AUTOREGRESSIVE PRETRAINING WITH MAMBA IN VISION

Anonymous authors
Paper under double-blind review

ABSTRACT

The vision community has started to build with the recently developed state space model, Mamba, as the new backbone for a range of tasks. This paper shows that Mamba's visual capability can be significantly enhanced through autoregressive pretraining, a direction not previously explored. Efficiency-wise, the autoregressive nature can well capitalize on the Mamba's unidirectional recurrent structure, enabling faster overall training speed compared to other training strategies like mask modeling. Performance-wise, autoregressive pretraining equips the Mamba architecture with markedly higher accuracy over its supervised-trained counterparts and, more importantly, successfully unlocks its scaling potential to large and even huge model sizes. For example, with autoregressive pretraining, a base-size Mamba attains 83.2% ImageNet accuracy, outperforming its supervised counterpart by 2.0%; our huge-size Mamba, the largest Vision Mamba to date, attains 85.0% ImageNet accuracy (85.5% when finetuned with 384×384 inputs), notably surpassing all other Mamba variants in vision. The code will be available soon.

1 INTRODUCTION

In natural language processing (NLP), state space models (SSMs) Gu et al. (2021a;b); Mehta et al. (2022); Gu et al. (2022) demonstrate strong potential for modeling long sequences with linear complexity. Among these, a recent variant, Mamba Gu & Dao (2023), has substantially advanced beyond traditional SSMs by synthesizing the best attributes of selective scanning. This innovation has also catalyzed its rapid adoption within the vision community, leading to its application across diverse visual tasks. These include the design of novel architectures Liu et al. (2024b); Zhu et al. (2024); Huang et al. (2024); Pei et al. (2024); Wang et al. (2024a), applications to segmentation Liu et al. (2024a); Wang et al. (2024b); Xing et al. (2024) and image synthesis Guo et al. (2024).

However, these prior studies are mostly in the setting of supervised visual representation learning. While such trained models exhibit promising results in different visual tasks, they generally suffer from limited transferability and encounter notable difficulties in scaling He et al. (2022); Bao et al. (2022); He et al. (2020); Chen et al. (2020b). For example, as illustrated in Figure 1, attempts to scale the Vision Mamba (Vim) under supervised conditions often lead to either performance plateauing or even training collapse when pushed to very large sizes. These issues, therefore, motivate us to alternatively explore self-supervised visual representation learning with Mamba architectures, a method that has demonstrated notable successes in helping models secure strong and scalable visual representations He et al. (2022); Bao et al. (2022); He et al. (2020); Chen et al. (2020b).

In this paper, we primarily focus on the autoregressive pretraining paradigm for self-supervised visual representation learning, which predicts the next token unidirectionally and autoregressively from the start to the end of the input sequence. This focus is driven by two reasons. First, autoregressive pretraining has already established itself as the de-facto standard in training large language models, with successful applications in various architectures including Transformers and Mamba Dosovitskiy et al. (2020); Radford & Narasimhan (2018); Gu & Dao (2023). The recent literature has also successfully, albeit preliminarily, confirmed its efficacy in the computer vision domain, *e.g.*, helping Vision Transformer (ViT) develop strong and scalable feature representations El-Nouby et al. (2024); Ren et al. (2023a). Secondly, Mamba architectures are inherently well-suited for autoregressive modeling due to their uniquely designed linear attention nature, which methodically constructs token-wise relationships in a strictly progressive and unidirectional manner. This configuration ensures that each

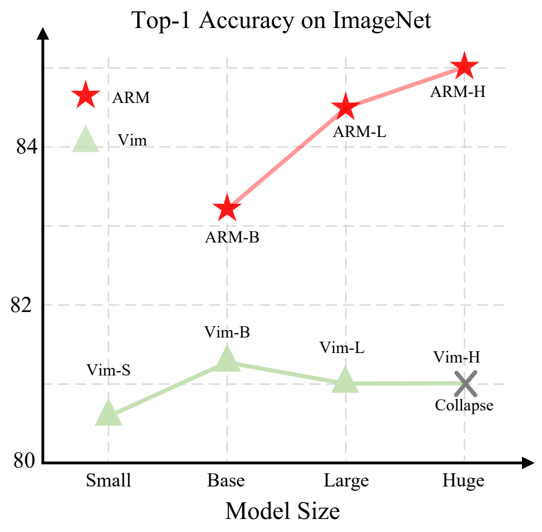

Figure 1: Compared to Vim, our ARM considerably boosts the ImageNet accuracy and, more critically, offers a stronger pathway for scaling up.

token can only attend to its preceding tokens, aligning perfectly with the underlying principles of autoregressive modeling. Additionally, this synergy practically leads to higher overall training efficiency. For example, under the setting of training the base-size Mamba for 300 epochs, autoregressive training requires only ~34 hours (measured by 8×A5000), a ~2× to ~10× improvement in training speed compared to other pretraining strategies (see Table 8 in Sec. 4.4).

