Time-SSM: Simplifying and Unifying State Space Models for Time Series Forecasting

Jiaxi Hu[†], Disen Lan[†], Ziyu Zhou[†], Qingsong Wen[‡], and Yuxuan Liang[†*]

[†]Hong Kong University of Science and Technology (Guangzhou)
[‡]Squirrel AI
[†]`jiaxihu@hkust-gz.edu.cn`, `yuxliang@outlook.com`

Abstract

State Space Models (SSMs) have emerged as a potent tool in sequence modeling tasks in recent years. These models approximate continuous systems using a set of basis functions and discretize them to handle input data, making them well-suited for modeling time series data collected at specific frequencies from continuous systems. Despite its potential, the application of SSMs in time series forecasting remains underexplored, with most existing models treating SSMs as a black box for capturing temporal or channel dependencies. To address this gap, this paper proposes a novel theoretical framework termed Dynamic Spectral Operator, offering more intuitive and general guidance on applying SSMs to time series data. Building upon our theory, we introduce Time-SSM, a novel SSM-based foundation model with only one-seventh of the parameters compared to Mamba. Various experiments validate both our theoretical framework and the superior performance of Time-SSM.

1 Introduction

State Space Models (SSMs) [7, 9, 11, 31] are recent deep learning models that have demonstrated significant potential in various sequence modeling tasks. Based on controlled differential equations and discretization techniques, SSMs naturally align with time series data, which are meticulously collected at predetermined sampling frequencies from underlying continuous-time dynamical systems. Recently, a series of models [2, 3, 16, 29, 34, 40] encapsulate the state-of-the-art variants (e.g., Mamba [7]) as black-box modules, attempting augmenting their temporal modeling prowess.

Despite their promising performance, the intricate mathematical foundations and the underlying physical interpretations of SSMs in the realm of time series forecasting (TSF) [18, 22] remain ambiguous. For instance, (i) SSM is mathematically interpreted as Generalized Orthogonal Basis Projection [12] (GOBP), representing a linear combination of convolution kernels parameterized by two matrices ($\mathbf{A}$ and $\mathbf{B}$) and controlled by a matrix $\mathbf{C}$. However, this formulation falls short of interpreting complex temporal dependencies when generalizing to time series. (ii) SSMs originate from Hippo [8], and the physical significance of various initialization in the field of TSF remains unclear. (iii) SSMs are initially devised for modeling ultra-long sequences (e.g., with a target of one million in Hippo), which prompts the adoption of an exponentially decaying basis, i.e., *S4-LegS* [12]. Nevertheless, the necessity of this decaying method for TSF tasks requires further justification. This work aims to provide a comprehensive theoretical exposition of the above questions, offering more intuitive and general guidance on applying SSMs to time series data. As immediate consequences of this paper:

- ***Clarity and Unity Spark Insight***. We firmly believe that a clear and unified theoretical framework will directly foster the advancement of SSMs in the TSF community. To this end, we extend the Generalized Orthogonal Basis Projection (GOBP) theory to the Dynamic Spectral Operator theory (see Figure 1),

(a) Target: Approximate Complex Nonlinear Mapping

u — y
$K: u \rightarrow y$

(b) Neural Spectral Operator

u — y
$\widehat{K}: U \rightarrow Y$

(c) Orthogonal Basis Projection & Reconstruction

$e^{A(t-s)}B$
u — x — y
C

(d) Trainable State Space Model (ABC)
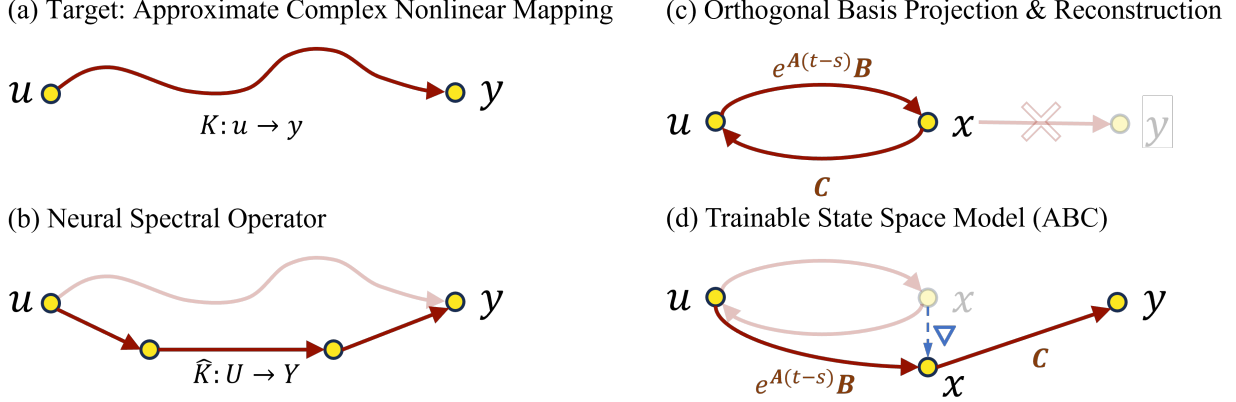
u — x — y
∇
$e^{A(t-s)}B$ x C

Figure 1: 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.

which is more appropriately tailored for TSF tasks. Building upon this, we introduce a novel variant of SSM basis called ***Hippo-LegP*** based on piecewise Legendre polynomials, along with various time-varying SSMs in complex-plane for temporal dynamic modeling.

- ***Never Train From Scratch***. Early SSMs are initialized based on matrices derived from Hippo theory [8], later evolving into negative real diagonal initialization [7]. This paper elucidates the physical significance of diagonal initialization from the perspective of time series dynamics and demonstrates that using specific matrices for initialization remains optimal in the context of TSF.

- ***A Dawn for Time Series***. Leveraging the above supports, we introduce **Time-SSM**, a novel SSM-based foundation model for TSF. Various experiments, including SSM variant ablation, autoregressive prediction, and long-term function reconstruction, empirically validate our theoretical framework and the superior performance of Time-SSM. We aim for this to mark the dawn of SSM's application in time series data, catalyzing vibrant growth and innovation within this community.

Apart from theoretical contribution, the important empirical findings include:
1. A unique phenomenon observed in the time series area that finite measure SSM and robust SSM demonstrate superior performance, in contrast to the results in Long Range Arena benchmarks.
2. S4D-real and LegP-complex initializations outperform others, and S4D-real is particularly efficient.
3. For complex-plane SSMs, the unitary matrix V serves as an inducing bias that enables ABC to possess coordinated dynamics, which is important for temporal modeling.
4. Preserving the specific initialization of A is crucial in SSM-based TSF models.
5. Autoregressive prediction is unnecessary in SSM-based models, LegP exhibits stronger representational capacity but is susceptible to noise in TSF tasks, the linear weights in Time-SSM play a crucial role, time-invariant B is beneficial for long-term prediction, and so on.

2 Theoretical Framework

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 B.

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 (1), referring to Figure 1a.

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 $\mathbb{T}$ to these domains as Equation (2) (see Figure 1b).

$$\mathcal{K}u(s) = \int_D K(t,s)u(s)ds, \qquad (1) \qquad\qquad K(t,s) = \mathbb{T} \circ \hat{K}(t,s) \circ \mathbb{T}^{-1}. \qquad (2)$$

This paper refers to these domains as spectral domains, such as Fourier [21], Laplace [4], and Wavelet [14] 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 (3), 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** [9]) SSM for simplicity, and our theoretical framework can be easily extended to multi-input/output (**MIMO** [31]) SSM with input $u \in D$ dimension.

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \qquad (3a) \qquad\qquad K(t) = \boldsymbol{C}e^{t\boldsymbol{A}}\boldsymbol{B} \qquad (4a)$$

$$y(t) = \boldsymbol{C}x(t) \qquad (3b) \qquad\qquad y(t) = (K * u)(t) \qquad (4b)$$

The two terms in Lemma 2.1 are also known as *zero-input response* component and *zero-state response* component [35]. 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 (4).

Lemma 2.1. *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 [8] 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 (5). When we differentiate this projection coefficient w.r.t. t (Equation (3a)), the self-similar relation [8] 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 $u(s)$.

$$\langle u, p_n \rangle_\mu = \int_{-\infty}^t u(s)p_n(t,s)\omega(t,s)\mathrm{d}s, \qquad (5) \qquad\qquad x_n(t) = \int u(s)K_n(t,s)\mathbb{I}(t,s)\mathrm{d}s. \qquad (6)$$

From Equation (5), it is evident that the projection coefficients depend on both the terms $\omega(t,s)$ and $p_n(t,s)$. The GOBP theory [12] combines them into a single term $K_n(t,s)$, called SSM kernel, and constrains the integration interval using indicator functions $\mathbb{I}(t,s)$ (see Equation 6).

Lemma 2.2. *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:*

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

Lemma 2.2 provides a theoretical guarantee in modeling non-linear and non-stationary temporal dependencies of real-world time series by linear SSM (Equation (3)) with limited error.
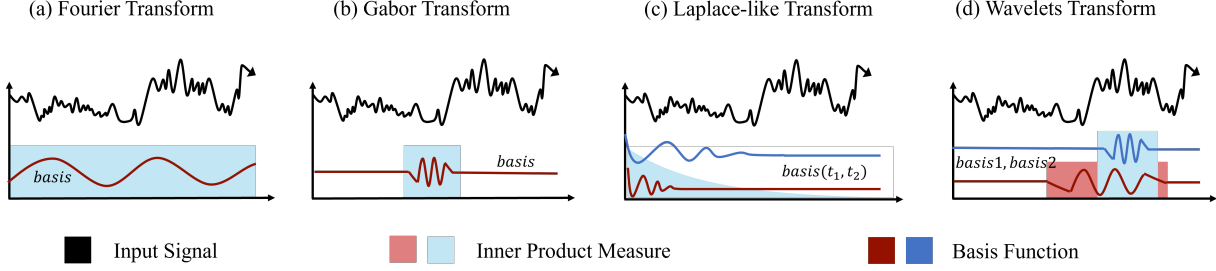
3

Figure 2: Classical spectral transform with basis function and inner product measure.

