Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model

Lianghui Zhu [1 *] **Bencheng Liao** [2 1 *] **Qian Zhang** [3] **Xinlong Wang** [4] **Wenyu Liu** [1] **Xinggang Wang** [1]

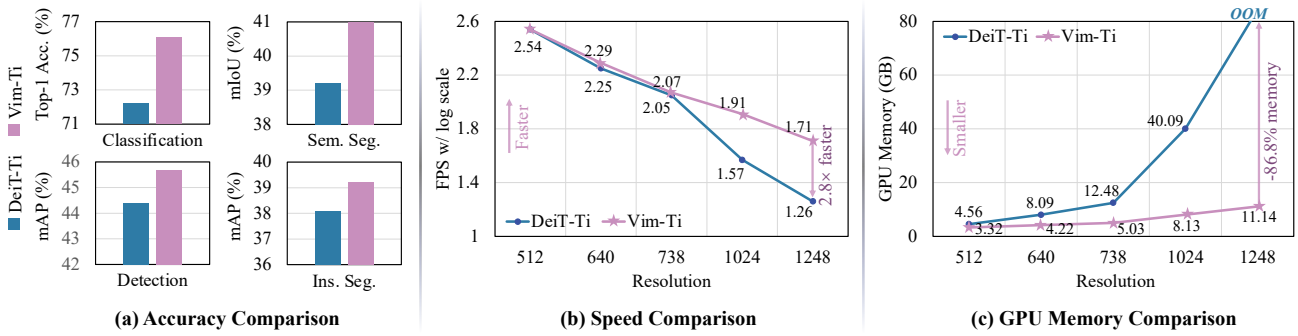

Figure 1: Performance and efficiency comparisons between DeiT (Touvron et al., 2021a) and our Vim model. Results show that Vim outperforms DeiT on both ImageNet classification and downstream detection and segmentation tasks and is more computation and memory efficient than DeiT in dealing with high-resolution images. For example, Vim is 2.8× faster than DeiT and saves 86.8% GPU memory when performing batch inference to extract features on images with a resolution of 1248×1248, *i.e.*, 6084 tokens per image.

Abstract

Recently the state space models (SSMs) with efficient hardware-aware designs, i.e., the Mamba deep learning model, have shown great potential for long sequence modeling. Meanwhile building efficient and generic vision backbones purely upon SSMs is an appealing direction. However, representing visual data is challenging for SSMs due to the position-sensitivity of visual data and the requirement of global context for visual understanding. In this paper, we show that the reliance on self-attention for visual representation learning is not necessary and propose a new generic vision backbone with bidirectional Mamba blocks (Vim), which marks the image sequences with position embeddings and compresses the visual representation with bidirectional state space models. On ImageNet classification, COCO object detection, and ADE20k semantic segmentation tasks, Vim achieves higher performance compared to well-established vision transformers like DeiT, while also demonstrating significantly improved computation & memory efficiency. For example, Vim is 2.8× faster than DeiT and saves 86.8% GPU memory when performing batch inference to extract features on images with a resolution of 1248×1248. The results demonstrate that Vim is capable of overcoming the computation & memory constraints on performing Transformer-style understanding for high-resolution images and it has great potential to be the next-generation backbone for vision foundation models. Code and models are released at https://github.com/hustvl/Vim

[*]Equal contribution [1]School of EIC, Huazhong University of Science & Technology [2]Institute of Artificial Intelligence, Huazhong University of Science & Technology [3]Horizon Robotics [4]Beijing Academy of Artificial Intelligence. Correspondence to: Xinggang Wang <xgwang@hust.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Recent research advancements have led to a surge of interest in the state space model (SSM). Originating from the classic Kalman filter model (Kalman, 1960), modern SSMs excel at capturing long-range dependencies and benefit from parallel training. Some SSM-based methods, such as the linear state-space layers (LSSL) (Gu et al., 2021b), structured state space sequence model (S4) (Gu et al., 2021a), diagonal state space (DSS) (Gupta et al., 2022), and S4D (Gu et al., 2022), are proposed to process sequence data across a

wide range of tasks and modalities, particularly on modeling long-range dependencies. They are efficient in processing long sequences because of convolutional computation and near-linear computation. 2-D SSM (Baron et al., 2023), SGConvNeXt (Li et al., 2022b), and ConvSSM (Smith et al., 2023a) combine SSM with CNN or Transformer architecture to process 2-D data. The recent work, Mamba (Gu & Dao, 2023), incorporates time-varying parameters into the SSM and proposes a hardware-aware algorithm to enable very efficient training and inference. The superior scaling performance of Mamba indicates that it is a promising alternative to Transformer in language modeling. Nevertheless, a generic pure-SSM-based backbone network has not been explored for processing visual data, such as images and videos.

Vision Transformers (ViTs) have achieved great success in visual representation learning, excelling in large-scale self-supervised pre-training and high performance on downstream tasks. Compared with convolutional neural networks, the core advantage lies in that ViT can provide each image patch with data/patch-dependent global context through self-attention. This differs from convolutional networks that use the same parameters, *i.e.*, the convolutional filters, for all positions. Another advantage is the modality-agnostic modeling by treating an image as a sequence of patches without 2D inductive bias, which makes it the preferred architecture for multimodal applications (Bavishi et al., 2023; Li et al., 2023; Liu et al., 2023). At the same time, the self-attention mechanism in Transformers poses challenges in terms of speed and memory usage when dealing with long-range visual dependencies, *e.g.*, processing high-resolution images.

Motivated by the success of Mamba in language modeling, it is appealing that we can also transfer this success from language to vision, *i.e.*, to design a generic and efficient visual backbone with the advanced SSM method. However, there are two challenges for Mamba, *i.e.*, unidirectional modeling and lack of positional awareness. To address these challenges, we propose the Vision Mamba (Vim) model, which incorporates the bidirectional SSMs for data-dependent global visual context modeling and position embeddings for location-aware visual recognition. We first split the input image into patches and linearly project them as vectors to Vim. Image patches are treated as the sequence data in Vim blocks, which efficiently compresses the visual representation with the proposed bidirectional selective state space. Furthermore, the position embedding in Vim block provides the awareness for spatial information, which enables Vim to be more robust in dense prediction tasks. In the current stage, we train the Vim model on the supervised image classification task using the ImageNet dataset and then use the pretrained Vim as the backbone to perform sequential visual representation learning for downstream dense prediction tasks, *i.e.*, semantic segmentation, object detection, and instance segmentation. Like Transformers, Vim can be pretrained on large-scale unsupervised visual data for better visual representation. Thanks to the better efficiency of Mamba, the large-scale pretraining of Vim can be achieved with lower computational cost.

Compared with other SSM-based models for vision tasks, Vim is a pure-SSM-based method and models images in a sequence manner, which is more promising for a generic and efficient backbone. Thanks to the bidirectional compressing modeling with positional awareness, Vim is the first pure-SSM-based model to handle dense prediction tasks. Compared with the most convincing Transformer-based model, *i.e.*, DeiT (Touvron et al., 2021a), Vim achieves superior performance on ImageNet classification. Furthermore, Vim is more efficient in terms of GPU memory and inference time for high-resolution images. The efficiency in terms of memory and speed empowers Vim to directly perform sequential visual representation learning without relying on 2D priors (such as the 2D local window in ViTDet (Li et al., 2022c)) for high-resolution visual understanding tasks while achieving higher accuracy than DeiT.

