 datasets, maintaining a fixed look-back window and forecasting horizon of 96. All other hyperparameters remain consistent with those reported in Table 1. The results, visualized in Fig. 11, highlight the following: (1) The standard deviation (σ) controls the extent of perturbation and mixing across channels, directly influencing the robustness of the model. Proper tuning is essential, as it governs the balance between stability and generalization. A standard normal distribution tends to yield stable performance across various scenarios, likely due to its balance in perturbation magnitude. (2) The expansion rate (r) controls the dimensionality of the hidden layers in the GDD-MLP. While larger hidden layers increase computational demands, they may also detrimentally affect model performance. This suggests that excessive emphasis on cross-channel dependencies can diminish the model's ability to capture cross-temporal dynamics, which are more critical for multivariate time series forecasting. Conversely, an expansion rate that is too small can result in underfitting, as the model may lack sufficient capacity to capture the necessary dependencies across channels.

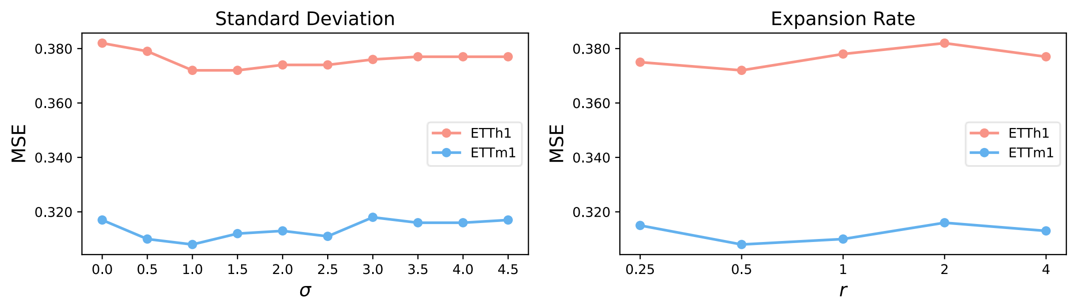


Figure 11: Hyperparameter sensitivity of the standard deviation σ for Channel Mixup (left) and the expansion rate r for GDD-MLP (right).

C.2 Upper Bound Evaluation

Considering that the performance of different models is influenced by the look-back length, we further compare our model with state-of-the-art frameworks under the **optimal** look-back length. As shown in Table 9, we compare the performance of each model using their best look-back window. For HIPPO-SSM, we search the look-back length in $\{96, 192, 336, 512\}$. For other benchmarks, we rerun iTransformer since its look-back length is fixed at 96 in the original paper, and we collect results for other models from tables in ModernTCN (Luo & Wang, 2024), TimeMixer (Wang et al., 2023), and TiDE (Das et al., 2023). The results indicate that our model still achieves state-of-the-art performance.

Table 9: Full results of the long-term forecasting task under the **optimal** look-back window. We search the look-back window of HIPPO-SSM in $\{96, 192, 336, 512\}$. We report the performance of four prediction lengths $T \in \{96, 192, 336, 720\}$. Avg means the average metrics for four prediction lengths. The best is highlighted in **red** and the runner-up in $\underline{\text{blue}}$.

Models		**HIPPO-SSM (My)**		ModernTCN (2024)		iTransformer (2023a)		TimeMixer (2023)		RLinear (2023)		PatchTST (2022)		Crossformer (2022)		TiDE (2023)		TimesNet (2022)		MICN (2022)		DLinear (2023)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.218**	**0.329**	0.292	0.346	0.295	0.344	0.291	$\underline{0.340}$	0.301	0.342	$\underline{0.290}$	0.342	0.316	0.373	0.306	0.349	0.338	0.375	0.314	0.360	0.299	0.343
	192	$\underline{0.267}$	**0.357**	$\underline{0.332}$	0.368	0.336	0.370	**0.327**	0.365	0.335	$\underline{0.363}$	$\underline{0.332}$	0.369	0.377	0.411	0.335	0.366	0.371	0.387	0.359	0.387	0.335	0.365
	336	**0.292**	**0.378**	0.365	0.391	0.373	0.391	$\underline{0.360}$	$\underline{0.381}$	0.370	0.383	0.366	0.392	0.431	0.442	0.364	0.384	0.410	0.411	0.398	0.413	0.369	0.386
	720	$\underline{0.349}$	**0.410**	0.416	0.417	0.432	0.427	0.415	0.417	0.425	0.414	0.416	0.420	0.600	0.547	**0.413**	$\underline{0.413}$	0.478	0.450	0.459	0.464	0.425	0.421
	Avg	**0.281**	**0.368**	0.351	0.381	0.359	0.383	$\underline{0.348}$	$\underline{0.375}$	0.358	0.376	0.351	0.381	0.431	0.443	0.355	0.378	0.400	0.406	0.383	0.406	0.357	0.379
ETTh1	96	**0.265**	**0.387**	0.368	0.394	0.386	0.405	$\underline{0.361}$	$\underline{0.390}$	0.366	0.391	0.370	0.399	0.386	0.429	0.375	0.398	0.384	0.402	0.396	0.427	0.375	0.399
	192	**0.334**	0.415	0.405	$\underline{0.413}$	0.441	0.436	0.409	0.414	$\underline{0.404}$	**0.412**	0.413	0.421	0.419	0.444	0.412	0.422	0.436	0.429	0.430	0.453	0.405	0.416
	336	0.367	0.437	**0.391**	**0.412**	0.461	0.452	0.430	0.429	$\underline{0.420}$	$\underline{0.423}$	0.422	0.436	0.440	0.461	0.435	0.433	0.491	0.469	0.433	0.458	0.439	0.443
	720	**0.384**	**0.451**	0.450	0.461	0.503	0.491	0.445	0.460	$\underline{0.442}$	$\underline{0.456}$	0.447	0.466	0.519	0.524	0.454	0.465	0.521	0.500	0.474	0.508	0.472	0.490
	Avg	**0.338**	0.422	$\underline{0.404}$	**0.420**	0.448	0.446	0.411	0.423	0.408	$\underline{0.421}$	0.413	0.431	0.441	0.465	0.419	0.430	0.458	0.450	0.433	0.462	0.423	0.437
Electricity	96	**0.123**	**0.220**	$\underline{0.129}$	0.226	0.132	0.228	$\underline{0.129}$	0.224	0.140	0.235	$\underline{0.129}$	$\underline{0.222}$	0.219	0.314	0.132	0.229	0.168	0.272	0.159	0.267	0.153	0.237
	192	0.137	$\underline{0.239}$	$\underline{0.143}$	$\underline{0.239}$	0.154	0.247	**0.140**	**0.220**	0.154	0.248	0.147	0.240	0.231	0.322	0.147	0.243	0.184	0.289	0.168	0.279	0.152	0.249
	336	$\underline{0.162}$	$\underline{0.258}$	**0.161**	0.259	0.172	0.266	**0.161**	**0.255**	0.171	0.264	0.163	0.259	0.246	0.337	**0.161**	0.261	0.198	0.300	0.196	0.308	0.169	0.267
	720	**0.175**	**0.278**	$\underline{0.191}$	$\underline{0.286}$	0.210	0.303	0.194	0.287	0.209	0.297	0.197	0.290	0.280	0.363	0.196	0.294	0.220	0.320	0.203	0.312	0.233	0.344
	Avg	**0.149**	$\underline{0.249}$	**0.156**	0.253	0.167	0.261	**0.156**	**0.246**	0.169	0.261	$\underline{0.159}$	0.253	0.244	0.334	$\underline{0.159}$	0.257	0.192	0.295	0.182	0.292	0.177	0.274
Weather	96	**0.141**	**0.180**	0.149	0.200	0.162	0.212	$\underline{0.147}$	$\underline{0.197}$	0.175	0.225	0.149	0.198	0.153	0.217	0.166	0.222	0.172	0.220	0.161	0.226	0.152	0.237
	192	**0.187**	**0.229**	0.196	0.245	0.204	0.252	$\underline{0.189}$	$\underline{0.239}$	0.218	0.260	0.194	0.241	0.197	0.269	0.209	0.263	0.219	0.261	0.220	0.283	0.220	0.282
	336	**0.238**	**0.267**	**0.238**	$\underline{0.277}$	0.256	0.290	$\underline{0.241}$	0.280	0.265	0.294	0.245	0.282	0.252	0.311	0.254	0.301	0.280	0.306	0.275	0.328	0.265	0.319
	720	**0.310**	**0.321**	0.314	0.334	0.326	0.338	**0.310**	$\underline{0.330}$	0.329	0.339	0.314	0.334	0.318	0.363	0.313	0.340	0.365	0.359	$\underline{0.311}$	0.356	0.323	0.362
	Avg	**0.220**	**0.257**	0.224	0.264	0.237	0.273	$\underline{0.222}$	$\underline{0.262}$	0.247	0.279	0.226	0.264	0.230	0.290	0.236	0.282	0.259	0.287	0.242	0.298	0.240	0.300

