## A. Related Works

### A.1. CNN-Based

For a long range of time, CNN-based architectures have dominated the computer vision domain. The prototype of CNN is presented in [31] and after the exciting success of AlexNet [30], a large number of methods adopt CNN architecture for higher performance [20, 52, 55]. Especially the ResNet [20], utilizes a residual connection among layers to alleviate the gradient vanishing. Owing to the hierarchical structure employed in CNNs, rich information can be effectively extracted from localized receptive fields. Yet, despite advantages, CNNs exhibit certain limitations, particularly in terms of capturing global contextual information, and inductive bias both could potentially impede CNNs being applied to downstream tasks.

### A.2. Transformer-Based

The foundation work [66] on Transformer-based architecture proposes the attention mechanism used to extract the relationship among the spatial positional features. Following the groundbreaking achievements of BERT [12] in natural language processing, numerous approaches have leveraged Transformer-based architecture for advanced performance [2, 3, 14, 40, 42, 45, 48, 49, 54, 63, 64, 74], both in natural language processing and computer vision. Notably with self-attention mechanism, Vision Transformer [14] has facilitated long-range dependencies, indicating that Transformers-based architectures have the ability to extract global context understanding.

Despite remarkable achievements across various domains, Transformers do have certain drawbacks that need to be considered. SoftMax$(\frac{Q\times K^T}{\sqrt{k}})V$, is the primary reason for heavy computational demands. Furthermore, the scalability is not only impeded by its quadratic computational complexity, but also by the necessity for extensive datasets and significant memory consumption.

## B. Compare SpiralMLP to CNNs and MHSA

In order to establish a comparison between **Multi-Head Self-Attention** (abbreviated as **MHSA**) and **SpiralMLP**, it is necessary to demonstrate the relationship between MHSA and **Convolutional Neural Networks** (abbreviated as **CNNs**). This is because SpiralMLP, unlike MHSA, does not incorporate self-attention layers and is more closely aligned with CNNs.

By examining the connection between MHSA and CNNs, we can provide a comprehensive understanding of the architectural differences and similarities between MHSA and SpiralMLP. This will enable us to highlight the unique features and advantages of each approach.

Thus we outline the following subsections in this way:

- We first elaborate how SpiralMLP is related to CNNs.
- Then we draw a relation between MHSA and CNNs using the proof of Cordonnier [9].
- Finally we provide the comparison between MHSA and SpiralMLP.

### B.1. How is SpiralMLP related to CNNs?

In order to demonstrate the functionality of CNNs, we define the standard convolution weight matrix as $W^{\text{cnn}} \in \mathbb{R}^{K_H\times K_W\times C_{\text{in}}\times C_{\text{out}}}$, where $K_H$, $K_W$, $C_{\text{in}}$, $C_{\text{out}}$ are the kernel height, kernel width, input channel dimension, output channel dimension, respectively. Given a position $(i, j, :)$ within the feature map $X \in \mathbb{R}^{H\times W\times C_{\text{in}}}$, where $H$, $W$ are the height and weight, and $\Omega = \{(x, y); 0 \le x \le K_W, 0 \le y \le K_H\}$ is defined as a set of coordinates in a rectangular, corresponding to the kernel. Thus the convolution operation is formulated as:

$$\text{Conv}_{i,j,:}(X) = \sum_{(x,y)\in\Omega} X_{i+x,j+y,:}W^{\text{cnn}}_{y,x,:,:} + b^{\text{cnn}} \tag{16}$$

where, $b^{\text{cnn}} \in \mathbb{R}^{C_{\text{out}}}$ is the CNNs bias, the overall output $\text{Conv}_{i,j,:}(\cdot)$ is a vector at each position $(i, j, :)$.

Next, we replace the dot product between $X_{i+x,j+y,:}$ and $W^{\text{cnn}}_{y,x,:,:}$ in Eq. (16) with a summation across the channel dimension to get:

$$\text{Conv}_{i,j,:}(X) = \sum_{(x,y)\in\Omega} \sum_{c=0}^{C_{\text{in}}} X_{i+x,j+y,c}W^{\text{cnn}}_{y,x,c,:} + b^{\text{cnn}} \tag{17}$$

Afterwards we change the order of the summation and obtain:

$$\text{Conv}_{i,j,:}(X) = \sum_{c=0}^{C_{\text{in}}} \sum_{(x,y)\in\Omega} X_{i+x,j+y,c}W^{\text{cnn}}_{y,x,c,:} + b^{\text{cnn}} \tag{18}$$

To relate Eq. (18) to the Spiral FC (Eqs. (3) to (5)), we first create a function:

$$g(x, y, c) = \begin{cases} 1, & \text{if } x = \phi_i(c) \text{ and } y = \phi_j(c) \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

Then we apply the Eq. (19) on the kernel weights $W^{\text{cnn}}$ of the CNNs to obtain:

$$\widetilde{W}^{\text{cnn}}_{y,x,c,:} = g(x, y, c) \cdot W^{\text{cnn}}_{y,x,c,:} \tag{20}$$