Definition 1. *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 1c), at all time t,*

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

Remark 2.3. *This paper also adheres to other definitions in HTTYH (GOBP) [12], such as OSSM (Orthogonal SSM) and TOSSM (Time-invariant Orthogonal SSM).*

Previous SSMs treat matrix $\boldsymbol{AB}$ and $\boldsymbol{C}$ in isolation, and $\boldsymbol{C}$ is typically understood as coefficients for linear combinations of kernel $e^{t\boldsymbol{A}}\boldsymbol{B}$ [12] or as a feature learning module [10]. 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 1. IOSSM defines a spectral transformation $\mathbb{T}$ and inverse transformation $\mathbb{T}^{-1}$ by IOSSM($\boldsymbol{ABC}$) in formula like (7) and (8).

$$\boldsymbol{A}_{nk} = - \begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & n > k \\ n+1 & n = k \\ 0 & n < k \end{cases} \tag{7}$$

$$\boldsymbol{B}_n = (2n+1)^{\frac{1}{2}} \quad \boldsymbol{C}_n = (2n+1)^{\frac{1}{2}} L_n(2n-1)$$
(**HiPPO-LegS matrix in IOSSM**)

$$\boldsymbol{A}_{nk} = -(2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}}$$
$$\cdot \begin{cases} 1 & k \le n \\ (-1)^{n-k} & k \ge n \end{cases} \tag{8}$$

$$\boldsymbol{B}_n = (2n+1)^{\frac{1}{2}} \quad \boldsymbol{C}_n = L_n(2n-1)$$
(**HiPPO-LegT matrix in IOSSM**)

$$\boldsymbol{A}_{nk}^{(N)} = - \begin{cases} (n+\frac{1}{2})^{1/2}(k+\frac{1}{2})^{1/2} & n > k \\ \frac{1}{2} & n = k \\ (n+\frac{1}{2})^{1/2}(k+\frac{1}{2})^{1/2} & n < k \end{cases} \tag{9}$$

$$\boldsymbol{A} = \boldsymbol{A}^{(N)} - \text{rank}(1), \qquad \boldsymbol{A}^{(D)} := \text{eig}(\boldsymbol{A}^{(N)})$$
(**Normal / DPLR form of HiPPO-LegS**)

$$\boldsymbol{A}_{nk}^{(N)} = - \begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & n < k, k \text{ odd} \\ 0 & else \\ (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & n > k, n \text{ odd} \end{cases}$$

$$\boldsymbol{A} = \boldsymbol{A}^{(N)} - \text{rank}(2), \qquad \boldsymbol{A}^{(D)} := \text{eig}(\boldsymbol{A}^{(N)})$$
(**Normal / DPLR form of HiPPO-LegT**)
(10)

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.

Proposition 1. *The IOSSM($\boldsymbol{A}$,$\boldsymbol{B}$) is a Fourier Transform (Figure 2a) with*

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

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.

Proposition 2. *The IOSSM($\boldsymbol{A}$,$\boldsymbol{B}$) is a Gabor Transform (Figure 2b) with*

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

4

The Gabor function $g(t-\tau)$ defines a window to truncate any orthogonal basis $\phi_n(t,s)$ and indirectly influence the $\omega(t) = \mathbb{I}[t, t+\tau]$, enabling the system to capture the local dynamic characteristics. **HiPPO-LegT** (8) use translated Legendre basis L_n, **S4-FouT** [12] utilize Fourier basis, known as the short-time Fourier transform, and Convolution Neural Network can be regarded as learnable basis.

Proposition 3. *The IOSSM(A,B) is a Laplace Transform with*

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

Exponential decay measures benefit long-term dynamics modeling. In the case of **S4-LegS(A,B)**, a variant of **Hippo-LegS**($\frac{1}{t}A, \frac{1}{t}B$) (7) without the $\frac{1}{t}$ term, it resembles a Laplace-like transform (Figure 2c) with exponential decay in both the measure and the Legendre basis function [12].

Proposition 4. *The IOSSM(A,B) is a Wavelet Transform (Figure 2d) with*

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

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 A.2 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 2.4 provides the approximation error of it with a finite basis. Appendix A.1 presents a detailed implementation.

Lemma 2.4. *[16] Suppose that the function $u : [0,1] \in \mathbb{R}$ is k times continuously differentiable, the piecewise polynomial $g \in \mathcal{G}_r^k$ approximates u with mean error bounded as follows:*

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

Corollary 2.5. *The SSM based on **HiPPO-LegP** have stronger dynamic representation ability.*

Remark 2.6. **HiPPO-LegP** *is a framework that can be extended to an arbitrary orthogonal basis. Since the piecewise-scale projection is a linear projection $\mathbb{T}^2 = \mathbb{T}$ onto a subspace, preserving the system dynamic characteristics determined by AB. Therefore, the AB matrix in **HiPPO-LegP** is consistent with that of **HiPPO-LegT** but with additional fixed or trainable $\mathbb{T}$ and $\mathbb{T}^{-1}$.*

Based on spectral neural operator and continuous-time SSM, we propose Dynamic Spectral Operator (**Definition** 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.

Definition 2. *SSM is a **Dynamic Spectral Operator** $\mathcal{K} : u \mapsto y$ by transform $\mathbb{T}$ with integral kernel $e^{A(t-s)}$. C can be regarded as $\hat{K} \circ \mathbb{T}^{-1}$ or directly linear projection: $: x \mapsto y$ (Figure 1d).*

Proposition 5. *The process of gradient updates in the IOSSM(ABC) is a spectral space skewing:*

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

Diagonal SSM. In practical scenarios, the naive recursive calculation of SSM Kernel (4) can be quite computationally intensive. Ideally, when matrix A is diagonal, the computation simplifies to exponentiating the diagonal elements only. In line with the Diagonal Plus Low-Rank (**DPLR**) representation in S4 [9], we introduce a low-rank correction term to the Hippo matrix in formula (7)(8) to obtain a skew-symmetric matrix $A^{(N)}$ in formula (9)(10) that can be diagonalized.

Lemma 2.7. *For the n-dimensional SSM(A,B,C), any transformation defined by a nonsingular matrix T will generate a transformed SSM($\hat{A},\hat{B},\hat{C}$):*

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

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

Lemma 2.7 defines a transformation rule for the SSM, which allows us to diagonalize matrix A into the complex domain using unitary matric V, where $A = V^{-1}\Lambda V$ and $\Lambda = A^{(D)} = \text{eig}(A^{(N)})$. A is unitarily equivalent to $A^{(D)}$, but the complex domain typically has a stronger expressive power.

Lemma 2.8. *The TOSSM(ABC) is K-Lipschitz with $\text{eig}(A) \leq K^2$.*

S4D introduces a method known as the left half-plane control to ensure that the diagonal elements of matrix $A^{(D)}$ remain negative. Lemma 2.8 provides an alternative, more intuitive explanation, which protects the model from suffering recursive gradient explosion.

Lemma 2.9. *[12] There is no TOSSM with the diagonal state matrix $A = \text{diag}\{-1, -2, \dots\}$.*

Lemma 2.10. *[43] Let A be an unitary matrix and ϵ_t be σ^2-subgaussian random noise. We have:*

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

Corollary 2.11. *The Diagonal SSM(A,B) is a rough approximation of complete dynamics.*

DSS [13] proposed a pure complex diagonal SSM that discards low-rank correction terms. By Lemma 2.2, 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*** (7). 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 2.10). Surprisingly, Table 3 reveal that rough dynamic approximations, such as S4D-real, actually yield better performance, in contrast to findings in Long Range Arena (LRA) benchmarks [32].

Discretization. For real-world time series data, it is necessary to convert continuous parameters (ΔAB) to discrete parameters ($\overline{A}, \overline{B}$). In this paper, a combination of the zero-order hold (Lemma 2.1) and the forward Euler [8] method is employed to achieve a more concise discrete representation.

$$(\textbf{ZOH}) : \overline{A} = \exp(\Delta A), \qquad (\textbf{Forward Euler}) : \overline{B} = \Delta B. \tag{11}$$

3   Time-SSM: State Space Model for Time Series

We adhere to TSF standards by applying normalization and de-normalization techniques to the input and output, addressing the distribution shift issues [24]. Furthermore, we employ linear embedding operations P on time series patches [28] to obtain a vectorized representation of the input function $u(s)$ and use linear mapping Q back to the numerical observation domain $y(s)$.

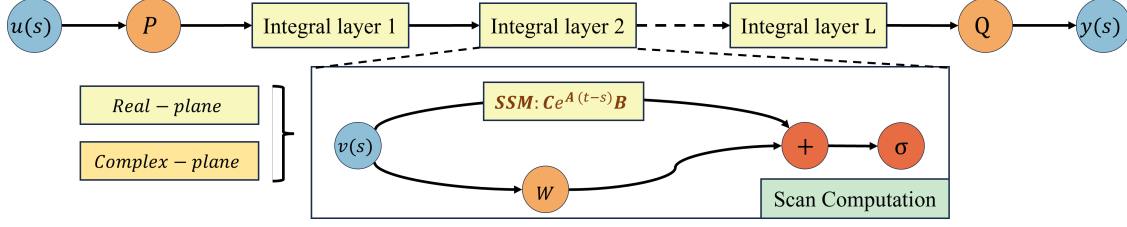**Proposition 6.** *Patch Operation is a simplified piecewise polynomial approximation approach.*

Figure 3: Architecture of Time-SSM, $\boldsymbol{PQ}$ represent vectorized representation of a function.

3.1 Operator-like Architecture

In accordance with Definition 2, Time-SSM utilizes a standard neural operator structure, where each layer combines linear weight and SSM integral kernel. Additionally, the incorporation of an activation function introduces non-linearity in the model. Empirical evidence in Table 2 showcases the substantial superiority of this simple architecture over Mamba [7] in the domain of TSF.

$$y^l = \sigma(\boldsymbol{W}^l(u) + \boldsymbol{SSM}^l(u)). \tag{12}$$

Table 1 provides a comprehensive list of all SSM kernel variants introduced in this paper, and detailed explanations will be presented in Sections 3.2 and Sections 3.3.

