Spectral State Space Model for Rotation-Invariant Visual Representation Learning

Sahar Dastani[1,2*] Ali Bahri[1] Moslem Yazdanpanah[1] Mehrdad Noori[1]
David Osowiechi[1] Gustavo Adolfo Vargas Hakim[1] Farzad Beizaee[1]
Milad Cheraghalikhani[1] Arnab Kumar Mondal[2,3†] Herve Lombaert[2,4]
Christian Desrosiers[1]
[1]LIVIA, ILLS, ÉTS Montréal, Canada, [2]Mila - Quebec AI Institute,
[3]McGill University, [4]Polytechnique Montreal

Abstract

State Space Models (SSMs) have recently emerged as an alternative to Vision Transformers (ViTs) due to their unique ability of modeling global relationships with linear complexity. SSMs are specifically designed to capture spatially proximate relationships of image patches. However, they fail to identify relationships between conceptually related yet not adjacent patches. This limitation arises from the non-causal nature of image data, which lacks inherent directional relationships. Additionally, current vision-based SSMs are highly sensitive to transformations such as rotation. Their predefined scanning directions depend on the original image orientation, which can cause the model to produce inconsistent patch-processing sequences after rotation. To address these limitations, we introduce Spectral VMamba, a novel approach that effectively captures the global structure within an image by leveraging spectral information derived from the graph Laplacian of image patches. Through spectral decomposition, our approach encodes patch relationships independently of image orientation, achieving patch traversal rotation invariance with our Rotational Feature Normalizer (RFN) module. Our experiments on classification tasks show that Spectral VMamba outperforms the leading SSM models in vision, such as VMamba, while maintaining invariance to rotations and a providing a similar runtime efficiency. The implementation is available at: https://github.com/Sahardastani/spectral_vmamba.git.

1. Introduction

Visual representation learning is fundamental to computer vision, where precise feature extraction is essential for tasks such as image classification, detection, and segmentation. The ability to extract meaningful patterns directly impacts performance and unlocks deeper insights into visual tasks. Significant advances in representation learning have been driven by the development of Convolutional Neural Networks (CNNs) [14, 16, 24, 34, 36] and Vision Transformers (ViTs) [7, 23, 38, 44]. Although CNNs are effective at capturing local patterns with linear scaling, they struggle to

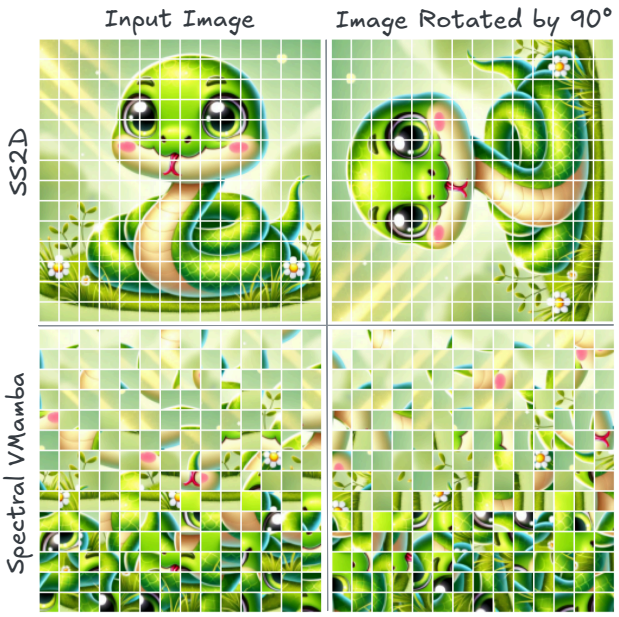

Figure 1. Effect of image rotation on patch processing in VMamba and Spectral VMamba networks. In VMamba networks (first row), patches are traversed using predefined horizontal and vertical scanning routes. As a result, rotating the image by 90° significantly changes the sequence in which patches are processed. In contrast, Spectral VMamba (second row) organizes patches using spectral information derived from a graph laplacian of the image patches. As shown in the second row, Spectral VMamba consistently processes background patches before snake patches in both the original and rotated images. The snake image in this analysis is AI-generated.

*correspondence: sahar.dastani-oghani.1@ens.etsmtl.ca
†currently at Apple

efficiently model global relationships. ViTs address this limitation by introducing self-attention mechanisms, which capture global dependencies by analyzing relationships between all input elements [7, 39]. However, the quadratic complexity of ViTs poses challenges for large-scale data processing, as computational costs increase with input size.

State Space Models (SSMs) [8, 11, 35] have recently gained attention as a compelling alternative to traditional architectures, offering global receptive fields with linear scalability. Inspired by the achievements of Mamba [10] in language modeling, recent studies [13, 21, 33, 47] have explored the application of linearly complex global receptive fields to the visual domain. However, unlike textual data, image pixels do not have a sequential dependency. Building on the parallelized selective scan operation in Mamba, known as S6 [10], which processes sequential one-dimensional data, VMamba [21] introduced 2D Selective Scan (SS2D), a novel four-way scanning mechanism tailored for spatial domain traversal. Vision Mamba (ViM) [47] further enhanced sequence modeling by employing a bidirectional processing strategy that sums both forward and reverse passes. This process enables the model to capture comprehensive dependencies across an image. Expanding upon these approaches, Multi-Scale VMamba (MSVMamba) [33] incorporates a multiscale 2D scanning technique, which processes both original and down-sampled feature maps. This strategy improves capturing long-range dependency while minimizing computational costs. Finally, MambaVision [13] introduces a hybrid approach. Early stages employ CNNs for fast feature extraction and later stages use Mamba and Transformer blocks to capture both local and global visual features.

Vision-based SSMs are primarily designed to capture relationships between spatially adjacent patches, focusing on spatial proximity or neighborhood-based interactions. However, SSMs are not inherently equipped to identify relationships between patches that are conceptually related but not spatially adjacent. This limitation arises due to the non-causal nature of image data, where pixels lack inherent temporal order or directional relationships.

In addition, current vision-based SSMs tend to exhibit high sensitivity to spatial information. For instance, VMamba uses predefined traversal routes that cause the network to indiscriminately switch between foreground and background regions within an image. This approach makes the network vulnerable to perturbations and leads to confusion in assessing feature importance. By treating all regions equally, VMamba fails to prioritize the features most relevant to the task, particularly when the foreground contains critical information. This can lead to poor performance when analyzing images with complex compositions.

Another notable challenge emerges when SSMs attempt to process images that undergo transformations, such as rotation. Vision-based SSMs typically process images by (1) unfolding image patches into sequences, and then (2) scanning these sequences in specific directions, such as horizontally and vertically. The sequence of image patches depends on the image orientation, meaning that rotating the image alters the order of the sequence. Similarly, the predefined scanning directions, which are fixed to the image original orientation, result in different scan paths when the image is rotated. Consequently, SS2D struggle to accommodate transformations such as rotation. Consider Figure 1 (first row): when the image is rotated by 90 degrees, the predefined and fixed traversal mechanism of VMamba significantly alters the sequence in which patches are processed.