C.3 Effectiveness of MAE Loss

The choice of the loss function plays a pivotal role in both the optimization process and the convergence behavior of time series forecasting models. While many prior models predominantly employ L2 loss, also known as Mean Squared Error (MSE), our proposed model adopts L1 loss, also referred to as Mean Absolute Error (MAE). The rationale behind this choice stems from empirical observations across various datasets, where the mean absolute error is typically small. In such cases, the gradients produced by L2 loss tend to be small, which can impede the model's ability to escape local minima during training, as depicted in Fig. 12. Moreover, time series data often exhibit outliers, and L2 loss is known to be more sensitive to these outliers, which can lead to instability during training. In contrast, L1 loss is more robust to outliers, promoting greater stability in model training and convergence.

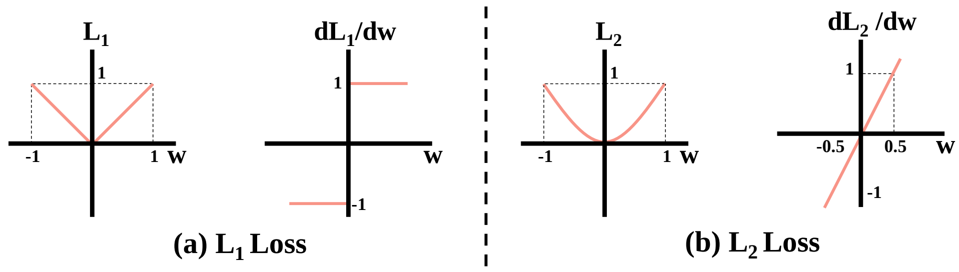


Figure 12: Visualization of L1 and L2 loss.

To substantiate this choice, we evaluate the performance of five state-of-the-art models from different architectural families: HIPPO-SSM (our proposed model, SSM-based), ModernTCN (Convolution-based), iTransformer (Transformer-based), TimeMixer (Linear-based), and PatchTST (Transformer-based). We assess the performance of these models using both MSE and MAE loss functions. The results are exhibited in Table 10, where **37** out of 40 cases have improved performance after switching from MSE loss to MAE loss. Even when considering different loss functions, our model consistently outperforms the alternatives, achieving the best average performance, which underscores the effectiveness and robustness of HIPPO-SSM under various training conditions.

Table 10: Performance promotion obtained by changing MSE loss to MAE loss. We fix the look-back window $L = 96$ and report the average performance of four prediction lengths $T \in \{96, 192, 336, 720\}$. The best under each loss is highlighted in **red**. ↑ indicates improved performance and ↓ denotes decreasing performance.

Method		HIPPO-SSM		ModernTCN		iTransformer		TimeMixer		PatchTST	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm2	MSE Loss	**0.273**	0.324	**0.278**	**0.322**	0.288	0.332	0.279	0.325	0.281	0.326
	MAE Loss	**0.218**↑	**0.316**↑	0.276↑	0.317↑	0.283↑	0.322↑	0.274↑	0.317↑	0.279↑	0.318↑
ETTh2	MSE Loss	**0.304**	**0.403**	0.381	0.404	0.383	0.407	0.389	0.409	0.387	0.407
	MAE Loss	0.245↑	**0.391**↑	0.379↑	0.398↑	0.379↑	0.400↑	0.377↑	0.396↑	**0.363**↑	**0.391**↑
Electricity	MSE Loss	**0.172**	**0.265**	0.197	0.282	0.178	0.270	0.183	0.272	0.216	0.304
	MAE Loss	**0.169**↑	**0.258**↑	0.211↓	0.290↓	0.175↑	0.259↑	0.185↓	0.271↑	0.206↑	0.285↑
Weather	MSE Loss	**0.240**	**0.270**	**0.240**	0.271	0.258	0.279	0.245	0.274	0.259	0.281
	MAE Loss	0.237↑	**0.259**↑	**0.236**↑	0.263↑	0.255↑	0.271↑	0.245↑	0.265↑	0.255↑	0.270↑

C.4 LIMITATIONS

In this work, we mainly focus on the multivariate time series forecasting task with endogenous variables, meaning that the values we aim to predict and the values treated as features only differ in terms of time steps. However, real-world scenarios often involve the influence of exogenous variables on the variables we seek to predict, a topic extensively discussed in prior research (Wang et al., 2024c). In addition, the experimental results show that our model exhibits significant improvements on some datasets with large-scale channels, such as Weather and Electricity. However, the improvements in MAE are relatively limited on the Traffic dataset, which contains 862 channels. This discrepancy could be attributed to the pronounced periodicity observed in traffic data compared to other domains. These periodic patterns are highly time-dependent, causing different channels to exhibit similar characteristics and obscuring their physical interconnections. Therefore, incorporating external variables and utilizing prior knowledge about the relationships between channels, such as the connectivity of traffic roads, might further enhance the prediction accuracy.