3.2 Real-plane SSM

The essence of SSM lies in computing the Krylov kernel of matrix $\boldsymbol{A}$, which determines the system's dynamic characteristics. Following Mamba's methodology, we parameterize $\boldsymbol{A}$ with a specific matrix and use linear layers to transform input u into a time-varying $(\boldsymbol{BC\Delta})$ representation. By discretizing the system, we indirectly introduce time-varying properties to $\boldsymbol{A}$ by $\boldsymbol{\Delta} \in \mathbb{R}^{B \times L \times D}$. This allows $\boldsymbol{A}$ to still satisfy some properties of TOSSM in section 2 within a time-varying SSM system.

$$\boldsymbol{A} \leftarrow \text{parameter}, \qquad \boldsymbol{B}, \boldsymbol{C} = \text{Linear}_{\boldsymbol{BC}}(u), \qquad \boldsymbol{\Delta} = \text{softplus}(\text{Linear}_{\boldsymbol{\Delta}}(u)). \tag{13}$$

S4D-real. we set matrix $\boldsymbol{A} = \text{diag}\{-1, -2, \dots\}$ and use $\boldsymbol{A} = -\exp(\log(\text{diag}\{1, 2, \dots\}))$ to stabilize the gradients [20, 33], which is the default setting in Mamba and our proposed Time-SSM.

LegS/LegT/LegP. $\boldsymbol{\Delta}$ is fixed at input data frequency $1/L$, and the model parameters are initialized using a complete set of dynamics IOSSM ($\boldsymbol{ABC}$). The gradients from $\boldsymbol{ABC}$ are subsequently employed to adaptively update the state spectral space (Lemma 5).

Remark 3.1. *we observe that retaining the complete dynamics of $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ in a time-varying $(\boldsymbol{BC\Delta})$ environment leads to gradient explosion. we adopt S4 to eliminate the time-varying factors.*

Robust SSM. Notably, the discretization process $\overline{\boldsymbol{B}} = \boldsymbol{\Delta B}$ also indirectly introduces time-varying factors from $\boldsymbol{\Delta}$. To this end, we propose a robust version of the SSM where both $(\boldsymbol{AB})$ are specifically parameterized. Section 4.2 presents a comprehensive performance analysis of this variant.

3.3 Complex-plane SSM

Following Lemma 2.7, we conjugate diagonalize matrix $\boldsymbol{ABC}$ from real-plane to complex-plane. We initialize $\boldsymbol{V} \in \mathbb{C}^{N \times N}$ with the diagonal unitary matrix of $\boldsymbol{A}$ and combine it with the time-varying $\boldsymbol{BC}$ generated by the linear layer, which allows SSM($\boldsymbol{ABC}$) to possess compatible dynamics.

$$\boldsymbol{B} = \boldsymbol{V}^T \text{Linear}_{\boldsymbol{B}}(u) \quad (\text{unitary matrix} : \boldsymbol{V}^T = \boldsymbol{V}^{-1}), \qquad \boldsymbol{C} = \boldsymbol{V}\text{Linear}_{\boldsymbol{C}}(u). \tag{14}$$

In the Experiment Section, we observe that this operation is necessary. Directly initializing the imaginary part $\Im(\text{Linear}_{\boldsymbol{BC}}(u)) = 0$ leads to significant performance degradation. Time-SSM provides three complex-plane SSM kernels in this paper: **LegS/LegT/LegP-complex**.

Table 1: Parameterization choices for structured SSM kernels

Method	Structure of $\boldsymbol{A}$	Structure of $\boldsymbol{B}$	Structure of $\boldsymbol{C}$	Time-varing factor
S4D-real	$\mathbb{R}(D,N) \leftarrow$ diagonal	$\mathbb{R}(B,L,N): \text{Linear}_B(x)$	$\mathbb{R}(B,L,N): \text{Linear}_C(x)$	$\Delta, \boldsymbol{B}, \boldsymbol{C}$
LegS/LegT/LegP-complex	$\mathbb{C}(D,N) \leftarrow$ diagonal	$\mathbb{C}(B,L,N): \boldsymbol{V}^{-1}\text{Linear}_B(x)$	$\mathbb{C}(B,L,N): \boldsymbol{V}\text{Linear}_C(x)$	$\Delta, \boldsymbol{B}, \boldsymbol{C}$
LegS/LegT	$\mathbb{R}(D,N,N) \leftarrow$ NPLR	$\mathbb{R}(D,N) \leftarrow$ parameter	$\mathbb{R}(D,N) \leftarrow$ parameter	None
Robust version	any above	$\mathbb{R}(N) \leftarrow$ parameter	$\mathbb{R}(B,L,N): \text{Linear}_C(x)$	$\Delta, \boldsymbol{C}$

Table 2: Average results of long-term forecasting with an input length of 96 and prediction horizons of {96, 192, 336, 720}. The best performance is in Red, and the second best is in Blue. Different colors represent SSM-based and other neural-network-based TSF models. Full results are in Appendix C.4.

Model	Time-SSM (Ours)		Mamba4TS (Temporal Emb.)		S-Mamba (Channel Emb.[34])		RWKV-TS [15]		Koopa [25]		InvTrm [26]		PatchTST [28]		DLinear [39]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.425**	**0.426**	0.444	0.438	0.459	0.453	0.454	0.446	0.450	0.443	0.463	0.454	**0.434**	**0.435**	0.462	0.458
ETTh2	**0.374**	**0.399**	0.386	0.410	0.381	0.407	**0.375**	**0.402**	0.397	0.417	0.383	0.407	0.380	0.406	0.564	0.520
ETTm1	**0.386**	**0.396**	0.396	0.406	0.399	0.407	**0.391**	0.403	0.395	0.403	0.407	0.412	0.403	**0.398**	0.403	0.406
ETTm2	**0.283**	**0.328**	0.299	0.343	0.289	0.333	0.285	0.330	**0.281**	**0.326**	0.291	0.335	**0.283**	0.329	0.345	0.396
Exchange	**0.352**	**0.398**	0.364	**0.405**	0.364	0.407	0.406	0.439	0.390	0.424	0.366	0.416	0.383	0.416	**0.346**	0.416
Crypto	**0.192**	**0.160**	0.193	0.162	0.198	0.163	**0.190**	**0.159**	0.199	0.165	0.196	0.164	**0.192**	0.161	0.201	0.176
Air-convection	**0.459**	**0.332**	0.471	0.343	0.484	0.352	0.464	0.336	**0.463**	0.337	0.493	0.363	0.483	0.354	**0.459**	**0.341**
Weather	**0.252**	**0.276**	0.258	0.280	**0.252**	0.277	0.256	0.280	**0.247**	**0.273**	0.260	0.280	0.258	0.280	0.267	0.319

4 Experiment

In this section, we commence by conducting a comprehensive performance comparison of Time-SSM against other state-of-the-art models, followed by ablation experiments pertaining to initialization methods and model architectures. Furthermore, considering the characteristics of TSF tasks, we perform autoregressive prediction experiments, complexity analysis, and representation ability analysis. Except for LegS and LegT, all other variants of SSM kernel can be implemented using the diagonal scanning algorithm like S5 [31]. For detailed information regarding baseline models, dataset descriptions, experimental settings, and hyper-parameter analysis, please refer to Appendix C.

4.1 Overall Performance

As depicted in Table 2, we can observe that (a) Time-SSM (S4D-real) achieved the best overall performance, followed by Koopa and RWKV-TS. This highlights the necessity of modeling temporal dynamics in the field of TSF. (b) Linear model DLinear surprisingly performs well on the Exchange and Air-convection datasets, where the former is believed to be influenced by external factors such as the economy, and the latter is considered a chaotic time series [16]. This observation further highlights the importance of robust modeling in TSF tasks. (c) Time-SSM eliminates redundant modules from Mamba while exhibiting significant performance enhancements compared to Mamba4TS. We envision Time-SSM as a novel foundational model that can drive further advancements in future research. (d) In datasets with a relatively small number of variables, both SSM-based (Mamba4TS *vs.* S-Mamba) and Transformer-based models (PatchTST *vs.* InvTrm) still consider temporal embedding as the optimal choice. Variable embedding may lead to improved results in datasets with a larger number of variables, which is evident in datasets like Traffic [37], where the number of variables (862) greatly exceeds the time size (96). In Section 4.2, we present a lightweight solution to balance the temporal dependency and variable dependency, which is not the focus (**temporal dynamic**) of this paper.

4.2 Ablation Study

Next, we will explore the practical performance of different SSM kernels and model architectures.

Table 3: Ablation study of **ABC** parameterization. Red/Blue denotes the best/second performance.

SSMs		S4D-real		LegS-complex		LegT-complex		LegP-complex		LegS(RNN)		LegT(RNN)		Full-Select	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.377	0.394	0.382	0.398	0.379	0.395	0.381	0.397	0.391	0.405	0.388	0.403	0.382	0.399
	192	0.423	0.424	0.429	0.427	0.426	0.426	0.424	0.427	0.438	0.436	0.434	0.433	0.435	0.429
	336	0.466	0.437	0.471	0.450	0.470	0.451	0.464	0.435	0.475	0.451	0.470	0.445	0.482	0.452
	720	0.452	0.448	0.474	0.471	0.469	0.467	0.460	0.455	0.488	0.483	0.484	0.477	0.483	0.476
	AVG	0.430	0.426	0.439	0.437	0.436	0.435	0.432	0.429	0.448	0.444	0.444	0.440	0.446	0.439
ETTm2	96	0.176	0.260	0.179	0.265	0.176	0.263	0.174	0.259	0.188	0.274	0.184	0.273	0.180	0.265
	192	0.246	0.305	0.246	0.308	0.242	0.307	0.244	0.308	0.264	0.321	0.262	0.318	0.247	0.309
	336	0.305	0.344	0.310	0.348	0.304	0.343	0.301	0.339	0.321	0.358	0.317	0.355	0.312	0.351
	720	0.406	0.405	0.411	0.410	0.409	0.407	0.410	0.408	0.431	0.436	0.430	0.428	0.412	0.410
	AVG	0.283	0.329	0.287	0.333	0.283	0.330	0.282	0.329	0.301	0.347	0.298	0.343	0.288	0.334
Weather	96	0.167	0.212	0.175	0.218	0.178	0.221	0.170	0.215	0.189	0.228	0.184	0.222	0.185	0.225
	192	0.217	0.255	0.222	0.253	0.226	0.259	0.216	0.257	0.238	0.269	0.236	0.263	0.230	0.262
	336	0.274	0.294	0.285	0.303	0.282	0.297	0.270	0.291	0.291	0.310	0.294	0.311	0.285	0.302
	720	0.351	0.345	0.354	0.346	0.355	0.345	0.353	0.348	0.366	0.358	0.362	0.355	0.360	0.351
	AVG	0.252	0.277	0.259	0.280	0.260	0.281	0.252	0.278	0.271	0.291	0.269	0.288	0.265	0.285

Table 4: Architecture ablation with various prediction horizons. *w/o* denotes without the corresponding module. B: Taime-varying **B** matrix; V: diagonal unitary matrix; W: Linear weights in each layer; VK: Variable kernel. Red denotes improved performance, while Blue denotes the declined performance.

