SiMBA: Simplified Mamba-based Architecture for Vision and Multivariate Time series

Badri N. Patro[1] and Vijay S, Agneeswaran[1]

Microsoft
{badripatro,vagneeswaran}@microsoft.com

Abstract. Transformers have widely adopted attention networks for sequence mixing and MLPs for channel mixing, playing a pivotal role in achieving breakthroughs across domains. However, recent literature highlights issues with attention networks, including low inductive bias and quadratic complexity concerning input sequence length. State Space Models (SSMs) like S4 and others (Hippo, Global Convolutions, liquid S4, LRU, Mega, and Mamba), have emerged to address the above issues to help handle longer sequence lengths. Mamba, while being the state-of-the-art SSM, has a stability issue when scaled to large networks for computer vision datasets. We propose SiMBA, a new architecture that introduces Einstein FFT (EinFFT) for channel modeling by specific eigenvalue computations and uses the Mamba block for sequence modeling. Extensive performance studies across image and time-series benchmarks demonstrate that SiMBA outperforms existing SSMs, bridging the performance gap with state-of-the-art transformers. Notably, SiMBA establishes itself as the new state-of-the-art SSM on ImageNet and transfer learning benchmarks such as Stanford Car and Flower as well as task learning benchmarks as well as seven time series benchmark datasets. The project page is available on this website https://github.com/badripatro/Simba

Keywords: Transformer · Mamba · Spectral Channel Mixing · State Space Model

1 Introduction

The evolution of language models in the technological landscape is transitioning from Large Language Models (LLMs) to the paradigm of Small Language Models (SLMs) inspired by cutting-edge SLM architectures like Mistral [22], Phi [28], Orca [38], among others. At the heart of both LLMs and SLMs lies the power of transformers, where the layers are not only scaled but also exhibit scaling in both the token and channel modeling. This has made transformers the building blocks of LLMs and SLMs.

Transformers operate through two fundamental mixing directions: sequential modeling, involving the interaction of one token with another within the input sequence, and channel modeling, facilitating interactions within the channel dimension or across input features. Traditionally, multi-headed self-attention (MHSA) was employed by transformers for sequence modeling, but its computational complexity of $O(N^2)$ posed inefficiencies and performance challenges, particularly for longer sequences. To address this limitation, and recognize the need for handling extended input sequences in domains like genomics or protein folding, a novel approach emerged with the introduction of the

Structured State Space model (S4) [15]. This model leverages state-space-based sequential modeling, offering enhanced efficiency and performance for processing longer input sequences.

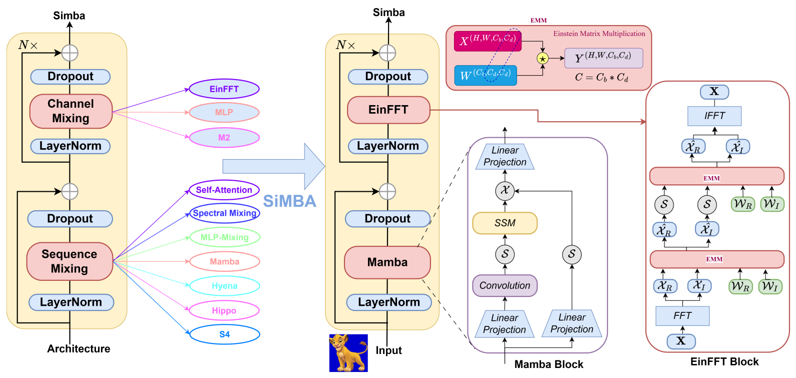

Fig. 1: Simplified Mamba Based Architecture.

However, S4 has been less effective in catering to modeling information-dense data, particularly in domains such as computer vision, and faces challenges in discrete scenarios like genomic data. In response to these limitations, several other SSMs, including Hippo [14], H3 [9], Hyena [48], RetNet [52], RWKV [47] and S4nd [39] and Gated State Spaces [37], have been proposed in the literature to address the shortcomings of S4, specifically focusing on enhancing its ability to handle long-range dependencies. Additionally, global convolutions-based [11] state-space models have been introduced as an alternative solution to mitigate issues related to long-range dependencies. Further expansions in the S4 model family include Liquid S4 models [18] and a variant known as Mega [36], which interprets S4 through an exponential moving average approach. These advancements collectively contribute to a more versatile and effective suite of models for diverse applications. Mamba [13] is a selective state space sequence modeling technique proposed recently to address difficulties typical state space models had in handling long sequences efficiently. The typical state space models have trouble propagating or forgetting information in long sequences. Mamba handles this difficulty, by incorporating the current token in the state space, achieving in-context learning. Mamba in general addresses the two concerns of transformers, namely its lack of inductive bias by using CNN and its quadratic complexity by using input-specific state space models.

The current instantiation of Mamba has stability issues i.e. the training loss is not converging while scaling to large-sized networks on the ImageNet dataset. It is not clear as to why Mamba has this instability while scaling to large networks. This leads to the problem of vanishing/exploding gradients commonly observed in Mamba in general. Existing literature, such as Oppenheim and Verghese's work [43], establishes that linear state space models exhibit stability when all eigenvalues of matrix A are negative real numbers. This motivates the need for a stable Mamba architecture, as presented in this paper, and specifically, we use Fourier Transforms followed by a learnable layer with non-linearity. SiMBA also introduces residual connections with dropouts, which help in handling solving the instability issues in Mamba, as illustrated in Figure-1. This strategic application of learnable Fourier Transforms aims to manipulate the eigenvalues, ensuring they are negative real numbers. Thus, The proposed channel modeling technique, named EinFFT, is a distinctive contribution to the field. This is a unique contribution, as SSMs in the literature have not addressed channel modeling explicitly.

Table 1: Overview of Large Vision Models for Image Recognition Task.

Method	Seqence Mixing	Channel Mixing	Models
Convolution	ConvNet	MLP	AlexNet, VGG, ResNet, RegNetY
Transformer	Attention	MLP	ViT, Deit, T2T, TnT
MLP-Mixer	MLP	MLP	Mlp-Mixer, GMLP, DynamicMLP
Spectral-Mixer	FFT/Wavelet	MLP	GFNet, AFNO,WaveMix
State Space	SSM	-	S4, Hyena, Hippo, H3, Mamba
Conv-Transformer	Attention+ ConvNet	MLP	Swin, CvT, CMT, CSwin
Spectral-Transformer	FFT +Attention	MLP	SiMBA, SVT, WaveViT
SiMBA	Mamba	EinFFT	SiMBA

Our architectural investigation explores diverse state space models, including S4 [15], Hippo [14], Hyena [48], and Mamba [13], alongside attention models such as DeIT [57], ViT [7], and Swin [33], as well as spectral models like GFNet [50], and AFNO [16] for sequence modeling. Additionally, we explore various channel modeling alternatives, including MLP, and Monarch Mixer, and introduce a novel technique called EinFFT. Through extensive experimentation, we have identified the most efficient and streamlined state space architecture to date, named SiMBA. This novel architecture incorporates Mamba for sequence modeling and introduces EinFFT as a new channel modeling technique. SiMBA effectively addresses the instability issues observed in Mamba when scaling to large networks. The architectural alternatives explored for large-scale sequence modeling are depicted in Figure-1. Table-1 provides an overview of large vision models used for image recognition tasks, categorizing them based on their sequence mixing and channel mixing techniques. It highlights a diverse range of models, including those based on convolutional models, transformers models, MLP-mixers, spectral-mixers models, and state space methods. Additionally, it introduces hybrid models combining convolution with transformers or spectral approaches. Lastly, it presents SiMBA, a novel model utilizing Mamba for sequence mixing and EinFFT for channel mixing.

The contribution of this paper is as follows:

- EinFFT: A new technique for channel modeling known as EinFFT is proposed, which solves the stability issue in Mamba. This uses Fourier transforms with non-linearity to model eigenvalues as negative real numbers, which solves instability [43]. We validate this technique on two data modalities time series and ImageNet dataset.
- SiMBA: We propose an optimized Mamba architecture for computer vision tasks, known as SiMBA which uses EinFFT for channel modeling and Mamba for token mixing to handle inductive bias and computational complexity. We also show the

importance of different architectural elements of SiMBA with ablation studies including residual connections and dropouts.

- Performance Gap: It must be noted that SiMBA is the first SSM to close the performance gap with state-of-the-art attention-based transformers on the ImageNet dataset and six standard time series datasets. We show with extensive performance analysis how SiMBA achieves state-of-art performance compared to V-Mamba and Vision Mamba in the vision domain. We have shown the generalization capability of SiMBA architecture in other domains like time series to handle long sequences.
- We show the performance of SiMBA on transfer learning datasets like CIFAR, Stanford Car, and Flower. We also show that SiMBA achieves comparable performance even in other tasks such as instance segmentation with the MS COCO dataset.

2 Related work

At the heart of SiMBA's transformer-based architecture lies a combination of multi-headed self-attention and a multi-layer perceptron (MLP). When comparing SiMBA with attention-based transformers like ViT [7], DeIT [57], Swin [34], CSwin [6], and CVT [64], it becomes evident that ViT, although pioneering attention-based transformers in computer vision, faced challenges due to its quadratic attention complexity ($O(N^2)$). Subsequent models like DeIT, Swin, TvT, CVT, and CSwin aimed at refining transformer performance in computer vision and specific NLP tasks. Alternatively, MLP Mixer architectures such as MLP Mixers [55] and DynamicMLP [62] sought to replace attention networks with MLP to address these challenges. Efforts to mitigate attention complexity led to the development of transformers like GFNet [50], AFNO [16], FNet [25], and FourierFormer [40], incorporating Fourier transforms to reduce complexity to $O(N \log(N))$. Despite this reduction, these models demonstrated a performance gap compared to state-of-the-art attention-based transformers. The emergence of transformers like SVT [46], WaveViT [69], and SpectFormer [45], combining initial spectral layers with deeper attention mechanisms, surpassed the performance of both attention-based and MLP-based transformers. This comprehensive comparative analysis highlights SiMBA's distinctive position in the evolving landscape of transformer architectures.

Attention-based transformers encounter limitations in modeling long input sequences, especially when dependencies extend beyond the attention window size. This constraint proves crucial in applications such as high-resolution imagery analysis and genomics. S4 pioneered state space models to address this issue by reducing complexity to $O(N \log(N))$, enabling the handling of long-range dependencies in tasks like the long-range arena path-X [54]. Subsequent efforts, including Hippo and Long Convolutions [10], aimed to enhance state space models' efficiency but demonstrated a performance gap compared to state-of-the-art transformers. Hyena [48], H3 [9], and related models further improved state space models' effectiveness but faced challenges due to a lack of consideration for the current input. Mamba attempted to bridge this gap by parameterizing the current input in the state space model. However, state-space models often encounter instability issues due to their sensitivity to recurrent dynamics.