Table 11: Full results of the long-term forecasting task. We fix the look-back window $L = 96$ and make predictions for $T = \{96, 192, 336, 720\}$. Avg means the average metrics for four prediction lengths. The best is highlighted in **red** and the runner-up in blue. We report the average results of three random seeds.

Models		**HIPPO-SSM (Ours)**		ModernTCN (2024)		iTransformer (2023a)		TimeMixer (2023)		RLinear (2023)		PatchTST (2022)		Crossformer (2022)		TiDE (2023)		TimesNet (2022)		MICN (2022)		DLinear (2023)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.218**	**0.338**	0.317	0.362	0.334	0.368	0.320	0.355	0.355	0.376	0.329	0.367	0.404	0.426	0.364	0.387	0.338	0.375	0.317	0.367	0.345	0.372
	192	**0.267**	**0.364**	0.363	0.389	0.377	0.391	0.362	0.382	0.391	0.392	0.367	0.385	0.450	0.451	0.398	0.404	0.374	0.387	0.382	0.413	0.380	0.389
	336	**0.292**	**0.389**	0.403	0.412	0.426	0.420	0.396	0.406	0.424	0.415	0.399	0.410	0.532	0.515	0.428	0.425	0.410	0.411	0.417	0.443	0.413	0.413
	720	**0.349**	**0.425**	0.461	0.443	0.491	0.459	0.458	0.445	0.487	0.450	0.454	0.439	0.666	0.589	0.487	0.461	0.478	0.450	0.511	0.505	0.474	0.453
	Avg	**0.281**	**0.379**	0.386	0.401	0.407	0.410	0.384	0.397	0.414	0.407	0.387	0.400	0.513	0.496	0.419	0.419	0.400	0.406	0.407	0.432	0.403	0.407
ETTm2	96	**0.171**	**0.248**	0.173	0.255	0.180	0.264	0.176	0.259	0.182	0.265	0.175	0.259	0.287	0.366	0.207	0.305	0.187	0.267	0.182	0.278	0.193	0.292
	192	**0.235**	**0.292**	**0.235**	0.296	0.250	0.309	0.242	0.303	0.246	0.304	0.241	0.302	0.414	0.492	0.290	0.364	0.249	0.309	0.288	0.357	0.284	0.362
	336	**0.296**	**0.334**	0.308	0.344	0.311	0.348	0.303	0.339	0.307	0.342	0.305	0.343	0.597	0.542	0.377	0.422	0.321	0.351	0.370	0.413	0.369	0.427
	720	**0.392**	**0.391**	0.398	0.394	0.412	0.407	0.396	0.399	0.407	0.398	0.402	0.400	1.730	1.042	0.558	0.524	0.408	0.403	0.519	0.495	0.554	0.522
	Avg	**0.273**	**0.316**	0.278	0.322	0.288	0.332	0.279	0.325	0.286	0.327	0.281	0.326	0.757	0.610	0.358	0.404	0.291	0.333	0.339	0.386	0.350	0.401
ETTh1	96	**0.265**	**0.386**	0.386	0.394	0.386	0.405	0.384	0.400	0.386	0.395	0.414	0.419	0.423	0.448	0.479	0.464	0.384	0.402	0.417	0.436	0.386	0.400
	192	**0.334**	**0.416**	0.436	0.423	0.441	0.436	0.437	0.429	0.439	0.424	0.460	0.445	0.471	0.474	0.525	0.492	0.436	0.429	0.488	0.476	0.437	0.432
	336	**0.367**	**0.438**	0.479	0.445	0.487	0.458	0.472	0.446	0.479	0.446	0.501	0.466	0.570	0.546	0.565	0.515	0.491	0.469	0.599	0.549	0.481	0.459
	720	**0.384**	**0.461**	0.481	0.469	0.503	0.491	0.586	0.531	0.481	0.470	0.500	0.488	0.653	0.621	0.594	0.558	0.521	0.500	0.730	0.634	0.519	0.516
	Avg	**0.338**	**0.425**	0.445	0.432	0.454	0.447	0.470	0.451	0.446	0.434	0.469	0.454	0.529	0.522	0.541	0.507	0.485	0.450	0.559	0.524	0.456	0.452
ETTh2	96	**0.245**	**0.329**	0.292	0.340	0.297	0.349	0.297	0.348	0.288	0.338	0.302	0.348	0.745	0.584	0.400	0.440	0.340	0.374	0.355	0.402	0.333	0.387
	192	**0.281**	**0.381**	0.377	0.395	0.380	0.400	0.369	0.392	0.374	0.390	0.388	0.400	0.877	0.656	0.528	0.509	0.402	0.414	0.511	0.491	0.477	0.476
	336	**0.326**	**0.419**	0.424	0.434	0.428	0.432	0.427	0.435	0.415	0.426	0.426	0.433	1.043	0.732	0.643	0.571	0.452	0.452	0.618	0.551	0.594	0.541
	720	**0.329**	**0.435**	0.433	0.448	0.427	0.445	0.462	0.463	0.420	0.440	0.431	0.446	1.104	0.763	0.874	0.679	0.462	0.468	0.835	0.660	0.831	0.657
	Avg	**0.295**	**0.391**	0.381	0.404	0.383	0.407	0.389	0.409	0.374	0.398	0.387	0.407	0.942	0.684	0.611	0.550	0.414	0.427	0.580	0.526	0.559	0.515
Electricity	96	**0.141**	**0.231**	0.173	0.260	0.148	0.240	0.153	0.244	0.201	0.281	0.195	0.285	0.219	0.314	0.237	0.329	0.168	0.272	0.172	0.285	0.197	0.282
	192	**0.157**	**0.245**	0.181	0.267	0.162	0.253	0.168	0.259	0.201	0.283	0.199	0.289	0.231	0.322	0.236	0.330	0.184	0.289	0.177	0.287	0.196	0.285
	336	**0.175**	**0.265**	0.196	0.283	0.178	0.269	0.185	0.275	0.215	0.298	0.215	0.305	0.246	0.337	0.249	0.344	0.198	0.300	0.186	0.297	0.209	0.301
	720	**0.203**	**0.289**	0.238	0.316	0.225	0.319	0.227	0.312	0.257	0.331	0.256	0.337	0.280	0.363	0.284	0.373	0.220	0.320	0.204	0.314	0.245	0.333
	Avg	**0.169**	**0.258**	0.197	0.282	0.178	0.270	0.183	0.272	0.219	0.298	0.216	0.304	0.244	0.334	0.251	0.344	0.192	0.295	0.185	0.296	0.212	0.300
Weather	96	**0.150**	**0.187**	0.155	0.203	0.174	0.214	0.162	0.208	0.192	0.232	0.177	0.218	0.158	0.230	0.202	0.261	0.172	0.220	0.194	0.253	0.196	0.255
	192	**0.200**	**0.236**	0.202	0.247	0.221	0.254	0.208	0.252	0.240	0.271	0.225	0.259	0.206	0.277	0.242	0.298	0.219	0.261	0.240	0.301	0.237	0.296
	336	**0.260**	**0.281**	0.263	0.293	0.278	0.296	0.263	0.293	0.292	0.307	0.278	0.297	0.272	0.335	0.287	0.335	0.280	0.306	0.284	0.334	0.283	0.335
	720	**0.347**	**0.339**	0.334	0.343	0.358	0.349	0.345	0.345	0.364	0.353	0.354	0.348	0.398	0.418	0.351	0.386	0.365	0.359	0.351	0.387	0.345	0.381
	Avg	**0.237**	**0.259**	0.240	0.271	0.258	0.279	0.245	0.274	0.272	0.291	0.259	0.281	0.259	0.315	0.271	0.320	0.259	0.287	0.267	0.318	0.265	0.317
Traffic	96	0.414	**0.251**	0.550	0.355	**0.395**	0.268	0.473	0.287	0.649	0.389	0.544	0.359	0.522	0.290	0.805	0.493	0.593	0.321	0.521	0.310	0.650	0.396
	192	0.432	**0.257**	0.527	0.337	**0.417**	0.276	0.486	0.294	0.601	0.366	0.540	0.354	0.530	0.293	0.756	0.474	0.617	0.336	0.536	0.314	0.598	0.370
	336	0.446	**0.265**	0.537	0.342	**0.433**	0.283	0.488	0.298	0.609	0.369	0.551	0.358	0.558	0.305	0.762	0.477	0.629	0.336	0.550	0.321	0.605	0.373
	720	0.485	**0.286**	0.570	0.359	**0.467**	0.302	0.536	0.314	0.647	0.387	0.586	0.375	0.589	0.328	0.719	0.449	0.640	0.350	0.571	0.329	0.645	0.394
	Avg	0.444	**0.265**	0.546	0.348	**0.428**	0.282	0.496	0.298	0.626	0.378	0.555	0.362	0.550	0.304	0.760	0.473	0.620	0.336	0.544	0.319	0.625	0.383
1st Count		**30**	**35**	1	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12: Full results of ablation studies for ETTm1, ETTm2, ETTh1, and ETTh2. We fix the look-back window $L = 96$ and make predictions for $T = \{96, 192, 336, 720\}$. The best is highlighted in **red** and the runner-up in blue.