Model		S4D-real		w/o B		w/o W		LegS-complex		w/o B		w/o V	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.377	0.394	0.382	0.401	0.380	0.400	0.382	0.398	0.385	0.403	0.385	0.396
	192	0.423	0.424	0.438	0.436	0.432	0.429	0.429	0.427	0.445	0.435	0.435	0.432
	336	0.466	0.437	0.483	0.452	0.476	0.452	0.471	0.450	0.489	0.456	0.478	0.453
	720	0.452	0.448	0.510	0.489	0.497	0.471	0.474	0.471	0.524	0.495	0.478	0.465
	AVG	0.425	0.426	0.453	0.445	0.446	0.438	0.439	0.437	0.461	0.447	0.444	0.437
ETTh2	96	0.290	0.341	0.298	0.349	0.288	0.339	0.298	0.347	0.308	0.358	0.305	0.357
	192	0.368	0.387	0.382	0.403	0.371	0.390	0.377	0.395	0.384	0.405	0.377	0.400
	336	0.416	0.430	0.420	0.427	0.412	0.425	0.421	0.434	0.433	0.442	0.423	0.434
	720	0.424	0.439	0.422	0.441	0.418	0.438	0.427	0.441	0.434	0.452	0.426	0.447
	AVG	0.374	0.399	0.381	0.405	0.372	0.398	0.381	0.404	0.390	0.414	0.383	0.410
Air	96	0.290	0.341	0.292	0.343	0.286	0.338	0.296	0.345	0.299	0.351	0.299	0.352
	192	0.368	0.387	0.366	0.386	0.372	0.391	0.371	0.388	0.377	0.394	0.376	0.392
	336	0.416	0.430	0.411	0.423	0.419	0.434	0.421	0.436	0.424	0.436	0.430	0.441
	720	0.424	0.439	0.419	0.434	0.432	0.447	0.430	0.444	0.428	0.442	0.428	0.449
	AVG	0.374	0.399	0.372	0.396	0.377	0.403	0.380	0.403	0.382	0.406	0.383	0.409

Model		S4D-real		*with* VK	
Metric		MSE	MAE	MSE	MAE
ETTh2	96	0.290	0.341	0.305	0.351
	192	0.368	0.387	0.372	0.386
	336	0.416	0.430	0.399	0.416
	720	0.424	0.439	0.418	0.429
	AVG	0.374	0.399	0.373	0.395
ETTm2	96	0.176	0.260	0.176	0.258
	192	0.246	0.305	0.247	0.309
	336	0.305	0.344	0.301	0.334
	720	0.406	0.405	0.392	0.397
	AVG	0.283	0.328	0.279	0.325
Weather	96	0.167	0.212	0.154	0.202
	192	0.217	0.255	0.204	0.242
	336	0.274	0.294	0.262	0.288
	720	0.351	0.345	0.339	0.341
	AVG	0.252	0.276	0.239	0.268

Various SSM Kernels. Table 3 demonstrates the results of Time-SSM using all SSM kernels from Table 1. Overall, S4D-real and LegP-complex parameterizations achieve the best performance, and S4D-real initialization remains the simplest and most effective choice for TSF tasks. Removing time-varying factors in LegS and LegT leads to poor performance, emphasizing their importance as attention or gating mechanisms. Surprisingly, generating **A** by linear layer (full select) results in decreased performance, highlighting the necessity of selecting specific dynamic matrices **A**.

Interestingly, LegS-complex generally outperforms other variants in LRA benchmark tasks (text, images, logical reasoning), with the support of exponential decay memory. However, in TSF tasks, the opposite phenomenon occurs, which suggests that *in TSF settings, where the lookback window is limited, robust approximation and finite window approximation yield better performance.*

Time-varying B. As shown in Table 4 (left), removing time-varying factors from matrix B (initialized by the Hippo matrix) generally leads to decreased performance. However, in datasets with prominent nonlinear features like Air-convection, especially in long-term prediction windows, it produces more robust predictions. This effect is somewhat weakened in the complex-plane version of Time-SSM.

Unitary matrix V. Removing the unitary matrix **V** from the **BC** matrix consistently leads to performance degradation. We view *V as an inductive bias that imparts BC with complex-plane dynamics that are compatible with A matrix, which is essential for temporal dynamical modeling.*

Linear Weights in Operator. Preserving the linear weights in each layer generally yields better results. We believe that the performance decline in the ETTh2 dataset may be attributed to overfitting by non-stationary

Table 5: Forecasting results in AR process with various prediction horizons. ∞ means gradient explosions.

| SSMs		S4D-real		LegS-complex		LegT-complex		LegP-complex		Full-Select		(S4D-real) Recurrent	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm2	96	**0.184**	**0.272**	0.189	0.278	0.188	0.275	**0.181**	**0.270**	0.192	0.276	0.198	0.296
	192	**0.251**	**0.313**	0.255	0.319	**0.254**	0.316	**0.251**	**0.312**	0.256	0.316	0.264	0.327
	336	**0.320**	0.360	**0.320**	**0.359**	0.324	0.362	0.324	0.360	**0.317**	**0.354**	0.337	0.368
	720	0.420	0.417	**0.417**	**0.413**	**0.418**	**0.415**	0.420	0.417	0.423	**0.413**	∞	∞
	AVG	**0.294**	**0.341**	**0.295**	0.342	0.296	0.342	**0.294**	**0.340**	0.297	**0.340**	∞	∞
Weather	96	**0.182**	**0.223**	0.190	0.229	0.194	0.234	**0.186**	0.228	0.190	0.229	**0.186**	**0.227**
	192	**0.227**	**0.260**	**0.234**	**0.266**	0.242	0.273	0.236	0.269	0.239	0.269	0.236	0.270
	336	**0.282**	**0.300**	0.289	0.304	0.298	0.312	**0.284**	**0.299**	0.295	0.308	0.315	0.330
	720	**0.361**	**0.351**	**0.362**	**0.351**	0.375	0.362	**0.361**	**0.350**	0.366	0.354	∞	∞
	AVG	**0.263**	**0.284**	0.269	0.288	0.277	0.295	**0.267**	**0.287**	0.273	0.290	∞	∞
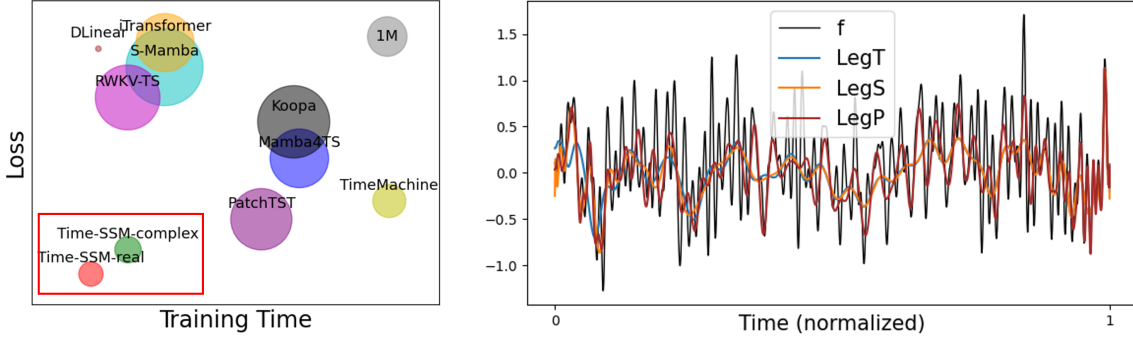
Figure 4: Left: Complexity analysis. Right: Long-range function approximation with different SSM basis

distributions, which remains consistent in both real-plane and complex-plane.

Multivariate SSM Kernel. In the era of LLM, we believe that excessive focus on modeling variable relationships within individual datasets becomes less meaningful. Models like UniTime [23], Timer [27], and MOIRAI [36] have already begun exploring unified time series models. Therefore, in this paper, ***we prioritize modeling temporal dependencies and aim to enhance model performance on small-scale datasets by leveraging inter-variable relationships at minimal cost.*** Two principles are proposed: (i) As for a sequence model, it is desirable to achieve a complexity of $\mathcal{O}(L)$. (ii) Given that the number of variables in different datasets may vary, it is preferable to impose an upper limit on the parameter count of variables rather than allowing unrestricted growth. To achieve this, we employ Fourier transformation to capture dominant k modes along variable dimensions and utilize complex-domain linear weights to represent the variable integral kernel: $\hat{y} = \mathsf{IFFT}(\boldsymbol{W} \cdot \mathsf{topk}(\mathsf{FFT}(y)))$. As shown in Table 4 (right), this simple and efficient module leads to a significant performance improvement, particularly on datasets with more variables.

4.3 Further Analysis

Auto Regression. SpaceTime [40] proposed a closed-loop control prediction method using input time series $u_{[1:L]}$ to construct a dynamical system and generate predictions $y_{[L+1:L+H]}$ through autoregressive modeling. However, this approach is not suitable for time-varying systems with changing matrices ($\boldsymbol{ABC}$). To address this, we introduced an implicit autoregressive method with zero padding: $y \in \mathbb{R}/\mathbb{C}^{L+H} = \text{Time-SSM } (u \in \mathbb{R}/\mathbb{C}^{L+O})$. The final prediction is obtained by truncating the last H steps of y. Experimental results (Table 5) show decreased performance for all initialization methods under this configuration, indicating redundancy of the additional autoregressive generation. Surprisingly, in longer sequences, the LegS-based variant outperforms LegT, contrary to Table 3 findings, supporting the original benefit of LegS for long-term sequence modeling. Moreover, selecting the $\boldsymbol{BC}$ matrix from the last time step for autoregressive generation leads to gradient explosions in time-varying environments due to the lack of dynamic constraints.