Our main contributions can be summarized as follows:

- We propose Vision Mamba (Vim), which incorporates bidirectional SSM for data-dependent global visual context modeling and position embeddings for location-aware visual understanding.
- Without the need of attention, the proposed Vim has the same modeling power as ViT while it only has subquadratic-time computation and linear memory complexity. Specifically, Vim is 2.8× faster than DeiT and saves 86.8% GPU memory when performing batch inference to extract features on images at the resolution of 1248×1248.
- We conduct extensive experiments on ImageNet classification and dense prediction downstream tasks. The results demonstrate that Vim achieves superior performance compared to the well-established and highly-optimized plain vision Transformer, *i.e.*, DeiT.

2. Related Work

Architectures for generic vision backbone. In the early eras, ConvNet (LeCun et al., 1998) serves as the de-facto standard network design for computer vision. Many convolutional neural architectures (Krizhevsky et al., 2012; Szegedy et al., 2015; Simonyan & Zisserman, 2014; He et al., 2016; Tan & Le, 2019; Wang et al., 2020a; Huang et al., 2017; Xie et al., 2017; Tan & Le, 2021; Radosavovic et al., 2020) have been proposed as the vision backbone for various visual applications. The pioneering work, Vision Transformer (ViT) (Dosovitskiy et al., 2020) changes the landscape. It treats an image as a sequence of flattened 2D

patches and directly applies a pure Transformer architecture. The surprising results of ViT on image classification and its scaling ability encourage a lot of follow-up works (Touvron et al., 2021b; Tolstikhin et al., 2021; Touvron et al., 2022; Fang et al., 2022). One line of works focuses on hybrid architecture designs by introducing 2D convolutional priors into ViT (Wu et al., 2021; Dai et al., 2021; d'Ascoli et al., 2021; Dong et al., 2022). PVT (Wang et al., 2021) proposes a pyramid structure Transformer. Swin Transformer (Liu et al., 2021) applies self-attention within shift windows. Another line of works focuses on improving traditional 2D ConvNets with more advanced settings (Wang et al., 2023b; Liu et al., 2022a). ConvNeXt (Liu et al., 2022b) reviews the design space and proposes pure ConvNets, which can be scalable as ViT and its variants. RepLKNet (Ding et al., 2022) proposes to scale up the kernel size of existing ConvNets to bring improvements.

Though these dominant follow-up works demonstrate superior performance and better efficiency on ImageNet (Deng et al., 2009) and various downstream tasks (Lin et al., 2014; Zhou et al., 2019) by introducing 2D priors, with the surge of large-scale visual pretraining (Bao et al., 2022; Fang et al., 2023; Caron et al., 2021) and multi-modality applications (Radford et al., 2021; Li et al., 2022a; 2023; Liu et al., 2023; Bavishi et al., 2023; Jia et al., 2021), vanilla Transformer-style model strikes back to the center stage of computer vision. The advantages of larger modeling capacity, unified multi-modality representation, being friendly to self-supervised learning *etc.*, make it the preferred architecture. However, the number of visual tokens is limited due to the quadratic complexity of Transformer. There are plenty of works (Choromanski et al., 2021; Wang et al., 2020b; Kitaev et al., 2020; Child et al., 2019; Ding et al., 2023; Qin et al., 2023; Sun et al., 2023) to address this long-standing and prominent challenge, but few of them focus on visual applications. Recently, LongViT (Wang et al., 2023c) built an efficient Transformer architecture for computational pathology applications via dilated attention. The linear computation complexity of LongViT allows it to encode the extremely long visual sequence. In this work, we draw inspiration from Mamba (Gu & Dao, 2023) and explore building a pure-SSM-based model as a generic vision backbone without using attention, while preserving the sequential, modality-agnostic modeling merit of ViT.

State space models for long sequence modeling. (Gu et al., 2021a) proposes a Structured State-Space Sequence (S4) model, a novel alternative to CNNs or Transformers, to model the long-range dependency. The promising property of linearly scaling in sequence length attracts further explorations. (Wang et al., 2022) proposes Bidirectional Gated SSM to replicate BERT (Devlin et al., 2018) results without attention. (Smith et al., 2023b) proposes a new S5 layer by introducing MIMO SSM and efficient parallel scan into S4 layer. (Fu et al., 2023) designs a new SSM layer, H3, that nearly fills the performance gap between SSMs and Transformer attention in language modeling. (Mehta et al., 2023) builds the Gated State Space layer on S4 by introducing more gating units to improve the expressivity. Recently, (Gu & Dao, 2023) proposes a data-dependent SSM layer and builds a generic language model backbone, Mamba, which outperforms Transformers at various sizes on large-scale real data and enjoys linear scaling in sequence length. In this work, we explore transferring the success of Mamba to vision, *i.e.*, building a generic vision backbone purely upon SSM without attention.

State space models for visual applications. (Islam & Bertasius, 2022) uses 1D S4 to handle the long-range temporal dependencies for video classification. (Nguyen et al., 2022) further extends 1D S4 to handle multi-dimensional data including 2D images and 3D videos. (Islam et al., 2023) combines the strengths of S4 and self-attention to build TranS4mer model, achieving state-of-the-art performance for movie scene detection. (Wang et al., 2023a) introduces a novel selectivity mechanism to S4, largely improving the performance of S4 on long-form video understanding with a much lower memory footprint. (Yan et al., 2023) supplants attention mechanisms with a more scalable SSM-based backbone to generate high-resolution images and process fine-grained representation under affordable computation. (Ma et al., 2024) proposes U-Mamba, a hybrid CNN-SSM architecture, to handle the long-range dependencies in biomedical image segmentation. The above works (Xing et al., 2024; Ma et al., 2024; Yan et al., 2023; Wang et al., 2023a; Islam et al., 2023; Nguyen et al., 2022; Islam & Bertasius, 2022) either apply SSM to specific visual applications or build a hybrid architecture by combining SSM with convolution or attention. Different from them, we build a pure-SSM-based model, which can be adopted as a generic vision backbone. It is noteworthy that VMamba (Liu et al., 2024), a concurrent work with our method, has demonstrated impressive results in visual recognition by incorporating Mamba with multi-directional scanning and a hierarchical network architecture. In contrast, Vim primarily concentrates on visual sequence learning and boasts a unified representation for multi-modality data.

3. Method

The goal of Vision Mamba (Vim) is to introduce the advanced state space model (SSM), *i.e.*, Mamba (Gu & Dao, 2023), to computer vision. This section begins with a description of the preliminaries of SSM. It is followed by an overview of Vim. We then detail how the Vim block processes input token sequences and proceed to illustrate the architecture details of Vim. The section concludes with an

analysis of the efficiency of the proposed Vim.