To address these limitations, we propose **Spectral VMamba**, which organizes input patches based on the spectral information derived from a graph Laplacian of image patches. By leveraging the eigenvectors from spectral decomposition of an affinity matrix, our approach captures the global structure and relationships within the image. This eigenvector-driven traversal clusters similar patches together, even if they are not spatially adjacent, thereby preventing indiscriminate transitions between foreground and background regions. Moreover, spectral decomposition encodes patch relationships independently of image orientation, enabling the extraction of rotation-invariant features. Hence, our approach defines relationships between patches by their content and mutual associations rather than by their spatial arrangement. Figure 1 (second row) demonstrates that rotation does not affect the order in which patches are processed. As we can see, the patches representing the background are processed first, followed by those representing the foreground (the snake in Fig. 1).

Our main contributions are summarized as follow:

- We propose a novel technique named Spectral VMamba based on spectral graph analysis for traversing image patches. By leveraging the eigenstructure of graphs that model the relationships between patches, our approach ensures that spatially-coherent structures in image data are maintained. Our method also improves the performance of SSMs in capturing and interpreting complex data patterns.
- We introduce the Rotational Feature Normalizer (RFN) module, designed to enhance our model robustness to isometric transformations, such as rotations.
- Our method demonstrates a notable improvement on the image classification task, outperforming the current state-of-the-art State Space Model (SSM)-based model in computer vision.

2. Related Work

State Space Models (SSMs). Transformers have significantly improved the ability to manage long-range dependencies. However, their self-attention mechanism is computationally expensive, especially for high-resolution images, due to the quadratic scaling with input size. To address these

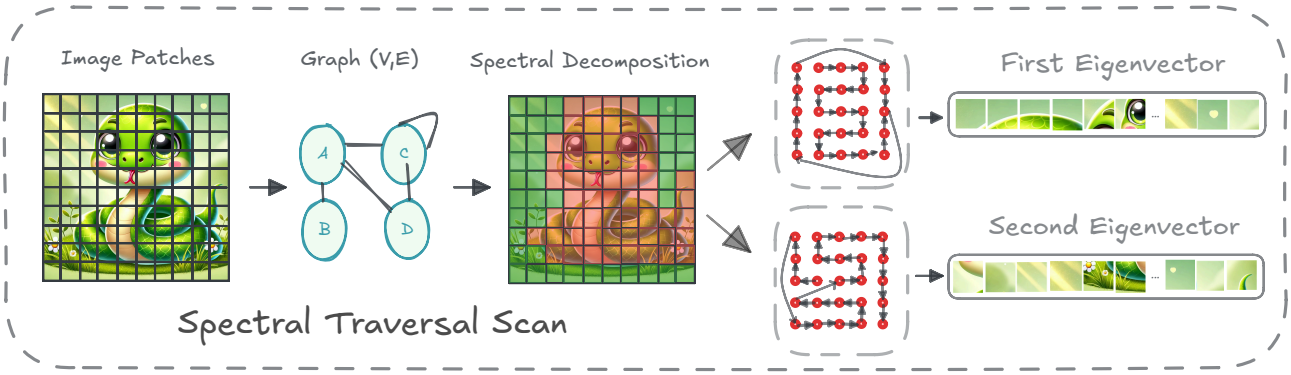

Figure 2. The Spectral Traversal Scan (STS) architecture begins by representing image patches as a graph. The Laplacian spectrum of this graph is then generated, and its eigenvectors are computed. The traversal path for the image patches is determined by the order of eigenvectors. The 1st eigenvector traverses the foreground/background regions, the 2nd eigenvector focuses on the next most salient structures, and each subsequent eigenvector progressively captures finer details in the image.

challenges, SSMs have emerged as an effective tool for long sequence modeling [11, 12, 26, 35]. In particular, the structured state-space sequence (S4) [11] model stands out for its efficient handling of long-range dependencies through a diagonal parameterization, which alleviates computational issues seen in earlier models.

Recent advancements in sequence modeling have been driven by innovations in parallel scanning techniques, inspired by models such as S4, S5 [35] and the H3 model [8]. Mamba [10] improved the S4 model by integrating a data-dependent selection mechanism. This mechanism allows Mamba to achieve linear scalability, ensuring robust performance while accommodating longer sequence lengths without the exponential increase in resource demands. Such improvements are essential for applications requiring real-time or large-scale sequence processing, including Natural Language Processing (NLP) and time-series forecasting.

SSMs in vision. The success of Mamba models in the NLP domain has stimulated significant progress in computer vision, leading to various adaptations tailored for visual data processing. Several studies have extended Mamba architectures to address specific challenges in vision tasks. For instance, VMamba [21] introduced an SS2D module to tackle direction sensitivity between non-causal 2D images and ordered 1D sequences. LocalMamba [17] focuses on localized feature extraction, utilizing state-space layers to enhance neighborhood interactions and reduce dependence on self-attention, leading to efficient high-resolution image analysis. QuadMamba [43] uses a quadrilateral traversal mechanism to manage directional dependencies in 2D images, integrating features from different image segments.

Hybrid architectures. Recent work has also focused on combining SSMs with traditional CNNs or ViTs, leveraging the respective strengths of these models. For example, U-Mamba [25] merges CNNs and state space models to address long-range dependencies in biomedical image segmentation. MambaVision [13] integrates Mamba architectures with convolutional neural networks to improve performance across various vision tasks by leveraging both sequential modeling and spatial feature extraction. Finally, the TranS4mer model [35] integrates S4 with self-attention mechanisms to achieve state-of-the-art performance in movie scene detection. These advancements collectively demonstrate the versatility and scalability of structured state space models across various applications in the vision domain.

Rotation-invariance in vision. Numerous approaches have been developed to embed equivariance and invariance directly within neural network architectures. Group-equivariant convolutional networks explicitly encode rotation symmetry into their design, providing intrinsic equivariance to rotations through steerable or group-convolutional filters [3, 4, 41, 42]. Learned canonicalization methods, such as Spatial Transformer Networks, address invariance by predicting transformations to consistently align input images into a canonical orientation, effectively standardizing pose variability [18, 19, 27]. Alternatively, symmetrization strategies explicitly aggregate feature responses across multiple rotated inputs or filters, achieving invariance by pooling orientation-specific activations [6, 20, 29, 46]. Additionally, capsule networks explicitly represent the pose of the object, leveraging pose-aware activations to enhance robustness against rotations [15, 31]. Within this landscape, our current work aligns with learned canonicalization approaches by leveraging spectral decomposition to canonicalize patch ordering and achieve patch traversal rotation invariance.

3. Method

This section begins with an overview of key concepts in SSMs and spectral graph analysis in Section 3.1, forming the foundation for our approach. Next, in Section 3.2, we in-

troduce our proposed architecture, Spectral VMamba, along with the RFN module designed to achieve rotation invariance (Section 3.3). Finally, we discuss the robustness of our method to rotational variations in Section 3.6.

3.1. Preliminaries

SSM formulation. SSM-based models [10, 11] map a one-dimensional function or an input sequence $x(t) \in \mathbb{R}$ to an output sequence $y(t) \in \mathbb{R}$ through a learnable hidden state $\mathbf{h}(t) \in \mathbb{R}^N$. This mapping follows a continuous dynamical system formulated as

$$\frac{d}{dt}\mathbf{h}(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}x(t), \quad y(t) = \mathbf{C}\mathbf{h}(t) \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$, $\mathbf{B} \in \mathbb{R}^{N\times 1}$ and $\mathbf{C} \in \mathbb{R}^{1\times N}$ are parameters governing the evolution of system.