Efficiency. As shown in Figure 4 (left), we present a comparison of the complexity between Time-SSM, and other advanced TSF models. The radius of the circle represents the number of model parameters. It can be observed that overall, the Time-SSM exhibits superior efficiency. Although the DLinear model has fewer model parameters, its performance significantly lags behind the Time-SSM. In practical applications, the Time-SSM in the real plane is undoubtedly the optimal choice.

Representation Ability. We utilize a 64-dimensional (same as in Time-SSM) fixed Hippo matrix to reconstruct the input function, in order to explore the representational capacity of different SSM bases. The input function is randomly sampled from a continuous-time band-limited white noise process with a length of 10^6, a sampling step size of 10^{-4}, and a signal band limit of 1Hz. From Figure 4 (right), we observe that: (a) Consistent with Corollary 2.5, LegP (with a piecewise scale of 3) exhibits stronger representational capacity, outperforming LegT and LegS in limited dimensions. (b) According to previous research [8, 16, 43], a spectral space of 256 dimensions is typically required to effectively represent temporal dynamics in single-channel SSMs. However, in our experiments, we found that Time-SSM with a 64-dimensional spectral space performs better. We believe this is because the Time-SSM first obtains a high-dimensional vectorized representation of the time series through patch embedding. In the context of MIMO SSMs, the representational capacity of the Time-SSM is multiplicatively accumulated (N plus D), which is sufficient for dynamic modeling.

5 Conclusion

This paper aims to address the theoretical and physical limitations of SSM in the TSF context, providing comprehensive theoretical guidance for its application in time series data. To achieve this, we propose the theory of dynamic spectral operators based on the concept of IOSSM, and introduce a simple and efficient foundational model called Time-SSM. Additionally, we explore various variants of SSM kernels in both the real and complex planes. Through an extensive range of experiments, including ablation studies, autoregressive testing, complexity analysis, and white noise reconstruction, we rigorously validate and explain the effectiveness of our theory and proposed Time-SSM model. Ultimately, our goal is to contribute to the flourishing development of SSM not only within the TSF community but also in other machine-learning domains. Limitations can be found in Appendix E.

References

[1] Bradley K Alpert. A class of bases in l^2 for the sparse representation of integral operators. *SIAM journal on Mathematical Analysis*, 24(1):246–262, 1993.

[2] Md Atik Ahamed and Qiang Cheng. Timemachine: A time series is worth 4 mambas for long-term forecasting. *arXiv e-prints*, pages arXiv–2403, 2024.

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

[4] Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Lno: Laplace neural operator for solving differential equations. *arXiv preprint arXiv:2303.10528*, 2023.

[5] Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. *arXiv preprint arXiv:2402.05956*, 2024.

[6] 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.

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

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

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

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

[11] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022.

[12] Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Ré. How to train your hippo: State space models with generalized orthogonal basis projections. *arXiv preprint arXiv:2206.12037*, 2022.

[13] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994, 2022.

[14] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34:24048–24062, 2021.

[15] Haowen Hou and F Richard Yu. Rwkv-ts: Beyond traditional recurrent neural network for time series tasks. *arXiv preprint arXiv:2401.09093*, 2024.

[16] Jiaxi Hu, Yuehong Hu, Wei Chen, Ming Jin, Shirui Pan, Qingsong Wen, and Yuxuan Liang. Attractor memory for long-term time series forecasting: A chaos perspective. *arXiv preprint arXiv:2402.11463*, 2024.

[17] Jiaxi Hu, Qingsong Wen, Sijie Ruan, Li Liu, and Yuxuan Liang. Twins: Revisiting non-stationarity in multivariate time series forecasting. *arXiv preprint arXiv:2406.03710*, 2024.

[18] Ming Jin, Qingsong Wen, Yuxuan Liang, Chaoli Zhang, Siqiao Xue, Xue Wang, James Zhang, Yi Wang, Haifeng Chen, Xiaoli Li, et al. Large models for time series and spatio-temporal data: A survey and outlook. *arXiv preprint arXiv:2310.10196*, 2023.

[19] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[20] Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning*, pages 3794–3803. PMLR, 2019.

[21] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[22] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. *arXiv preprint arXiv:2403.14735*, 2024.

[23] Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. Unitime: A language-empowered unified model for cross-domain time series forecasting. *arXiv preprint arXiv:2310.09751*, 2023.

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

[25] Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with koopman predictors. In *Advances in Neural Information Processing Systems*, volume 36, pages 12271–12290, 2023.

[26] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, pages 1–25, 2024.

[27] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Transformers for time series analysis at scale. *arXiv preprint arXiv:2402.02368*, 2024.

[28] 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.

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

[30] Mohammad Amin Shabani, Amir H. Abdi, Lili Meng, and Tristan Sylvain. Scaleformer: Iterative multi-scale refining transformers for time series forecasting. In *ICLR*, 2023.

[31] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

[32] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[33] Shida Wang and Qianxiao Li. Stablessm: Alleviating the curse of memory in state-space models through stable reparameterization. *arXiv preprint arXiv:2311.14495*, 2023.

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

[35] Robert L Williams, Douglas A Lawrence, et al. *Linear state-space control systems*. John Wiley & Sons, 2007.

[36] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592*, 2024.

[37] 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.

[38] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *ICLR*, pages 1–23, 2023.

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

[40] Michael Zhang, Khaled K Saab, Michael Poli, Tri Dao, Karan Goel, and Christopher Ré. Effectively modeling time series with simple discrete state spaces. *arXiv preprint arXiv:2303.09489*, 2023.

[41] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *ICLR*, 2023.

[42] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, et al. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021.

[43] Tian Zhou, Ziqing Ma, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, Rong Jin, et al. Film: Frequency improved legendre memory model for long-term time series forecasting. *Advances in Neural Information Processing Systems*, 35:12677–12690, 2022.

[44] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, 2022.

[45] Tian Zhou, Peisong Niu, xue wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained lm. In *NeurIPS*, 2023.

A Model Details

A.1 Computation for SSM-LegP

It is first proposed by Attraos [16]. We can perform computations either explicitly or implicitly.
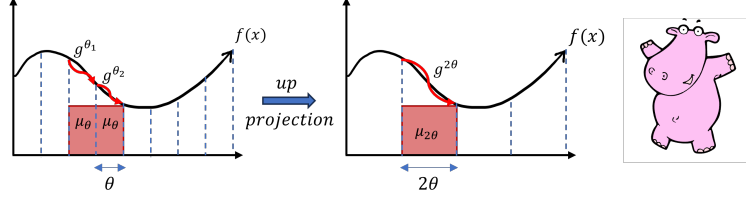
Figure 5: [16] Elaboration of Hippo-LegP

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:

$$\mathcal{G}_{(r)}^k = \begin{cases} g \mid \deg(g) < k, & x \in (2^{-r}l, 2^{-r}(l+1)) \\ 0, & \text{otherwise} \end{cases}, \tag{15}$$

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 $\mu_{2\theta}$:

$$\phi_i^{2\theta}(x) = \sum_{j=0}^{k-1} H_{ij}^{\theta_1} \phi_i^\theta(x)_{x \in [\theta_1]} + \sum_{j=0}^{k-1} H_{ij}^{\theta_2} \phi_i^\theta(x)_{x \in [\theta_2]}, \tag{16}$$

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. 16, 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:

$$\begin{aligned} x_t^{2\theta} &= H^{\theta_1} x_t^{\theta_1} + H^{\theta_2} x_t^{\theta_2}. \quad x^{2\theta} \in \mathbb{R}^{B \times L/2 \times D \times N} \\ x_t^{\theta_1} &= H^{\dagger \theta_1} x_t^{2\theta}, \quad x_t^{\theta_2} = H^{\dagger \theta_2} x_t^{2\theta}. \end{aligned} \tag{17}$$

H and $H^\dagger$ can be achieved by Gaussian Quadrature [14]. Iteratively repeating this process enables us to model the temporal dynamic from a more macroscopic perspective.

Remark A.1. *Although $\boldsymbol{\Delta}$ as a kind of attention mechanism leads to different measure windows, ie., $\theta_1 \neq \theta_2$, it still maintains the linear projection property for up and down projection: $\mathcal{G}_{\theta_1}, \mathcal{G}_{\theta_2} \leftrightarrow \mathcal{G}_{\theta_1 + \theta_2}$. In our illustration, we have used a unified measure window for simplicity.*

We can firstly obtain the state representation $x \in \mathbb{R}/\mathbb{C}^L$ by Equation 3a, and iteratively perform up projection based on the sequence positions. In each scale, we use a unified $\boldsymbol{C}$ matrix to get output. Then, the back projection is employed to recover the sequence in length L. H, and $H^\dagger$ are gradients opened to be adaptively optimized. It can be seen as an additional block after the SSM kernel to capture multi-scale information.
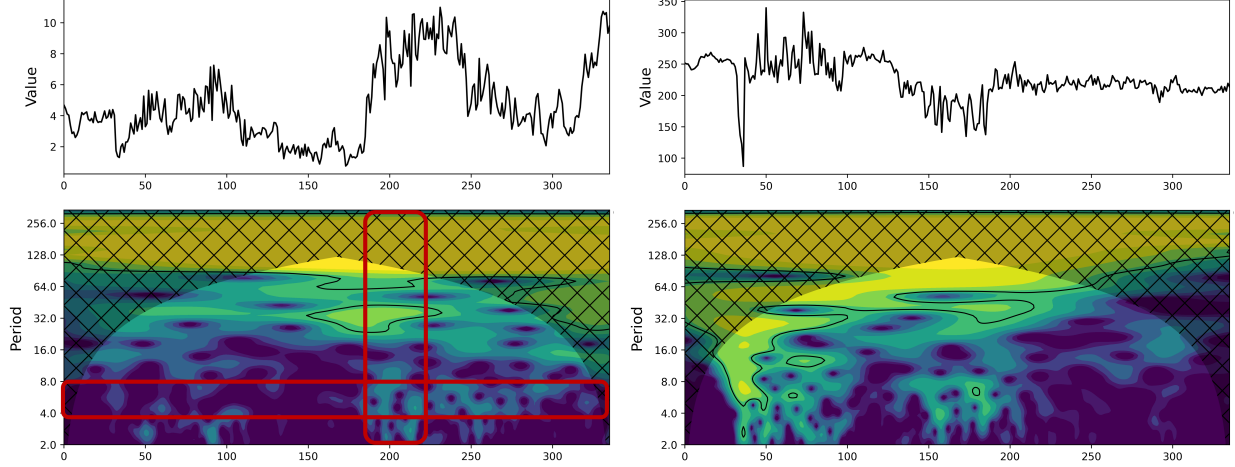
Figure 6: [17] Examples of the wv and wd variables in the Weather dataset. The x-axis is the shared time steps. The graph above shows the variable values over time. The one below is the wavelet level plot, representing the strength of the signal's energy at different time-frequency scales.