3.1. Preliminaries

The SSM-based models, *i.e.*, structured state space sequence models (S4) and Mamba are inspired by the continuous system, which maps a 1-D function or sequence $x(t) \in \mathbb{R} \mapsto y(t) \in \mathbb{R}$ through a hidden state $h(t) \in \mathbb{R}^{\mathtt{N}}$. This system uses $\mathbf{A} \in \mathbb{R}^{\mathtt{N}\times\mathtt{N}}$ as the evolution parameter and $\mathbf{B} \in \mathbb{R}^{\mathtt{N}\times 1}$, $\mathbf{C} \in \mathbb{R}^{1\times\mathtt{N}}$ as the projection parameters. The continuous system works as follows: $h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$ and $y(t) = \mathbf{C}h(t)$.

The S4 and Mamba are the discrete versions of the continuous system, which include a timescale parameter $\mathbf{\Delta}$ to transform the continuous parameters $\mathbf{A}$, $\mathbf{B}$ to discrete parameters $\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$. The commonly used method for transformation is zero-order hold (ZOH), which is defined as follows:

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

After the discretization of $\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$, the discretized version using a step size $\mathbf{\Delta}$ can be rewritten as:

$$\begin{aligned} h_t &= \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, \\ y_t &= \mathbf{C}h_t. \end{aligned} \tag{2}$$

At last, the models compute output through a global convolution.

$$\begin{aligned} \overline{\mathbf{K}} &= (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}}\overline{\mathbf{B}}, \dots, \mathbf{C}\overline{\mathbf{A}}^{\mathtt{M}-1}\overline{\mathbf{B}}), \\ \mathbf{y} &= \mathbf{x} * \overline{\mathbf{K}}, \end{aligned} \tag{3}$$

where $\mathtt{M}$ is the length of the input sequence $\mathbf{x}$, and $\overline{\mathbf{K}} \in \mathbb{R}^{\mathtt{M}}$ is a structured convolutional kernel.

3.2. Vision Mamba

An overview of the proposed Vim is shown in Fig. 2. The standard Mamba is designed for the 1-D sequence. To process the vision tasks, we first transform the 2-D image $\mathbf{t} \in \mathbb{R}^{\mathtt{H}\times\mathtt{W}\times\mathtt{C}}$ into the flattened 2-D patches $\mathbf{x_p} \in \mathbb{R}^{\mathtt{J}\times(\mathtt{P}^2\cdot\mathtt{C})}$, where $(\mathtt{H}, \mathtt{W})$ is the size of input image, $\mathtt{C}$ is the number of channels, $\mathtt{P}$ is the size of image patches. Next, we linearly project the $\mathbf{x_p}$ to the vector with size $\mathtt{D}$ and add position embeddings $\mathbf{E}_{pos} \in \mathbb{R}^{(\mathtt{J}+1)\times\mathtt{D}}$, as follows:

$$\mathbf{T}_0 = [\mathbf{t}_{cls}; \mathbf{t}_p^1\mathbf{W}; \mathbf{t}_p^2\mathbf{W}; \cdots; \mathbf{t}_p^{\mathtt{J}}\mathbf{W}] + \mathbf{E}_{pos}, \tag{4}$$

where $\mathbf{t}_p^{\mathtt{j}}$ is the $\mathtt{j}$-th patch of $\mathbf{t}$, $\mathbf{W} \in \mathbb{R}^{(\mathtt{P}^2\cdot\mathtt{C})\times\mathtt{D}}$ is the learnable projection matrix. Inspired by ViT (Dosovitskiy et al., 2020) and BERT (Kenton & Toutanova, 2019), we also use class token to represent the whole patch sequence, which is denoted as $\mathbf{t}_{cls}$. We then send the token sequence ($\mathbf{T}_{\mathtt{l}-1}$) to the $\mathtt{l}$-th layer of the Vim encoder, and get the output $\mathbf{T}_{\mathtt{l}}$. Finally, we normalize the output class token $\mathbf{T}_{\mathtt{L}}^0$ and feed it to the multi-layer perceptron (MLP) head to get the final prediction $\hat{p}$, as follows: $\mathbf{T}_l = \mathbf{Vim}(\mathbf{T}_{\mathtt{l}-1}) + \mathbf{T}_{\mathtt{l}-1}$, $\mathbf{f} = \mathbf{Norm}(\mathbf{T}_{\mathtt{L}}^0)$, and $\hat{p} = \mathbf{MLP}(\mathbf{f})$, where $\mathbf{Vim}$ is the proposed vision mamba block, $\mathtt{L}$ is the number of layers, and $\mathbf{Norm}$ is the normalization layer.

3.3. Vim Block

The original Mamba block is designed for the 1-D sequence, which is not suitable for vision tasks requiring spatial-aware understanding. In this section, we introduce the Vim block, which incorporates the bidirectional sequence modeling for the vision tasks. The Vim block is shown in Fig. 2.

Specifically, we present the operations of Vim block in Algo. 1. The input token sequence $\mathbf{T}_{\mathtt{l}-1}$ 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. Finally, we compute the $\mathbf{y}_{forward}$ and $\mathbf{y}_{backward}$ through the SSM. The $\mathbf{y}_{forward}$ and $\mathbf{y}_{backward}$ are then gated by the $\mathbf{z}$ and added together to get the output token sequence $\mathbf{T}_{\mathtt{l}}$.

3.4. Architecture Details

In summary, the hyper-parameters of our architecture are listed as follows: $\mathtt{L}$ denotes the number of blocks, $\mathtt{D}$ denotes the hidden state dimension, $\mathtt{E}$ denotes the expanded state dimension, and $\mathtt{N}$ denotes the SSM dimension. Following ViT (Dosovitskiy et al., 2020) and DeiT (Touvron et al., 2021b), we first employ 16×16 kernel size projection layer to get a 1-D sequence of non-overlapping patch embeddings. Subsequently, we directly stack $\mathtt{L}$ Vim blocks. By default, we set the number of blocks $\mathtt{L}$ to 24, SSM dimension $\mathtt{N}$ to 16. To align with the model sizes of DeiT series, we set the hidden state dimension $\mathtt{D}$ to 192 and expanded state dimension $\mathtt{E}$ to 384 for the tiny-size variant. For the small-size variant, we set $\mathtt{D}$ to 384 and $\mathtt{E}$ to 768.

3.5. Efficiency Analysis

Traditional SSM-based methods leverage the fast Fourier transform to boost the convolution operation as shown in Eq. (3). For data-dependent methods, such as Mamba, the SSM operation in Line 11 of Algo. 1 is no longer equivalent to convolution. To address this problem, Mamba and the proposed Vim choose a modern-hardware-friendly way to ensure efficiency. The key idea of this optimization is to avoid the IO-bound and memory-bound of modern hardware accelerators (GPUs).