Channel Mixup	GDD MLP	Metric	ETTm1				ETTm2				ETTh1				ETTh2			
			96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720
-	-	MSE	0.320	0.373	0.401	0.455	0.175	0.244	0.304	0.404	0.374	**0.422**	**0.458**	0.471	0.284	**0.361**	**0.410**	**0.415**
		MAE	0.345	0.373	0.398	0.434	0.254	0.300	0.339	0.397	0.390	0.418	0.439	0.463	0.331	**0.380**	**0.419**	**0.435**
✓	-	MSE	0.312	0.370	0.399	0.453	0.177	0.242	0.301	0.402	**0.372**	0.423	0.462	**0.462**	0.284	0.363	0.428	0.419
		MAE	0.339	0.369	0.390	0.429	0.255	0.297	0.337	0.396	0.390	0.419	**0.437**	0.459	0.331	**0.380**	0.427	**0.435**
-	✓	MSE	0.317	0.363	0.391	0.481	0.172	0.238	**0.293**	0.397	0.382	0.430	0.470	0.465	0.285	0.364	0.417	0.423
		MAE	0.341	0.372	0.394	0.441	0.250	0.294	**0.333**	0.393	0.394	0.420	0.440	**0.457**	0.332	0.382	0.422	0.439
✓	✓	MSE	**0.308**	**0.359**	**0.390**	**0.447**	**0.171**	**0.235**	0.296	**0.392**	**0.372**	**0.422**	0.466	0.470	**0.281**	**0.361**	0.413	0.419
		MAE	**0.338**	**0.364**	**0.389**	**0.425**	**0.248**	**0.292**	0.334	**0.391**	**0.386**	**0.416**	0.438	0.461	**0.329**	0.381	**0.419**	**0.435**

Table 13: Full results of ablation studies for Electricity, Weather, and Traffic. We fix the look-back window $L = 96$ and make predictions for $T = \{96, 192, 336, 720\}$. The best is highlighted in **red** and the runner-up in blue.

Channel Mixup	GDD MLP	Metric	Electricity				Weather				Traffic			
			96	192	336	720	96	192	336	720	96	192	336	720
-	-	MSE	0.163	0.173	0.190	0.233	0.173	0.219	0.276	0.352	0.454	0.467	0.479	0.514
		MAE	0.243	0.253	0.270	0.306	0.207	0.247	0.289	0.339	**0.251**	**0.257**	**0.264**	**0.284**
✓	-	MSE	0.164	0.173	0.190	0.229	0.173	0.218	0.273	0.352	0.467	0.463	0.483	0.518
		MAE	0.243	0.252	0.270	0.302	0.206	0.246	0.287	0.339	0.257	0.259	0.268	0.285
-	✓	MSE	0.147	0.166	**0.174**	0.212	**0.150**	**0.199**	0.269	0.345	0.471	0.499	0.546	0.585
		MAE	0.240	0.258	**0.265**	0.299	**0.186**	**0.236**	0.287	0.337	0.272	0.281	0.289	0.300
✓	✓	MSE	**0.141**	**0.157**	0.175	**0.203**	**0.150**	0.200	**0.260**	**0.339**	**0.414**	**0.432**	**0.446**	**0.485**
		MAE	**0.231**	**0.245**	**0.265**	**0.289**	0.187	**0.236**	**0.281**	**0.334**	**0.251**	**0.257**	0.265	0.286

Table 14: Full results of the long-term forecasting task with SSM variants. We fix the look-back window $L = 96$ and make predictions for $T = \{96, 192, 336, 720\}$. Avg means the average metrics for four prediction lengths. The best is highlighted in **red** and the runner-up in blue.