A.2 Visual Explanation

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.

B Proofs

Lemma B.1. *For a differential equation of $x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$, its general solution is:*

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

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

$$\dot{x}(t) = ax(t) + bu(t) \quad x(t_0) = x_0 \tag{18}$$

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

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

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

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

16

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

$$x(t) = e^{a(t-t_0)}x_0 + \int_{t_0}^t e^{a(t-s)}bu(s)ds \tag{19}$$

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 (3) given a specified initial state $x(t_0) = x_0$ and input vector $u(t)$. We begin with a related homogeneous matrix differential equation

$$\dot{X}(t) = AX(t) \quad X(t_0) = I \tag{20}$$

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

$$X(t) = \sum_{k=0}^{\infty} X_k (t - t_0)^k \tag{21}$$

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 (20) yields $X(t_0) = X_0 = I$. Substituting Equation (21) into Equation (20), formally differentiating term by term with respect to time, and shifting the summation index gives

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

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 (21) yields

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

We note here that the infinite power series (21) has the requisite convergence properties so that the infinite power series resulting from term by term differentiation converges to $\dot{X}(t)$, and Equation (20) is satisfied.

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

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

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

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 (21) yields

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

Note that the infinite power series (21) has the requisite convergence properties so that the infinite power series resulting from term-by-term differentiation converges to $\dot{X}(t)$, and Equation (20) is satisfied.

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

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

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

$$e^{At} = I + At + \frac{1}{2} A^2 t^2 + \frac{1}{6} A^3 t^3 + \cdots$$
$$= \sum_{k=0}^{\infty} \frac{1}{k!} A^k t^k \tag{22}$$

Now, we can finally generalize (19) to Lemma 2.1.

It is important to point out that e^{At} is merely a notation used to represent the power series in Equation (22). 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,

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

$\square$

Lemma B.2. *Any nonlinear, continuous differentiable dynamic*

$$\dot{x}(t) = f(x(t), u(t), t) \tag{23}$$

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

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

Proof. Expanding the nonlinear maps in Equation (23) in a multivariate Taylor series about $[\tilde{x}(t), \tilde{u}(t), t]$ we obtain
$$\dot{x}(t) = f[x(t), u(t), t]$$
$$= f[\tilde{x}(t), \tilde{u}(t), t] + \frac{\partial f}{\partial x}[\tilde{x}(t), \tilde{u}(t), t][x(t) - \tilde{x}(t)]$$
$$+ \frac{\partial f}{\partial u}[\tilde{x}(t), \tilde{u}(t), t][u(t) - \tilde{u}(t)] + \text{ higher-order terms}$$

18

On defining the Jacobian matrix as the coefficient matrix

$$A(t) = \frac{\partial f}{\partial x}(\tilde{x}(t), \tilde{u}(t), t)$$

$$B(t) = \frac{\partial f}{\partial u}(\tilde{x}(t), \tilde{u}(t), t)$$

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

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

$\square$

Lemma B.3. *(**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:*

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

Proof. Similar to [1], 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. $\square$

Proposition 7. *The process of gradient updates in the IOSSM(**ABC**) is a spectral space skewing:*

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

Proof. 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 [8].

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)

$$\left\|\zeta(t)^{\frac{1}{2}}p_n^{(t)}\chi^{(t)}\right\|_{\nu^{(t)}}^2 = \int \left(\zeta(t)^{\frac{1}{2}}p_n^{(t)}\chi^{(t)}\right)^2 \frac{\omega^{(t)}}{\zeta(t)\left(\chi^{(t)}\right)^2}$$

$$= \int \left(p_n^{(t)}\right)^2 \omega^{(t)}$$

$$= \left\|p_n^{(t)}\right\|_{\mu^{(t)}}^2 = 1.$$

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.

$\square$

Lemma B.4. *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.

Proof. For the transformation

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

We have

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

$$= \boldsymbol{T}^{-1}[\boldsymbol{A}x(t) + \boldsymbol{B}u(t)]$$

$$= \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}z(t) + \boldsymbol{T}^{-1}\boldsymbol{B}u(t)$$

$\square$

Definition 3. (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:*

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

Lemma B.5. *The TOSSM($\boldsymbol{ABC}$) is K-Lipschitz with $\mathrm{eig}(\boldsymbol{A}) \le K^2$.*

Proof. For a complex function, it has

$$\|f \circ g\|_{\mathrm{Lip}} \le \|f\|_{\mathrm{Lip}} \cdot \|g\|_{\mathrm{Lip}}$$

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

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

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 $\vec{v_i}$ for $i = 1, 2, \ldots, n$, with corresponding eigenvalues λ_i for $i = 1, 2, \ldots, n$. Since $\vec{v_i}$ are pairwise orthogonal (not rigorously, as there may not be exactly n of them), we can assume that $\vec{v_i}$ 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 $\vec{x} = \sum_{i=1}^n x_i \vec{v_i}$. we have:

$$
\begin{aligned}
& \left\langle \left(A^T A - K^2 I \right) \vec{x}, \vec{x} \right\rangle \le 0 \\
\Longleftrightarrow \quad & \left\langle \left(A^T A - K^2 I \right) \sum_{i=1}^n x_i \vec{v_i}, \sum_{i=1}^n x_i \vec{v_i} \right\rangle \le 0 \\
\Longleftrightarrow \quad & \sum_{i=1}^n \sum_{j=1}^n x_i x_j \left\langle \left(A^T A - K^2 I \right) \vec{v_i}, \vec{v_j} \right\rangle \le 0 \\
\Longleftrightarrow \quad & \sum_{i=1}^n \sum_{j=1}^n x_i x_j \left[\left(A^T A - K^2 I \right) \vec{v_i} \right]^T \vec{v_j} \le 0
\end{aligned}
$$

because

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

so we have

$$\sum_{i=1}^n \sum_{j=1}^n x_i x_j \left[\left(A^T A - K^2 I \right) \vec{v_i} \right]^T \vec{v_j} \le 0 \Longleftrightarrow \sum_{i=1}^n x_i^2 \left(\lambda_i - K^2 \right) \le 0$$

specifically

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

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

C Additional Experiments

C.1 Datasets

We conduct experiments on 8 real-world datasets to evaluate the performance of our model, and the details of these datasets are provided in Table 6. Dimension denotes the variate number of each dataset. Dataset Size

denotes the total number of time points in the train, Validation, and Test) split, respectively. Forecasting Length denotes the future time points to be predicted, and four prediction settings are included in each dataset. Frequency denotes the sampling interval of time points.

Specifically, **ETT** [42] contains 7 factors of electricity transformer from July 2016 to July 2018. We use four subsets where ETTh1 and ETTh2 is recorded every hour, ETTm1 and ETTm2 is recorded every 15 minutes. **Exchange** [37] collects the panel data of daily exchange rates from 8 countries from 1990 to 2016. **Crypots** comprises historical transaction data for various cryptocurrencies, including Bitcoin and Ethereum. We select samples with *Asset_ID* set to 0, and remove the column *Count*. We download the data from https://www.kaggle.com/competitions/g-research-crypto-forecasting/data?select=supplemental_train.csv. **Weather** [37] includes 21 meteorological factors collected every 10 minutes from the Weather Station of the Max Planck Bio-geochemistry Institute in 2020. **Air Convection**[1] dataset is obtained by scraping the data from NOAA, and it includes the data for the entire year of 2023. The dataset consists of 10 variables, including air humidity, pressure, convection characteristics, and others. The data was sampled at intervals of 15 minutes and averaged over the course of the entire year.

Table 6: Detailed dataset descriptions.

Dataset	Dimension	Forecasting Length	Dataset Size	Information (Frequency)
ETTm1	7	{96, 192, 336, 720}	(34369, 11425, 11425)	Electricity (15 min)
ETTh1	7	{96, 192, 336, 720}	(8445, 2785, 2785)	Electricity (Hourly)
ETTm2	7	{96, 192, 336, 720}	(34369, 11425, 11425)	Electricity (15 min)
ETTh2	7	{96, 192, 336, 720}	(8545, 2881, 2881)	Electricity (Hourly)
Exchange	8	{96, 192, 336, 720}	(5120, 665, 1422)	Exchange rate (Daily)
Crypto	7	{96, 192, 336, 720}	(125699, 17891, 35873)	Finance (1 min)
Weather	21	{96, 192, 336, 720}	(36696, 5175, 10440)	Weather (10 min)
Air Convection	10	{96, 192, 336, 720}	(14647, 4882, 4882)	Weather (15 min)

C.2   Baselines

Mamba4TS: Mamba4TS is a novel SSM architecture tailored for TSF tasks, featuring a parallel scan (https://github.com/alxndrTL/mamba.py/tree/main). Additionally, this model adopts a patching operation with both patch length and stride set to 16. We use the recommended configuration as our experimental settings with a batch size of 32, and the learning rate is 0.0001.

S-Mamba [34]: S-Mamba utilizes a linear tokenization of variates and a bidirectional Mamba layer to efficiently capture inter-variate correlations and temporal dependencies. This approach underscores its potential as a scalable alternative to Transformer technologies in TSF. We download the source code from: https://github.com/wzhwzhwzh0921/S-D-Mamba and adopt the recommended setting as its experimental configuration.

RWKV-TS [15]: RWKV-TS is an innovative RNN-based architecture for TSF that offers linear time and memory efficiency. We download the source code from: https://github.com/howard-hou/RWKV-TS. We follow the recommended settings as experimental configuration.

TimeMachine [2]: TimeMachine utilizes a unique quadruple-Mamba architecture to handle both channel-mixing and channel-independence effectively, allowing for precise content selection based on contextual cues at multiple scales. We download the source code from: https://github.com/Atik-Ahamed/TimeMachine. We use the recommended configuration as the experimental settings where the batch size is 64, the learning rate is 0.001 to train the model in 10 epochs for fair comparison.