Vision Mamba [76] and V-Mamba [32] adapted the Mamba architecture for computer vision tasks, utilizing bi-directional and visual state space models. However, the performance study section reveals a performance gap between Vision Mamba, V-Mamba, and state-of-the-art transformer models like SpectFormer [45], SVT [46], WaveViT [69], and Volo [72]. SiMBA addresses this gap by incorporating Mamba for token mixing, replacing attention networks, and leveraging Einstein FFT (EinFFT) for channel mixing. SiMBA introduces the Einstein blending method for channel mixing, offering a novel approach without the constraints of requiring perfect square dimensions for sequence length N and channel dimensions. Furthermore, SiMBA adopts the pyramid version of the transformer architecture, providing a significant performance boost compared to vanilla state space models. While many state space models reduce complexity to $O(N\log(N))$, they often fall short of achieving the performance levels seen in state-of-the-art attention-based transformers.

3 Method

In this study, we introduce EinFFT, a novel approach for frequency-domain channel mixing utilizing Einstein Matrix multiplication. EinFFT is specifically designed for complex number representations of frequency components, enabling the effective capture of key patterns in Image patch data with a global view and energy compaction. It must be noted that EinFFT is also applicable for other sequence data modalities like time series or speech or even text data. We have validated EinFFT-based SiMBA for image and time series benchmark datasets.

3.1 Channel Mixing: Einstein FFT (EinFFT)

The Channel Mixing component of SiMBA (EinFFT) encompasses three main components: Spectral Transformation, Spectral Gating Network using Einstein Matrix multiplication, and Inverse Spectral Transformation as depicted in Figure 1.

Spectral Transformation For a given input function $\mathbf{f}(x)$ and its corresponding frequency domain conversion function $\mathcal{F}(k)$, by using Discrete Fourier Transform (DFT). Let $\mathcal{F}$ denote the Fourier transform of a function $f(x)$ and $\mathcal{F}^{-1}$ its inverse then

$$\mathcal{F}(k) = \int_D f(x)e^{-j2\pi kx}\mathrm{d}x = \int_D f(x)\cos(2\pi kx)\mathrm{d}x + j\int_D f(x)\sin(2\pi kx)\mathrm{d}x \qquad (1)$$

Here, k is the frequency variable, x is the spatial variable, and j is the imaginary unit. The real part of $\mathcal{F}$ is denoted as $Re(\mathcal{F})$, and the imaginary part as $Im(\mathcal{F})$. The complete conversion is expressed as $\mathcal{F} = Re(\mathcal{F}) + jIm(\mathcal{F})$. Fourier transform is employed to decompose the input signal into its constituent frequencies. This allows the identification of periodic or aperiodic patterns which are crucial for image recognition or detection tasks.

Convolution Theorem 1 *The convolution theorem states that the Fourier transform of the convolution of two functions in the spatial domain is equal to the pointwise product of their Fourier transforms. Mathematically, if $f(x)$ and $g(x)$ are two functions in the* *spatial domain, and $F(u)$ and $G(u)$ are their respective Fourier transforms, then the convolution theorem can be expressed as:*

$$\mathcal{F}\{f * g\}(u) = F(u) \cdot G(u)$$

where:$\mathcal{F}$ denotes the Fourier transform operator. $*$ denotes the convolution operator.

Rayleigh's Theorem 1 *Rayleigh's Theorem states that the total energy (or power) of a signal in the spatial domain is equal to the total energy in its frequency domain representation. Rayleigh's Theorem for continuous signals, and is used to express the equivalence of energy between an image patch in the spatial domain ($|\mathbf{g}(x)|^2$) and its representation in the frequency domain ($|\mathcal{G}(f)|^2$) as defined as:*

$$\int_{-\infty}^{\infty} |\mathbf{g}(x)|^2 \, dx = \int_{-\infty}^{\infty} |\mathcal{G}(f)|^2 \, df$$

Here, $\mathcal{G}(f) = \int_{-\infty}^{\infty} \mathbf{g}(x) e^{-j2\pi f x} \mathrm{d}x$, where x represents the channel dimension, and f denotes the frequency dimension. Rayleigh's Theorem describes the conservation of energy between the spatial domain and the frequency domain. It indicates that the integral of the squared magnitude of the signal in the time domain is equal to the integral of the squared magnitude of its Fourier transform in the frequency domain.

The implications of this theorem, emphasize that if the majority of the energy of an input image is concentrated in a limited number of frequency components, then an accurate representation of the image can be achieved by focusing on those specific components. This is a key insight from the theorem, emphasizing the concentration of energy in certain frequency components and the potential benefits of selectively considering those components for more efficient and informative representations in the frequency domain.

Frequency-domain Channel Mixing For a given input $\mathbf{X} \in \mathbb{R}^{N \times D}$ and its corresponding frequency domain conversion $\mathcal{X} \in \mathbb{R}^{N \times D}$, by using Fourier transform. The convolution theorem states that the Fourier transform of the convolutions on $\mathbf{X}$ with its kernel in the spatial domain can be equivalently represented as the product operations of their frequency-domain representations. This equivalence is expressed as follows:

$$\mathcal{F}(\mathbf{X} * W + B) = \mathcal{F}(\mathbf{X}) \cdot \mathcal{F}(W) + \mathcal{F}(B) = \mathcal{X}\mathcal{W} + \mathcal{B} \quad (2)$$

where $*$ denotes circular convolution, $\mathcal{W}$ and $\mathcal{B}$ represent the complex number weight and bias in the frequency domain, while W and B denote the weight and bias in the spatial domain, and $\mathcal{F}$ signifies the Discrete Fourier Transform (DFT). The output DFT operation is a complex number value $\mathcal{X} \in \mathbb{R}^{N \times D}$, multiplied with a complex number weight matrix $\mathcal{W} \in \mathbb{R}^{D \times D}$ and added with a complex number bias $\mathcal{B} \in \mathbb{R}^{D}$ using the Spectral Gating Network (SGN). To reduce complexity we perform Einstein Matrix Multiplication (EMM) to have an efficient block diagonal matrix. The SGN is expressed by the following formulation:

$$h^{\ell} = \sigma(h^{\ell-1}\mathcal{W}^{\ell} + \mathcal{B}^{\ell}), h^0 = \mathcal{X} \quad (3)$$

Here, $h^{\ell} \in \mathbb{R}^{N\times D}$ represents the final output, ℓ denotes the ℓ-th layer, and σ is the activation function. Considering both $\mathcal{X}$ and $\mathcal{W}$ as complex numbers, we extend Equation (3) by employing the multiplication rule for complex numbers. The extended formulation is as follows:

$$\begin{aligned}
Re(h)^{\ell} &= EMM(Re(h^{\ell-1})\mathcal{W}_r^{\ell}) - EMM(Im(h^{\ell-1})\mathcal{W}_i^{\ell}) + \mathcal{B}_r^{\ell} \\
Im(h)^{\ell} &= EMM(Re(h^{\ell-1})\mathcal{W}_i^{\ell}) + EMM(Im(h^{\ell-1})\mathcal{W}_r^{\ell}) + \mathcal{B}_i^{\ell} \\
h^{\ell} &= \sigma(Re(h)^{\ell}) + j\sigma(Im(h)^{\ell})
\end{aligned} \tag{4}$$

Here, $\mathcal{W}^{\ell} = \mathcal{W}_r^{\ell} + j\mathcal{W}_i^{\ell}$ and $\mathcal{B}^{\ell} = \mathcal{B}_r^{\ell} + j\mathcal{B}_i^{\ell}$ represent the real and imaginary parts, respectively. The implications of this theorem, emphasize that if the energy of an image patch is concentrated in a limited number of frequency components, then with the help of Rayleigh's Theorem a representation focusing on those specific frequency components can accurately capture the signal. this can be modeled with the help of a learnable spectral gating mechanism. We also apply a non-linear activation function to handle stability in the sequence modeling block (Mamba) by allowing a specific Eigenvalue that is required for convergence. We then apply EMM between input **I** and the weight $W \in \mathbb{R}^{C_b\times C_d\times C_d}$ along the channel dimensions, where the channels are arranged in number of blocks and number of subchannels in each block to make it block diagonal matrix (**I** from $\mathbb{R}^{N\times C}$ to $\mathbb{R}^{N\times C_b\times C_d}$, where $C = C_b \times C_d$, and $b << d$). This results in a blended feature tensor $Y \in \mathbb{R}^{N\times C_b\times C_d}$ as defined where $N = H \times W$. The formula for EMM is:

$$\mathbf{Y}^{N\times C_b\times C_d} = \mathbf{I}^{N\times C_b\times C_d} \boxed{\circledast} \mathbf{W}^{C_b\times C_d\times C_d}$$

Where $\boxed{\circledast}$ represents an Einstein matrix multiplication

The channel mixing in the frequency domain, denoted as EinFFT, is performed by separately computing the real and imaginary parts of frequency components. Subsequently, these parts are stacked to form a complex number, with the final result obtained by concatenating the real and imaginary parts. This methodology enhances the extraction of patterns while ensuring efficient utilization of computational resources.

Inverse Spectral Transformation : After learning in the frequency domain, $\mathcal{F}$ can be converted back into the time domain using the inverse conversion formulation:

$$f(x) = \int_D \mathcal{F}(k)e^{j2\pi kx}\mathrm{d}f = \int_D (Re(\mathcal{F}(k)) + jIm(\mathcal{F}(k))e^{j2\pi kx}\mathrm{d}f \tag{5}$$

The frequency spectrum, expressed as a combination of cos and sin waves, aids in discerning prominent frequencies and periodic patterns in time series signals. The terms FFT and IFFT are used for brevity. The Spectral Transformation stage leverages the Fourier transform for enhanced signal analysis, capturing important periodic patterns crucial for the Image classification task.

EinFFT Architecture for SiMBA Our method, EinFFT, leverages frequency-domain channel mixing through the application of Einstein Matrix multiplication on complex number representations. This enables the extraction of intricate data patterns with enhanced global visibility and energy concentration.