To adapt continuous SSMs in deep learning frameworks, discretization is required. This can be achieved using the zero-order hold (ZOH) rule with a time scale parameter $\Delta \in \mathbb{R}$. Based on this rule, the discretized versions of $\mathbf{A}$ and $\mathbf{B}$ are defined as

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

and the system becomes

$$\mathbf{h}(t) = \overline{\mathbf{A}}\mathbf{h}(t-1) + \overline{\mathbf{B}}x(t), \quad y(t) = \mathbf{C}\mathbf{h}(t). \tag{3}$$

As $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ are linear operators, the auto-regressive formulation of Eq. (3), computing the output at each time step t in a recurrent manner, can be unrolled and expressed as single convolution with kernel $\overline{\mathbf{K}} \in \mathbb{R}^L$:

$$\mathbf{y} = \mathbf{x} \odot \overline{\mathbf{K}}, \quad \overline{\mathbf{K}} = \left(\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, \ldots, \mathbf{C}\overline{\mathbf{A}}^{L-1}\overline{\mathbf{B}}\right). \tag{4}$$

During training, this convolutional can be employed to compute all output values in parallel.

Spectral Graph Analysis studies the properties of a graph by analyzing the eigen-spectrum of its Laplacian matrix, which can be seen as a discrete version of the continuous Laplace-Beltrami operator obtained by a finite difference method [30]. Given a real-valued function $f \in C^2$ on a Riemannian manifold $\mathcal{M}$, the Laplace-Beltrami operator Δ of f is defined as $\Delta f = div(grad\ f)$ where with $grad\ f$ is the gradient of f and div denotes the divergence.

The Laplacian Eigenvalue Problem corresponds to solving the Helmholtz equation $\Delta f = -\lambda f$, whose solutions (eigenfunctions) are directly related to the vibration of a physical membrane [2]. Following Courant's Nodal Line Theorem [5], the Laplacian eigen-spectrum is composed of non-negative eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots$, each one repeating according to its multiplicity. Moreover, the eigenfunction corresponding to λ_n has at most n poles of vibration, hence smaller eigenvalues and their associated eigenvector encode lower frequency structural information which captures relationships at a more global scale.

The Laplacian matrix of a graph $G = (V, E)$ with nodes V and edges E is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the diagonal degree matrix with entries $D_{ii} = \sum_j W_{ij}$. A normalized version of the Laplacian is typically employed to address differences in the weight scales W_{ij} and variations in the distribution of node degrees D_{ii}. In this work, we utilize the symmetric normalized Laplacian, defined as

$$\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}, \tag{5}$$

which is positive semi-definite and possesses $|V|$ non-negative real-valued eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_{|V|}$.

A key property of the Laplace eigen-spectrum is that it is invariant to transformations preserving distance (i.e., isometry), which include translations and rotations. In our work, we use this property to design an SSM that enhances rotational robustness, where the order in which patches are traversed is determined based on the eigenvectors corresponding to the smallest eigenvalues of the Laplacian matrix.

3.2. Spectral VMamba

Existing vision-based Mamba methods, such as VMamba [21], are based on SS2D, a four-way scanning mechanism designed for spatial domain traversal. SS2D organizes input patches along four predefined scanning routes. However, two key challenges emerge: (1) connections between semantically related but non-neighboring patches are overlooked; and (2) scanning is sensitive to isometric transformations, such as rotation. To address these limitations, we introduce the Spectral VMamba model whose architecture is summarized in Figure 3. This architecture consists of two main components: a RFN module ensuring consistent feature representation across orientations and a sequence of Spectral VMamba blocks forming the network body.

Data forwarding in our Spectral VMamba block comprises three key steps: Spectral Traversal Scan (STS), S6 block (selective scan), and Spectral Traversal Merge (STM). In the STS step, input patches are reordered based on the Laplacian spectrum of their underlying graph structure, ensuring traversal aligns with spectral coherence (see Section 3.4). The selective scan step then processes each reordered patch sequence in parallel using dedicated S6 blocks, effectively capturing localized patterns. Finally, in the STM step, the reordered traversal paths are restored to their original spatial format, merging the resulting sequences to generate the final output map. The detailed architecture of STS is depicted in Figure 2. The sequence of patch processing is determined by the order of eigenvectors derived from the spectral decomposition of the graph Laplacian. It is important to note that STS is only performed once for the initial layer.

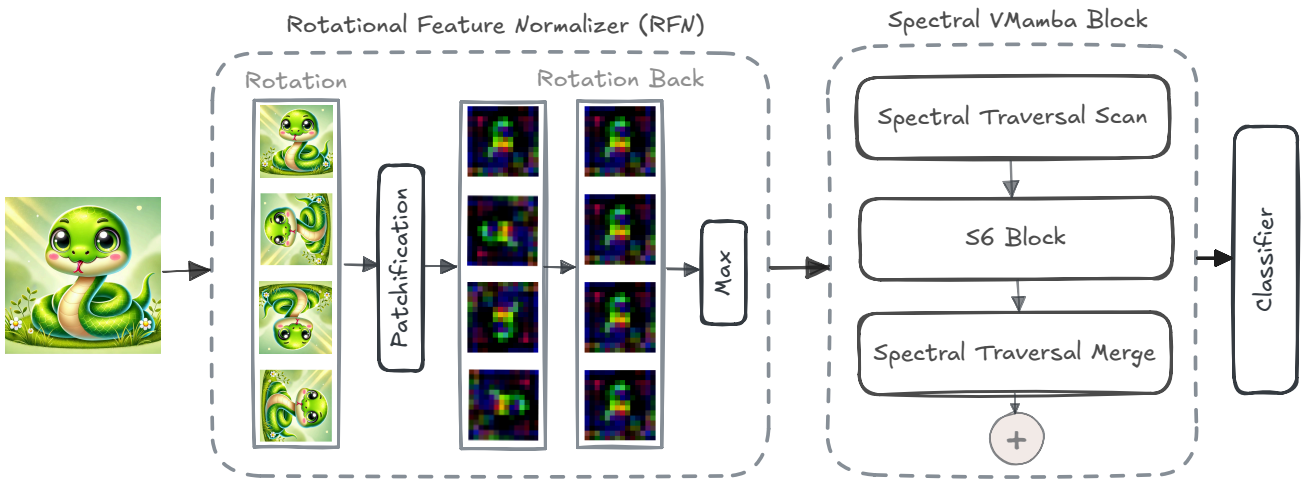

Figure 3. An overview of the proposed method. Our network consists of two primary components: the RFN module, which normalizes feature orientation to ensure consistency across different perspectives, and a series of Spectral VMamba blocks that process data in three stages: (1) Spectral Traversal Scan (STS), (2) S6 block (selective scan), and (3) Spectral Traversal Merge (STM). In STS, patches are reordered based on spectral coherence derived from the Laplacian spectrum. Each sequence is then processed in parallel by dedicated S6 blocks, capturing localized patterns, and finally merged in STM to reconstruct the spatial layout, producing the final feature map.