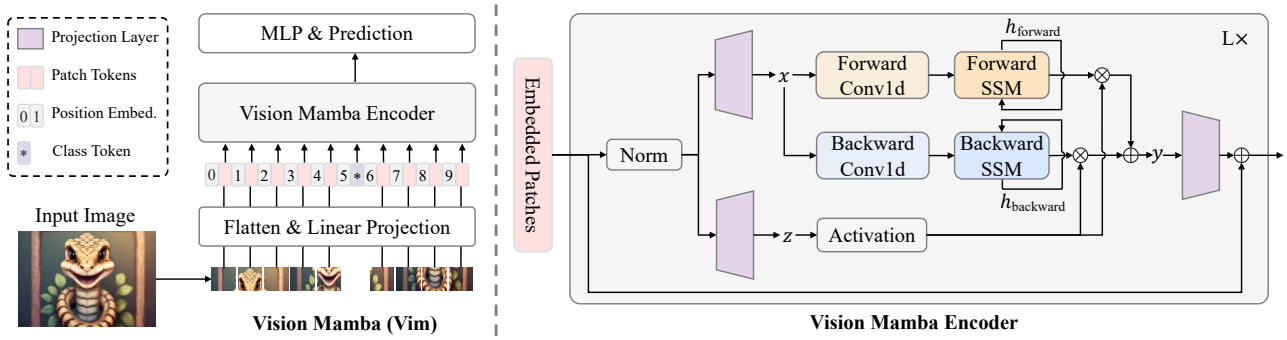

Figure 2: The overview of the proposed Vim model. We first split the input image into patches, and then project them into patch tokens. Last, we send the sequence of tokens to the proposed Vim encoder. To perform ImageNet classification, we concatenate an extra learnable classification token to the patch token sequence. Different from Mamba for text sequence modeling, Vim encoder processes the token sequence with both forward and backward directions.

IO-Efficiency. The high bandwidth memory (HBM) and SRAM are two important components for GPUs. Among them, SRAM has a larger bandwidth and HBM has a bigger memory size. The standard implementation of Vim's SSM operation with HBM requires the number of memory IO on the order of $O(\texttt{BMEN})$. Inspired by Mamba, Vim first reads in $O(\texttt{BME}+\texttt{EN})$ bytes of memory $(\mathbf{\Delta_o}, \mathbf{A_o}, \mathbf{B_o}, \mathbf{C_o})$ from slow HBM to fast SRAM. Then, Vim gets the discrete $\overline{\mathbf{A}}_\mathbf{o}$, $\overline{\mathbf{B}}_\mathbf{o}$ of a size of $(\texttt{B}, \texttt{M}, \texttt{E}, \texttt{N})$ in SRAM. Last, Vim performs SSM operations in SRAM and writes the output of a size of $(\texttt{B}, \texttt{M}, \texttt{E})$ back to HBM. This method can help to reduce IOs from $O(\texttt{BMEN})$ to $O(\texttt{BME}+\texttt{EN})$.

Memory-Efficiency. To avoid out-of-memory problems and achieve lower memory usage when dealing with long sequences, Vim chooses the same recomputation method as Mamba. For the intermediate states of size $(\texttt{B}, \texttt{M}, \texttt{E}, \texttt{N})$ to calculate the gradient, Vim recomputes them at the network backward pass. For intermediate activations such as the output of activation functions and convolution, Vim also recomputes them to optimize the GPU memory requirement, as the activation values take a lot of memory but are fast for recomputation.

Computation-Efficiency. SSM in Vim block (Line 11 in Algo.1) and self-attention in Transformer both play a key role in providing global context adaptively. Given a visual sequence $\mathbf{T} \in R^{1\times\texttt{M}\times\texttt{D}}$ and the default setting $\texttt{E}=2\texttt{D}$, the computation complexity of a global self-attention and SSM are:

$$\Omega(\text{self-attention}) = 4\texttt{MD}^2 + 2\texttt{M}^2\texttt{D}, \tag{5}$$

$$\Omega(\text{SSM}) = 3\texttt{M}(2\texttt{D})\texttt{N} + \texttt{M}(2\texttt{D})\texttt{N}, \tag{6}$$

where self-attention is quadratic to sequence length $\texttt{M}$, and SSM is linear to sequence length $\texttt{M}$ ($\texttt{N}$ is a fixed parameter, set to 16 by default). The computational efficiency makes Vim scalable for gigapixel applications with large sequence lengths.

4. Experiment

Method	image size	#param.	ImageNet top-1 acc.
Convnets			
ResNet-18	224^2	12M	69.8
ResNet-50	224^2	25M	76.2
ResNet-101	224^2	45M	77.4
ResNet-152	224^2	60M	78.3
ResNeXt50-32×4d	224^2	25M	77.6
RegNetY-4GF	224^2	21M	80.0
Transformers			
ViT-B/16	384^2	86M	77.9
ViT-L/16	384^2	307M	76.5
DeiT-Ti	224^2	6M	72.2
DeiT-S	224^2	22M	79.8
DeiT-B	224^2	86M	81.8
SSMs			
S4ND-ViT-B	224^2	89M	80.4
Vim-Ti	224^2	7M	76.1
Vim-Ti†	224^2	7M	78.3 +2.2
Vim-S	224^2	26M	80.3
Vim-S†	224^2	26M	81.4 +1.1
Vim-B	224^2	98M	81.9
Vim-B†	224^2	98M	83.2 +1.3

Table 1: Comparison with different backbones on ImageNet-1K validation set. † represents the model is fine-tuned with our long sequence setting.

4.1. Image Classification

Settings. We benchmark Vim on the ImageNet-1K dataset (Deng et al., 2009), which contains 1.28M training