Koopa [25]: Koopa is a novel forecasting model that tackles non-stationary time series using Koopman theory to differentiate time-variant dynamics. It features a Fourier Filter and Koopman Predictors within a stackable block architecture, optimizing hierarchical dynamics learning. We download the source code from:

---

[1]https://www.psl.noaa.gov/

https://github.com/thuml/Koopa. We set the lookback window to fixed values of {96, 192, 336, 720} instead of twice the output length as in the original experimental settings.

iTransformer [26]: iTransformer modifies traditional Transformer models for time series forecasting by inverting dimensions and applying attention and feed-forward networks across variate tokens. We download the source code from: https://github.com/thuml/iTransformer. We follow the recommended settings as experimental configuration.

PatchTST [28]: PatchTST introduces a novel design for Transformer-based models tailored to time series forecasting. It incorporates two essential components: patching and channel-independent structure. We obtain the source code from: https://github.com/PatchTST. This code serves as our baseline for long-term forecasting, and we follow the recommended settings for our experiments.

LTSF-Linear [39]: In LTSF-Linear family, DLinear decomposes raw data into trend and seasonal components and NLinear is just a single linear models to capture temporal relationships between input and output sequences. We obtain the source code from: https://github.com/cure-lab/LTSF-Linear, using it as our long-term forecasting baseline and adhering to recommended settings for experimental configuration.

TimesNet [38]: TimesNet is a task-generic framework for time series analysis that processes complex temporal variations by transforming one-dimensional time series into two-dimensional tensors based on the observation of multi-periodicity in time series. We obtain the source code from: https://github.com/thuml/Time-Series-Library and follow the recommended settings.

GPT4TS [45]: This study explores the application of pre-trained language models to time series analysis tasks, demonstrating that the Frozen Pretrained Transformer (FPT), without modifications to its core architecture, achieves state-of-the-art results across various tasks. We download the source code from: https://github.com/DAMO-DI-ML/NeurIPS2023-One-Fits-All. We follow the recommended settings as experimental configuration.

Crossformer [41]: Crossformer explicitly exploits dependencies across channel dimensions and time steps. It introduces a two-stage attention mechanism for learning dependencies and achieves linear complexity in its operations. We train the model integrated in: https://github.com/thuml/Time-Series-Library.

Scaleformer [30]: Scaleformer introduces a multi-scale framework that improves the performance of transformer-based time series forecasting models, achieving higher accuracy with minimal additional computational cost. We download the source code from: https://github.com/BorealisAI/scaleformer and use the recommended configuration as the experimental settings.

TiDE [6]: TiDE is an MLP-based encoder-decoder model that combines the lightweight and fast training attributes of linear models while also handling covariates and nonlinear dependencies. We train the model integrated in: https://github.com/thuml/Time-Series-Library.

Stationary [24]: Non-stationary Transformer focuses on exploring the stabilization issues in time series forecasting tasks. We train the model integrated in: https://github.com/thuml/Time-Series-Library and use the recommended configuration as the experimental settings.

FEDformer [44]: FEDformer introduces an attention mechanism with a low-rank approximation in the frequency domain, along with a mixed decomposition to control distribution shifts. We adheres to the recommended settings for the experimental configuration and download the code from https://github.com/MAZiqing/FEDformer.

C.3 Experiments Setting

All experiments are conducted on the NVIDIA RTX3090-24G and A6000-48G GPUs. The Adam optimizer is chosen. A grid search is performed to determine the optimal hyperparameters, including the learning rate from {0.0001, 0.0005, 0.001} and patch length from {8, 16, 24}. The stride is equal to the patch length. The primary mode in the variable kernel is 64, the hidden dim is 256, the state space dim is 64, the piecewise scale is based on $log2(L)$ (L is the patches number), the model layers are 2, and the input length is 96. All experimental results are averaged over two runs.

C.4 Full Results

As shown in Table 7, we provide the full experiment results.

Table 7: Multivariate long-term series forecasting results in mainstream datasets with an input length of 96 and prediction horizons of {96, 192, 336, 720}. **Red**: best, **Blue**: second best. The bottom part introduces the variable Kernel for multivariate variables.

Model		Time-SSM (Ours)		Mamba4TS (Temporal Emb.)		S-Mamba (Channel Emb.[34])		RWKV-TS [15]		TimeMachine [2]		Koopa [25]		InvTrm [26]		PatchTST [28]		DLinear [39]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.377	0.394	0.386	0.400	0.388	0.407	0.383	0.401	0.373	0.393	0.385	0.408	0.393	0.409	0.375	0.396	0.396	0.410
	192	0.423	0.424	0.426	0.430	0.443	0.439	0.441	0.431	0.430	0.421	0.441	0.431	0.448	0.442	0.429	0.426	0.449	0.444
	336	0.466	0.437	0.484	0.451	0.492	0.467	0.493	0.465	0.480	0.443	0.474	0.454	0.491	0.465	0.461	0.448	0.487	0.465
	720	0.452	0.448	0.481	0.472	0.511	0.499	0.501	0.487	0.463	0.456	0.501	0.480	0.518	0.501	0.469	0.471	0.515	0.512
	AVG	0.425	0.426	0.444	0.438	0.459	0.453	0.454	0.446	0.437	0.428	0.450	0.443	0.463	0.454	0.434	0.435	0.462	0.458
ETTh2	96	0.290	0.341	0.297	0.347	0.296	0.347	0.290	0.342	0.284	0.336	0.317	0.359	0.302	0.351	0.295	0.344	0.353	0.405
	192	0.368	0.387	0.392	0.409	0.377	0.398	0.372	0.393	0.363	0.389	0.375	0.399	0.379	0.399	0.375	0.399	0.482	0.479
	336	0.416	0.430	0.424	0.436	0.425	0.435	0.417	0.431	0.399	0.419	0.436	0.446	0.423	0.432	0.420	0.429	0.588	0.539
	720	0.424	0.439	0.431	0.448	0.427	0.446	0.421	0.442	0.427	0.443	0.460	0.463	0.429	0.447	0.431	0.451	0.833	0.658
	AVG	0.374	0.399	0.386	0.410	0.381	0.407	0.375	0.402	0.368	0.397	0.397	0.417	0.383	0.407	0.380	0.406	0.564	0.520
ETTm1	96	0.329	0.365	0.331	0.368	0.332	0.368	0.328	0.366	0.313	0.352	0.322	0.360	0.343	0.377	0.326	0.365	0.345	0.372
	192	0.370	0.379	0.376	0.391	0.378	0.393	0.372	0.389	0.362	0.382	0.378	0.393	0.379	0.394	0.365	0.383	0.382	0.391
	336	0.396	0.402	0.406	0.413	0.409	0.414	0.401	0.409	0.391	0.400	0.405	0.413	0.418	0.418	0.396	0.405	0.413	0.413
	720	0.449	0.440	0.469	0.452	0.476	0.453	0.462	0.446	0.446	0.435	0.473	0.447	0.488	0.458	0.458	0.439	0.472	0.450
	AVG	0.386	0.396	0.396	0.406	0.399	0.407	0.391	0.403	0.378	0.392	0.395	0.403	0.407	0.412	0.403	0.398	0.403	0.406
ETTm2	96	0.176	0.260	0.186	0.268	0.182	0.267	0.181	0.264	0.175	0.255	0.180	0.261	0.184	0.269	0.177	0.260	0.192	0.291
	192	0.246	0.305	0.261	0.320	0.248	0.309	0.245	0.307	0.236	0.297	0.244	0.304	0.253	0.313	0.246	0.308	0.284	0.360
	336	0.305	0.344	0.331	0.366	0.312	0.350	0.306	0.344	0.302	0.340	0.300	0.340	0.313	0.351	0.302	0.343	0.371	0.420
	720	0.406	0.405	0.418	0.416	0.412	0.406	0.406	0.406	0.394	0.394	0.398	0.400	0.413	0.406	0.407	0.405	0.532	0.511
	AVG	0.283	0.328	0.299	0.343	0.289	0.333	0.285	0.330	0.277	0.323	0.281	0.326	0.291	0.335	0.283	0.329	0.345	0.396
Exchange	96	0.083	0.202	0.086	0.205	0.087	0.209	0.129	0.256	0.087	0.203	0.093	0.215	0.097	0.222	0.093	0.212	0.094	0.227
	192	0.170	0.295	0.173	0.297	0.180	0.303	0.231	0.346	0.180	0.300	0.189	0.313	0.201	0.319	0.185	0.325	0.185	0.325
	336	0.334	0.418	0.340	0.423	0.330	0.417	0.380	0.448	0.346	0.423	0.371	0.443	0.327	0.416	0.338	0.422	0.330	0.437
	720	0.824	0.677	0.855	0.696	0.860	0.700	0.883	0.704	0.943	0.720	0.908	0.726	0.885	0.715	0.900	0.711	0.774	0.673
	AVG	0.352	0.398	0.364	0.405	0.364	0.407	0.406	0.439	0.389	0.412	0.390	0.424	0.366	0.416	0.383	0.416	0.346	0.416
Crypto	96	0.177	0.142	0.179	0.143	0.187	0.147	0.176	0.139	0.182	0.138	0.181	0.143	0.183	0.144	0.177	0.141	0.183	0.155
	192	0.188	0.152	0.188	0.154	0.191	0.153	0.186	0.151	0.191	0.153	0.191	0.153	0.190	0.155	0.188	0.152	0.195	0.169
	336	0.195	0.162	0.197	0.166	0.197	0.163	0.192	0.162	0.197	0.164	0.208	0.173	0.199	0.167	0.195	0.164	0.206	0.180
	720	0.206	0.183	0.207	0.186	0.216	0.190	0.205	0.184	0.208	0.183	0.215	0.189	0.212	0.189	0.208	0.185	0.219	0.201
	AVG	0.192	0.160	0.193	0.162	0.198	0.163	0.190	0.159	0.195	0.160	0.199	0.165	0.196	0.164	0.192	0.161	0.201	0.176
Air	96	0.441	0.307	0.451	0.314	0.468	0.329	0.447	0.308	0.441	0.305	0.443	0.307	0.470	0.337	0.465	0.331	0.441	0.325
	192	0.461	0.325	0.472	0.331	0.481	0.340	0.467	0.328	0.465	0.331	0.451	0.329	0.485	0.349	0.477	0.341	0.460	0.338
	336	0.463	0.339	0.468	0.342	0.485	0.351	0.461	0.339	0.473	0.348	0.468	0.342	0.499	0.363	0.484	0.353	0.461	0.341
	720	0.470	0.358	0.492	0.379	0.501	0.386	0.482	0.367	0.488	0.374	0.488	0.369	0.516	0.401	0.504	0.392	0.474	0.359
	AVG	0.459	0.332	0.471	0.342	0.484	0.352	0.464	0.336	0.467	0.340	0.463	0.337	0.493	0.363	0.483	0.354	0.459	0.341
Weather	96	0.167	0.212	0.175	0.215	0.165	0.208	0.175	0.217	0.170	0.215	0.158	0.203	0.175	0.216	0.176	0.217	0.197	0.259
	192	0.217	0.255	0.223	0.257	0.215	0.254	0.219	0.256	0.214	0.253	0.211	0.252	0.225	0.257	0.223	0.257	0.238	0.299
	336	0.274	0.294	0.278	0.297	0.273	0.297	0.275	0.298	0.266	0.292	0.267	0.292	0.280	0.298	0.277	0.296	0.282	0.331
	720	0.351	0.345	0.355	0.349	0.354	0.349	0.353	0.349	0.347	0.345	0.351	0.346	0.358	0.350	0.354	0.348	0.350	0.388
	AVG	0.252	0.276	0.258	0.280	0.252	0.277	0.256	0.280	0.249	0.276	0.247	0.273	0.260	0.280	0.258	0.280	0.267	0.319
$1^{st}/2^{st}$ Count		31	34	4	3	5	4	14	12	-	-	12	13	1	1	16	14	9	3