Importantly, to further unleash the power of autoregressive visual representation learning with Mamba architectures, we highlight two key recipes for forming input sequences. First, instead of naively taking 16×16 patches as basic units of prediction, we opt for a more strategic approach by grouping spatially neighboring patches to form larger clusters; empirically, we find the cluster size of 64×64 reaches the best performance. Secondly, in our ablation of mapping 2D images into 1D visual sentences with various orderings, we note that vanilla ordering, which simply orders clusters with the row-by-row and forward scan approach, is already an effective choice. We term this method ARM.

Extensive results are provided showing our proposed ARM achieves substantially stronger performance. As shown in Figure 1, ARM helps our base-size model attain 83.2% ImageNet accuracy, outperforming the supervised counterpart by 2.0% and achieves 85.2% Top-1 accuracy with the input resolution of 448×448. Moreover, ARM enables the training of the first successful huge-size model (ARM-H), marking it as the largest vision Mamba model to date. Specifically, ARM-H achieves an impressive 85.0% ImageNet accuracy, significantly outperforming all previous Mamba variants. Additionally, ARM also improves the performance on out-of-domain datasets by a large margin: ARM-B outperforms supervised Vim-B by 4.4% on ImageNet-A, 2.9% on ImageNet-R, and 3.3% on ImageNet-S.

2 Related Work

State space model. The state space model (SSM) Gu et al. (2021a) stands as a novel alternative to Transformers for long-range dependency modeling with linear complexity. Linear attention Katharopoulos et al. (2020); Choromanski et al. (2020); Peng et al. (2021) recurrently approximates self-attention via a softmax-free attention matrix with linear complexity, which can be viewed as a degenerate linear SSM. The Structured State-Space Sequence (S4) model Gu et al. (2021a) computes more efficiently than prior approaches while preserving their theoretical strengths based on a new parameterization. S5 Smith et al. (2022) extends S4 by adding multi-input multi-output (MIMO) SSM and efficient parallel scan. RWKV Peng et al. (2023) is a recent RNN with its key

"WKV" components that operate similarly to a system with two SSMs. Its updated version Peng et al. (2024) incorporates state expansion and input-dependent gating for more flexible sequence modeling. Following this, Mamba Gu & Dao (2023) proposes a data-dependent SSM layer with hidden state expansion and builds a generic language model backbone, which performs comparably to transformers at various sizes and enjoys linear scaling in sequence length. This work focuses on Mamba in vision, aiming to enhance it via autoregressive visual pretraining.

Mamba in vision. The successful application of Mamba in NLP has inspired its adoption in vision applications. Vision Mamba (Vim) Zhu et al. (2024) utilizes Vim blocks composed of pure Mamba layers: each Vim block leverages both forward and backward scans to model bidirectional representations and mitigate the direction-sensitive problem in Mamba. Alternatively, Vmamba Liu et al. (2024b) employs Visual State Space (VSS) blocks that integrate both Mamba and 2D convolution layers, supported by a pyramid architecture akin to the Swin Transformer Liu et al. (2021): each VSS block first models 2D local information via 2D depth-wise convolution as the token mixer, followed by a CrossScan Module that processes 2D global information both horizontally and vertically. Mamba-ND Li et al. (2024) further expands Mamba's capabilities to multi-dimensional data, including images and videos. LocalMamba Huang et al. (2024) splits the input image into several local windows and performs SSM in various directions within these windows, enhancing local processing. EfficientVMamba Pei et al. (2024) introduces an efficient 2D scanning technique using atrous sampling on feature map patches to reduce computational demands. Compared to these newly designed Mamba architectures, ours is *less novel*, which closely follows the design of ViT, but substituting the self-attention with the Mamba module. With this *naive* Mamba architecture, our main focus is to show autoregressive pretraining can enhance its visual capabilities.

Self-supervised visual representation learning. Self-supervised visual representation learning aims to learn strong and transferable representations without labels, including contrastive learning Chen et al. (2020c); He et al. (2020); Chen et al. (2021; 2020b), position prediction Zhai et al. (2022), masked image modeling He et al. (2022); Bao et al. (2022); Ren et al. (2023b), *etc*. This paper focuses on autoregressive pretraining, which is highly successful in NLP but still less explored in computer vision. iGPT Chen et al. (2020a) is the first work to introduce Generative Pretrained Transformer to vision and highlights the potential of autoregressive pretraining as a general self-supervised visual representation learning strategy. SAIM Qi et al. (2023) and RandSAC Hua et al. (2022) further enhance autoregressive pretraining, achieving performance on par with MAE He et al. (2022) by utilizing the ViT architecture and a stochastic sequence permutation strategy. D-iGPT Ren et al. (2023a) slightly modifies the learning objective to predict not only the next token but also visible tokens. AIM El-Nouby et al. (2024) demonstrates that, with autoregressive pretraining, ViT scales effectively with increased model capacity and data quantity. Different from these prior works, which focus on Transformer architecture, we provide the first study of exploring autoregressive visual pretraining with Mamba architectures.