In subsequent layers, we apply a downsampling strategy for generating the traversal path. For a detailed explanation of the downsampling process, please refer to the supplementary material.

The graph Laplacian eigen-spectrum is isometric invariant, guaranteeing that the order in which patches are traversed is not affected by changes to their position in the image. However, the values in the Laplacian matrix are computed using patch features which are themselves sensitive to rotation of the patch. A simple solution to this problem is to use patches containing a single pixel, however this would be impractical for two reasons. First, individual pixels encode little information (RGB color) and thus using their similarity to construct the Laplacian matrix would not be useful. More importantly, this would severely increase the computational cost of downstream operations such as the spectral decomposition of the Laplacian. To address this issue, we compute patch features using our RFN module (Section 3.3) designed to normalize the feature extraction process by aggregating features from multiple orientations of the input image. This approach allows for effectively handling of feature changes during rotations, preserving the desired order and mitigating orientation-dependent discrepancies.

3.3. Rotational Feature Normalizer (RFN)

An overview of the module is illustrated in Figure 3. This module rotates the input image by a pre-determined set of angles $\{\theta_i, \ldots, \theta_R\}$ and processes them with a *stem module* which serves as a feature extraction backbone for patchification. This backbone partitions the input image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$ into patches $\mathbf{x} \in \mathbb{R}^{\frac{H}{p}\times\frac{W}{p}\times C}$, where (H, W) denote the dimensions of the input image, C is the number of feature channels, and p represents the patch size. To consolidate features from all rotated versions, each feature map is rotated back to its original orientation. This unrotation ensures that all extracted features align with the original image coordinate system. Finally, an element-wise max operation is applied across the unrotated feature maps to generate a single aggregated feature. The final aggregated feature map can be expressed as:

$$\mathbf{F}_{i,j} = \max_{r\in\{1,\ldots,R\}} \left[\mathcal{R}_{-\theta_r}\left(\text{Patchify}\left(\mathcal{R}_{\theta_r}(\mathbf{I})\right)\right)\right]_{i,j} \quad (6)$$

where θ_r is the angle of the r-th rotation and $\mathcal{R}_{\theta_r}$ is the transformation applying this rotation.

The choice of rotation angles for the RFN module represents a trade-off between rotation invariance and computational overhead. Using a larger number of angles ensures a greater coverage in terms of rotation, hence improving the invariance of the model at the cost of increased computations. In our implementation, we used four *canonical* angles, $\{0°, 90°, 180°, 270°\}$, corresponding to quarter turns. By rotating the image at these specific angles, the pixel grid alignment is preserved, eliminating the need for interpolation. This alignment not only simplifies the back-rotation process but also guarantees that the extracted features from all rotated images are comparable.

3.4. Spectral Traversal Scan

In this section, we describe how the graph Laplacian eigenspectrum is leveraged to define the travesal order of patches.

We begin by representing the image as a graph $G = (V, E)$, where each node $v_i \in V$ corresponds to a patch $\mathbf{x}_i$

of the image, and edges $e_{ij} \in E$ connect pairs of nodes based on their spatial proximity. Following methods for spectral clustering [28] and normalized cuts [32], we consider a weighted adjacency matrix $\mathbf{W}$ of the graph, which is constructed using Euclidean distances between patches. Specifically, the weight W_{ij} between the nodes v_i and v_j is given by

$$W_{ij} = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{f}_i - \mathbf{f}_j\|^2\right) \tag{7}$$

where $\mathbf{f}_i$ is the feature vector of patch $\mathbf{x}_i$ and σ is a scaling parameter determined as the average of the Euclidean distances between patches. We set $W_{ij} = 0$ for pairs (v_i, v_j) that are not among the top-k nearest neighbors of each other, resulting in a sparse adjacency matrix.

We compute the first m smallest eigenvectors of the symmetric Laplacian $\mathbf{L}_{\text{sym}}$ using an iterative method known as the Arnoldi algorithm [9], which leverages the sparsity of the adjacency matrix to provide a low complexity. Each eigenvector $\mathbf{u}^{(j)} \in \mathbb{R}^{N_p}$, where j ranges from 1 to m, assigns an eigenfunction value $\mathbf{u}_i^{(j)}$ to each image patch i. In every Mamba block of our model, we perform two traversals per eigenvector: one in the order of increasing $\mathbf{u}_i^{(j)}$ values and another in decreasing order. At the end of the block, the resulting features from these $2m$ traversals are concatenated together.

Patches $\mathbf{x}_i$ are ordered based on these eigenvectors, generating multiple traversal sequences:

$$\mathbf{X}^{(j,\text{rev})} = \left\{\mathbf{x}_{i_1}^{(j,\text{rev})}, \mathbf{x}_{i_2}^{(j,\text{rev})}, \ldots, \mathbf{x}_{i_m}^{(j,\text{rev})}\right\}, \text{ where} \tag{8}$$

$$i_k^{(j,\text{rev})} = \arg\min_{i \in \{1,\ldots,m\}} \left\{(-1)^{\text{rev}} \cdot u_i^{(j)} \mid i \notin \{i_1, \ldots, i_{k-1}\}\right\}$$

where $\text{rev} \in \{0, 1\}$ indicates if the traversal is reversed ($\text{rev} = 1$) or not ($\text{rev} = 0$). This concatenated sequence $\mathbf{X}$ is then fed into the Mamba network for further processing, allowing the model to incorporate rich spectral relationships while preserving global coherence.

3.5. Spectral Canonicalization

Eigenvectors inherently have a sign ambiguity as, if $\mathbf{u}$ is an eigenvector of a matrix $\mathbf{A}$ with eigenvalue λ, then $-\mathbf{u}$ is also an eigenvector with the same eigenvalue. To resolve this ambiguity, we standardize the sign of eigenvectors by ensuring that the first non-zero element of each eigenvector is positive. If the first element is negative, we flip the sign of the entire eigenvector. This adjustment does not alter the underlying structure captured by the eigenvectors, as flipping the sign consistently maintains the same eigenvalue λ and preserves the relationships between eigenvectors.

3.6. Rotation Invariance

Spectral decomposition leverages the eigen-structure of the Laplacian matrix, which remains consistent regardless of rotation. As our traversal strategy is directly guided by the spectral decomposition of input features, the rotation-invariant nature of spectral decomposition extends to our method. This property is demonstrated in the following theorem.