Considering channel dependencies is crucial for accurate class prediction as it allows the model to capture interactions and correlations between different variables. The frequency channel learner facilitates communication between different channels, and learning channel dependencies by sharing the same weights across L timestamps. For the l-th timestamp $\mathbf{X}_c^{:,(l)} \in \mathbb{R}^{N \times d}$, the frequency channel learner operates as follows:

$$\begin{aligned}
\mathcal{X}^{:,(l)} = \mathcal{X}_R^{:,(l)} + j\mathcal{X}_I^{:,(l)} &= \mathrm{FFT}(\mathbf{X}_c^{:,(l)}) \\
Re(h)^{\ell} + jIm(h)^{\ell} &= \mathrm{EMM}(\mathcal{X}^{:,(l)}, \mathcal{W}^c, \mathcal{B}^c) \text{ as in eq-4} \\
\mathcal{Y}^{:,(l)} = \mathcal{Y}_R^{:,(l)} + j\mathcal{Y}_I^{:,(l)} &= \sigma(Re(h)^{\ell}) + j\sigma(Im(h)^{\ell}) \\
\mathcal{Z}^{:,(l)} = \mathcal{Z}_R^{:,(l)} + j\mathcal{Z}_I^{:,(l)} &= \mathrm{EMM}(\mathcal{Y}^{:,(l)}, \mathcal{W}^c, \mathcal{B}^c) \text{ as in eq-4} \\
\mathbf{Z}_c^{:,(l)} &= \mathrm{IFFT}(\mathcal{Z}_R^{:,(l)} + j\mathcal{Z}_I^{:,(l)})
\end{aligned} \tag{6}$$

Here, $\mathcal{X}^{:,(l)} \in \mathbb{R}^{\frac{N}{2} \times d}$ represents the frequency components of $\mathbf{X}_c^{:,(l)}$. The operations FFT and IFFT are performed along the channel dimension. EinFFT refers to the frequency-domain channel Mixing introduced, taking $\mathcal{W}^c$ and $\mathcal{B}^c$ as the complex number weight matrix and biases, respectively. The output $\mathcal{Z}^{:,(l)} \in \mathbb{R}^{\frac{N}{2} \times d}$ of the EinFFT is then converted back to the time domain as $\mathbf{Z}^{:,(l)} \in \mathbb{R}^{N \times d}$. Finally, the outputs $\mathbf{Z}^{:,(l)}$ across L timestamps are ensembled to produce the overall output $\mathbf{Z}_t \in \mathbb{R}^{N \times L \times d}$.

3.2 Sequence Modeling: Mamba based SSM

To model a large sequence we use state space models instead of Multi-headed self-attention due to its complexity. The state space model [13, 15] is commonly known as a linear time-invariant system that map the input stimulation $x(t) \in \mathcal{R}^L$ to a response $y(t)$ through a hidden space $h(t) \in \mathcal{R}^N$. Structured state space sequence models (S4) are a recent class of sequence models for deep learning that are broadly related to RNNs, CNNs, and classical state space models. Mathematically, The Continuous-time Latent State spaces can be modeled as linear ordinary differential equations that use evolution parameter $A \in \mathcal{R}^{N \times N}$ and projection parameter $B \in \mathcal{R}^{N \times 1}$ and $C \in \mathcal{R}^{N \times 1}$ as follows:

$$\begin{aligned}
x'(t) &= \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \\
y(t) &= \boldsymbol{C}x(t) + \boldsymbol{D}u(t)
\end{aligned} \tag{7}$$

Discrete-time SSM: . The discrete form of SSM uses a time-scale parameter Δ to transform continuous parameters A, B, and C to discrete parameters $\bar{A}$, $\bar{B}$ and $\bar{C}$ using fixed formula $\bar{A} = f_A(\Delta, A)$, $\bar{B} = f_B(\Delta, A, B)$. The pair f_A, f_B is the discretization rule that uses a zero-order hold (ZOH) for this transformation. The equations are as follows.

$$\begin{aligned}
x_k &= \overline{\boldsymbol{A}}x_{k-1} + \overline{\boldsymbol{B}}u_k & \overline{\boldsymbol{A}} &= (\boldsymbol{I} - \Delta/2 \cdot \boldsymbol{A})^{-1}(\boldsymbol{I} + \Delta/2 \cdot \boldsymbol{A}) & & \\
y_k &= \overline{\boldsymbol{C}}x_k & \overline{\boldsymbol{B}} &= (\boldsymbol{I} - \Delta/2 \cdot \boldsymbol{A})^{-1}\Delta\boldsymbol{B} & \overline{\boldsymbol{C}} &= \boldsymbol{C}.
\end{aligned} \tag{8}$$

Convolutional Kernel Representation The discretized form of recurrent SSM in equation-8 is not practically trainable due to its sequential nature. To get efficient representation, we model continuous convolution as discrete convolution as it is a linear time-invariant system. For simplicity let the initial state be $x_{-1} = 0$. Then unrolling (8) explicitly yields:

$$\begin{array}{llll} x_0 = \overline{\boldsymbol{B}} u_0 & x_1 = \overline{\boldsymbol{A}\boldsymbol{B}} u_0 + \overline{\boldsymbol{B}} u_1 & x_2 = \overline{\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{A}\boldsymbol{B}} u_1 + \overline{\boldsymbol{B}} u_2 & \dots \\ y_0 = \overline{\boldsymbol{C}\boldsymbol{B}} u_0 & y_1 = \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}\boldsymbol{B}} u_1 & y_2 = \overline{\boldsymbol{C}\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}} u_1 + \overline{\boldsymbol{C}\boldsymbol{B}} u_2 & \dots \end{array}$$

This can be vectorized into a convolution (9) with an explicit formula for the convolution kernel (10).

$$\begin{aligned} y_k &= \overline{\boldsymbol{C}\boldsymbol{A}}^k \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}\boldsymbol{A}}^{k-1} \overline{\boldsymbol{B}} u_1 + \cdots + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}} u_{k-1} + \overline{\boldsymbol{C}\boldsymbol{B}} u_k \\ y &= \overline{\boldsymbol{K}} * u. \end{aligned} \tag{9}$$

$$\overline{\boldsymbol{K}} \in \mathbb{R}^L := \mathcal{K}_L(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}) := \left(\overline{\boldsymbol{C}\boldsymbol{A}}^i \overline{\boldsymbol{B}} \right)_{i \in [L]} = (\overline{\boldsymbol{C}\boldsymbol{B}}, \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}}, \dots, \overline{\boldsymbol{C}\boldsymbol{A}}^{L-1} \overline{\boldsymbol{B}}). \tag{10}$$

$\overline{\boldsymbol{K}}$ in equation (9) can be represented as a single (non-circular) convolution which can be computed very efficiently with FFTs.However, computing $\overline{\boldsymbol{K}}$ in (10) is non-trivial and is modelled as a $\overline{\boldsymbol{K}}$ the **SSM convolution kernel** or filter.

Specifically, we model input sequence using the state-of-the-art state space model Mamba [13]. Mamba identifies a critical weakness in existing models: their inability to perform content-based reasoning. To address this, Mamba introduces selective state spaces (SSMs) that allow the model to selectively propagate or forget information along the sequence length dimension based on the current token. While we apply the Mamba block for the vision task, we face the problem of stability issues (loss convergence issue) compared to other models like S4 or Hippo. We are providing one type of solution for the instability issue by preserving only negative eigenvalue. To perform this we need an extra module which we call the channel mixing, which was missing in the mamba block. We combine the channel mixing module with the Sequence mixing module and make a Simplified Mamba Based Architecture (SiMBA) as shown in the figure-1. The input token sequence $\mathbf{X}_{1-1}$ is first normalized by the normalization layer. Next, we linearly project the normalized sequence to the $\mathbf{x}$ and $\mathbf{z}$ with dimension size E. Then, we process the $\mathbf{x}$ from the forward and backward directions. For each direction, we first apply the 1-D convolution to the $\mathbf{x}$ and get the $\mathbf{x}'_o$. We then linearly project the $\mathbf{x}'_o$ to the $\mathbf{B}_o$, $\mathbf{C}_o$, $\mathbf{\Delta}_o$, respectively. The $\mathbf{\Delta}_o$ is then used to transform the $\overline{\mathbf{A}}_o$, $\overline{\mathbf{B}}_o$, respectively.

SiMBA uses Mamba block for sequence modeling, L denotes the number of layers, and **Norm** stands for the normalization layer and EinFFT for channel modeling. The process involves applying the Mamba block to the previous time step $\mathbf{X}_{1-1}$, followed by a residual connection and dropout (**DP**). The resulting tensor is then normalized (**Norm**) to obtain the final sequence vector. The subsequent operation involves applying EinFFT, which is the proposed frequency-domain channel mixing operation. Finally, the tensor undergoes another dropout and is added to the previous state $\mathbf{X}_l$.

SiMBA illustrates an important trade-off between performance and scalability. Mamba by itself may have stability issues for large networks. Mamba combined with MLP for channel mixing bridges the performance gap for small-scale networks, but may have the same stability issues for large networks. Mamba combined with EinFFT solves stability issues for both small-scale and large networks. There may still be a performance gap with state-of-the-art transformers for large networks, which we shall address in future work.

4 Experiment

We conducted a comprehensive evaluation of SiMBA on key computer vision tasks, including image recognition, and instance segmentation as well as on other data modalities such as time series. Our assessments for SiMBA model on standard datasets involved the following: a) Training and evaluating ImageNet1K [5] from scratch for the image recognition task. b) Ablation analysis comparing SiMBA with various state space architectures for sequence modeling and various architectures for channel modeling. c) Evaluation of SiMBA on other data modalities such as time series datasets - namely the multi-variate time series benchmark [66]. d) Transfer learning on CIFAR-10 [24], CIFAR-100 [24], Stanford Cars [23], and Flowers-102 [42] for image recognition. e) Fine-tuning SiMBA for downstream instance segmentation tasks. All experiments were conducted on a hierarchical transformer architecture, currently a state-of-the-art model, with an image size of $224 \times 224 \times 3$.