3 METHOD

3.1 MAMBA PRELIMINARIES

The Mamba architecture inherits from state space sequence models Gu et al. (2021a), which models a 1-D function or sequence $x(t) \in \mathbb{R} \rightarrow y(t) \in \mathbb{R}$ at time t via expanded hidden states $h_t \in \mathbb{R}^N$. The hidden state is evolved through time driven by parameters $\mathbf{A}, \mathbf{B}, \mathbf{C}$ following linear ordinary differential equations (ODEs):

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

To discretize parameters in this continuous system, a common solution is to introduce a time scale parameter $\mathbf{\Delta}$ to transform continuous $\mathbf{A}, \mathbf{B}$ to discrete $\overline{\mathbf{A}}, \overline{\mathbf{B}}$ using zero-order hold (ZOH) model Oppenheim et al. (1997):

$$\begin{aligned} \overline{\mathbf{A}} &= \exp(\mathbf{\Delta A}), \\ \overline{\mathbf{B}} &= (\mathbf{\Delta A})^{-1}(\exp(\mathbf{\Delta A}) - \mathbf{I}) \cdot \mathbf{\Delta B}. \end{aligned} \tag{2}$$

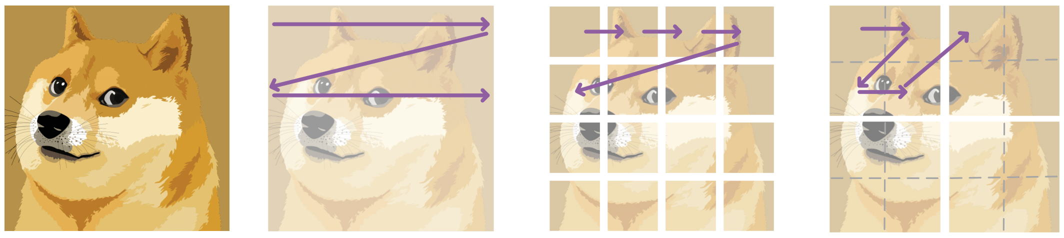

Figure 2: Different prediction units in the autoregressive modeling.

By applying such transformation, we can rewrite Eq. 1 as:

$$\begin{aligned} h'(t) &= \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, \\ y_t &= \mathbf{C}h_t. \end{aligned} \tag{3}$$

We then employ a matrix $\overline{\mathbf{K}}$ for fast computation:

$$\begin{aligned} \overline{\mathbf{K}} &= (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}}\overline{\mathbf{B}}, ..., \mathbf{C}\overline{\mathbf{A}}^k\overline{\mathbf{B}}, ...), \\ \mathbf{y} &= \mathbf{x} * \overline{\mathbf{K}}, \end{aligned} \tag{4}$$

where $k \in [0, L)$ and L is the input sequence length. We also have $\mathbf{y} = \{y_1, ..., y_L\}$, $\mathbf{x} = \{x_1, ..., x_L\}$, while $\overline{\mathbf{K}} \in \mathbb{R}^L$ can be regarded as the convolutional kernel. Note this computing structure allows Mamba to model the input sequence that perfectly matches the unidirectional, next-word prediction in autoregressive modeling.

3.2 Autoregressive Pretraining

We first briefly revisit autoregressive pretraining in NLP. Then, we shift our attention to autoregressive pretraining with mamba in vision, including the prediction unit and prediction order design. Lastly, we present the model variants.

3.2.1 Autoregressive Pretraining in NLP

Autoregressive pretraining models the probability of the next word one by one given a corpus $\mathcal{U} = \{u_1, ..., u_n\}$. This can be formulated as:

$$p(u) = \prod_{i=1}^{n} p(u_i | u_1, ..., u_{i-1}, \Theta) \tag{5}$$

Here, autoregressive pertaining computes the likelihood of each word u_i based on the context of all preceding words from u_1 to u_{i-1} and minimizes the negative log-likelihood:

$$\mathcal{L} = -log\ p(u) \tag{6}$$

This strategy plays a fundamental role in training large language models like ChatGPT Brown et al. (2020) and GPT-4 OpenAI (2023) in NLP.

3.2.2 Autoregressive Pretraining with Mamba in Vision

Prediction unit. Transitioning from 1D sentences to 2D images introduces the challenge of defining a suitable autoregressive prediction unit. We start with the vanilla strategy presented in iGPT Chen et al. (2020a) which considers each individual pixel as the prediction unit, as illustrated in Figure 2(b). For an image $X = \{p_1, ..., p_n\}$, our objective is to minimize the loss function:

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^{n-1} l(f([p_1, ..., p_i]), p_{i+1}), \\ l(\hat{y}, y) &= |\hat{y} - y|^2. \end{aligned} \tag{7}$$