Theorem 3.1. *Under the assumption that patch features* $\mathbf{f}$ *are invariant to rotation, i.e.* $\mathbf{f}(\mathbf{I}) = \mathbf{f}(\mathcal{R}_\theta(\mathbf{I}))$ *for any* θ*, our spectral traversal scan is also rotation-invariant.*

Proof. As rotations are a type of isometry, we can prove the more general property of isometric invariance. This invariance derives from two properties: 1) the nearest neighbors and values of the weighted adjacency matrix are defined based on distance alone, i.e., $W_{ij} = \exp(-\frac{1}{2\sigma}^2\|\mathbf{f}_i - \mathbf{f}_j\|^2)$, thus they do not change if distances remain the same; 2) the eigen-spectrum is invariant under cyclic permutation. This second property can be demonstrated as follows. Let $\mathbf{P}$ be a permutation matrix and $\mathbf{u}$ be an eigenvector of Laplacian matrix $\mathbf{L}$ corresponding to eigenvalue λ. Using $\mathbf{u}' = \mathbf{P}\mathbf{u}$ (thus $\mathbf{u} = \mathbf{P}^T\mathbf{u}'$) we have that

$$\begin{aligned} \mathbf{L}\mathbf{u} = \lambda\mathbf{u} &\Rightarrow \mathbf{L}\mathbf{P}^T\mathbf{u}' = \lambda\mathbf{P}^T\mathbf{u}' \\ &\Rightarrow \mathbf{P}\mathbf{L}\mathbf{P}^T\mathbf{u}' = \mathbf{P}(\lambda\mathbf{P}^T\mathbf{u}') \\ &\Rightarrow \mathbf{P}\mathbf{L}\mathbf{P}^T\mathbf{u}' = \lambda\mathbf{u}' \end{aligned} \tag{9}$$

Hence, $\mathbf{P}\mathbf{u}$ is an eigenvector of the permuted Laplacian matrix $\mathbf{P}\mathbf{L}\mathbf{P}^T$. Since we traverse patches by their eigenvector values (from smallest to largest or vice-versa), we will therefore get the same ordering regardless of the permutation. □

4. Experiments

We start by comparing our method against recently-proposed ViTs and SSMs on an image classification task. We then present an ablation study to thoroughly investigate the impact of various components on the performance of our method.

4.1. Image Classification

We conduct classification experiments on the *mini*ImageNet [40] dataset. This dataset includes $50,000$ training images and $10,000$ validation images across 100 categories. Following previous works [21, 22, 47], we train our models for 300 epochs with a batch size of 128 per GPU. The models are optimized using the AdamW optimizer with a momentum of 0.9. A cosine decay scheduler manages the learning rate, which starts at 5×10^{-4}, alongside a weight decay of 0.05. Additionally, we apply an exponential moving average (EMA) to stabilize training. For input images of size 224×224, our data augmentation techniques include color jittering, AutoAugment, random erasing, mixup, and cutmix.

Table 1 compares our Spectral VMamba method against ViTs and SSM-based models on the *mini*ImageNet dataset. As can be seen, our method outperforms all other approaches

Table 1. Classification performance comparison on *mini*ImageNet. All images are of size 224×224. T, S, and B denote the tiny, small, and base scales, respectively.

Model	FLOPs (G)	Top-1 (%)	Top-5 (%)
Transformer-Based			
DeiT-S [37]	4.6G	70.83	89.74
DeiT-B [37]	17.5G	72.43	90.14
Swin-T [22]	4.5G	83.25	95.54
Swin-S [22]	8.7G	84.10	95.53
Swin-B [22]	15.4G	82.77	95.33
XCiT-S24 [1]	9.2G	85.79	96.31
XCiT-M24 [1]	16.2G	86.80	96.38
SSM-Based			
Vim-T [47]	1.5G	67.30	87.97
Vim-S [47]	5.1G	79.70	93.20
LocalVim-T [17]	1.5G	82.12	94.60
LocalVim-S [17]	4.8G	81.68	93.63
MSVMamba-N [33]	0.9G	82.16	95.10
MSVMamba-M [33]	1.5G	83.72	95.81
MSVMamba-T [33]	4.6G	86.48	96.43
VMamba-T [21]	4.9G	86.25	96.64
VMamba-S [21]	8.7G	86.48	96.79
VMamba-B [21]	8.7G	87.17	96.96
Ours-T	3.9G	87.86 (+1.61)	97.25 (+0.61)
Ours-S	6.3G	88.09 (+1.61)	97.08 (+0.29)
Ours-B	6.3G	88.17 (+1.00)	97.27 (+0.31)

in both top-1 and top-5 EMA accuracy. Compared to VMamba, its closest competitor, our method improves top-1 accuracy by 1.61% for the tiny and small models (T and S) and by 1% for the base model (B), while requiring less FLOPs (see 4.2 for explanation). Moreover, our Spectral VMamba (B) model yields a 1.37% higher top-1 accuracy than the strongest ViT-based model, XCiT-M24, and an improvement of 5.4% over Swin-B, one of the most popular architectures based on ViT.

4.2. Ablation Study

Rotational invariance. To evaluate the robustness to rotation of our method trained on unrotated images, we evaluated its performance on test images rotated at angles from $0°$ to $360°$ with $15°$ increments. As a reminder, our RFN module extracts and aggregates features for four canonical rotations: $\{0°, 90°, 180°, 270°\}$. We therefore expect a greater robustness at those rotation angles, for which our method is fully invariant.

As shown in Figure 4, our model with the RFN (blue line) maintains a high accuracy near 87% at those canonical angles, demonstrating its rotational invariance property. In contrast, VMamba (orange line) suffers from a significant drop in performance for images rotated at $90°$ (∼30% drop), $180°$ (∼20% drop) and $270°$ (∼30% drop). Although our method does not guarantee full rotational invariance for non-canonical angles, we observe a remarkably consistent performance across all rotation angles, stabilizing at around 78% in top-1 accuracy. On the other hand, the performance

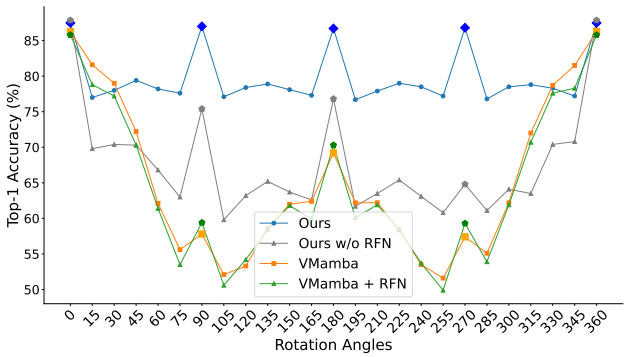

Figure 4. Model performance comparison across various rotation angles. Our method demonstrates consistent accuracy across all rotation angles, while VMamba exhibits significant fluctuations.

of VMamba fluctuates drastically, reaching accuracy values near 50% at certain angles (e.g., $105°$ or $255°$).

Impact of RFN. To assess the usefulness of our RFN module, we also tested an ablation variant of our method without this module (Ours w/o RFN in Figure 4). As mentioned before, while the Laplacian eigen-spectrum is isometric-invariant, and thus invariant to rotation, the Laplacian matrix is constructed using patch features which are themselves sensitive to rotation. As we expect, removing the RFN causes the performance to drop for all angles, this drop growing as we increase the rotation (away from $0°$ or $360°$). Interestingly, our method without the RFN module still achieves a higher accuracy than VMamba for angles between $60°$ and $300°$, which represent severe rotations. This could be explained by the small size of patches (16×16) compared to the image, minimizing the impact of rotations (full invariance would be achieved for 1×1 patches).

Impact of STS. Since the STS is a critical part of our Spectral VMamba method, we cannot remove it. To evaluate its impact, we instead add the RFN to VMamba, the resulting approach now only differing from our method by not having a STS. As shown in Figure 4, the VMamba + RFN model (green line) offers a low robustness to rotation, its accuracy similar to that of VMamba. These results demonstrate that improving rotational invariance at the patch level is not sufficient to achieve a high performance.