Models		**HIPPO-SSM (Ours)**		Bi-Mamba+ (2024)		SiMBA (2024)		Time-SSM (2024)		SAMBA (2024)		S-Mamba (2024d)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.218**	**0.338**	0.320	0.360	0.324	0.360	0.329	0.365	0.315	0.357	0.331	0.368
	192	**0.267**	**0.364**	0.361	0.383	0.363	0.382	0.370	0.379	0.360	0.383	0.371	0.387
	336	0.390	**0.292**	**0.386**	0.402	0.395	0.405	0.396	0.402	0.389	0.405	0.417	0.418
	720	0.349	**0.425**	**0.445**	0.437	0.451	0.437	0.449	0.440	0.448	0.440	0.471	0.448
	Avg	**0.281**	**0.379**	0.378	0.396	0.383	0.396	0.386	0.397	0.378	0.396	0.398	0.405
ETTm2	96	**0.171**	**0.248**	0.176	0.263	0.177	0.263	0.176	0.260	0.172	0.259	0.179	0.263
	192	**0.235**	**0.292**	0.242	0.304	0.245	0.306	0.246	0.305	0.238	0.301	0.253	0.310
	336	**0.296**	**0.334**	0.304	0.344	0.304	0.343	0.305	0.344	0.300	0.340	0.312	0.348
	720	**0.392**	**0.391**	0.402	0.402	0.400	0.399	0.406	0.405	0.394	0.394	0.412	0.408
	Avg	**0.273**	**0.316**	0.281	0.328	0.282	0.328	0.283	0.329	0.276	0.324	0.289	0.332
ETTh1	96	**0.265**	**0.386**	0.378	0.395	0.379	0.395	0.377	0.394	0.376	0.400	0.386	0.406
	192	**0.334**	**0.416**	0.427	0.428	0.432	0.424	0.423	0.424	0.432	0.429	0.448	0.444
	336	**0.367**	0.438	0.471	0.445	0.473	0.443	**0.466**	**0.437**	0.477	**0.437**	0.494	0.468
	720	0.384	0.461	0.470	0.457	0.483	0.469	**0.452**	**0.448**	0.488	0.471	0.493	0.488
	Avg	0.338	**0.425**	0.437	0.431	0.442	0.433	**0.430**	0.426	0.443	0.434	0.455	0.452
ETTh2	96	**0.245**	**0.329**	0.291	0.342	0.290	0.339	0.290	0.341	0.288	0.340	0.298	0.349
	192	**0.281**	**0.381**	0.368	0.392	0.373	0.390	0.368	0.387	0.373	0.390	0.379	0.398
	336	0.326	0.419	0.407	0.424	**0.376**	**0.406**	0.416	0.430	0.380	**0.406**	0.417	0.432
	720	0.329	0.435	0.421	0.439	**0.407**	**0.431**	0.424	0.439	0.412	0.432	0.431	0.449
	Avg	0.295	**0.391**	0.372	0.399	**0.362**	0.392	0.375	0.399	0.363	0.392	0.381	0.407
Electricity	96	0.141	**0.231**	**0.140**	0.238	0.165	0.253	-	-	0.146	0.244	0.142	0.238
	192	0.157	**0.245**	**0.155**	0.253	0.173	0.262	-	-	0.164	0.260	0.169	0.267
	336	0.175	**0.265**	**0.170**	0.269	0.188	0.277	-	-	0.179	0.274	0.178	0.275
	720	0.203	**0.289**	**0.197**	0.293	0.214	0.305	-	-	0.206	0.300	0.207	0.303
	Avg	0.169	**0.258**	**0.166**	0.263	0.185	0.274	-	-	0.174	0.270	0.174	0.271
Weather	96	**0.150**	**0.187**	0.159	0.205	0.176	0.219	0.167	0.212	0.165	0.211	0.166	0.210
	192	**0.200**	**0.236**	0.205	0.249	0.222	0.260	0.217	0.255	0.214	0.255	0.215	0.253
	336	**0.260**	**0.281**	0.264	0.291	0.275	0.297	0.274	0.291	0.271	0.297	0.276	0.298
	720	**0.347**	**0.339**	0.343	0.344	0.350	0.349	0.351	0.345	0.346	0.347	0.353	0.349
	Avg	**0.237**	**0.259**	0.243	0.272	0.256	0.281	0.252	0.277	0.249	0.278	0.253	0.278
Traffic	96	0.414	**0.251**	**0.375**	0.258	0.468	0.268	-	-	0.403	0.270	0.381	0.261
	192	0.432	**0.257**	**0.394**	0.269	0.413	0.317	-	-	0.427	0.278	0.397	0.267
	336	0.446	**0.265**	**0.406**	0.274	0.529	0.284	-	-	0.440	0.284	0.423	0.276
	720	0.485	**0.286**	**0.440**	0.288	0.564	0.297	-	-	0.470	0.302	0.458	0.300
	Avg	0.444	**0.265**	**0.404**	0.272	0.494	0.292	-	-	0.435	0.284	0.415	0.276
1stCount		**18**	**31**	12	0	3	2	3	2	0	2	0	0

D Full Results

D.1 Full Main Results

Here, we present the complete results of all chosen models and our HIPPO-SSM under four different prediction lengths in Table 11. Generally, the proposed HIPPO-SSM demonstrates stable performance across various datasets and prediction lengths, consistently ranking among the top performers. Specifically, our model ranks top 1 in **65** out of 70 settings and ranks top 2 in **all** settings, while the runner-up, iTransformer (Liu et al., 2023a) ranks top 1 in only 5 settings and top 2 in 18 settings.

D.2 Full Ablation Results of Module Design

In the main text, we only present the improvements brought by the proposed modules in the average case. To validate the effectiveness of our design, we provide the complete results in Table 12 and Table 13. Consistent with our claims, GDD-MLP alone can easily lead to oversmoothing. However, when combined with the Channel Mixup, our model consistently achieves state-of-the-art performance.

D.3 Comparison with Other SSMs

Since Mamba (Gu & Dao, 2023) was proposed, there have been many remarkable works trying to apply it to multivariate time series prediction, e.g., Bi-Mamba+ (Liang et al., 2024), SiMBA (Patro & Agneeswaran, 2024), HIPPO-SSM (Hu et al., 2024), SAMBA (Weng et al., 2024), and S-Mamba (Wang et al., 2024d). In Table 14, we compare our HIPPO-SSM with these methods on the same 7 real-world datasets. It is worth noting that since some methods are tested on other datasets in the original paper, there will be missing results on some of the datasets we use. Among all these Mamba variants, our model achieves the best average performance. Specifically, our model ranks top 1 in **49** out of 70 settings and ranks top 2 in **59** settings, while the runner-up, Bi-Mamba+ ranks top 1 in only 12 settings and top 2 in 39 settings. The results demonstrate the superiority of our model.