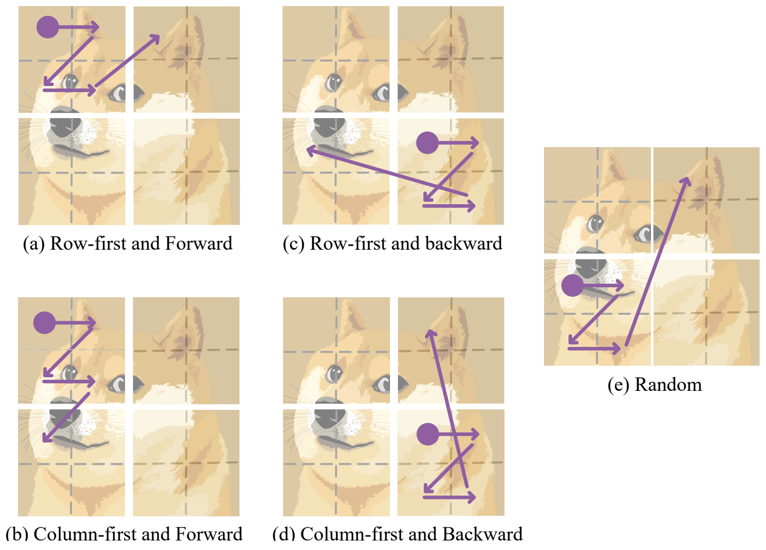
(a) Row-first and Forward
(c) Row-first and backward
(e) Random
(b) Column-first and Forward
(d) Column-first and Backward

Figure 3: Different prediction orderings of a visual sentence.

Here $f(\cdot)$ denotes the Mamba model, and p_i represents the i_{th} pixel of the image. This pixel-based approach, while straightforward, imposes significant computational demands, particularly for high-resolution images. Therefore, as shown in the original iGPT paper Chen et al. (2020a), this constraint necessitates the use of low-resolution images for computationally feasible autoregressive pretraining.

Patchifying Dosovitskiy et al. (2020) images into non-overlapped regions and then mapping them into visual tokens can address this computation challenge. For example, with an image size of 224×224, the sequence length would reduce significantly from 50,176 in the iGPT framework to just 196 patches with the 16×16 patchifying operation. Intuitively, shifting the prediction unit from pixels Chen et al. (2020a) to patches Dosovitskiy et al. (2020); Zhu et al. (2024); El-Nouby et al. (2024), as shown in Figure 2(c), adjusts the autoregressive input to $X = \{P_1, ..., P_n\}$:

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{n-1} l(f([P_1, ..., P_i]), P_{i+1}), \\ l(\hat{y}, y) &= |\hat{y} - y|^2.\end{aligned} \tag{8}$$

Here $P_i \in \mathcal{R}^{16\times16}$ is the i_{th} patch. Moreover, to encapsulate the 2D spatial information at the token level, we propose grouping spatially adjacent patches into larger clusters to serve as the prediction unit, illustrated in Figure 2(d). The clustered input $X = \{c_1, ..., c_n\}$ aims to be optimized by:

$$\begin{aligned}\mathcal{L}_{\text{ARM}} &= \sum_{i=1}^{n-1} l(f([c_1, ..., c_i]), c_{i+1}), \\ l(\hat{y}, y) &= |\hat{y} - y|^2.\end{aligned} \tag{9}$$

Here, each $c_i \in \mathcal{R}^{H_c \times W_c}$ is a cluster formed by grouping $\frac{H_c}{16} \times \frac{W_c}{16}$ patches. Our ablation studies (Section 4.4, Table 4) show that using clusters as prediction targets significantly enhances performance compared to the use of individual pixels or patches. Next, we explore the strategies for sequencing these clusters into a coherent visual sentence.

Prediction order. Unlike the 1D sentences in NLP, which inherently have a clear sequence order for autoregressive modeling, we hereby explore four different prediction orders when projecting 2D images into 1D visual sentences, *e.g.*, how these clusters should be arranged given a cluster size of s, with $\frac{W}{s}$ clusters per row and $\frac{H}{s}$ clusters per column. We hereby explore four primary prediction orders: 1) *Row-first and forward* orders the clusters row by row, processing from the first to the last cluster within each row sequentially, as depicted in Figure 3(a). 2) *Row-first and backward* similarly orders the clusters row by row but inverts the processing direction, starting with the last

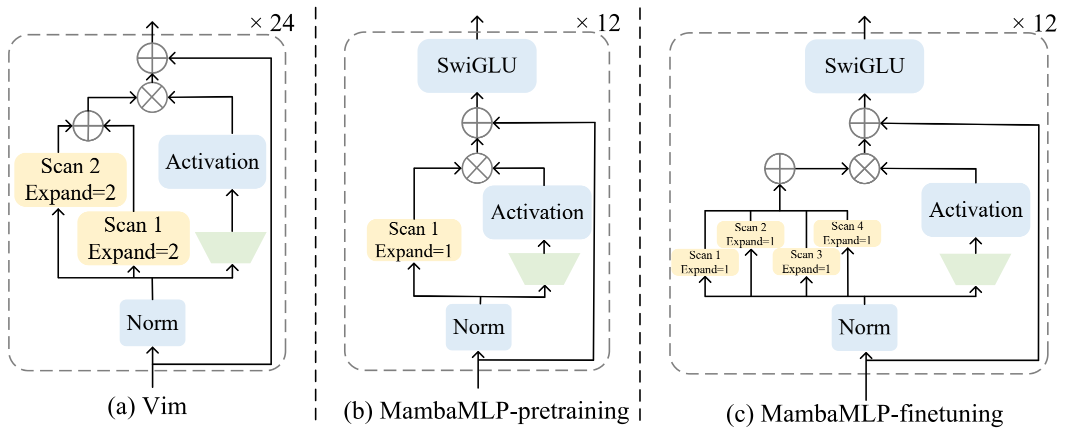

Figure 4: The comparison of block architectures between Vim, and MambaMLP in pretraining and in finetuning.

cluster and moving to the first within each row, illustrated in Figure 3(b). 3) *Column-first and forward* organizes the clusters column by column, processing sequentially within each column from top to bottom, shown in Figure 3(c). 4) *Column-first and backward* similarly sequences the clusters column by column but starts with the bottom-most cluster, moving upwards, as seen in Figure 3(c). To consider an approach free from pre-defined sequential biases, we also experimented with a *Random* permutation Yang et al. (2019) of cluster order, visualized in Figure 3(e).