Number of eigenvectors. In Figure 5, we assess the effect on performance of varying the number of eigenvectors used in the STS ($m \in \{1, 2, 3, 4\}$). As can be seen, accuracy improves consistently with more eigenvectors, peaking at 87.48% for $m = 4$. This suggests that eigenvectors offers complementary information and that their combination enhances the recognition of complex global structures. In the same plot, we also report the performance of the standard VMamba model and a variant using a random traversal strategy. As can be observed, the improvement provided by

our STS over VMamba, for any number of eigenvectors, is greater than the one offered by the raster scan of VMamba over a random traversal.

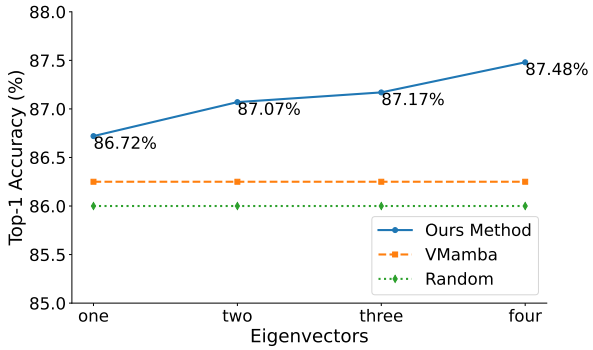

Figure 5. Model performance for different number of eigenvectors. Four eigenvectors exhibit the best performance.

Number of nearest neighbors. The K-Nearest Neighbors (KNN) algorithm constructs a sparse adjacency matrix by selecting the top k nearest neighbors for each image patch. In Tab. 2, we report our model's performance across k values from 5 to 25. As can be seen, our model is not very sensitive to this parameter and achieves its highest accuracy of 87.48% for $k=5$ neighbors. This suggests that only a few neighbors are necessary to capture the global structure of an image. This has a positive impact on the efficiency of our method as the computational complexity of the Laplacian eigen-decomposition step, using the Arnoldi algorithm, is proportional to the number of non-zero entries in the Laplacian matrix (which is directly proportional to k).

k	5	10	15	20	25
Top-1 Accuracy (%)	**87.48**	86.98	87.14	86.91	87.15

Table 2. Model performance for different values of k (number of nearest neighbors). The best performance is achieved at $k = 5$.

Memory Usage and Runtime. Next, we analyze the computational cost and time efficiency of the STS. As mentioned before, the proposed strategy minimizes computations by leveraging the sparsity of the Laplacian matrix and limiting the number of computed eigenvectors. In the following experiments, we use $m = 4$ eigenvectors and batch size = 1.
Memory Usage: As depicted in Fig. 6, our STS approach (blue line) is highly efficient in terms of memory usage compared to the VMamba (orange line). Even when the number of patches increases, the STS based on a sparse eigen-solver, requires only a modest increase in memory compared to the VMamba. The stars along each line indicate the token lengths used in our method, highlighting the specific points where processing occurs across different sequence lengths.
Runtime: Our STS strategy, which can run on a CPU, also shows a favorable runtime scaling with increased patch

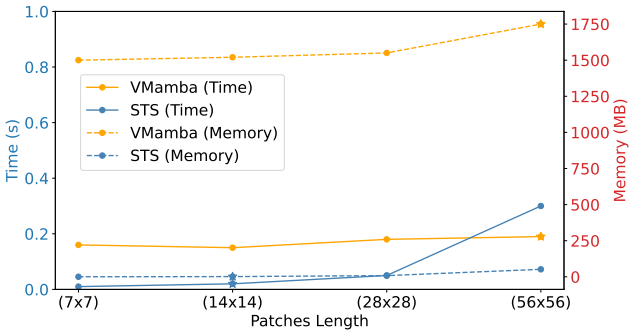

Figure 6. Runtime and memory usage for different patches lengths.

length (Fig. 6). Specifically, for a patch length of 196 (marked by a star), which is the setting used in our main experiments, the increase in runtime is minimal when compared to the VMamba backbone, highlighting the efficiency of our approach.
FLOPs: The amount of FLOPs for our traversing strategy include the computations required for constructing the adjacency matrix, calculating the Laplacian, and obtaining its first eigenvectors. This calculation results in approximately 2 MB of FLOPs, which is significantly lower than the 3.9 GB FLOPs needed by VMamba's backbone. Interestingly, as reported in Tab. 1, the total number of FLOPs performed by our Spectral VMamba method is *less* than that of VMamba when using the same model size. This is because, after processing patches, the input to our Spectral VMamba has a size of 14×14 tokens, compared to 56×56 tokens for VMamba.

5. Conclusion

In this paper, we introduce Spectral VMamba, a novel approach to visual state space models that achieves patch traversal rotation invariance. Our method leverages spectral decomposition from the graph Laplacian of image patches to define traversal paths for patch processing. In addition, we develop the RFN module to ensure consistent feature representation in different orientations. This module serves as a foundation for our STS module, enabling patch features that are robust to rotational variations. By incorporating the RFN and STS modules, Spectral VMamba achieves a robust transformation-invariant feature extraction pipeline that maintains high accuracy across a variety of orientations. Our experiments demonstrate that Spectral VMamba consistently outperform popular models including Swin transformer, VMamba, MSVmamba, and LocalVim in image classification tasks. Our method does not yet address generalization to the medical domain. In future work, we aim to handle disconnected graph components, common in medical images with distinct anatomical structures or pathologies. Adapting spectral traversal for these regions will enhance pattern capture and robustness across imaging modalities.

Acknowledgement

We appreciate the computational resources and support provided by Compute Canada and the Digital Research Alliance of Canada.

References

[1] Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *NeurIPS*, 34:20014–20027, 2021. 7

[2] Alvin Bayliss, Charles I Goldstein, and Eli Turkel. An iterative method for the helmholtz equation. *Journal of Computational Physics*, 49(3):443–457, 1983. 4

[3] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, pages 2990–2999, 2016. 3

[4] Taco S Cohen and Max Welling. Steerable cnns. In *International Conference on Learning Representations (ICLR)*, 2017. 3

[5] Richard Courant and David Hilbert. *Methods of mathematical physics: partial differential equations*. Interscience Publishers, 1937. 4

[6] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. *International Conference on Machine Learning (ICML)*, 2016. 3

[7] 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. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2

[8] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022. 2, 3

[9] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013. 6

[10] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 2, 3, 4

[11] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021. 2, 3, 4

[12] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021. 3

[13] Ali Hatamizadeh and Jan Kautz. Mambavision: A hybrid mamba-transformer vision backbone. *arXiv preprint arXiv:2407.08083*, 2024. 2, 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[15] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Matrix capsules with em routing. In *International Conference on Learning Representations (ICLR)*, 2018. 3

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 1

[17] 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. 3, 7

[18] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2017–2025, 2015. 3

[19] Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *International Conference on Learning Representations (ICLR)*, 2023. 3

[20] Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 289–297, 2016. 3

[21] 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*, 2024. 2, 3, 4, 6, 7

[22] 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*, pages 10012–10022, 2021. 6, 7

[23] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1

[24] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 1

[25] Jun Ma, Feifei Li, and Bo Wang. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *arXiv preprint arXiv:2401.04722*, 2024. 3