E Proofs

For a differential equation of $x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$, its general solution is:

$$x(t) = e^{\boldsymbol{A}(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{\boldsymbol{A}(t-s)}\boldsymbol{B}u(s)\mathrm{d}s.$$

Let's start with the one-dimensional system represented by the scalar differential equation

$$\dot{x}(t) = ax(t) + bu(t) \quad x(t_0) = x_0 \tag{28}$$

in which a and b are scalar constants, and $u(t)$ is a given scalar input signal. A traditional approach for deriving a solution formula for the scalar state $x(t)$ is to multiply both sides of the differential equation by the integrating factor $e^{-a(t-t_0)}$ to yield

$$\begin{aligned}\frac{d}{dt}\left(e^{-a(t-t_0)}x(t)\right) &= e^{-a(t-t_0)}\dot{x}(t) - e^{-a(t-t_0)}ax(t)\\ &= e^{-a(t-t_0)}bu(t)\end{aligned}$$

We next integrate from t_0 to t and invoke the fundamental theorem of calculus to obtain

$$\begin{aligned}e^{-a(t-t_0)}x(t) - e^{-a(t-t_0)}x(t_0) &= \int_{t_0}^{t} \frac{d}{dt}\left(e^{-a(s-t_0)}x(s)\right)ds\\ &= \int_{t_0}^{t} e^{-a(s-t_0)}bu(s)ds.\end{aligned}$$

After multiplying through by $e^{a(t-t_0)}$ and some manipulation, we get

$$x(t) = e^{a(t-t_0)}x_0 + \int_{t_0}^{t} e^{a(t-s)}bu(s)ds \tag{29}$$

which expresses the state response $x(t)$ as a sum of terms, the first owing to the given initial state $x(t_0) = x_0$ and the second owing to the specified input signal $u(t)$. Notice that the first component characterizes the state response when the input signal is identically zero. We therefore refer to the first term as the zero-input response component. Similarly, the second component characterizes the state response for zero initial state, referred to as the zero-state response component.

Now, let us derive a closed-form solution to the n-dimensional linear time-invariant state equation equation 7 given a specified initial state $x(t_0) = x_0$ and input vector $u(t)$. We begin with a related homogeneous matrix differential equation

$$\dot{X}(t) = AX(t) \quad X(t_0) = I \tag{30}$$

where I is the $n \times n$ identity matrix. We assume an infinite power series form for the solution

$$X(t) = \sum_{k=0}^{\infty} X_k (t - t_0)^k \tag{31}$$

Each term in the sum involves an $n \times n$ matrix X_k to be determined and depends only on the elapsed time $t - t_0$, reflecting the time-invariance of the state equation. The initial condition for Equation equation 30 yields $X(t_0) = X_0 = I$. Substituting Equation equation 31 into Equation equation 30, formally differentiating term by term with respect to time, and shifting the summation index gives

$$\begin{aligned} \sum_{k=0}^{\infty}(k+1)X_{k+1}(t-t_0)^k &= A\left(\sum_{k=0}^{\infty} X_k (t-t_0)^k\right) \\ &= \sum_{k=0}^{\infty} AX_k (t-t_0)^k \end{aligned}$$

By equating like powers of $t - t_0$, we obtain the recursive relationship

$$X_{k+1} = \frac{1}{k+1} AX_k \quad k \geq 0$$

which, when initialized with $X_0 = I$, leads to

$$X_k = \frac{1}{k!} A^k \quad k \geq 0$$

Substituting this result into the power series equation 31 yields

$$X(t) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k (t-t_0)^k$$

We note here that the infinite power series equation 31 has the requisite convergence properties so that the infinite power series resulting from term by-term differentiation converges to $\dot{X}(t)$, and Equation equation 30 is satisfied.

$$\begin{aligned} \sum_{k=0}^{\infty}(k+1)X_{k+1}(t-t_0)^k &= A\left(\sum_{k=0}^{\infty} X_k (t-t_0)^k\right) \\ &= \sum_{k=0}^{\infty} AX_k (t-t_0)^k \end{aligned}$$

By equating like powers of $t - t_0$, we obtain the recursive relationship

$$X_{k+1} = \frac{1}{k+1} AX_k \quad k \geq 0$$

which, when initialized with $X_0 = I$, leads to

$$X_k = \frac{1}{k!} A^k \quad k \geq 0$$

Substituting this result into the power series equation 31 yields

$$X(t) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k (t - t_0)^k$$

Note that the infinite power series equation 31 has the requisite convergence properties so that the infinite power series resulting from term-by-term differentiation converges to $\dot{X}(t)$, and Equation equation 30 is satisfied.

Recall that the scalar exponential function is defined by the following infinite power series

$$\begin{aligned} e^{at} &= 1 + at + \frac{1}{2}a^2t^2 + \frac{1}{6}a^3t^3 + \cdots \\ &= \sum_{k=0}^{\infty} \frac{1}{k!} a^k t^k \end{aligned}$$

Motivated by this, we define the so-called matrix exponential via

$$\begin{aligned} e^{At} &= I + At + \frac{1}{2}A^2t^2 + \frac{1}{6}A^3t^3 + \cdots \\ &= \sum_{k=0}^{\infty} \frac{1}{k!} A^k t^k \end{aligned} \tag{32}$$

Now, we can finally generalize equation 29 to Lemma 4.2.