Detailed empirical comparisons of these four predefined orders alongside the random order are presented in Section 4.4. Our findings reveal that while the predefined orders exhibit minimal differences in performance, employing a random order leads to severe performance degradation. Consequently, the straightforward and effective *row-first and forward order* (Figure 3(a)) is adopted as our standard ordering strategy for autoregressive modeling.

3.3 MambaMLP

We hereby introduce our newly developed MambaMLP blocks. Specifically, our MambaMLP block uses Mamba as the token mixer and the multi-layer perceptron (MLP) as the channel mixer, drawing inspiration from the self-attention block in Transformer Dosovitskiy et al. (2020); Vaswani et al. (2017). Note that the configuration of the MambaMLP block varies between pretraining and fine-tuning phases to cater to their different requirements. During pretraining, as illustrated in Figure 4(b), the MambaMLP block contains the Mamba layer with only 1 scan Liu et al. (2024b) to match the uni-directional modeling manner in autoregressive pertaining; while in finetuning (displayed in Figure 4(c)), the block is then adapted to contains the Mamba layer with 4 scans, thus enabling bi-directional modeling of global information analogous to that in Vmamba Liu et al. (2024b). The other architectural components in the pretraining and the finetuning stay the same: the block utilizes SwiGLU Touvron et al. (2023) as the MLP layer, and the $expand$ is set to 1 to enhance scanning efficiency. Additionally, we provide a visual comparison between our MambaMLP block and the Vim block in Figure 4. We can see that the Vim block contains Mamba layers with 2 scans Liu et al. (2024b) for bi-directional global information processing and has no MLP layer, and the $expand$ of each scan is set to 2. Practically, this larger $expand$ in each scan results in higher performance but slower inference speeds.

By stacking multiple MambaMLP blocks and training with our autoregressive strategy developed in Section 3.2.2, we name the resulting model ARM. As detailed in Table 1, ARM is designed to match the depth and width of ViT in its base and large configurations. For the huge model size, ARM adopts the structure of AIM-600M El-Nouby et al. (2024), which is wider but less deep compared to ViT-H, balancing performance and computational efficiency. In the next section, we will extensively validate the efficacy of ARM.

Table 1: The configuration of different architecture variants.

Model	Block	Width	Depth	Param.(M)
ViT-B	(Attention+MLP)	768	12	86
Vim-B	Mamba	768	24	98
ARM-B	(Mamba+MLP)	768	12	85
ViT-L	(Attention+MLP)	1024	24	307
Vim-L	Mamba	1024	48	340
ARM-L	(Mamba+MLP)	1024	24	297
ViT-H	(Attention+MLP)	1280	32	632
Vim-H	Mamba	1536	48	755
ARM-H	(Mamba+MLP)	1536	24	662

4 EXPERIMENT

4.1 IMPLEMENTATION DETAILS

Pretraining. We pretrain ARM using the ImageNet-1K dataset Deng et al. (2009). Specifically, ARM-B and ARM-L are pre-trained for 1600 epochs, and ARM-H is pre-trained for 800 epochs. We use a batch size of 2048/1024/512 for ARM-B/L/H, respectively, and a learning rate of lr = 1.5e-4$\times\frac{\text{batchsize}}{256}$. We adopt a cosine decay schedule with a warm-up for 5 epochs. We adopt the AdamW Loshchilov & Hutter (2019) optimizer with a weight decay of 0.05. We use random resized cropping and random horizontal flipping. The pretraining input size is set to 192×192.

Finetuning. Following pretraining, we finetune the ARM models on the ImageNet classification task. Specifically, we finetune all models for 100 epochs with a batch size of 1024, with the input size set at 224×224. We use the same data augmentation as MAE He et al. (2022). We adopt AdamW as an optimizer, and the peak learning rate is lr=5e-4$\times\frac{\text{batchsize}}{256}$ with a cosine decay schedule and a warm-up for 5 epochs. Additionally, we employ the exponential moving average (EMA) Izmailov et al. (2018) for stronger performance.

Further, we evaluate model robustness on various out-of-domain ImageNet variants, including natural adversarial examples (ImageNet-A Hendrycks et al. (2021b)), semantic shifts (ImageNet-R Hendrycks et al. (2021a)), image sketches (ImageNet-S Wang et al. (2019)), ImageNet-V2 Recht et al. (2019), and ImageNet-Real Beyer et al. (2020).