4.1 SOTA Comparison on ImageNet 1K

We conducted performance evaluations on the ImageNet 1K dataset, comprising 1.2 million training images and 50,000 validation images distributed across 1000 categories. Our results, presented in Table 2, categorize architectures based on their size: small (<5 GFlops), base (5-10 GFlops), and large (>10 GFlops). In the small category, SiMBA(MLP) demonstrates remarkable performance with an 84.0% top-1 accuracy, surpassing prominent convolutional networks like ResNet-101 and ResNet-152, as well as leading transformers such as EffNet, ViT, Swin, and DeIT. Our comparison extends to state space models (SSMs) like S4ND, HyenaViT [48], VMambaa [32], and Vim [76], where SiMBA not only outperforms them significantly but also maintains comparable complexity. Specifically, SiMBA-S(EinFFT) and SiMBA-S(MLP) stand out among other small models, including SSMs, small convolutional networks, and transformers, achieving accuracies of 82.7% and 84.0%, respectively. Furthermore, it surpasses competitor transformers like Wave-ViT-S, Max-ViT-T, and iFormer-S with lower GFlops and parameters while being on par with SpectFormer. Notably, SiMBA-S outperforms SSMs like Vim-T (76.1%) and V-Mamba-T (82.2%). We compare among our variants such as SiMBA(EinFFT), SiMBA(Monarch), and SiMBA(MLP), where Mamba is used for sequence modeling, and EinFFT, Monarch Mixing, and standard MLP are used for channel modeling, respectively. Although there is a slight difference in performance between SiMBA with EinFFT as a channel mixer compared to using MLP as a channel mixer, the MLP version demonstrates superior performance in terms of top-1 accuracy. However, it is noted

Method	Image Size	#Param.	FLOPs	Top-1 acc.
Convnets				
ResNet-101	224^2	45M	-	77.4
RegNetY-8G [49]	224^2	39M	8.0G	81.7
ResNet-152	224^2	60M	-	78.3
RegNetY-16G [49]	224^2	84M	16.0G	82.9
Transformers				
DeiT-S [57]	224^2	22M	4.6G	79.8
Swin-T [33]	224^2	29M	4.5G	81.3
EffNet-B4 [27]	380^2	19M	4.2G	82.9
WaveViT-H-S⋆ [69]	224^2	22.7M	4.1G	82.9
SVT-H-S⋆ [46]	224^2	22M	3.9G	84.2
EffNet-B5 [27]	456^2	30M	9.9G	83.6
Swin-S [33]	224^2	50M	8.7G	83.0
CMT-B [17]	224^2	45M	9.3G	84.5
MaxViT-S [58]	224^2	69M	11.7G	84.5
iFormer-B [51]	224^2	48M	9.4G	84.6
Wave-ViT-B⋆ [69]	224^2	33M	7.2G	84.8
SVT-H-B⋆ [46]	224^2	32.8M	6.3G	85.2
ViT-b + Monarch [8]	224^2	33M	-	78.9
M2-ViT-b [8]	224^2	45M	-	79.5
DeiT-B [57]	224^2	86M	17.5G	81.8
Swin-B [33]	224^2	88M	15.4G	83.5
M2-Swin-B [8]	224^2	50M	-	83.5
EffNet-B6 [27]	528^2	43M	19.0G	84.0
MaxViT-B [58]	224^2	120M	23.4G	85.0
VOLO-D2⋆ [72]	224^2	58M	14.1G	85.2
VOLO-D3⋆ [72]	224^2	86M	20.6G	85.4
Wave-ViT-L⋆ [69]	224^2	57M	14.8G	85.5
SVT-H-L⋆ [46]	224^2	54.0M	12.7G	85.7
SSMs				
Vim-Ti [76]	224^2	7M	-	76.1
VMamba-T [32]	224^2	22M	5.6G	82.2
SiMBA-S(Monarch) (Ours)	224^2	18.5M	3.6G	81.1
SiMBA-S(EinFFT) (Ours)	224^2	15.3M	2.4G	81.7
SiMBA-S(MLP) (Ours)	224^2	26.5M	5.0G	84.0
Vim-S [76]	224^2	26M	-	80.5
VMamba-S [32]	224^2	44M	11.2G	83.5
SiMBA-B(Monarch) (Ours)	224^2	26.9M	5.5G	82.6
SiMBA-B(EinFFT) (Ours)	224^2	22.8M	4.2G	83.5
SiMBA-B(MLP) (Ours)	224^2	40.0M	9.0G	84.7
HyenaViT-B [8]	224^2	88M	-	78.5
S4ND-ViT-B [39]	224^2	89M	-	80.4
VMamba-B [32]	224^2	75M	18.0G	83.2
SiMBA-L(Monarch) (Ours)	224^2	42M	8.7G	83.8
SiMBA-L(EinFFT) (Ours)	224^2	36.6M	7.6G	84.4
SiMBA-L(MLP)[†] (Ours)	224^2	66.6M	16.3G	49.4

Table 2: SOTA on ImageNet-1K The table shows the performance of various vision backbones on the ImageNet1K [5] dataset for image recognition tasks. ⋆ indicates additionally trained with the Token Labeling [59] for patch encoding. We have grouped the vision models into three categories based on their GFLOPs (Small, Base, and Large). The GFLOP ranges: Small (GFLOPs<5), Base (5≤GFLOPs<10), and Large (10≤GFLOPs<30). [†] indicates that instability in the training SSM.

that the MLP version is less efficient in terms of parameters and FLOPS. The SiMBA (MLP) variant outperforms other SSM models in terms of top-1 accuracy, parameters, and FLOPS.

Models		SiMBA		TimesNet		Crossformer		PatchTST		ETSFormer		DLinear		FEDFormer		Autoformer		Pyraformer		MTGNN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.324**	**0.360**	0.338	0.375	0.349	0.395	0.339	0.377	0.375	0.398	0.345	0.372	0.379	0.419	0.505	0.475	0.543	0.510	0.379	0.446
	192	**0.363**	**0.382**	0.374	0.387	0.405	0.411	0.376	0.392	0.408	0.410	0.380	0.389	0.426	0.441	0.553	0.496	0.557	0.537	0.470	0.428
	336	**0.395**	**0.405**	0.410	0.411	0.432	0.431	0.408	0.417	0.435	0.428	0.413	0.413	0.445	0.459	0.621	0.537	0.754	0.655	0.473	0.430
	720	**0.451**	**0.437**	0.478	0.450	0.487	0.463	0.499	0.461	0.499	0.462	0.474	0.453	0.543	0.490	0.671	0.561	0.908	0.724	0.553	0.479
ETTm2	96	**0.177**	**0.263**	0.187	0.267	0.208	0.292	0.192	0.273	0.189	0.280	0.193	0.292	0.203	0.287	0.255	0.339	0.435	0.507	0.203	0.299
	192	**0.245**	**0.306**	0.249	0.309	0.263	0.332	0.252	0.314	0.253	0.319	0.284	0.362	0.269	0.328	0.281	0.340	0.730	0.673	0.265	0.328
	336	**0.304**	**0.343**	0.321	0.351	0.337	0.369	0.318	0.357	0.314	0.357	0.369	0.427	0.325	0.366	0.339	0.372	1.201	0.845	0.365	0.374
	720	**0.400**	**0.399**	0.408	0.403	0.429	0.430	0.413	0.416	0.414	0.413	0.554	0.522	0.421	0.415	0.433	0.432	3.625	1.451	0.461	0.459
ETTh1	96	**0.379**	**0.395**	0.384	0.402	0.384	0.428	0.385	0.408	0.494	0.479	0.386	0.400	0.376	0.419	0.449	0.459	0.664	0.612	0.515	0.517
	192	0.432	**0.424**	0.436	0.429	0.438	0.452	**0.431**	0.432	0.538	0.504	0.437	0.432	0.420	0.448	0.500	0.482	0.790	0.681	0.553	0.522
	336	**0.473**	**0.443**	0.491	0.469	0.495	0.483	0.485	0.462	0.574	0.521	0.481	0.459	0.459	0.465	0.521	0.496	0.891	0.738	0.612	0.577
	720	**0.483**	**0.469**	0.521	0.500	0.522	0.501	0.497	0.483	0.562	0.535	0.519	0.516	0.506	0.507	0.514	0.512	0.963	0.782	0.609	0.597
ETTh2	96	**0.290**	**0.339**	0.340	0.374	0.347	0.391	0.343	0.376	0.340	0.391	0.333	0.387	0.358	0.397	0.346	0.388	0.645	0.597	0.354	0.454
	192	**0.373**	**0.390**	0.402	0.414	0.419	0.427	0.405	0.417	0.430	0.439	0.477	0.476	0.429	0.439	0.456	0.452	0.788	0.683	0.457	0.464
	336	**0.376**	**0.406**	0.452	0.452	0.449	0.465	0.448	0.453	0.485	0.479	0.594	0.541	0.496	0.487	0.482	0.486	0.907	0.747	0.515	0.540
	720	**0.407**	**0.431**	0.462	0.468	0.479	0.505	0.464	0.483	0.500	0.497	0.831	0.657	0.463	0.474	0.515	0.511	0.963	0.783	0.532	0.576
Electricity	96	0.165	**0.253**	0.168	0.272	0.185	0.288	**0.159**	0.268	0.187	0.304	0.197	0.282	0.193	0.308	0.201	0.317	0.386	0.449	0.217	0.318
	192	**0.173**	**0.262**	0.184	0.289	0.201	0.295	0.177	0.278	0.199	0.315	0.196	0.285	0.201	0.315	0.222	0.334	0.378	0.443	0.238	0.352
	336	**0.188**	**0.277**	0.198	0.300	0.211	0.312	0.195	0.296	0.212	0.329	0.209	0.301	0.214	0.329	0.231	0.338	0.376	0.443	0.260	0.348
	720	**0.214**	**0.305**	0.220	0.320	0.223	0.335	0.215	0.317	0.233	0.345	0.245	0.333	0.246	0.355	0.254	0.361	0.376	0.445	0.290	0.369
Traffic	96	**0.468**	**0.268**	0.593	0.321	0.591	0.329	0.583	0.319	0.607	0.392	0.650	0.396	0.587	0.366	0.613	0.388	0.867	0.468	0.660	0.437
	192	**0.413**	**0.317**	0.617	0.336	0.607	0.345	0.591	0.331	0.621	0.399	0.598	0.370	0.604	0.373	0.616	0.382	0.869	0.467	0.649	0.438
	336	**0.529**	**0.284**	0.629	0.336	0.613	0.339	0.599	0.332	0.622	0.396	0.605	0.373	0.621	0.383	0.622	0.337	0.881	0.469	0.653	0.472
	720	**0.564**	**0.297**	0.640	0.350	0.620	0.348	0.601	0.341	0.632	0.396	0.645	0.394	0.626	0.382	0.660	0.408	0.896	0.473	0.639	0.437
Weather	96	0.176	**0.219**	0.172	0.220	0.191	0.251	**0.171**	0.230	0.197	0.281	0.196	0.255	0.217	0.296	0.266	0.336	0.622	0.556	0.230	0.329
	192	0.222	**0.260**	**0.219**	0.261	0.219	0.279	0.219	0.271	0.237	0.312	0.237	0.296	0.276	0.336	0.307	0.367	0.739	0.624	0.263	0.322
	336	**0.275**	**0.297**	0.280	0.306	0.287	0.332	0.277	0.321	0.298	0.353	0.283	0.335	0.339	0.380	0.359	0.395	1.004	0.753	0.354	0.396
	720	**0.350**	**0.349**	0.365	0.359	0.368	0.378	0.365	0.367	0.352	0.288	0.345	0.381	0.403	0.428	0.419	0.428	1.420	0.934	0.409	0.371

Table 3: Multivariate long-term forecasting results with supervised SiMBA. We use prediction lengths $T \in \{96, 192, 336, 720\}$ for all the datasets for lookup window 96. The best results are in **bold** and the second best is underlined.