Algorithm 1 Vim Block Process

Require: token sequence $\mathbf{T}_{l-1}$: $(\mathtt{B}, \mathtt{M}, \mathtt{D})$
Ensure: token sequence $\mathbf{T}_l$: $(\mathtt{B}, \mathtt{M}, \mathtt{D})$
/* normalize the input sequence $\mathbf{T}'_{l-1}$ */
$\mathbf{T}'_{l-1}$: $(\mathtt{B}, \mathtt{M}, \mathtt{D}) \leftarrow \mathbf{Norm}(\mathbf{T}_{l-1})$
$\mathbf{x}$: $(\mathtt{B}, \mathtt{M}, \mathtt{E}) \leftarrow \mathbf{Linear}^{\mathbf{x}}(\mathbf{T}'_{l-1})$
$\mathbf{z}$: $(\mathtt{B}, \mathtt{M}, \mathtt{E}) \leftarrow \mathbf{Linear}^{\mathbf{z}}(\mathbf{T}'_{l-1})$
/* process with different direction */
for o in {forward, backward} **do**
  $\mathbf{x}'_o$: $(\mathtt{B}, \mathtt{M}, \mathtt{E}) \leftarrow \mathbf{SiLU}(\mathbf{Conv1d}_o(\mathbf{x}))$
  $\mathbf{B}_o$: $(\mathtt{B}, \mathtt{M}, \mathtt{N}) \leftarrow \mathbf{Linear}^{\mathbf{B}}_o(\mathbf{x}'_o)$
  $\mathbf{C}_o$: $(\mathtt{B}, \mathtt{M}, \mathtt{N}) \leftarrow \mathbf{Linear}^{\mathbf{C}}_o(\mathbf{x}'_o)$
  /* softplus ensures positive $\mathbf{\Delta}_o$ */
  $\mathbf{\Delta}_o$: $(\mathtt{B}, \mathtt{M}, \mathtt{E}) \leftarrow \log(1 + \exp(\mathbf{Linear}^{\mathbf{\Delta}}_o(\mathbf{x}'_o) + \mathbf{Parameter}^{\mathbf{\Delta}}_o))$
  /* shape of $\mathbf{Parameter}^{\mathbf{A}}_o$ is $(\mathtt{E}, \mathtt{N})$ */
  $\overline{\mathbf{A}_o}$: $(\mathtt{B}, \mathtt{M}, \mathtt{E}, \mathtt{N}) \leftarrow \mathbf{\Delta}_o \bigotimes \mathbf{Parameter}^{\mathbf{A}}_o$
  $\overline{\mathbf{B}_o}$: $(\mathtt{B}, \mathtt{M}, \mathtt{E}, \mathtt{N}) \leftarrow \mathbf{\Delta}_o \bigotimes \mathbf{B}_o$
  /* initialize h_o and $\mathbf{y}_o$ with 0 */
  h_o : $(\mathtt{B}, \mathtt{E}, \mathtt{N}) \leftarrow$ zeros $(\mathtt{B}, \mathtt{E}, \mathtt{N})$
  $\mathbf{y}_o$: $(\mathtt{B}, \mathtt{M}, \mathtt{E}) \leftarrow$ zeros $(\mathtt{B}, \mathtt{M}, \mathtt{E})$
  /* SSM recurrent */
  for i in {0, ..., M-1} **do**
    $h_o = \overline{\mathbf{A}_o}[:, i, :, :] \bigodot h_o + \overline{\mathbf{B}_o}[:, i, :, :] \bigodot \mathbf{x}'_o[:, i, :, \mathtt{None}]$
    $\mathbf{y}_o[:, i, :] = h_o \bigotimes \mathbf{C}_o[:, i, :]$
  end for
end for
/* get gated $\mathbf{y}$ */
$\mathbf{y}'_{forward}$: $(\mathtt{B}, \mathtt{M}, \mathtt{E}) \leftarrow \mathbf{y}_{forward} \bigodot \mathbf{SiLU}(\mathbf{z})$
$\mathbf{y}'_{backward}$: $(\mathtt{B}, \mathtt{M}, \mathtt{E}) \leftarrow \mathbf{y}_{backward} \bigodot \mathbf{SiLU}(\mathbf{z})$
/* residual connection */
$\mathbf{T}_l$: $(\mathtt{B}, \mathtt{M}, \mathtt{D}) \leftarrow \mathbf{Linear}^{\mathbf{T}}(\mathbf{y}'_{forward} + \mathbf{y}'_{backward}) + \mathbf{T}_{l-1}$
Return: $\mathbf{T}_l$

images and 50K validation images from 1,000 categories. All models are trained on the training set, and top-1 accuracy on the validation set is reported. For fair comparisons, our training settings mainly follow DeiT (Touvron et al., 2021b). Specifically, we apply random cropping, random horizontal flipping, label-smoothing regularization, mixup, and random erasing as data augmentations. When training on 224^2 input images, we employ AdamW (Loshchilov & Hutter, 2019) with a momentum of 0.9, a total batch size of 1024, and a weight decay of 0.05 to optimize models. We train the Vim models for 300 epochs using a cosine schedule, 1×10^{-3} initial learning rate, and EMA. During testing, we apply a center crop on the validation set to crop out 224^2 images. Experiments are performed on 8 A800 GPUs.

Long Sequence Fine-tuning To make full use of the efficient long sequence modeling power of Vim, we continue to fine-tune Vim with a long sequence setting for 30 epochs after pretraining. Specifically, we set a patch extraction stride of 8 while keeping the patch size unchanged, a constant learning rate of 10^{-5}, and a weight decay of 10^{-8}.

Results. Tab. 1 compares Vim with ConvNet-based, Transformer-based and SSM-based backbone networks. Compared to ConvNet-based ResNet (He et al., 2016), Vim demonstrates superior performance. For example, when the parameters are roughly similar, the top-1 accuracy of Vim-Small reaches 80.3, which is 4.1 points higher than that of ResNet50. Compared with the conventional self-attention-based ViT (Dosovitskiy et al., 2020), Vim outperforms it by considerable margins in terms of both parameter numbers and classification accuracy. When compared to the highly-optimized ViT-variant, *i.e.*, DeiT (Touvron et al., 2021b), Vim surpasses it at different scales with comparable parameter numbers: 3.9 points higher for Vim-Tiny over DeiT-Tiny, 0.5 points higher for Vim-Small over DeiT-Small, and 0.1 points higher for Vim-Base over DeiT-Base. Compared with SSM-based S4ND-ViT-B (Nguyen et al., 2022), Vim achieves similar top-1 accuracy with 3× fewer parameters. After long sequence fine-tuning, Vim-Tiny†, Vim-S†, and Vim-B† all achieve higher results. Among them, Vim-S† even achieves similar results with DeiT-B. The results demonstrate that Vim can be adapted to longer sequence modeling easily and extract stronger visual representation.

Method	Backbone	image size	#param.	*val* mIoU
DeepLab v3+	ResNet-101	512^2	63M	44.1
UperNet	ResNet-50	512^2	67M	41.2
UperNet	ResNet-101	512^2	86M	44.9
UperNet	DeiT-Ti	512^2	11M	39.2
UperNet	DeiT-S	512^2	43M	44.0
UperNet	Vim-Ti	512^2	13M	41.0
UperNet	Vim-S	512^2	46M	44.9

Table 2: Results of semantic segmentation on the ADE20K *val* set.

Fig. 1 (b) and (c) compare the FPS and GPU memory of tiny-size Vim and DeiT. Vim demonstrates better efficiency in speed and memory as image resolution grows. Specifically, when the image size is 512×512, Vim achieves similar FPS and memory as DeiT. As the image size grows to 1248×1248, Vim is 2.8× faster than DeiT and saves 86.8% GPU memory. The pronounced superiority of Vim's linear scaling in sequence length makes it ready for high-resolution downstream vision applications and long-sequence multi-modality applications.

4.2. Semantic Segmentation

Settings. We conduct experiments for semantic segmentation on the ADE20K (Zhou et al., 2019) and use UperNet (Xiao et al., 2018b) as the segmentation framework. We provide detailed settings in Sec. B. **Results.** As shown in Tab. 2, Vim consistently outperforms DeiT across different scales: 1.8 mIoU higher for Vim-Ti over DeiT-Ti, and

Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{s}	AP^{box}_{m}	AP^{box}_{l}
DeiT-Ti	44.4	63.0	47.8	26.1	47.4	61.8
Vim-Ti	45.7	63.9	49.6	26.1	49.0	63.2
Backbone	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{s}	AP^{mask}_{m}	AP^{mask}_{l}
DeiT-Ti	38.1	59.9	40.5	18.1	40.5	58.4
Vim-Ti	39.2	60.9	41.7	18.2	41.8	60.2

Table 3: Results of object detection and instance segmentation on the COCO *val* set using Cascade Mask R-CNN (Cai & Vasconcelos, 2019) framework.