4.2 MAIN RESULTS

In Table 2, we compare our ARM with convolution-based RegNet Radosavovic et al. (2020), Attention-based ViT, and different Mamba architectures in vision. For the base-size model, our ARM achieves 83.2% accuracy, making a substantial 2.0% improvement over its supervised MambaMLP counterpart. Additionally, we note that ARM outperforms Vim by 2.0%, and is the only Mamba architecture that attains stronger performance than convolution-based RegNetY-16G (*i.e.*, by 0.3%). Further enhancements are observed when ARM-B is finetuned with increased input sizes of 384$\times$384 and 448$\times$448 with the patchify stride of 8, where performance improves to 84.2% and 85.2%, respectively. We also report the comparison to VMamba-B, which takes a hybrid architecture: When configured with inputs of 224×224, ARM-B slightly underperforms VMamba-B by 0.7% but enjoys a much faster throughput, *i.e.*, ~4$\times$ faster; ARM-B with the inputs of 384$\times$384 outperforms Vmamba-B by 0.3% and still maintains a faster throughput, *i.e.*, 440 imgs/s *vs.* 315 imgs/s.

Next, we scale the Mamba architectures to much larger model sizes. First, we observe that Mamba-based Vim sees a performance dip with the large size and fails to train stably at the huge size. This observation suggests that these prior Mamba-based architectures grapple with scaling challenges. Contrarily, ARM models excel in scalability — ARM-L achieves an accuracy of 84.5%, marking a 3.5% improvement over Vim-L, and ARM-H sets a new benchmark for the largest Mamba architecture in vision to date by reaching 85.0% accuracy. Moreover, by tuning ARM at a larger resolution of 384$\times$384, further leveraging the model's capacity to handle long sequences at a linear complexity, we

Table 2: Performance comparison on ImageNet-1K. Throughputs are measured with an A5000 GPU. † denotes we extend the training of Vim to the large-size model, using its original GitHub repo. ‡ indicates the stride is 8. Hybrid architectures are in Gray.

Model	Token Mixer	Image Size	Param. (M)	Throughputs (imgs/s)	Top-1 (%)
Base-size models					
RegNetY-16G	2D Conv.	224^2	84	870	82.9
DeiT-B	Attention	224^2	21	1073	81.2
Vim-B†	Mamba	224^2	98	890	81.2
MambaMLP-B	Mamba	224^2	85	1301	81.2
VMamba-B	Mamba+2D Conv.	224^2	89	315	83.9
ARM-B	Mamba	224^2	85	1301	83.2
ARM-B	Mamba	384^2	85	440	84.2
ARM-B ‡	Mamba	448^2	85	86	85.2
Large-size models					
Vim-L†	Mamba	224^2	340	345	81.0
MambaMLP	Mamba	224^2	297	445	81.4
ARM-L	Mamba	224^2	297	445	84.5
ARM-L	Mamba	384^2	297	154	85.1
Huge-size models					
Vim-H†	Mamba	224^2	755	211	collapsed
ARM-H	Mamba	224^2	662	275	85.0
ARM-H	Mamba	384^2	662	94	85.5

Table 3: Robustness and Generalization evaluation on out-of-domain datasets.

Method	IN-1K ↑	IN-V2 ↑	IN-Real ↑	IN-Adv.↑	IN-Ren.↑	IN-Ske.↑
Vim-S Zhu et al. (2024)	80.6	69.4	86.0	20.3	45.8	33.4
Vim-B Zhu et al. (2024)	81.2	70.0	86.2	27.5	46.0	33.9
ARM-B	83.2	72.3	88.0	31.9	48.9	37.2
Vim-L Zhu et al. (2024)	81.0	69.8	86.0	27.9	44.7	31.8
ARM-L	84.5	74.0	88.6	41.4	52.1	39.2
ARM-H	85.0	75.6	89.2	42.3	53.2	40.5

observe additional gains: a 0.6% increase with ARM-L and a 0.5% increase with ARM-H. Notably, ARM-H attains the best Mamba accuracy of 85.5% on ImageNet classification.

4.3 Robustness and Generalization

We report the robustness evaluation of Mamba architectures in Table 3. We can observe that ARM consistently shows much stronger robustness than the supervised Vim by, *e.g.*, ARM-B exhibits improvements ranging from 1.8% to 4.4% over supervised Vim-B across these robustness benchmarks. More impressively, ARM-L extends these gains even further, showing enhancements ranging between 2.6% and 7.4% when compared to supervised Vim-L. In addition, ARM-H, our largest model variant, not only continues this trend but also shows an average performance superiority of 1.1% over ARM-L, reaffirming the efficacy of scaling up the model size on enhancing robustness.

4.4 Ablation Study

This section provides different ablations on ARM. Unless otherwise specified, all ablation studies are performed on ARM-B under 300 epochs pretraining.

Table 4: Ablation on the number of predictions units.

Num of Prediction unit	Cluster size	Top-1 (%)
0 (Supervised)	N/A	81.2
144 (iGPT)	1×1 (Pixel)	79.8
4	96×96	82.0
9	64×64	82.5
16	48×48	82.2
36	32×32	81.9
144	16×16	81.7

Table 5: Ablation on prediction orders.