[26] Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022. 3

[27] Arnab Kumar Mondal, Siba Smarak Panigrahi, Oumar Kaba, Sai Rajeswar Mudumba, and Siamak Ravanbakhsh. Equivariant adaptation of large pretrained models. *Advances in Neural Information Processing Systems*, 36:50293–50309, 2023. 3

[28] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001. 6

[29] Omri Puny, Matan Atzmon, Heli Ben-Hamu, Ishan Misra, Aditya Grover, Edward J Smith, and Yaron Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021. 3

[30] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace–beltrami spectra as 'shape-dna'of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006. 4

[31] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 3

[32] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. 6

[33] Yuheng Shi, Minjing Dong, and Chang Xu. Multi-scale vmamba: Hierarchy in hierarchy visual state space model. *arXiv preprint arXiv:2405.14174*, 2024. 2, 7

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International conference on learning representations*, 2014. 1

[35] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022. 2, 3

[36] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1

[37] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Int. Conf. Machine Learning*, pages 10347–10357, 2021. 7

[38] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 1

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[40] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016. 6

[41] Maurice Weiler and Gabriele Cesa. General E(2)-Equivariant Steerable CNNs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 3

[42] Maurice Weiler, Fred A. Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 849–858, 2018. 3

[43] Fei Xie, Weijia Zhang, Zhongdao Wang, and Chao Ma. Quadmamba: Learning quadtree-based selective scan for visual state space model. *arXiv preprint arXiv:2410.06806*, 2024. 3

[44] Xiaosong Zhang, Yunjie Tian, Lingxi Xie, Wei Huang, Qi Dai, Qixiang Ye, and Qi Tian. Hivit: A simpler and more efficient design of hierarchical vision transformer. In *The Eleventh International Conference on Learning Representations*, 2023. 1

[45] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2019. 2

[46] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 519–528, 2017. 3

[47] 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. 2, 6, 7

Spectral State Space Model for Rotation-Invariant Visual Representation Learning

Supplementary Material

A. Implementation Details

In this section, we give the pseudo-code for the RFN and STS modules. This pseudo-code provides a concise summary of the key steps involved in our approach, offering a high-level abstraction of the implementation. It is designed to complement the detailed explanations in the main paper and can serve as a reference for reproducing our results.

We begin with Algorithm 1, which outlines the steps for implementing the RFN module. As discussed in the paper, this module ensures a consistent representation of image features across different orientations. This module comprises four key steps: rotation, patchification, back-rotation, and a max operation to aggregate features across all orientations.

Algorithm 1 Rotational Feature Normalizer Module

Require: Input image $\mathcal{I} \in \mathbb{R}^{H\times W\times 3}$; rotation angles: $\{\theta_r \mid r = 1, \ldots, R\}$; stem module: Stem; patch size: p
Ensure: Aggregated feature map $\mathcal{F}$.
Initialize: $\{\theta_r\}$ and $\{-\theta_r\}$.
for $r \in \{1, \ldots, R\}$ **do**
  $\mathcal{I}_r \leftarrow \mathcal{R}_{\theta_r}(\mathcal{I})$ ▷ Rotate image by θ_r
  $\mathcal{F}_r \leftarrow \text{Stem}(\mathcal{I}_r)$ ▷ Extract features
  $\mathcal{F}_r^{\text{unrotated}} \leftarrow \mathcal{R}_{-\theta_r}(\mathcal{F}_r)$ ▷ Back-rotate feature map
  Append $\mathcal{F}_r^{\text{unrotated}}$ to $\mathcal{F}_r$.
end for
$\mathcal{F} \leftarrow \max_{r\in\{1,\ldots,R\}} \mathcal{F}_r$ ▷ Max operation
Output: Aggregated feature map $\mathcal{F}$.

Algorithm 2 represents the STS strategy for determining the traversal order of image patches based on spectral decomposition. The process begins by constructing the adjacency matrix $\mathbf{W}$ using the k-Nearest Neighbors (KNN) algorithm, based on the Euclidean distances between image patches. This matrix captures the relationships and similarities between the patches. Subsequently, the degree matrix $\mathbf{D}$ is computed, where each diagonal entry D_{ii} represents the sum of the weights of edges connected to node i. Using $\mathbf{W}$ and $\mathbf{D}$, the symmetric normalized Laplacian matrix $\mathbf{L}_{\text{sym}}$ is calculated. Spectral decomposition is then applied to $\mathbf{L}_{\text{sym}}$, yielding the eigenvalues U and eigenvectors V. Finally, the patches are reordered based on the spectral information, resulting in the ordered sequence of patches P.

Algorithm 2 Spectral Traversal Scan Module

Require: patch feature $\mathbf{f}$, number of neighbors k, and eigenvectors m
Ensure: Traversal sequence $\mathcal{P}$
Step 1: *Compute Weighted Adjacency Matrix*
for each pair of patches $(\mathbf{x}_i, \mathbf{x}_j)$ **do**
  if $i \in knn_j$ OR $j \in knn_i$ **then**
    $W_{ij} = \exp\left(-\frac{\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2\sigma^2}\right)$
  else
    $W_{ij} = 0$
  end if
end for
Step 2: *Compute Normalized Laplacian*
$\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$
Step 3: *Compute m Smallest Eigenvectors of $\mathbf{L}_{sym}$*
$[U, V] = \text{eigenSolver}(L_{\text{sym}}, m)$
Step 4: *Building Traversal Sequences*
Sort U from smallest to largest.
Select the corresponding m eigenvalues from V.
for $j = 1$ to m **do**
  P_1^j = sorting f using increasing order of $\mathbf{v}^{(j)}$
  P_2^j = sorting f using decreasing order of $\mathbf{v}^{(j)}$
  $\mathcal{P} = \left[P_1^{(j)},\ P_2^{(j)}\right]^{j=1,\ldots,m}$
end for
Output: Traversal sequence $\mathcal{P}$.

B. Downsampling Strategy

Similar to VMamba, the architecture of our network consists of four layers. Each layer contains a different number of Spectral VMamba blocks. For example, in the tiny scale configuration, we use the setup [2, 2, 5, 2], indicating that the first layer has two Spectral VMamba blocks, the second layer also has two blocks, the third layer has five blocks, and the final layer contains two blocks. As previously mentioned, the STS module is applied only once, in the first Spectral VMamba block of the first layer. Additionally, downsampling occurs at the end of each layer.

The eigenvectors are initially computed in the STS using the original 14×14 spatial features. However, after downsampling, the eigenvectors derived from the 14×14 patches no longer align with the downsampled patches in subsequent layers, which requires careful handling to maintain consistency. To resolve this alignment issue, we save the indices used during the max pooling operation, which is responsible for downsampling the spatial features. By leveraging the

saved indices, we extract the corresponding eigenvectors, generating a new set that matches the shape of the downsampled patches. This alignment ensures that the eigenvectors and patches remain correctly paired, enabling us to order the downsampled patches using eigenvectors of compatible dimensions. We repeat this process at each subsequent layer to maintain proper alignment throughout the network.