Moving to the base size, SiMBA-B demonstrates superior performance compared to other SSMs such as Hyena-ViT-B (78.5%), Vim-S (80.5%), VMamba-S (83.5%), and S4ND-ViT-B (80.4%), while maintaining similar parameters and GFlops. SiMBA-B achieves an impressive accuracy of 84.7%, which is on par with transformer-based models like Wave-ViT-B (84.8%), iFormer-B (84.6%), and Max-ViT-S (84.5%), but with notably fewer parameters and GFlops. Notably, the SiMBA (MLP) variant outperforms other SSM models and the current state-of-the-art model in terms of top-1 accuracy, while also reducing the number of parameters and FLOPS, thus marking a significant advancement in state-of-the-art performance.

In the large category, SiMBA-L achieves an 83.9% top-1 accuracy, surpassing VMamba-B (83.2%) with comparable computational complexity and parameters. However, a performance gap remains compared to state-of-the-art transformers such as SVT-H-L (85.7%). Future work aims to address this gap by further scaling SiMBA. However, SiMBA-L (EinFFT) outperforms other SSMs in this range. Notably, the SiMBA(MLP) variant exhibits instability issues in large network sizes, prompting us to solely report results from the SiMBA(EinFFT) variant for large network sizes. SiMBA(EinFFT) mitigates stability concerns at large network sizes and demonstrates greater efficiency in terms of FLOPS and parameters, outperforming all other SSMs, while with certain trade-offs.

4.2 SOTA for Multi-Variate Time Series Forecasting

We conducted a comprehensive evaluation of our State Space model, SiMBA, on seven benchmark standard datasets widely used for Multivariate Time Series Forecasting, including Electricity, Weather, Traffic, and four ETT datasets (ETTh1, ETTh2, ETTm1, and ETTm2), as presented in Table 3. Our evaluation compares SiMBA with various state-of-the-art models, including Transformer-based methods like PatchTST [41], CrossFormer [74], FEDFormer [75], ETSFormer [63], PyraFormer [31], and AutoFormer [3]. Additionally, CNN-based methods such as TimeNet [65], graph-based methods like MTGNN [67], and MLP-based models like DLinear [73] are included in the comparison.

SiMBA demonstrates superior performance across multiple evaluation metrics, including Mean Squared Error (MSE) and Mean Absolute Error (MAE), outperforming the state-of-the-art models. These results underscore the versatility and effectiveness of the SiMBA architecture in handling diverse time series forecasting tasks and modalities, solidifying its position as a leading model in the field. While presenting our results, it's important to note that due to space limitations in Table 3, we couldn't include some recent methods in the Time series domain, such as FourierGNN [70], CrossGNN [21], TiDE [4], SciNet [30], and FreTS [71]. For a fair comparison, we utilized the code from PatchTST [41], and the results were based on a lookup window of size 96 for all datasets.

Model	Param	Top-1%	Top-5%	Description(Seq-mix, Channel-mix)
ResNet-152	60M	78.6	94.3	ConvNet, MLP
ViT-b	87M	78.5	93.6	Attention, MLP
s4 only†	13.2M	58.9	82.5	S4, NA
Hippo only†	16.4M	63.2	89.2	Hippo, NA
Mamba only†	15.3M	39.1	67.1	Mamba, NA
s4+MLP	23.5M	75.9	93.6	S4, MLP
Hippo+MLP	24.8M	77.9	94.0	Hippo, MLP
SiMBA(Mamba +Monach)	18.5M	81.1	95.5	Mamba, Monach
SiMBA(Mamba +EinFFT)	15.3M	81.7	95.9	Mamba, EinFFT
SiMBA(Mamba +MLP)	26.6M	84.0	96.7	Mamba, MLP

Table 4: Ablation Analysis on ImageNet-1k for small size model. † indicates that instability is encountered during the training of the SSMs

4.3 Ablation Analysis of Model SiMBA

In our ablation analysis on the ImageNet-1k dataset for small-sized models (Table 4), we explored various sequence modeling configurations, focusing on SiMBA and comparing it with other state space models (SSMs) like S4 and Hyena. Notably, Mamba alone faced stability issues and was non-trainable for the ImageNet dataset for large-scale networks, prompting us to integrate it with different channel mixing components. The results highlight that both S4 and Hyena when utilized as standalone models without channel mixing, exhibit a performance gap compared to state-of-the-art vision transformers like SVT, WaveViT, Volo, and MaxViT. To address this, our SiMBA architecture combines Mamba as the sequence modeling component with different channel mixing strategies. In the ablation study for channel mixing, we explored three components: MLP, Monarch Mixing (M2), and our EinFFT. The findings in Table 4 indicate

Mask R-CNN 1× schedule								
Backbone	AP^b	AP$^b_{50}$	AP$^b_{75}$	AP^m	AP$^m_{50}$	AP$^m_{75}$	#param.	FLOPs
ResNet-101	38.2	58.8	41.4	34.7	55.7	37.2	63M	336G
Swin-S	44.8	66.6	48.9	40.9	63.2	44.2	69M	354G
ConvNeXt-S	45.4	67.9	50.0	41.8	65.2	45.1	70M	348G
PVTv2-B3	47.0	68.1	51.7	42.5	65.7	45.7	65M	397G
SiMBA-S	46.9	**68.6**	**51.7**	**42.6**	**65.9**	**45.8**	60M	382G

Table 5: Object detection and instance segmentation results on COCO dataset. The performances of various vision models on the COCO val2017 dataset for the downstream tasks of object detection and instance segmentation. RetinaNet is used as the object detector for the object detection task, and the Average Precision (AP) at different IoU thresholds or two different object sizes (i.e., small and base) are reported for evaluation. For instance segmentation task, we adopt Mask R-CNN as the base model, and the bounding box and mask Average Precision (i.e., AP^b and AP^m) are reported for evaluation. "1×" indicates models fine-tuned for 12 epochs.

that EinFFT-based SiMBA is a channel modeling component which is an alternative method to solve the stability issue, showcasing superior performance with other SSMs when coupled with Mamba as the sequence modeling component. This analysis underscores the significance of channel mixing strategies in enhancing the effectiveness of SiMBA on ImageNet-1k, providing valuable insights for optimizing small-sized models in sequence modeling tasks.

Table 6: Results on transfer learning datasets. We report the top-1 accuracy on the four datasets as well as the number of parameters and FLOPs.

Model	CIFAR-10	CIFAR-100	Flowers-102	Cars-196
ResNet50 [20])	-	-	96.2	90.0
EfficientNet-B7 [53]	98.9	91.7	98.8	92.7
ViT-B/16 [7]	98.1	87.1	89.5	-
ViT-L/16 [7]	97.9	86.4	89.7	-
Deit-B/16 [57]	**99.1**	**90.8**	98.4	92.1
ResMLP-24 [56]	98.7	89.5	97.9	89.5
GFNet-XS ([50])	98.6	89.1	98.1	92.8
GFNet-H-B ([50])	99.0	90.3	98.8	93.2
SiMBA-B	98.7	89.3	98.4	92.7

4.4 Task Learning: Object Detection

In our experiments on the MS COCO 2017 dataset, a widely used benchmark for object detection and instance segmentation, consisting of approximately 118,000 training images and 5,000 validation images, we employed two widely-used detection frameworks: RetinaNet [29] and Mask R-CNN [19]. Model performance was assessed using Average Precision (AP). SiMBA-s, pre-trained on the ImageNet-1K dataset, served as the backbone architecture, and Xavier initialization was applied to additional layers. The results, detailed in Table 5, showcase SiMBA's competitive performance in comparison to RetinaNet [29] and Mask R-CNN [19]. Notably, SiMBA surpasses the latest models, including LITv2 [44], RegionViT, and PVT [60] transformer models, outperforming ResNet in

Method	size	mIoU (SS)	mIoU (MS)	#Param.	FLOPs
ResNet-101	512^2	42.9	44.0	85M	1030G
DeiT-B + MLN	512^2	45.5	47.2	144M	2007G
Swin-S	512^2	47.6	49.5	81M	1039G
ConvNeXt-S	512^2	48.7	49.6	82M	1027G
SiMBA-S	512^2	49.0	49.6	62M	1040G

Table 7: Semantic segmentation results on ADE20K using UperNet [68]. We evaluate the performance of semantic segmentation on the ADE20K dataset with UperNet [68]. The FLOPs are calculated with input sizes of 512×2048 based on the crop size. "SS" and "MS" denote single-scale and multi-scale testing, respectively.

terms of AP. SiMBA exhibits superior performance over vanilla ViT models and hierarchical transformer models, achieving the highest AP. Furthermore, we extend SiMBA's capabilities to semantic segmentation on the ADE20K dataset using UperNet [68], with results summarized in Table 7. The outcomes underscore SiMBA's versatility and effectiveness across diverse computer vision tasks.

4.5 Transfer Learning Comparison

To evaluate the efficacy of the SiMBA architecture and learned representations, we conducted assessments on various transfer learning benchmark datasets, including CIFAR-10 [24], CIFAR-100 [24], Stanford Cars [23], and Flowers-102 [42]. The performance of SiMBA, pre-trained on ImageNet-1K and fine-tuned for image classification on these datasets, was compared against ResMLP models and state-of-the-art models such as GFNet [50]. Table 6 highlights the competitive performance of SiMBA on downstream datasets, outperforming ResMLP models and demonstrating comparable results to GFNet [50]. Additionally, SiMBA's performance was competitive against popular models like ViT-b, HyenaViT-b, and ViT-b-Monarch on ImageNet-1k. Notably, SiMBA achieved superior results compared to ResNet-152, despite having fewer parameters, emphasizing its effectiveness in transfer learning scenarios.

5 Conclusion

This paper has proposed a new channel modeling technique EinFFT which solves the key stability problem in Mamba. We have also proposed SiMBA, a new architecture that leverages EinFFT for channel modeling and Mamba for sequence modeling. SiMBA allows for the exploration of various alternatives for sequence modeling like S4, long conv, Hyena, H3, RWKV, and even newer state space models. Importantly, SiMBA allows the exploration of alternatives for channel modeling like M2, and EinFFT as well as other possible spectral techniques. SiMBA also bridges the performance gap that most state space models have with state-of-art transformers on both vision and time series datasets. We plan to explore a few alternatives within the SiMBA framework such as long conv for sequence modeling with M2 or EinFFT for channel modeling.