Order	Direction	Top-1 (%)
Row-first	Forward	82.5
Row-first	Backward	82.3
Column-first	Forward	82.5
Column-first	Backward	82.4
Random	Random	81.5

4.4.1 Number of prediction units.

Table 4 reports the ablation on the number of prediction units. We start from the cluster size equal to the patch size (*i.e.*, each cluster contains only one patch), resulting in a total of 144 prediction units. We note that, even with this vanilla setup, autoregressive pretraining successfully helps MambaMLP improve performance from 81.2% (via supervised training) to 81.7%. Then, we gradually group multiple patches into one cluster, thereby reducing the total number of prediction units. We note that the performance first increases and then decreases — the best performance is achieved when the number of the prediction units is set to 9, corresponding to a cluster size of 64×64. Specifically, this setup provides a performance improvement of 1.3% over the supervised counterpart and 0.8% over the vanilla autoregressive pretrained counterpart (*i.e.*, with a cluster size of 144). We also report the comparison to MambaMLP trained under the iGPT-style autoregressive pretraining — with the input image size at 144×144 and setting per pixel as the prediction unit, it underperforms our best setup by 2.7% (*i.e.*, 79.8% *vs*. 82.5%).

4.4.2 Prediction Order.

As shown in Table 5, we find different pre-defined orders only lead to minor performance variances. For example, both row-first and column-first forward prediction orders achieve an identical performance of 82.5%; even the least favorable case, where the prediction order was row-first and backward, only underperforms the best case by 0.2%. Nonetheless, interestingly, if we do not predefine the prediction order and pick a random permutation, the performance significantly drops to 81.5%.

4.4.3 Decoder Design.

Our exploration into decoder design is summarized in Table 6. We first focus on the design of *decoder depth*, finding that increasing the depth up to 4 progressively enhanced performance up to 82.5%; further increasing the decoder depth to 8 sees a performance saturation. With this 4-layer decoder setup, we next study the width of the decoder. By ablating these three options {384, 512, 1024}, we empirically observe that setting the decoder depth to 512 yields optimal accuracy.

4.4.4 Prediction targets.

We hereby explore different prediction targets for our ARM. By default, we use per-patch normalized pixels with mean square error (MSE) loss. For comparison, we ablate it against two setups: 1) unnormed pixels with MSE loss, and 2) discretized tokens of the patches derived from dVAE Bao et al. (2022) with cross-entropy loss. The results, presented in Table 7, show that employing normalized pixels as the target with MSE loss yields the best performance, achieving an accuracy of 82.5%. Comparatively, this configuration outperforms the model using discrete tokens from dVAE by 0.3% and the model leveraging unnormed pixels which trailed by 0.6%.

4.4.5 Pretraining paradigm.

As shown in Table 8, we evaluate different pretraining paradigms, including contrastive learning Chen et al. (2021), MAE He et al. (2022), and our ARM. Firstly, we note that all pretraining methods result in performance gains over the supervised counterpart, demonstrating the benefits of self-supervised visual pretraining on Mamba architectures. However, using MAE or contrastive learning, the performance is only moderately improved by 0.4% and 0.2%, respectively, over the supervised

Table 6: Ablation on decoder designs.

Dec. Depth	Dec. Width	Top-1 (%)
1	512	82.1
2	512	82.4
4	512	82.5
8	512	82.5
4	384	82.3
4	512	82.5
4	1024	82.2

Table 7: Ablation on prediction targets.

Targets	Top-1 (%)
dVAE Bao et al. (2022)	82.2
Pixel He et al. (2022)	81.9
Normed Pixel	82.5

Table 8: Comparison of architecture and pretraining paradigms. FPS represents the inference speed after supervised finetuning of the model. The † symbol indicates that Vim, when subjected to contrastive learning, experiences poor performance, potentially due to mode collapse.

Architecture	Pretraining paradigm	Training Cost (h) ↓	FPS (imgs/s) ↑	Top-1 (%)
MambaMLP	Supervised	110	1330	81.2
MambaMLP	Contrastive	330	1330	81.4
MambaMLP	MAE	70	1330	81.6
MambaMLP	ARM	34	1330	82.5
Vim	Supervised	165	923	81.2
Vim	Contrastive	510	923	80.2†
Vim	MAE	106	923	81.4
Vim	ARM	57	923	82.2

baseline. In contrast, our ARM achieves significant improvements of 1.3% over the supervised baseline, as well as achieves higher accuracy than both contrastive learning and MAE. Additionally, in terms of efficiency, ARM requires just 34 hours of pretraining, cutting the training duration in half compared to MAE, which is already noted for its relatively low pretraining demands.

4.4.6 Architecture design.

Exploring further into architectural impacts, Table 8 (from the 5th row to the 8th row) presents our investigation into whether Vim, another variant within the Mamba architecture, benefits from autoregressive pretraining. Results indicate a positive response as ARM-trained Vim reaches an 82.2% accuracy on ImageNet, marking a 1.0% improvement over its supervised-only counterpart. Contrastingly, other pretraining paradigms did not fare as well for Vim: when subjected to contrastive learning, Vim experiences training instability, falling below the supervised baseline; MAE pretraining on Vim only slightly improved over the supervised method, with a marginal gain of 0.2%. These results further support the effectiveness of ARM in pretraining Mamba in Vision.