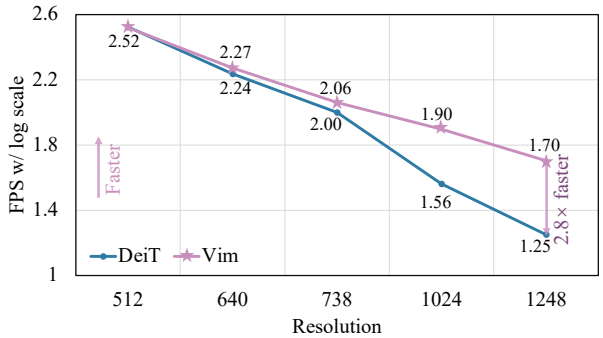

Figure 3: FPS comparison between DeiT-Ti (Touvron et al., 2021a) and our Vim-Ti on the commonly used downstream framework. We perform batch inference and benchmark the log-scaled FPS on the architecture with the backbone and FPN. Vim achieves comparable performance to DeiT with a small resolution, *i.e.*, 512×512. As the input image resolution increases, Vim has a higher FPS.

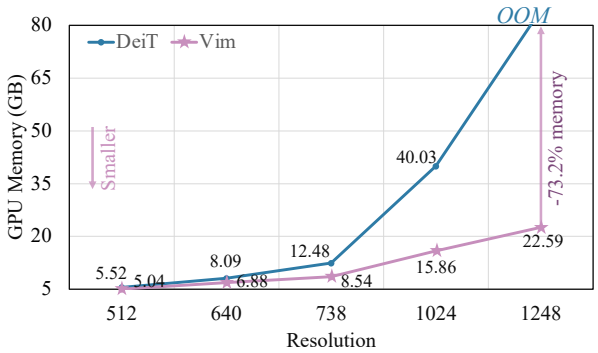

Figure 4: GPU memory efficiency comparison between DeiT-Ti (Touvron et al., 2021a) and our Vim-Ti on the commonly used downstream framework. We perform batch inference and benchmark the GPU memory on the architecture with the backbone and FPN. Vim requires comparable GPU memory to DeiT with a small resolution, *i.e.*, 512×512. As the input image resolution increases, Vim will use significantly less GPU memory.

0.9 mIoU higher for Vim-S over DeiT-S. Compared to the ResNet-101 backbone, our Vim-S achieves the same segmentation performance with nearly 2× fewer parameters.

To further evaluate the efficiency for downstream tasks, *i.e.*, segmentation, detection, and instance segmentation, we combine the backbones with a commonly used feature pyramid network (FPN) module and benchmark their FPS and GPU memory. As shown in Fig. 3 and Fig. 4, the efficiency curves demonstrate similar comparison results of the pure backbone (Fig. 1), though we append a heavy FPN on the backbones. The exceptional linear scaling performance is attributed to our proposed efficient backbone Vim, which builds the foundation for learning gigapixel-level visual representation in an end-to-end manner without the need for multi-stage encoding (*e.g.*, aerial image, medical image, and computational pathology).

4.3. Object Detection and Instance Segmentation

Settings. We conduct experiments for object detection and instance segmentation on the COCO 2017 dataset (Lin et al., 2014) and use ViTDet (Xiao et al., 2018b) as the basic framework. We provide detailed settings in Sec. B.

Results. Tab. 3 compares Vim-Ti with DeiT-Ti using Cascade Mask R-CNN framework (Cai & Vasconcelos, 2019). Vim-Ti surpasses DeiT-Ti by 1.3 box AP and 1.1 mask AP. For the middle-size and large-size objects, Vim-Ti outperforms DeiT-Ti by 1.6 AP^{box}_{m}/1.3 AP^{mask}_{m} and 1.4 AP^{box}_{l}/1.8 AP^{mask}_{l}, demonstrating better long-range context learning than DeiT (Fig. 5).

We highlight that the accuracy superiority is non-trivial since DeiT is equipped with window attention while Vim works in a pure sequence modeling manner. Specifically, to perform representation learning on high-resolution images (*i.e.*, 1024×1024), we follow ViTDet (Li et al., 2022c) and modify the DeiT backbone with the use of 2D window attention, which injects 2D prior and breaks the sequential modeling nature of Transformer. Thanks to the efficiency illustrated in Sec. 3.5, Fig. 1 and Fig. 4, we can directly apply Vim on 1024×1024 input images and learn sequential visual representation for object detection and instance segmentation without need for 2D priors in the backbone.

4.4. Ablation Study

Bidirectional SSM. We ablate the key bidirectional design of Vim, using ImageNet-1K classification and the Segmenter (Strudel et al., 2021) semantic segmentation framework on ADE20K. To fully evaluate the power of learned representation on ImageNet, we use a simple Segmenter head with only 2 layers to perform transfer learning on semantic segmentation. We study the following bidirectional strategies. *None*: We directly adopt the Mamba block to

Bidirectional strategy	ImageNet top-1 acc.	ADE20K mIoU
None	73.2	32.3
Bidirectional Layer	70.9	33.6
Bidirectional SSM	72.8	33.2
Bidirectional SSM + Conv1d	73.9	35.9

Table 4: Ablation study on the bidirectional design. To ensure a fair comparison, we do not use the class token for each experiment. The default setting for Vim is marked in blue.

Classification strategy	ImageNet top-1 acc.
Mean pool	73.9
Max pool	73.4
Head class token	75.2
Double class token	74.3
Middle class token	76.1

Table 5: Ablation study on the classification design. The default setting for Vim is marked in blue.

process visual sequence with only the forward direction. *Bidirectional Sequence*: During training, we randomly flip the visual sequence. This works like data augmentation. *Bidirectional Block*: We pair the stacked blocks. The first block of each pair processes visual sequence in the forward direction and the second block of each pair processes in the backward direction. *Bidirectional SSM*: We add an extra SSM for each block to process the visual sequence in the backward direction. *Bidirectional SSM + Conv1d*: Based on Bidirectional SSM, we further add a backward Conv1d before the backward SSM (Fig. 2).

As shown in Tab. 4, directly adopting the Mamba block achieves good performance in classification. However, the unnatural unidirectional manner poses challenges in downstream dense prediction. Specifically, the preliminary bidirectional strategy of using Bidirectional Block achieves 7 points lower top-1 accuracy on classification. Yet, it outperforms the vanilla unidirectional Mamba block by 1.3 mIoU on semantic segmentation. By adding extra backward SSM and Conv1d, we achieve superior classification accuracy (73.9 top-1 acc *vs.* 73.2 top-1 acc) and exceptional segmentation superiority (35.9 mIoU *vs.* 32.3 mIoU). We use the strategy of Bidirectional SSM + Conv1d as the default setting in our Vim block.