References

1. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6154–6162 (2018) 20, 23, 24
2. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., et al.: Mmdetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019) 23, 24
3. Chen, M., Peng, H., Fu, J., Ling, H.: Autoformer: Searching transformers for visual recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 12270–12280 (2021) 13
4. Das, A., Kong, W., Leach, A., Mathur, S.K., Sen, R., Yu, R.: Long-term forecasting with tide: Time-series dense encoder. Transactions on Machine Learning Research (2023) 13
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 10, 11
6. Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., Guo, B.: Cswin transformer: A general vision transformer backbone with cross-shaped windows. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12124–12134 (2022) 4
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2020) 3, 4, 14, 23
8. Fu, D., Arora, S., Grogan, J., Johnson, I., Eyuboglu, E.S., Thomas, A., Spector, B., Poli, M., Rudra, A., Ré, C.: Monarch mixer: A simple sub-quadratic gemm-based architecture. Advances in Neural Information Processing Systems **36** (2024) 11
9. Fu, D.Y., Dao, T., Saab, K.K., Thomas, A.W., Rudra, A., Ré, C.: Hungry hungry hippos: Towards language modeling with state space models. arXiv preprint arXiv:2212.14052 (2022) 2, 4
10. Fu, D.Y., Epstein, E.L., Nguyen, E., Thomas, A.W., Zhang, M., Dao, T., Rudra, A., Re, C.: Simple hardware-efficient long convolutions for sequence modeling. In: ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models (2023) 4
11. Fu, D.Y., Kumbong, H., Nguyen, E., Ré, C.: Flashfftconv: Efficient convolutions for long sequences with tensor cores. arXiv preprint arXiv:2311.05908 (2023) 2
12. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256. JMLR Workshop and Conference Proceedings (2010) 24
13. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023) 2, 3, 8, 9
14. Gu, A., Dao, T., Ermon, S., Rudra, A., Ré, C.: Hippo: Recurrent memory with optimal polynomial projections. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1474–1487. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper_files/paper/2020/file/102f0bb6efb3a6128a3c750dd16729be-Paper.pdf 2, 3
15. Gu, A., Goel, K., Re, C.: Efficiently modeling long sequences with structured state spaces. In: International Conference on Learning Representations (2021) 2, 3, 8
16. Guibas, J., Mardani, M., Li, Z., Tao, A., Anandkumar, A., Catanzaro, B.: Efficient token mixing for transformers via adaptive fourier neural operators. In: International Conference on Learning Representations (2022) 3, 4

17. Guo, J., Han, K., Wu, H., Tang, Y., Chen, X., Wang, Y., Xu, C.: Cmt: Convolutional neural networks meet vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12175–12185 (2022) 11
18. Hasani, R., Lechner, M., Amini, A., Rus, D., Grosu, R.: Liquid time-constant networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 7657–7666 (2021) 2
19. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017) 14, 24
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 14, 23
21. Huang, Q., Shen, L., Zhang, R., Ding, S., Wang, B., Zhou, Z., Wang, Y.: Crossgnn: Confronting noisy multivariate time series via cross interaction refinement. In: Thirty-seventh Conference on Neural Information Processing Systems (2023) 13
22. Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M.A., Stock, P., Scao, T.L., Lavril, T., Wang, T., Lacroix, T., Sayed, W.E.: Mistral 7b (2023) 1
23. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: Proceedings of the IEEE international conference on computer vision workshops. pp. 554–561 (2013) 10, 15, 23, 24
24. Krizhevsky, A., et al.: Learning multiple layers of features from tiny images (2009) 10, 15, 23, 24
25. Lee-Thorp, J., Ainslie, J., Eckstein, I., Ontanon, S.: Fnet: Mixing tokens with fourier transforms. arXiv preprint arXiv:2105.03824 (2021) 4
26. Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., Yang, J.: Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. Advances in Neural Information Processing Systems **33**, 21002–21012 (2020) 20, 23, 24
27. Li, Y., Yuan, G., Wen, Y., Hu, E., Evangelidis, G., Tulyakov, S., Wang, Y., Ren, J.: Efficientformer: Vision transformers at mobilenet speed. arXiv preprint arXiv:2206.01191 (2022) 11
28. Li, Y., Bubeck, S., Eldan, R., Giorno, A.D., Gunasekar, S., Lee, Y.T.: Textbooks are all you need ii: phi-1.5 technical report (2023) 1
29. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017) 14, 24
30. Liu, M., Zeng, A., Chen, M., Xu, Z., Lai, Q., Ma, L., Xu, Q.: Scinet: Time series modeling and forecasting with sample convolution and interaction. Advances in Neural Information Processing Systems **35**, 5816–5828 (2022) 13
31. Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A.X., Dustdar, S.: Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In: International conference on learning representations (2021) 13
32. Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., Ye, Q., Liu, Y.: Vmamba: Visual state space model. arXiv preprint arXiv:2401.10166 (2024) 4, 10, 11
33. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al.: Swin transformer v2: Scaling up capacity and resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12009–12019 (2022) 3, 11
34. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10012–10022 (2021) 4, 23, 24
35. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2018) 24

36. Ma, X., Zhou, C., Kong, X., He, J., Gui, L., Neubig, G., May, J., Zettlemoyer, L.: Mega: Moving average equipped gated attention. In: The Eleventh International Conference on Learning Representations (2022) 2
37. Mehta, H., Gupta, A., Cutkosky, A., Neyshabur, B.: Long range language modeling via gated state spaces. In: The Eleventh International Conference on Learning Representations (2022) 2
38. Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., Awadallah, A.: Orca: Progressive learning from complex explanation traces of gpt-4 (2023) 1
39. Nguyen, E., Goel, K., Gu, A., Downs, G., Shah, P., Dao, T., Baccus, S., Ré, C.: S4nd: Modeling images and videos as multidimensional signals with state spaces. Advances in neural information processing systems **35**, 2846–2861 (2022) 2, 11
40. Nguyen, T.M., Pham, M., Nguyen, T.M., Nguyen, K., Osher, S., Ho, N.: Fourierformer: Transformer meets generalized fourier integral theorem. In: Advances in Neural Information Processing Systems (2022) 4
41. Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J.: A time series is worth 64 words: Long-term forecasting with transformers. In: The Eleventh International Conference on Learning Representations (2022) 13
42. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing. pp. 722–729. IEEE (2008) 10, 15, 23, 24
43. Oppenheim, A.V., Verghese, G.: Signals, Systems and Inference. Pearson, USA (2010) 2, 3
44. Pan, Z., Cai, J., Zhuang, B.: Fast vision transformers with hilo attention. In: Advances in Neural Information Processing Systems (2022) 14
45. Patro, B.N., Namboodiri, V.P., Agneeswaran, V.S.: Spectformer: Frequency and attention is what you need in a vision transformer. arXiv preprint arXiv:2304.06446 (2023) 4, 5
46. Patro, B.N., Agneeswaran, V.S.: Scattering vision transformer: Spectral mixing matters. In: Thirty-seventh Conference on Neural Information Processing Systems (2023) 4, 5, 11
47. Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Cao, H., Cheng, X., Chung, M., Grella, M., GV, K.K., et al.: Rwkv: Reinventing rnns for the transformer era. arXiv preprint arXiv:2305.13048 (2023) 2
48. Poli, M., Massaroli, S., Nguyen, E., Fu, D.Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., Re, C.: Hyena hierarchy: Towards larger convolutional language models (2023) 2, 3, 4, 10
49. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10428–10436 (2020) 11
50. Rao, Y., Zhao, W., Zhu, Z., Lu, J., Zhou, J.: Global filter networks for image classification. Advances in Neural Information Processing Systems **34**, 980–993 (2021) 3, 4, 14, 15, 23
51. Si, C., Yu, W., Zhou, P., Zhou, Y., Wang, X., YAN, S.: Inception transformer. In: Advances in Neural Information Processing Systems (2022) 11
52. Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., Wei, F.: Retentive network: A successor to transformer for large language models. ICLR (2023) 2
53. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019) 14, 23
54. Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., Metzler, D.: International conference on learning representations (iclr) (2021) 4
55. Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al.: Mlp-mixer: An all-mlp architecture for vision. Advances in neural information processing systems **34**, 24261–24272 (2021) 4
56. Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Izacard, G., Joulin, A., Synnaeve, G., Verbeek, J., et al.: Resmlp: Feedforward networks for image clas-

sification with data-efficient training. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022) 14, 23
57. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. pp. 10347–10357. PMLR (2021) 3, 4, 11, 14, 23
58. Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., Li, Y.: Maxvit: Multi-axis vision transformer. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV. pp. 459–479. Springer (2022) 11
59. Wang, P., Wang, X., Luo, H., Zhou, J., Zhou, Z., Wang, F., Li, H., Jin, R.: Scaled relu matters for training vision transformers. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 2495–2503 (2022) 11
60. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 568–578 (2021) 14
61. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pvt v2: Improved baselines with pyramid vision transformer. Computational Visual Media **8**(3), 415–424 (2022) 23
62. Wang, Z., Jiang, W., Zhu, Y.M., Yuan, L., Song, Y., Liu, W.: Dynamixer: a vision mlp architecture with dynamic mixing. In: International Conference on Machine Learning. pp. 22691–22701. PMLR (2022) 4
63. Woo, G., Liu, C., Sahoo, D., Kumar, A., Hoi, S.: Etsformer: Exponential smoothing transformers for time-series forecasting. ICLR (2022) 13
64. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., Zhang, L.: Cvt: Introducing convolutions to vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 22–31 (2021) 4
65. Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., Long, M.: Timesnet: Temporal 2d-variation modeling for general time series analysis. In: The Eleventh International Conference on Learning Representations (2022) 13
66. Wu, H., Xu, J., Wang, J., Long, M.: Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In: Advances in Neural Information Processing Systems (2021) 10
67. Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C.: Connecting the dots: Multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 753–763 (2020) 13
68. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: Proceedings of the European conference on computer vision (ECCV). pp. 418–434 (2018) 15
69. Yao, T., Pan, Y., Li, Y., Ngo, C.W., Mei, T.: Wave-vit: Unifying wavelet and transformers for visual representation learning. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXV. pp. 328–345. Springer (2022) 4, 5, 11
70. Yi, K., Zhang, Q., Fan, W., He, H., Hu, L., Wang, P., An, N., Cao, L., Niu, Z.: Fouriergnn: Rethinking multivariate time series forecasting from a pure graph perspective. In: Thirty-seventh Conference on Neural Information Processing Systems (2023) 13
71. Yi, K., Zhang, Q., Fan, W., Wang, S., Wang, P., He, H., An, N., Lian, D., Cao, L., Niu, Z.: Frequency-domain mlps are more effective learners in time series forecasting. Advances in Neural Information Processing Systems **36** (2024) 13
72. Yuan, L., Hou, Q., Jiang, Z., Feng, J., Yan, S.: Volo: Vision outlooker for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022) 5, 11

73. Zeng, A., Chen, M., Zhang, L., Xu, Q.: Are transformers effective for time series forecasting? In: Proceedings of the AAAI conference on artificial intelligence. vol. 37, pp. 11121–11128 (2023) 13
74. Zhang, Y., Yan, J.: Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: The Eleventh International Conference on Learning Representations (2022) 13
75. Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., Jin, R.: Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In: International Conference on Machine Learning. pp. 27268–27286. PMLR (2022) 13
76. Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., Wang, X.: Vision mamba: Efficient visual representation learning with bidirectional state space model. arXiv preprint arXiv:2401.09417 (2024) 4, 10, 11, 22

A Appendix

This section includes comprehensive details regarding training configurations for both transfer learning and task learning tasks. Specifically, Table-10 provides a breakdown of the dataset information utilized for transformer learning, offering insights into the datasets employed in the training process. For a thorough understanding of the SiMBA algorithm, Algorithm-32 is presented, providing a complete architectural view of the SiMBA model. This algorithm outlines the key operations and processes involved in the SiMBA architecture, facilitating a deeper understanding of its inner workings. Furthermore, detailed implementation insights into the EinFFT channel mixing method are provided, shedding light on the mechanism utilized for frequency-domain channel mixing within the SiMBA model. This implementation detail offers clarity on how the frequency-domain information is leveraged to enhance the model's performance in capturing intricate relationships within sequential data. This comparison aids in understanding the different approaches employed by various vision models in handling sequential data and exploiting inter-channel relationships. Moreover, the document reports the object detection performance of two prominent models, GFL [26] and Cascade Mask R-CNN [1], on the MS COCO val2017 dataset. The performance metrics presented in Table- 9 demonstrate a notable improvement in performance, showcasing the effectiveness of the proposed SiMBA model.