As a side note, it is important to highlight that although Vim's performance improves with ARM pretraining, it operates ~45% slower during inference compared to MambaMLP. Additionally, MambaMLP incurs only ~66% of the training cost required for pretraining Vim under the ARM framework. These points underscore the superior efficiency of our default ARM framework.

5 Conclusion

In this study, we introduced a novel autoregressive visual pretraining strategy tailored for Mamba architectures, known as ARM. This approach enhances pretraining efficiency and effectiveness by strategically treating groups of spatially neighboring image patches as prediction units. Through our method, we have significantly improved the scalability and benchmark performance of Mamba-based models, setting new standards in their operational functionality. We hope this work can lay a strong foundation for future explorations and potential expansions in the usage of autoregressive pretraining strategies for Mamba architectures within the vision community.

References

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022.

Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICLR*, 2020b.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *preprint arXiv:2003.04297*, 2020c.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *ArXiv*, abs/2104.02057, 2021.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

Alaaeldin El-Nouby, Michal Klein, Shuangfei Zhai, Miguel Angel Bautista, Alexander Toshev, Vaishaal Shankar, Joshua M Susskind, and Armand Joulin. Scalable pre-training of large autoregressive image models. *arXiv preprint arXiv:2401.08541*, 2024.

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

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

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

Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Re. How to train your hippo: State space models with generalized orthogonal basis projections. In *ICLR*, 2022.

Hang Guo, Jinmin Li, Tao Dai, Zhihao Ouyang, Xudong Ren, and Shu-Tao Xia. Mambair: A simple baseline for image restoration with state-space model. *arXiv preprint arXiv:2402.15648*, 2024.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021b.

Tianyu Hua, Yonglong Tian, Sucheng Ren, Michalis Raptis, Hang Zhao, and Leonid Sigal. Self-supervision through random segments with autoregressive coding (randsac). In *ICLR*, 2022.

Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan. *arXiv preprint arXiv:2403.09338*, 2024.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

Shufan Li, Harkanwar Singh, and Aditya Grover. Mamba-nd: Selective state space modeling for multi-dimensional data. *arXiv preprint arXiv:2402.05892*, 2024.

Jiarun Liu, Hao Yang, Hong-Yu Zhou, Yan Xi, Lequan Yu, Yizhou Yu, Yong Liang, Guangming Shi, Shaoting Zhang, Hairong Zheng, et al. Swin-umamba: Mamba-based unet with imagenet-based pretraining. *arXiv preprint arXiv:2402.03302*, 2024a.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024b.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL `https://api.semanticscholar.org/CorpusID:257532815`.

Alan V Oppenheim, Alan S Willsky, Syed Hamid Nawab, and Jian-Jiun Ding. *Signals and systems*. Prentice hall Upper Saddle River, NJ, 1997.

Xiaohuan Pei, Tao Huang, and Chang Xu. Efficientvmamba: Atrous selective scan for light weight visual mamba. *arXiv preprint arXiv:2403.09977*, 2024.

Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *EMNLP*, 2023.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, et al. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

Yu Qi, Fan Yang, Yousong Zhu, Yufei Liu, Liwei Wu, Rui Zhao, and Wei Li. Exploring stochastic autoregressive image modeling for visual representation. In *AAAI*, 2023.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019.

Sucheng Ren, Zeyu Wang, Hongru Zhu, Junfei Xiao, Alan Yuille, and Cihang Xie. Rejuvenating image-gpt as strong visual representation learners. *arXiv preprint arXiv:2312.02147*, 2023a.

Sucheng Ren, Fangyun Wei, Zheng Zhang, and Han Hu. Tinymim: An empirical study of distilling mim pre-trained models. In *CVPR*, 2023b.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Feng Wang, Jiahao Wang, Sucheng Ren, Guoyizhe Wei, Jieru Mei, Wei Shao, Yuyin Zhou, Alan Yuille, and Cihang Xie. Mamba-r: Vision mamba also needs registers. *arXiv preprint arXiv:2405.14858*, 2024a.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *NeurIPS*, 2019.

Ziyang Wang, Jian-Qing Zheng, Yichi Zhang, Ge Cui, and Lei Li. Mamba-unet: Unet-like pure visual mamba for medical image segmentation. *arXiv preprint arXiv:2402.05079*, 2024b.

Zhaohu Xing, Tian Ye, Yijun Yang, Guang Liu, and Lei Zhu. Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation. *arXiv preprint arXiv:2401.13560*, 2024.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

Shuangfei Zhai, Navdeep Jaitly, Jason Ramapuram, Dan Busbridge, Tatiana Likhomanenko, Joseph Yitan Cheng, Walter Talbott, Chen Huang, Hanlin Goh, and Joshua Susskind. Position prediction as an effective pretraining strategy. *arXiv preprint arXiv:2207.07611*, 2022.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.