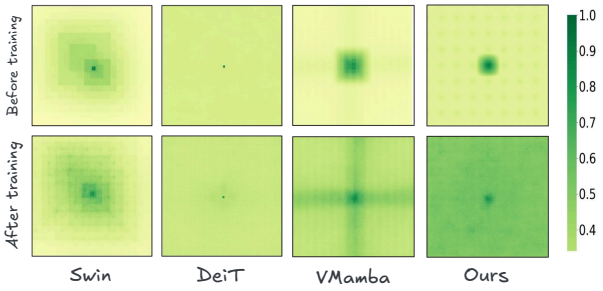

Figure 7. Comparison of Effective Receptive Fields (ERF) Before and After Training.

C. Visualization of Activation Maps

Effective Receptive Field (ERF) in CNNs refer to the region of the input image that significantly influences the activation of a particular neuron in a deeper layer of the network. Unlike the theoretical receptive field, which considers the full extent of influence regardless of intensity, the ERF focuses on the practical impact, often showing that only a central portion of the theoretical receptive field has substantial influence. Analyzing ERFs helps in refining network designs to ensure that neurons capture relevant information efficiently, enhancing performance in tasks such as object detection and image segmentation.

We conducted experiments to compare our model with VMamba, focusing on the ERF of the central pixel before and after training. Our results in Figure 7 indicate that our model exhibits more global ERFs compared to VMamba. Specifically, after training, the areas colored dark green, which represent regions of high influence, are more extensive in our method than in VMamba. This suggests that our model is better at capturing broader contextual information from the input image. The increased dark green regions in our model demonstrate its enhanced capability to integrate information over larger portions of the input, potentially leading to improved performance in tasks requiring a comprehensive understanding of the image content.

D. Training Dynamics

Figure 9 illustrates the comparison of training dynamics, measured using the maximum accuracy with an Exponential Moving Average, across three model scales: Tiny, Small, and Base. Each plot represents the accuracy progression over 300 epochs for our proposed method and the VMamba model. The plots represents our method demonstrates faster convergence across all scales, achieving high accuracy earlier in training compared to VMamba. The faster convergence not only highlights the efficiency of our approach but also reduces training time, making it highly suitable for practical applications where computational resources or time are constrained.

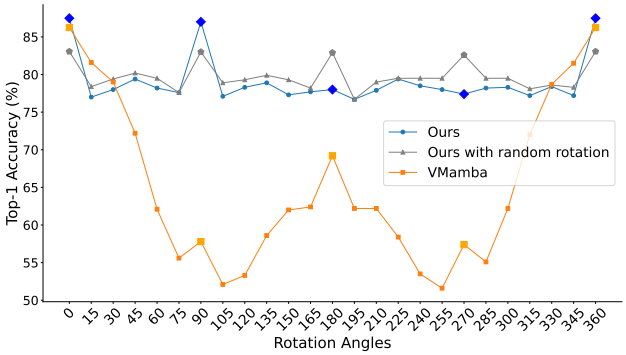

Figure 8. Effect of Random Rotation in RFN Module.

E. RFN with Random Rotations

For the RFN module, we utilized four *canonical* angles: $0°, 90°, 180°, 270°$, corresponding to quarter turns. Additionally, we tested the method with four random rotations selected at each iteration from the ranges [0, 90], [91, 180], [181, 270], and [271, 360]. Unlike quarter turns, most random rotations necessitate interpolation. The results, presented in Figure 8, demonstrate that even with random rotations, the model can achieve a high level of invariance. However, a slight accuracy drop was observed when using random rotations (83.06%) compared to the canonical quarter turns (87.48%), which can be due to the loss of details resulting from interpolation.

F. Downstream tasks

To demonstrate the versatility of our pre-trained model and its applicability to tasks beyond classification, we extended its use to semantic segmentation. Specifically, we fine-tuned the model—originally pre-trained on the *mini*ImageNet dataset—to perform segmentation tasks on ADE20K dataset [45]. ADE20K includes 150 fine-grained semantic categories and comprises 20,000 training images, 2,000 validation images, and 3,000 test images. For optimization, we use AdamW with a weight decay of 0.01 and a total batch size of 2 per GPU. The training schedule features an initial learning rate of 6×10^{-5}, linear decay, a 1500-iteration linear warmup, and a total of 160,000 iterations. We apply standard data augmentations such as random horizontal flipping, random scaling within a ratio range of 0.5 to 2.0, and

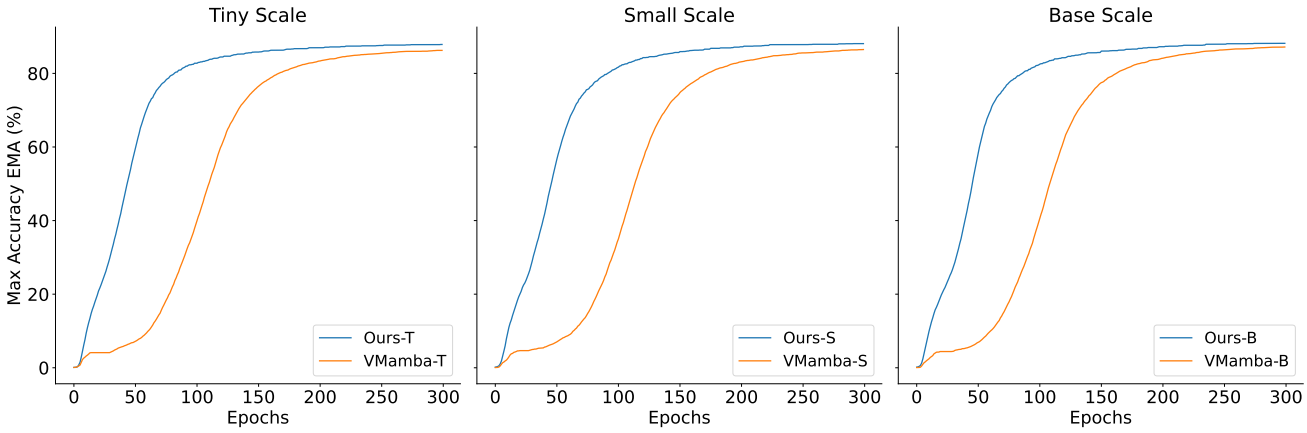

Figure 9. Training Speed Comparison.

random photometric distortion.

Table 3 shows that our method outperforms VMamba by +2.81 (tiny), +1.68 (small), and +1.49 (base) on single-scale (SS) testing, and by +3.74 (tiny), +2.21 (small), and +0.58 (base) on multi-scale testing. The backbone used for the segmentation task was initialized from a classification checkpoint which was trained on the mini-ImageNet dataset. This explains the relatively lower range of segmentation performance compared to other papers, which often use classification models pre-trained on ImageNet-1k.

Method	**mIoU (SS)**	**mIoU (MS)**
VMamba-T	22.77	23.98
VMamba-S	25.84	27.13
VMamba-B	26.32	28.41
Ours-T	25.58 **(+2.81)**	27.72 **(+3.74)**
Ours-S	27.52 **(+1.68)**	29.34 **(+2.21)**
Ours-B	27.81 **(+1.49)**	28.99 **(+0.58)**

Table 3. Results of semantic segmentation on ADE20K. SS and MS denote single-scale and multi-scale testing, respectively.