Remark C.1. *TimeMachine is an ensemble learning model that falls outside the scope of our comparison. However, we still include the experimental results in the table for future reference in research. It is worth mentioning that TimeMachine has several times the number of parameters and complexity compared to Time-SSM, yet it still performs worse than Time-SSM on five datasets.*

C.5 Hyper-parameter

As shown in Figure 7, we conducted a hyperparameter analysis of the Time-SSM state spectral space dimensions. It can be observed that for the chunked version of Time-SSM, 64 dimensions yield optimal results, which further validates our inference in Section 4.3 (Representation Ability). However, the absence of patch operations necessitates higher spatial dimensions for representation.

D Discussion

Although some recent models, such as Pathformer [5] and Time-LLM [19], have surpassed Time-SSM in performance, we believe that operations such as multi-scale patching and large model enhancement can be integrated into Time-SSM. This integration can lead to the emergence of Time-multipatch-SSM or even Time-LargeSSM. It is important to note that the focus of this paper is to explore the feasibility of SSM in
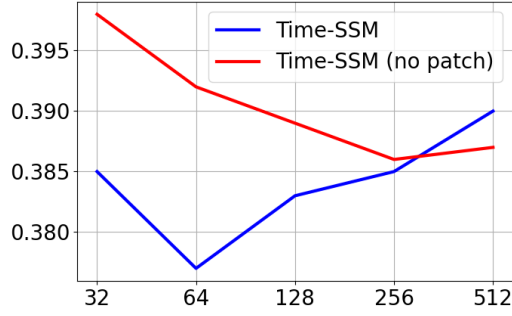
Figure 7: [?] Hyper-parameter analysis of MSE in ETTh1(96-96)

Table 8: Additional Baselines.

Model		Time-SSM (Ours)		GPT4TS [45]		TimesNet [38]		Crossformer [41]		Scaleformer [30]		TiDE [6]		NLinear [39]		Stationary [24]		FEDformer [44]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.377	0.394	0.398	0.424	0.384	0.402	0.391	0.417	0.396	0.440	0.479	0.464	0.386	0.400	0.513	0.419	0.376	0.419
	192	0.423	0.424	0.441	0.436	0.436	0.429	0.449	0.452	0.434	0.460	0.524	0.490	0.400	0.430	0.533	0.505	0.420	0.448
	336	0.466	0.437	0.492	0.466	0.491	0.469	0.510	0.489	0.462	0.476	0.566	0.513	0.480	0.443	0.586	0.535	0.460	0.465
	720	0.452	0.448	0.487	0.483	0.521	0.500	0.594	0.567	0.494	0.500	0.595	0.558	0.486	0.472	0.644	0.616	0.505	0.507
	AVG	0.374	0.399	0.457	0.450	0.458	0.450	0.486	0.481	0.447	0.469	0.541	0.506	0.438	0.436	0.569	0.519	0.440	0.460
ETTh2	96	0.290	0.341	0.312	0.360	0.340	0.374	0.641	0.549	0.364	0.407	0.400	0.440	0.324	0.369	0.476	0.458	0.358	0.397
	192	0.368	0.387	0.387	0.405	0.402	0.414	0.896	0.656	0.466	0.458	0.528	0.509	0.413	0.421	0.512	0.493	0.429	0.439
	336	0.396	0.402	0.424	0.437	0.452	0.452	0.936	0.690	0.479	0.476	0.627	0.560	0.455	0.455	0.552	0.551	0.496	0.487
	720	0.449	0.440	0.433	0.453	0.462	0.468	1.390	0.863	0.487	0.492	0.874	0.679	0.457	0.466	0.562	0.560	0.463	0.474
	AVG	0.374	0.399	0.389	0.414	0.414	0.427	0.436	0.453	0.445	0.458	0.607	0.547	0.412	0.428	0.526	0.516	0.437	0.449
ETTm1	96	0.329	0.365	0.335	0.369	0.338	0.375	0.366	0.400	0.355	0.398	0.364	0.387	0.339	0.369	0.386	0.398	0.379	0.419
	192	0.370	0.379	0.374	0.385	0.374	0.387	0.396	0.414	0.428	0.455	0.400	0.402	0.379	0.386	0.460	0.445	0.426	0.440
	336	0.396	0.402	0.407	0.406	0.410	0.411	0.439	0.443	0.524	0.487	0.428	0.424	0.411	0.407	0.495	0.464	0.445	0.459
	720	0.449	0.440	0.469	0.442	0.478	0.450	0.540	0.509	0.558	0.517	0.486	0.460	0.478	0.442	0.584	0.515	0.543	0.491
	AVG	0.386	0.396	0.396	0.401	0.400	0.406	0.435	0.442	0.466	0.464	0.420	0.418	0.402	0.401	0.481	0.456	0.448	0.452
ETTm2	96	0.176	0.260	0.190	0.275	0.187	0.267	0.273	0.346	0.182	0.275	0.207	0.305	0.177	0.257	0.192	0.274	0.203	0.287
	192	0.246	0.305	0.253	0.313	0.249	0.309	0.350	0.421	0.251	0.318	0.290	0.364	0.241	0.297	0.280	0.339	0.269	0.328
	336	0.305	0.344	0.321	0.360	0.321	0.351	0.474	0.505	0.340	0.375	0.377	0.422	0.302	0.337	0.334	0.361	0.325	0.336
	720	0.406	0.405	0.411	0.406	0.408	0.403	1.347	0.812	0.435	0.433	0.558	0.524	0.405	0.396	0.417	0.413	0.421	0.415
	AVG	0.283	0.328	0.294	0.339	0.291	0.333	0.611	0.521	0.302	0.350	0.358	0.404	0.281	0.322	0.306	0.347	0.305	0.349
Exchange	96	0.083	0.202	0.091	0.212	0.107	0.234	0.256	0.367	0.155	0.285	0.094	0.218	0.089	0.208	0.111	0.237	0.148	0.278
	192	0.170	0.295	0.183	0.304	0.226	0.344	0.469	0.508	0.274	0.384	0.184	0.307	0.180	0.301	0.219	0.335	0.271	0.315
	336	0.334	0.418	0.328	0.417	0.367	0.448	0.901	0.741	0.452	0.498	0.349	0.431	0.320	0.409	0.421	0.476	0.460	0.427
	720	0.824	0.677	0.880	0.704	0.964	0.746	1.398	0.965	1.172	0.839	0.852	0.698	0.832	0.685	1.092	0.769	0.964	0.746
	AVG	0.352	0.398	0.371	0.409	0.416	0.443	0.756	0.645	0.513	0.502	0.370	0.413	0.355	0.401	0.461	0.454	0.519	0.429
Weather	96	0.167	0.212	0.203	0.244	0.172	0.220	0.164	0.232	0.288	0.365	0.202	0.261	0.168	0.208	0.173	0.223	0.217	0.296
	192	0.217	0.255	0.247	0.277	0.219	0.261	0.211	0.276	0.368	0.425	0.242	0.298	0.217	0.255	0.245	0.285	0.219	0.261
	336	0.274	0.294	0.297	0.311	0.280	0.306	0.269	0.327	0.447	0.469	0.287	0.335	0.267	0.292	0.321	0.338	0.339	0.380
	720	0.351	0.345	0.368	0.356	0.365	0.359	0.355	0.404	0.640	0.574	0.355	0.404	0.351	0.386	0.414	0.410	0.403	0.428
	AVG	0.252	0.276	0.279	0.297	0.259	0.287	0.250	0.310	0.436	0.458	0.271	0.320	0.251	0.275	0.288	0.314	0.309	0.360
$1^{st}/2^{st}$Count		25	28	11	13	1	4	5	-	1	-	1	-	17	19	-	-	3	-

the field of time series prediction, with a commitment to the simplest SSM framework and smallest model size. Furthermore, we are the first to provide a detailed theoretical basis for applying SSM to TSF tasks in our article. We provide guidance for future studies on how to apply SSM effectively.

E   Limitation

This paper provides comprehensive theoretical guidance for the application of State Space Models (SSMs) to time series data. However, there are still areas that warrant further exploration. For instance, while this paper intuitively explains the S4-real initialization method as a rough approximation of dynamics, it lacks rigorous mathematical proof. Furthermore, experimental results demonstrate a significant performance improvement of the Hippo-LegP method compared to Hippo-LegT. As this framework is built upon a finite measure approximation, it remains unclear whether it can be extended to Hippo-LegS, which employs an exponential decay approximation.