Classification Design. We ablate the classification design of Vim, benchmarking on ImageNet-1K classification. We study the following classification strategies. *Mean pool*: We adopt mean pooling on the output feature from the last Vim block and perform classification on this pooled feature. *Max pool*: We first adapt the classification head on each token of the visual sequence and then perform max pooling on the sequence to get the classification prediction result. *Head class token*: Following DeiT (Touvron et al., 2021b), we concatenate the class token at the head of the visual sequence and perform classification. *Double class token*: Based on the head class token strategy, we additionally add a class token at the tail of the visual sequence. *Middle class token*: We add a class token at the middle of the visual sequence and then perform classification on the final middle class token.

As shown in Tab. 5, experiments show that the middle class token strategy can fully exploit the recurrent nature of SSM and the central object prior in ImageNet, demonstrating the best top-1 accuracy of 76.1.

5. Conclusion and Future Work

We have proposed Vision Mamba (Vim) to explore the very recent efficient state space model, *i.e.*, Mamba, as generic vision backbones. Unlike prior state space models for vision tasks which use hybrid architecture or equivalent global 2D convolutional kernel, Vim learns visual representation in the sequence modeling manner and does not introduce image-specific inductive biases. Thanks to the proposed bidirectional state space modeling, Vim achieves data-dependent global visual context and enjoys the same modeling power as Transformer, while having lower computation complexity. Benefiting from the hardware-aware designs of Mamba, the inference speed and memory usage of Vim are significantly better than ViTs when processing high-resolution images. Experiment results on standard computer vision benchmarks have verified the modeling power and high efficiency of Vim, showing that Vim has great potential to be the next-generation vision backbone.

In future works, Vim with the bidirectional SSM modeling with position embeddings is suitable for unsupervised tasks such as mask image modeling pretraining and the similar architecture with Mamba enables multimodal tasks such as CLIP-style pretraining. Based on the pretrained Vim weights, exploring the usefulness of Vim for analyzing high-resolution medical images, remote sensing images, and long videos, which can be regarded as downstream tasks, is very straightforward.

Impact Statement

We advance the efficiency of the generic vision backbone. Any societal consequences or impacts that typically relate to work focused on increased efficiency also apply here, as such work necessarily improves the practicality of vision backbone for an array of visual applications with high-resolution input images.

Acknowledgement

This work was partially supported by the National Science and Technology Major Project under Grant No. 2023YFF0905400 and National Natural Science Foundation of China (NSFC) under Grant No. 62276108.

We would like to acknowledge Tianheng Cheng, Yuxin Fang, Shusheng Yang, Bo Jiang, and Jingfeng Yao for their helpful feedback on the draft.

References

Bao, H., Dong, L., Piao, S., and Wei, F. Beit: BERT pre-training of image transformers. In *ICLR*, 2022. URL https://openreview.net/forum?id=p-BhZSz59o4.

Baron, E., Zimerman, I., and Wolf, L. 2-d ssm: A general spatial layer for visual transformers. *arXiv preprint arXiv:2306.06635*, 2023.

Bavishi, R., Elsen, E., Hawthorne, C., Nye, M., Odena, A., Somani, A., and Taşırlar, S. Introducing our multimodal models, 2023. URL https://www.adept.ai/blog/fuyu-8b.

Cai, Z. and Vasconcelos, N. Cascade r-cnn: High quality object detection and instance segmentation. *TPAMI*, 2019.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In *ICLR*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Dai, Z., Liu, H., Le, Q. V., and Tan, M. Coatnet: Marrying convolution and attention for all data sizes. *NeurIPS*, 34, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ding, J., Ma, S., Dong, L., Zhang, X., Huang, S., Wang, W., Zheng, N., and Wei, F. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023.

Ding, X., Zhang, X., Han, J., and Ding, G. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *CVPR*, 2022.

Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., and Guo, B. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *CVPR*, 2022.

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 *ICLR*, 2020.

d'Ascoli, S., Touvron, H., Leavitt, M. L., Morcos, A. S., Biroli, G., and Sagun, L. Convit: Improving vision transformers with soft convolutional inductive biases. In *ICML*, 2021.

Fang, J., Xie, L., Wang, X., Zhang, X., Liu, W., and Tian, Q. Msg-transformer: Exchanging local spatial information by manipulating messenger tokens. In *CVPR*, 2022.

Fang, Y., Wang, W., Xie, B., Sun, Q., Wu, L., Wang, X., Huang, T., Wang, X., and Cao, Y. Eva: Exploring the limits of masked visual representation learning at scale. In *CVPR*, 2023.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Re, C. Hungry hungry hippos: Towards language modeling with state space models. In *ICLR*, 2023. URL https://openreview.net/forum?id=COZDy0WYGg.

Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.-Y., Cubuk, E. D., Le, Q. V., and Zoph, B. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, 2021.

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

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021a.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In *NeurIPS*, 2021b.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. In *NeurIPS*, 2022.

Gupta, A., Gu, A., and Berant, J. Diagonal state spaces are as effective as structured state spaces. In *NeurIPS*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *CVPR*, 2017.

Islam, M. M. and Bertasius, G. Long movie clip classification with state-space video models. In *ECCV*, 2022.

Islam, M. M., Hasan, M., Athrey, K. S., Braskich, T., and Bertasius, G. Efficient movie scene detection using state-space transformers. In *CVPR*, 2023.

Jia, C., Yang, Y., Xia, Y., Chen, Y.-T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H., Li, Z., and Duerig, T. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, 2021.

Kalman, R. E. A new approach to linear filtering and prediction problems. 1960.

Kenton, J. D. M.-W. C. and Toutanova, L. K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *ICLR*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, J., Li, D., Xiong, C., and Hoi, S. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022a.

Li, J., Li, D., Savarese, S., and Hoi, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

Li, Y., Cai, T., Zhang, Y., Chen, D., and Dey, D. What makes convolutional models great on long sequence modeling? In *ICLR*, 2022b.

Li, Y., Mao, H., Girshick, R., and He, K. Exploring plain vision transformer backbones for object detection. In *ECCV*, 2022c.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

Liu, S., Chen, T., Chen, X., Chen, X., Xiao, Q., Wu, B., Kärkkäinen, T., Pechenizkiy, M., Mocanu, D., and Wang, Z. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*, 2022a.

Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., Ye, Q., and Liu, Y. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *CVPR*, 2022b.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR*, 2019.

Ma, J., Li, F., and Wang, B. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *arXiv preprint arXiv:2401.04722*, 2024.

Mehta, H., Gupta, A., Cutkosky, A., and Neyshabur, B. Long range language modeling via gated state spaces. In *ICLR*, 2023. URL https://openreview.net/forum?id=5MkYIYCbva.

Nguyen, E., Goel, K., Gu, A., Downs, G., Shah, P., Dao, T., Baccus, S., and Ré, C. S4nd: Modeling images and videos as multidimensional signals with state spaces. In *NeurIPS*, 2022.

Qin, Z., Yang, S., and Zhong, Y. Hierarchically gated recurrent neural network for sequence modeling. In *NeurIPS*, 2023. URL https://openreview.net/forum?id=P1TCHxJwLB.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces. In *CVPR*, 2020.