It is important to point out that e^{At} is merely a notation used to represent the power series in Equation equation 32. Beyond the scalar case, the matrix exponential never equals the matrix of scalar exponentials corresponding to the individual elements in the matrix A. That is,

$$e^{At} \neq \left[e^{a_{ij}t} \right]$$

Any nonlinear, continuous differentiable dynamic

$$\dot{x}(t) = f(x(t), u(t), t) \tag{33}$$

can be represented by its linear nominal SSM $(\tilde{x}, \tilde{u})$ plus a third-order infinitesimal quantity:

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) + \mathcal{O}(x, u); \quad \boldsymbol{A} = \frac{\partial f}{\partial x}(x, u)\Big|_{\tilde{x}, \tilde{u}}; \boldsymbol{B} = \frac{\partial f}{\partial u}(x, u)\Big|_{\tilde{x}, \tilde{u}}.$$

Expanding the nonlinear maps in Equation equation 33 in a multivariate Taylor series about $[\tilde{x}(t), \tilde{u}(t), t]$ we obtain

$$\begin{aligned} \dot{x}(t) =& f[x(t), u(t), t] \\ =& f[\tilde{x}(t), \tilde{u}(t), t] + \frac{\partial f}{\partial x}[\tilde{x}(t), \tilde{u}(t), t][x(t) - \tilde{x}(t)] \\ &+ \frac{\partial f}{\partial u}[\tilde{x}(t), \tilde{u}(t), t][u(t) - \tilde{u}(t)] + \text{ higher-order terms} \end{aligned}$$

On defining the Jacobian matrix as the coefficient matrix

$$\begin{aligned} A(t) &= \frac{\partial f}{\partial x}(\tilde{x}(t), \tilde{u}(t), t) \\ B(t) &= \frac{\partial f}{\partial u}(\tilde{x}(t), \tilde{u}(t), t) \end{aligned}$$

rearranging slightly we have

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) + \mathcal{O}(x, u).$$

Deviations of the state, input, and output from their nominal trajectories are denoted by δ subscripts

$$x_\delta(t) = x(t) - \tilde{x}(t)$$
$$u_\delta(t) = u(t) - \tilde{u}(t)$$

We also have

$$\dot{x}_\delta(t) = \boldsymbol{A}(t)x_\delta(t) + \boldsymbol{B}(t)u_\delta(t) + \mathcal{O}(x,u)$$

(Approximation Error Bound) Suppose that the function $f : [0,1] \in \mathbb{R}$ is k times continuously differentiable, the piecewise polynomial g approximates f with mean error bounded as follows:

$$\|f - g\| \leq 2^{-rk} \frac{2}{4^k k!} \sup_{x\in[0,1]} \left| f^{(k)}(x) \right| .$$

Similar to **?**, we divide the interval $[0,1]$ into subintervals on which g is a polynomial; the restriction of g to one such subinterval $I_{r,l}$ is the polynomial of degree less than k that approximates f with minimum mean error. Also, the optimal g can be regarded as the orthonormal projection $Q_r^k f$ onto $\mathcal{G}_{(r)}^k$. We then use the maximum error estimate for the polynomial, which interpolates f at Chebyshev nodes of order k on $I_{r,l}$. We define $I_{r,l} = [2^{-r}l, 2^{-r}(l+1)]$ for $l = 0, 1, \ldots, 2^r - 1$, and obtain

$$\begin{aligned}
\left\|Q_r^k f - f\right\|^2 &= \int_0^1 \left[\left(Q_r^k f\right)(x) - f(x)\right]^2 dx \\
&= \sum_l \int_{I_{r,l}} \left[\left(Q_r^k f\right)(x) - f(x)\right]^2 dx \\
&\leq \sum_l \int_{I_{r,l}} \left[\left(C_{r,l}^k f\right)(x) - f(x)\right]^2 dx \\
&\leq \sum_l \int_{I_{r,l}} \left(\frac{2^{1-rk}}{4^k k!} \sup_{x\in I_{r,l}} \left|f^{(k)}(x)\right|\right)^2 dx \\
&\leq \left(\frac{2^{1-rk}}{4^k k!} \sup_{x\in[0,1]} \left|f^{(k)}(x)\right|\right)^2
\end{aligned}$$

and by taking square roots, we have bound (7). Here $C_{r,l}^k f$ denotes the polynomial of degree k, which agrees with f at the Chebyshev nodes of order k on $I_{r,l}$, and we have used the well-known maximum error bound for Chebyshev interpolation.

The error of the approximation $Q_r^k f$ of f therefore decays like 2^{-rk} and, since S_r^k has a basis of $2^r k$ elements, we have convergence of order k. For the generalization to m dimensions in the dynamic structure modeling, a similar argument shows that the rate of convergence is of order k/m.

The process of gradient updates in the IOSSM($\boldsymbol{ABC}$) is a spectral space skewing:

$$span\{\phi_i\} \to span\{\psi_i\}, \quad \psi_i = \chi(t,s) \circ \phi_i.$$

It can be regarded as a basis skewing from an orthogonal basis to an arbitrary basis by skewing in the basis itself or the measure, as shown in Hippo **?**.

For any scaling function

$$\chi(t,x) = \chi^{(t)}(x),$$

the functions $p_n(x)\chi(x)$ are orthogonal with respect to the density ω/χ^2 at every time t. Thus, we can choose this alternative basis and measure to perform the projections.

To formalize this tilting with χ, define $\nu^{(t)}$ to be the normalized measure with density proportional to $\omega^{(t)}/\left(\chi^{(t)}\right)^2$. We will calculate the normalized measure and the orthonormal basis for it. Let

$$\zeta(t) = \int \frac{\omega}{\chi^2} = \int \frac{\omega^{(t)}(x)}{\left(\chi^{(t)}(x)\right)^2} \, \mathrm{d}x$$

be the normalization constant, so that $\nu^{(t)}$ has density $\frac{\omega^{(t)}}{\zeta(t)\left(\chi^{(t)}\right)^2}$. If $\chi(t,x) = 1$ (no tilting), this constant is $\zeta(t) = 1$. In general, we assume that ζ is constant for all t; if not, it can be folded into χ directly. Next, note that (dropping the dependence on x inside the integral for shorthand)

$$\begin{aligned}\left\|\zeta(t)^{\frac{1}{2}} p_n^{(t)} \chi^{(t)}\right\|_{\nu^{(t)}}^2 &= \int \left(\zeta(t)^{\frac{1}{2}} p_n^{(t)} \chi^{(t)}\right)^2 \frac{\omega^{(t)}}{\zeta(t)\left(\chi^{(t)}\right)^2} \\ &= \int \left(p_n^{(t)}\right)^2 \omega^{(t)} \\ &= \left\|p_n^{(t)}\right\|_{\mu^{(t)}}^2 = 1.\end{aligned}$$

Thus we define the orthogonal basis for $\nu^{(t)}$

$$g_n^{(t)} = \lambda_n \zeta(t)^{\frac{1}{2}} p_n^{(t)} \chi^{(t)}, \quad n \in \mathbb{N}.$$

We let each element of the basis be scaled by a λ_n scalar, for reasons discussed soon, since arbitrary scaling does not change orthogonality:

$$\left\langle g_n^{(t)}, g_m^{(t)} \right\rangle_{\nu^{(t)}} = \lambda_n^2 \delta_{n,m}$$

Note that when $\lambda_n = \pm 1$, the basis $\left\{g_n^{(t)}\right\}$ is an orthonormal basis with respect to the measure $\nu^{(t)}$, at every time t. Notationally, let $g_n(t,x) := g_n^{(t)}(x)$ as usual.