B EinFFT Implementation

We proposed EinFFT module introduces a novel approach to model channels more efficiently, with complex-valued weights and advanced signal processing techniques. The architecture leverages Fast Fourier Transform (FFT) operations, which transform the physical domain to the Frequency Domain. The module's distinctive formulation employs complex-valued weights initialized with a scale factor, promoting enhanced representation learning. The incorporation of soft thresholding introduces sparsity constraints, facilitating noise reduction and feature selection. The complex-valued weights contribute to a more expressive model, enriching the representational space. The modular and extensible implementation of these operations in the code presents a valuable tool

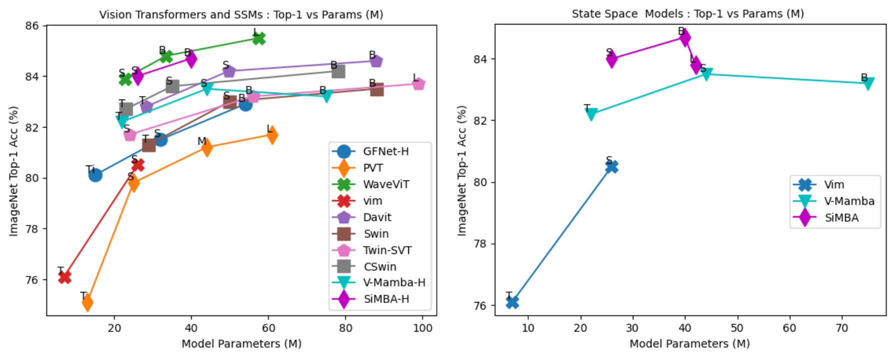

Fig. 2: Comparison of ImageNet Top-1 Accuracy (%) vs Params(M) of Transformer architectures and State space architectures. SiMBA is a better state space model compared to other state space models for Vision data.

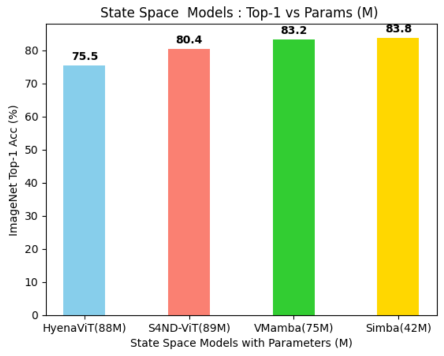

Fig. 3: Comparison of ImageNet Top-1 Accuracy (%) vs Params(M) of State space architectures. SiMBA is a better state space model compared to other state space models for Vision data.

for researchers and practitioners in sequence modeling, offering a unique perspective on capturing intricate dependencies within sequential data.

Mathematically, the core computations of EinFFT can be represented as follows. Let x denote the input tensor of shape $B \times N \times C$, where B is the batch size, N is the sequence length, and C is the number of channels. The FFT operation is denoted as $\text{FFT}(x, \text{dim} = 1, \text{norm} =' ortho')$. Complex-valued weights W_1, W_2, B_1, B_2 are utilized in a series of operations involving complex multiplication, rectified linear units (ReLU), and soft thresholding:

$$\begin{aligned}
X &= \text{FFT}(x) \\
X_{\text{real}_1} &= \max(\text{Re}(X) \cdot W_{1,\text{real}} - \text{Im}(X) \cdot W_{1,\text{imag}} + B_{1,\text{real}}, 0) \\
X_{\text{imag}_1} &= \max(\text{Re}(X) \cdot W_{1,\text{imag}} + \text{Im}(X) \cdot W_{1,\text{real}} + B_{1,\text{imag}}, 0) \\
X_{\text{real}_2} &= \text{Re}(X_{\text{real}_1} \cdot W_{2,\text{real}} - X_{\text{imag}_1} \cdot W_{2,\text{imag}} + B_{2,\text{real}}) \\
X_{\text{imag}_2} &= \text{Im}(X_{\text{real}_1} \cdot W_{2,\text{imag}} + X_{\text{imag}_1} \cdot W_{2,\text{real}} + B_{2,\text{imag}}) \\
X_{\text{shrink}} &= \text{softshrink}(X_{\text{real}_2} \,\&\, X_{\text{imag}_2}, \lambda = \text{sparsity\_threshold}) \\
x_{\text{ifft}} &= \text{IFFT}(X_{\text{shrink}}, \text{dim} = 1, \text{norm} ='\, ortho') \\
x_{\text{reshaped}} &= \text{reshape}(x_{\text{ifft}}, (B, N, C))
\end{aligned}$$

This sequence of operations encapsulates the essence of the EinFFT module, providing a comprehensive mathematical description of its functionality.

The SiMBA Block Process, depicted in Algorithm 32 inspire from ViM [76], unfolds as follows: Initially, the input token sequence $\mathbf{X}_{1-1}$ undergoes normalization via a dedicated layer. Subsequently, a linear transformation is employed to project the normalized sequence onto $\mathbf{x}$ and $\mathbf{z}$, both of dimension size P. Moving forward, the algorithm proceeds to process $\mathbf{x}$ in both forward and backward directions. Within each direction, a 1-D convolution operation is applied to $\mathbf{x}$ to yield $\mathbf{x}_o$. Subsequently, linear projections are applied to $\mathbf{x}_o$, resulting in $\mathbf{B}_o$, $\mathbf{C}_o$, and $\mathbf{\Delta}_o$. Utilizing $\mathbf{\Delta}_o$, transformations are performed on $\overline{\mathbf{A}}_o$ and $\overline{\mathbf{B}}_o$. Following this, the SSM mechanism computes $\mathbf{y}_{forward}$ and $\mathbf{y}_{backward}$. Gating by $\mathbf{z}$ is applied to $\mathbf{y}_o$, culminating in the generation of the output token sequence $\mathbf{X}_1$. This output sequence is the aggregation of the gated $\mathbf{y}_o$ from both directions. In essence, the SiMBA Block Process encompasses a series of operations, including normalization, linear projections, convolution, and attention mechanisms, ultimately leading to the generation of the final token sequence $\mathbf{X}_1$.

C Experiment

C.1 Dataset and Training setup for Image classification task

We describe the training process of the image recognition task using the ImageNet1K benchmark dataset, which includes 1.28 million training images and 50K validation images belonging to 1,000 categories. The vision backbones are trained from scratch using data augmentation techniques like RandAug, CutOut, and Token Labeling objectives with MixToken. The performance of the trained backbones is evaluated using both top-1 and top-5 accuracies on the validation set. The optimization process involves using the AdamW optimizer with a momentum of 0.9, 10 epochs of linear warm-up, and 310 epochs of cosine decay learning rate scheduler. The batch size is set to 128 and is distributed on 8 A100 GPUs. The learning rate and weight decay are fixed at 0.00001 and 0.05, respectively.

Table 8: SiMBA training settings.

	ImageNet-1k	CIFAR-10
Optimizer	AdamW	
Optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$	
Learning rate schedule	Cosine decay w/ linear warmup	
Dropout rate	0.2	
Label smoothing	0.1	
Image size	224 x 224	32 x 32
Base learning rate	1e-3	{1e-4, 3e-4, 1e-3}
Batch size	128	64
Training epochs	300	up to 1000
Warmup epochs	10	5
Stochastic depth rate	0.1	{0, 0.1}
Weight decay	0.05	{0, 0.1}

Table 9: The performances of various vision backbones on COCO val2017 dataset for the down-stream task of object detection. Four kinds of object detectors, i.e, GFL [26], and Cascade Mask R-CNN [1] in mmdetection [2], are adopted for evaluation. We report the bounding box Average Precision (AP^b) in different IoU thresholds.

Backbone	Method	AP^b	AP^b_{50}	AP^b_{75}
ResNet50 [20]	GFL [26]	44.5	63.0	48.3
Swin-T [34]		47.6	66.8	51.7
PVTv2-B2 [61]		50.2	69.4	54.7
SiMBA-S (Ours)		**50.3**	**69.0**	**55.1**
ResNet50 [20]	Cascade Mask [1] R-CNN	46.3	64.3	50.5
Swin-T [34]		50.5	69.3	54.9
PVTv2-B2 [61]		51.1	69.8	55.3
SiMBA-S (Ours)		**51.4**	**70.1**	**56.0**

C.2 Training setup for Transfer Learning

To test the effectiveness of our architecture and learned representation, we evaluated vanilla SiMBA on commonly used transfer learning benchmark datasets, including CIFAR-10 [24], CIFAR100 [24], Oxford-IIIT-Flower [42] and Standford Cars [23]. Our approach followed the methodology of previous studies [7,50,53,56,57], where we initialized the model with ImageNet pre-trained weights and fine-tuned it on the new datasets. In Table-6 of the main paper, we have presented a comparison of the transfer learning performance of our basic and best models with state-of-the-art CNNs and vision transformers. The transfer learning setup employs a batch size of 64, a learning rate (lr) of 0.0001, a weight-decay of 1e-4, a clip-grad of 1, and warmup epochs of 5. We have utilized a pre-trained model trained on the Imagenet-1K dataset, which we have fine-tuned on the transfer learning dataset specified in table-10 for 1000 epochs.