Rao, Y., Zhao, W., Zhu, Z., Lu, J., and Zhou, J. Global filter networks for image classification. *Advances in neural information processing systems*, 34:980–993, 2021.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Smith, J. T., De Mello, S., Kautz, J., Linderman, S., and Byeon, W. Convolutional state space models for long-range spatiotemporal modeling. In *NeurIPS*, 2023a.

Smith, J. T., Warrington, A., and Linderman, S. Simplified state space layers for sequence modeling. In *ICLR*, 2023b. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Strudel, R., Garcia, R., Laptev, I., and Schmid, C. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language modelss. *arXiv preprint arXiv:2307.08621*, 2023.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *CVPR*, 2015.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

Tan, M. and Le, Q. Efficientnetv2: Smaller models and faster training. In *ICML*, 2021.

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. In *NeurIPS*, 2021.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021a.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021b.

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 classification with data-efficient training. *TPAMI*, 2022.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2020a.

Wang, J., Yan, J. N., Gu, A., and Rush, A. M. Pretraining without attention. *arXiv preprint arXiv:2212.10544*, 2022.

Wang, J., Zhu, W., Wang, P., Yu, X., Liu, L., Omar, M., and Hamid, R. Selective structured state-spaces for long-form video understanding. In *CVPR*, 2023a.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020b.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

Wang, W., Dai, J., Chen, Z., Huang, Z., Li, Z., Zhu, X., Hu, X., Lu, T., Lu, L., Li, H., et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *CVPR*, 2023b.

Wang, W., Ma, S., Xu, H., Usuyama, N., Ding, J., Poon, H., and Wei, F. When an image is worth 1,024 x 1,024 words: A case study in computational pathology. *arXiv preprint arXiv:2312.03558*, 2023c.

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. Cvt: Introducing convolutions to vision transformers. In *ICCV*, 2021.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., and Sun, J. Unified perceptual parsing for scene understanding. In *ECCV*, 2018a.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., and Sun, J. Unified perceptual parsing for scene understanding. In *ECCV*, 2018b.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Xing, Z., Ye, T., Yang, Y., Liu, G., and Zhu, L. Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation. *arXiv preprint arXiv:2401.13560*, 2024.

Yan, J. N., Gu, J., and Rush, A. M. Diffusion models without attention. *arXiv preprint arXiv:2311.18257*, 2023.

Yang, J., Li, C., Zhang, P., Dai, X., Xiao, B., Yuan, L., and Gao, J. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*, 2021.

Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., and Yan, S. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference*

on computer vision and pattern recognition, pp. 10819–10829, 2022.

Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., and Torralba, A. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 2019.

A. Visualization

GT DeiT-Ti Vim-Ti

Figure 5: Visualization comparison of DeiT-Ti (Touvron et al., 2021b) and our Vim-Ti on the Cascade Mask R-CNN (Cai & Vasconcelos, 2019) framework. Thanks to the long-range context learning of SSM, we can capture the very large object in the image, which the DeiT-Ti counterpart fails to perceive.

B. Additional Setting

Settings for Semantic Segmentation. We conduct experiments for semantic segmentation on the ADE20K (Zhou et al., 2019) dataset. ADE20K contains 150 fine-grained semantic categories, with 20K, 2K, and 3K images for training, validation, and testing, respectively. We choose UperNet (Xiao et al., 2018a) as our base framework. In training, we employ AdamW with a weight decay of 0.01, and a total batch size of 16 to optimize models. The employed training schedule uses an initial learning rate of 6×10^{-5}, linear learning rate decay, a linear warmup of $1,500$ iterations, and a total training of 160K iterations. The data augmentations follow common settings, including random horizontal flipping, random re-scaling within the ratio range $[0.5, 2.0]$, and random photometric distortion. During evaluation, we rescale the image to have a shorter side of 512.

Settings for Object Detection and Instance Segmentation. We conduct experiments for object detection and instance segmentation on the COCO 2017 dataset (Lin et al., 2014). The COCO 2017 dataset contains 118K images for training, 5K images for validating, and 20K images for testing. We use the canonical Cascade Mask R-CNN (Cai & Vasconcelos, 2019) as the base framework. For ViT-based backbones, we apply extra configurations (*e.g.*, interleaved window & global attention) to handle the high-resolution images following ViTDet (Li et al., 2022c). For SSM-based Vim, we directly use it without any modifications. Other training and evaluation settings are just the same. During training, we employ AdamW with a weight decay of 0.1, and a total batch size of 64 to optimize models. The employed training schedule uses an initial learning rate of 1×10^{-4}, linear learning rate decay, and a total training of 380K iterations. The data augmentations use large-scale jitter data augmentation (Ghiasi et al., 2021) to 1024×1024 input images. During evaluation, we rescale the image to have a shorter side of 1024.

C. Extended Comparison on Hierarchical Architecture

To further compare with hierarchical architectures, we propose another variant Hier-Vim by replacing shifted local window attention in SwinTransformer with the proposed global bidirectional SSM. We detail the configuration in Tab. 6

Model	#Blocks	#Channels	Params
Hier-Vim-T	[2, 2, 5, 2]	[96, 192, 384, 768]	30M
Hier-Vim-S	[2, 2, 15, 2]	[96, 192, 384, 768]	50M
Hier-Vim-B	[2, 2, 15, 2]	[128, 256, 512, 1024]	89M

Table 6: Detailed configurations of different variants of Hier-Vim. We provide the number of channels and blocks in 4 stages.

Classification on ImageNet. Following the standard training and validation protocols (Liu et al., 2021; 2024), we compare

Method	image size	#param.	ImageNet top-1 acc.
Swin-T (Liu et al., 2021)	224^2	28M	81.2
FocalTransformer-T (Yang et al., 2021)	224^2	29M	82.2
CVT-21 (Wu et al., 2021)	224^2	32M	82.5
MetaFormer-S35 (Yu et al., 2022)	224^2	31M	81.4
GFNet-H-S (Rao et al., 2021)	224^2	32M	81.5
Hier-Vim-T	224^2	**30M**	**82.5**
Swin-S (Liu et al., 2021)	224^2	50M	83.2
FocalTransformer-S (Yang et al., 2021)	224^2	51M	83.5
MetaFormer-S35 (Yu et al., 2022)	224^2	73M	82.5
GFNet-H-B (Rao et al., 2021)	224^2	54M	82.9
Hier-Vim-S	224^2	**50M**	**83.4**
Swin-B (Liu et al., 2021)	224^2	88M	83.5
FocalTransformer-B (Yang et al., 2021)	224^2	90M	83.8
Hier-Vim-B	224^2	**89M**	**83.9**

Table 7: Comparison with hierarchical architectures on ImageNet-1K validation set.

Hier-Vim with popular hierarchical architectures across tiny, small, and base model sizes in Tab. 7. The results indicate that Hier-Vim outperforms Swin Transformer by 1.3% at the tiny size, 0.2% at the small size, and 0.4% at the base size, demonstrating competitive performance against well-established and highly-optimized modern hierarchical architectures.