For the n-dimensional SSM($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$), any transformation defined by a nonsingular matrix $\boldsymbol{T}$ will generate a transformed SSM($\hat{\boldsymbol{A}}$,$\hat{\boldsymbol{B}}$,$\hat{\boldsymbol{C}}$):

$$\hat{\boldsymbol{A}} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T} \quad \hat{\boldsymbol{B}} = \boldsymbol{T}^{-1}\boldsymbol{B} \quad \hat{\boldsymbol{B}} = \boldsymbol{C}\boldsymbol{T},$$

where the dynamic matrix $\hat{\boldsymbol{A}}$ has the same characteristic polynomial and eigenvalues (same dynamics) as $\boldsymbol{A}$, but its eigenvectors are different. For the transformation

$$x(t) = \boldsymbol{T}z(t) \quad z(t) = \boldsymbol{T}^{-1}x(t)$$

We have

$$\begin{aligned}\dot{z}(t) &= \boldsymbol{T}^{-1}\dot{x}(t) \\ &= \boldsymbol{T}^{-1}[\boldsymbol{A}x(t) + \boldsymbol{B}u(t)] \\ &= \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}z(t) + \boldsymbol{T}^{-1}\boldsymbol{B}u(t)\end{aligned}$$

(K-Lipschitz Continuity) Suppose that the function $f : [0,1] \in \mathbb{R}$ is k times continuously differentiable, the piecewise polynomial $g \in \mathcal{G}_r^k$ approximates f with mean error bounded as follows:

$$\|f(x_1) - f(x_2)\| \leq K \|x_1 - x_2\|, \forall x_1, x_2 \in dom f$$

The TOSSM($\boldsymbol{ABC}$) is K-Lipschitz with $\mathrm{eig}(\boldsymbol{A}) \leq K^2$. For a complex function, it has

$$\|f \circ g\|_{\text{Lip}} \leq \|f\|_{\text{Lip}} \cdot \|g\|_{\text{Lip}}$$

The essence of SSM lies in computing the Krylov kernel of matrix $\boldsymbol{A}$, so if we want a multi-layer SSM to be 1-Lipschitz, the $\boldsymbol{A}$ needs to be 1-Lipschitz.

For any $\boldsymbol{A}$, we have

$$\begin{aligned}&\boldsymbol{A} \in \text{K} - \text{ Lipschitz}, \forall \boldsymbol{A} : R^n \to R^m / \forall \boldsymbol{A} \in R^{m \times n} \\ \Longleftrightarrow &\|\boldsymbol{A}\vec{x}\| \leq K\|\vec{x}\|, \forall \vec{x} \in R^n \\ \Longleftrightarrow &\langle \boldsymbol{A}\vec{x}, \boldsymbol{A}\vec{x} \rangle \leq K^2 \langle \vec{x}, \vec{x} \rangle \\ \Longleftrightarrow &\vec{x}^T \left(\boldsymbol{A}^T\boldsymbol{A} - K^2 I\right) \vec{x} \leq 0 \\ \Longleftrightarrow &\left\langle \left(\boldsymbol{A}^T\boldsymbol{A} - K^2 I\right) \vec{x}, \vec{x} \right\rangle \leq 0\end{aligned}$$

Since $\boldsymbol{A}^T\boldsymbol{A}$ is a real symmetric matrix, the eigenvectors corresponding to different eigenvalues are orthogonal to each other. Moreover, since $\boldsymbol{A}^T\boldsymbol{A}$ is positive semidefinite, all its eigenvalues are non-negative. Let's assume that the eigenvectors of $\boldsymbol{A}^T\boldsymbol{A} \in \mathbb{R}^{n\times n}$ are denoted as $\overrightarrow{v_i}$ for $i = 1, 2, \ldots, n$, with corresponding eigenvalues λ_i for $i = 1, 2, \ldots, n$. Since $\overrightarrow{v_i}$ are pairwise orthogonal (not rigorously, as there may not be exactly n of them), we can assume that $\overrightarrow{v_i}$ has already been normalized to unit length, thus forming an orthonormal basis for the $\mathbb{R}^n$ space. Therefore, for any $\vec{x} \in \mathbb{R}^n$, we can express it as $\vec{x} = \sum_{i=1}^n x_i \overrightarrow{v_i}$. we have:

$$\begin{array}{ll} & \left\langle \left(A^TA - K^2I\right)\vec{x}, \vec{x}\right\rangle \leq 0 \\ \iff & \left\langle \left(A^TA - K^2I\right)\sum_{i=1}^n x_i\overrightarrow{v_i}, \sum_{i=1}^n x_i\overrightarrow{v_i}\right\rangle \leq 0 \\ \iff & \sum_{i=1}^n\sum_{j=1}^n x_ix_j \left\langle \left(A^TA - K^2I\right)\overrightarrow{v_i}, \overrightarrow{v_j}\right\rangle \leq 0 \\ \iff & \sum_{i=1}^n\sum_{j=1}^n x_ix_j \left[\left(A^TA - K^2I\right)\overrightarrow{v_i}\right]^T\overrightarrow{v_j} \leq 0 \end{array}$$

because

$$\begin{aligned} \left[\left(A^TA - K^2I\right)\overrightarrow{v_i}\right]^T\overrightarrow{v_j} &= \overrightarrow{v_i}^T\left(A^TA - K^2I\right)\overrightarrow{v_j} = \overrightarrow{v_i}^TA^TA\overrightarrow{v_j} - K^2\overrightarrow{v_i}^T\overrightarrow{v_j} \\ &= \begin{cases} 0, & \text{if } i = 1j \\ \left(\lambda_i - K^2\right)\langle\overrightarrow{v_i}, \overrightarrow{v_i}\rangle = \lambda_i - K^2, & \text{if } i = j \end{cases} \end{aligned}$$

so we have

$$\sum_{i=1}^n\sum_{j=1}^n x_ix_j \left[\left(A^TA - K^2I\right)\overrightarrow{v_i}\right]^T\overrightarrow{v_j} \leq 0 \iff \sum_{i=1}^n x_i^2\left(\lambda_i - K^2\right) \leq 0$$

specifically

$$A \in \text{ K-Lipschitz }, \forall A : R^n \to R^m / \forall A \in R^{m\times n} \iff \sum_{i=1}^n x_i^2\left(K^2 - \lambda_i\right) \geq 0$$

So the TOSSM($\boldsymbol{ABC}$) is K-Lipschitz with eig($\boldsymbol{A}$) $\leq K^2$.