C.3 Task Learning: Object Detection

Training setup: In this section, we examine the pre-trained SiMBA-H-small behavior on COCO dataset for two downstream tasks that localize objects ranging from bounding-box level to pixel level, i.e., object detection and instance segmentation. Two mainstream detectors, i.e., RetinaNet [29] and Mask R-CNN [19] as shown in table-8 of the main paper, and two state-of-the-art detectors i.e., GFL [26], and Cascade Mask R-CNN [1] in mmdetection [2] in this supplementary doc. We are employed for each downstream task, and we replace the CNN backbones in each detector with our SiMBA-H-small for evaluation. Specifically, each vision backbone is first pre-trained over ImageNet1K, and the newly added layers are initialized with Xavier [12]. Next, we follow the standard setups in [34] to train all models on the COCO train2017 (∼118K images). Here the batch size is set as 16, and AdamW [35] is utilized for optimization (weight decay: 0.05, initial learning rate: 0.0001, betas=(0.9, 0.999)). We used learning rate (lr) configuration with step lr policy, linear warmup at every 500 iterations with warmup ration 0.001. All models are finally evaluated on the COCO val2017 (5K images). For state-of-the-art models like GFL [26], and Cascade Mask R-CNN [1], we utilize 3 × schedule (i.e., 36 epochs) with the multi-scale strategy for training, whereas for RetinaNet [29] and Mask R-CNN [19] we utilize 1 × schedule (i.e., 12 epochs).

Table 10: This table presents information about datasets used for transfer learning. It includes the size of the training and test sets, as well as the number of categories included in each dataset such as CIFAR-10 [24], CIFAR-100 [24], Flowers-102 [42], Stanford Cars [23].

Dataset	CIFAR-10	CIFAR-100	Flowers-102	Stanford Cars
Train Size	50,000	50,000	8,144	2,040
Test Size	10,000	10,000	8,041	6,149
#Categories	10	100	196	102

C.4 SiMBA Architecture Details

The architectural details of the simplified SiMBA-based context can be expressed as follows:

$$\begin{aligned} \mathbf{X}_l &= \mathbf{DP}(\mathbf{Mamba}(\mathbf{X}_{\mathtt{l}-1})) + \mathbf{X}_{\mathtt{l}-1}, \\ \mathbf{X}_l &= \mathbf{DP}(\mathbf{EinFFT}(\mathbf{Norm}(\mathbf{X}_{\mathtt{l}}))) + \mathbf{X}_l, \end{aligned} \tag{11}$$

Here, **SiMBA** represents the proposed Mamba block, $\mathtt{L}$ denotes the number of layers, and **Norm** stands for the normalization layer. The process involves applying the Mamba block to the previous time step $\mathbf{X}_{\mathtt{l}-1}$, followed by a residual connection and dropout (**DP**). The resulting tensor is then normalized (**Norm**) and by applying the EinFFT operation subsequently, which is the proposed frequency-domain channel mixing operation. Finally, the tensor undergoes another dropout and is added to the previous state $\mathbf{X}_l$. This overall structure is iteratively applied for multiple layers ($\mathtt{l}$).

In our proposed SiMBA model, we employ a series of operations to process sequential data effectively. Here's a breakdown of the key components and their interactions:

- Mamba Block (**Mamba**): The Mamba block is the fundamental building block of our model, responsible for capturing temporal dependencies within the input time series. By applying this block to the previous time step $\mathbf{X}_{\mathtt{l-1}}$, we aim to extract relevant features and patterns crucial for forecasting. The residual connection with dropout (**DP**) ensures the retention of essential information from preceding steps, enhancing the model's ability to learn complex temporal dynamics.
- Normalization (**Norm**): Normalizing the tensor $\mathbf{X}_{\mathtt{L}}^{0}$ enhances the stability and convergence of the model during training. This step helps mitigate issues related to gradient vanishing or explosion, promoting smoother learning dynamics.
- Frequency-Domain Channel Mixing (**EinFFT**): The operation **EinFFT** represents a novel approach to exploit frequency-domain information for capturing intricate relationships among different channels within the data. By transforming the normalized tensor into the frequency domain, our model can effectively capture periodic patterns and complex inter-channel dependencies, thereby improving its forecasting capabilities.
- Dropout (**DP**): Dropout is applied after the frequency-domain channel mixing to prevent overfitting and enhance the model's generalization ability. This regularization technique aids in robust feature learning by reducing the model's sensitivity to noise in the data.
- Residual Connection and Iteration: The residual connection adds the result of the dropout operation to the previous state $\mathbf{X}_l$, facilitating iterative refinement of temporal representations across multiple layers ($\mathtt{l}$). This iterative process enables the model to progressively learn hierarchical features and capture increasingly complex temporal dynamics.

As for the hyperparameters governing our architecture, we define:

- $\mathtt{L}$: The number of Mamba blocks, determining the depth of the model. - $\mathtt{D}$: The hidden state dimension, representing the dimensionality of the model's hidden states. - $\mathtt{P}$: The expanded state dimension, determining the dimensionality of the expanded states within the model. - $\mathtt{N}$: The sequence length of the input data sequence.

our SiMBA architecture leverages Mamba blocks, frequency-domain channel mixing, normalization, and dropout to effectively process sequential data and capture intricate temporal dependencies. The defined hyperparameters control the model's depth, hidden state dimension, expanded state dimension, and attention mechanism dimensionality, thereby shaping its overall characteristics and performance.

Algorithm 1 SiMBA Block Process

```
1:  Input: Patch sequence $\mathbf{X}_{l-1}$ : $(\mathtt{B}, \mathtt{N}, \mathtt{D})$
2:  Output: Patch sequence $\mathbf{X}_l$ : $(\mathtt{B}, \mathtt{N}, \mathtt{D})$
3:  /* normalize the input sequence $\mathbf{X}'_{l-1}$ */
4:  $\mathbf{X}'_{l-1} : (\mathtt{B}, \mathtt{N}, \mathtt{D}) \leftarrow \mathbf{Norm}(_{l-1})$
5:  $\mathbf{x} : (\mathtt{B}, \mathtt{N}, \mathtt{P}) \leftarrow \mathbf{Linear}^{\mathbf{x}}(\mathbf{X}'_{l-1})$
6:  $\mathbf{z} : (\mathtt{B}, \mathtt{N}, \mathtt{P}) \leftarrow \mathbf{Linear}^{\mathbf{z}}(\mathbf{X}'_{l-1})$
7:  /* process with different direction */
8:  for $o$ in {loop} do
9:      $\mathbf{x}'_o : (\mathtt{B}, \mathtt{N}, \mathtt{P}) \leftarrow \mathbf{SiLU}(\mathbf{Conv1d}_o(\mathbf{x}))$
10:     $\mathbf{B}_o : (\mathtt{B}, \mathtt{N}, \mathtt{K}) \leftarrow \mathbf{Linear}^{\mathbf{B}}_o(\mathbf{x}'_o)$
11:     $\mathbf{C}_o : (\mathtt{B}, \mathtt{N}, \mathtt{K}) \leftarrow \mathbf{Linear}^{\mathbf{C}}_o(\mathbf{x}'_o)$
12:     /* softplus ensures positive $\boldsymbol{\Delta}_o$ */
13:     $\boldsymbol{\Delta}_o : (\mathtt{B}, \mathtt{N}, \mathtt{P}) \leftarrow \log(1 + \exp(\mathbf{Linear}^{\boldsymbol{\Delta}}_o(\mathbf{x}'_o) + \mathbf{Parameter}^{\boldsymbol{\Delta}}_o))$
14:     /* shape of $\mathbf{Parameter}^{\mathbf{A}}_o$ is $(\mathtt{P}, \mathtt{K})$ */
15:     $\overline{\mathbf{A}}_o : (\mathtt{B}, \mathtt{N}, \mathtt{P}, \mathtt{K}) \leftarrow \boldsymbol{\Delta}_o \bigotimes \mathbf{Parameter}^{\mathbf{A}}_o$
16:     $\overline{\mathbf{B}}_o : (\mathtt{B}, \mathtt{N}, \mathtt{P}, \mathtt{K}) \leftarrow \boldsymbol{\Delta}_o \bigotimes \mathbf{B}_o$
17:     $\mathbf{y}_o : (\mathtt{B}, \mathtt{N}, \mathtt{P}) \leftarrow \mathbf{SSM}(\overline{\mathbf{A}}_o, \overline{\mathbf{B}}_o, \mathbf{C}_o)(\mathbf{x}'_o)$
18: end for
19: /* get gated $\mathbf{y}_o$ */
20: $\mathbf{y}_o : (\mathtt{B}, \mathtt{N}, \mathtt{P}) \leftarrow \mathbf{y}_o \bigodot \mathbf{SiLU}(\mathbf{z})$
21: /* residual connection */
22: $\mathbf{X}_l : (\mathtt{B}, \mathtt{N}, \mathtt{D}) \leftarrow \mathbf{DP}(\mathbf{Linear}^{\mathbf{X}}(\mathbf{y}_o)) + \mathbf{X}_{l-1}$
23: $\mathbf{Res} : (\mathtt{B}, \mathtt{N}, \mathtt{D}) \leftarrow \mathbf{X}_l$
24: $\mathbf{X}_l : (\mathtt{B}, \mathtt{N}, \mathtt{D}) \leftarrow \mathbf{FFT}(\mathbf{X}_l)$
25: $\mathbf{X}_l : (\mathtt{B}, \mathtt{N}, \mathtt{C_b}, \mathtt{C_d}) \leftarrow \mathbf{rearrange}(\mathbf{X}_l), \mathtt{D} = \mathtt{C_b} \times \mathtt{C_d}$
26: $\mathbf{W}_1, \mathbf{W}_2 : (\mathtt{C_b}, \mathtt{C_d}, \mathtt{C_d}) \leftarrow$ shape of complex value $\mathbf{Parameter}^{\mathbf{W}_1, \mathbf{W}_2}_l$
27: $\mathbf{B}_1, \mathbf{B}_2 : (\mathtt{C_b}, \mathtt{C_d}) \leftarrow$ shape of complex value $\mathbf{Parameter}^{\mathbf{B}_1, \mathbf{B}_2}_l$
28: $\mathbf{X}^{Real}_l : (\mathtt{B}, \mathtt{N}, \mathtt{C_b}, \mathtt{C_d}) \leftarrow ReLU(\mathbf{EMM}(\mathbf{W}_1, \mathbf{B}^{Real}_1)(\mathbf{x}^{Real}_l))$
29: $\mathbf{X}^{Img}_l : (\mathtt{B}, \mathtt{N}, \mathtt{C_b}, \mathtt{C_d}) \leftarrow ReLU(\mathbf{EMM}(\mathbf{W}_1, \mathbf{B}^{Img}_1)(\mathbf{x}^{Img}_l))$
30: $\mathbf{X}_l : (\mathtt{B}, \mathtt{N}, \mathtt{C_b}, \mathtt{C_d}) \leftarrow \mathbf{IFFT}(\mathbf{X}^{Real}_l + j\mathbf{X}^{Img}_l)$
31: $\mathbf{X}_l : (\mathtt{B}, \mathtt{N}, \mathtt{D}) \leftarrow \mathbf{X}_l + \mathbf{Res}$
32: Return: $\mathbf{X}_l$
```