Subsequently, we substitute the weights defined in Eq. (18) with the weights in Eq. (20):

$$\begin{aligned}
&\text{Conv}_{i,j,:}(X)\\
&= \sum_{c=0}^{C_{\text{in}}} \sum_{(x,y)\in\Omega} X_{i+x,j+y,c} \widetilde{W}^{\text{cnn}}_{y,x,c,:} + b^{\text{cnn}}\\
&= \sum_{c=0}^{C_{\text{in}}} \sum_{(x,y)\in\Omega} g(x,y,c) X_{i+x,j+y,c} W^{\text{cnn}}_{y,x,c,:} + b^{\text{cnn}} \qquad (21)\\
&= \sum_{c=0}^{C_{\text{in}}} X_{i+\phi_i(c),j+\phi_j(c),c} W^{\text{spiral}}_{c,:} + b^{\text{spiral}}\\
&= \text{Spiral FC}_{i,j,:}(X)
\end{aligned}$$

where, $W^{\text{spiral}}$ and $b^{\text{spiral}}$ are already introduced in Eq. (3).

Consequently, we have demonstrated how the Spiral FC can be derived from convolutions. It is noteworthy that Eqs. (19) to (21) reveal that the Spiral FC assumes sparsity within the convolution operation, indicating its potential advantages and distinctive characteristics compared to traditional dense convolution layers.

### B.2. How is MHSA related to CNNs?

To formulate, the multi-head self-attention weight matrix for each head is denoted as $W^{\text{mhsa},h} \in \mathbb{R}^{C_{\text{in}}\times C_{\text{out}}}$, and the bias is denoted as $b^{\text{mhsa}} \in \mathbb{R}^{C_{\text{out}}}$. Furthermore, $h$ refers to each individual head and $N_h$ represents the total number of heads. The process of MHSA applied to the input $X \in \mathbb{R}^{H\times W\times C_{\text{in}}}$ on position $(i,j,:)$ can be arranged as:

$$\text{MHSA}_{i,j,:}(X) = \sum_{h=1}^{N_h} X_{:,:,C^h_{\text{in}}} W^{\text{mhsa},h}_{C^h_{\text{in}},:} + b^{\text{mhsa}} \qquad (22)$$

where, $C^h_{\text{in}}$ illustrates the division of the input channel corresponding to the $h^{\text{th}}$ head, and it is incorporated into the computation for each head $h$.

While in the work [9], it is demonstrated that MHSA can be represented in a manner similar to CNNs:

$$\begin{aligned}
&\text{MHSA}_{i,j,:}(X)\\
&= \sum_{h=1}^{N_h} X_{i+\Delta_i(h),j+\Delta_j(h),C^h_{\text{in}}} W^{\text{mhsa},h}_{C^h_{\text{in}},:} + b^{\text{mhsa}} \qquad (23)\\
&= \sum_{h=1}^{N_h} \sum_{c=0}^{C^h_{\text{in}}} X_{i+\Delta_i(h),j+\Delta_j(h),c} W^{\text{mhsa},h}_{c,:} + b^{\text{mhsa}}
\end{aligned}$$

where, all relative positional shifts for a kernel of size $\sqrt{N_h} \times \sqrt{N_h}$ at position $(i,j)$ are contained in $\{\Delta_i(h), \Delta_j(h)\} = \{(-1,0),(-1,1),(-1,2),\ldots\}$.

### B.3. How is SpiralMLP related to MHSA?

Through a similar technique to Eq. (20), we establish a connection between MHSA and SpiralMLP through CNNs. Firstly, we augment $W^{\text{mhsa},h}$ by multiplying with the Eq. (19) to obtain:

$$\widetilde{W}^{\text{mhsa},h}_{c,:} = g(x,y,c) \cdot W^{\text{mhsa},h}_{c,:} \qquad (24)$$

where, $\widetilde{W}^{\text{mhsa},h}$ is dependent on the position along the kernel height, width and input channel. Substitute it into Eq. (23):

$$\begin{aligned}
&\text{MHSA}_{i,j,:}(X)\\
&= \sum_{h=1}^{N_h} \sum_{c=0}^{C^h_{\text{in}}} X_{i+\Delta_i(h),j+\Delta_j(h),c} \widetilde{W}^{\text{mhsa},h}_{c,:} + b^{\text{mhsa}}\\
&= \sum_{h=1}^{N_h} \sum_{c=0}^{C^h_{\text{in}}} X_{i+\phi_i(c),j+\phi_j(c),c} W^{\text{mhsa},h}_{c,:} + b^{\text{mhsa}} \qquad (25)\\
&= \sum_{c=0}^{C_{\text{in}}} X_{i+\phi_i(c),j+\phi_j(c),c} W^{\text{spiral}}_{c,:} + b^{\text{spiral}}\\
&= \text{Spiral FC}_{i,j,:}(X)
\end{aligned}$$

where, $W^{\text{spiral}}$ and $b^{\text{spiral}}$ are already introduced in Eq. (3).

Eq. (25) demonstrates the relationship between Spiral FC and MHSA. Drawing a parallel with the relationship between CNNs and Spiral FC, we can conclude that Spiral FC exhibits a significantly sparser receptive field compared to MHSA. This highlights the distinctive characteristic of Spiral FC in terms of its sparse attention mechanism compared to the dense attention mechanism of MHSA.

## C. Model Zoo and Training Details

### C.1. Model Zoo Configurations and Performances

We implement five models according to PVT style, named **SpiralMLP B1** to **B5**, and three models based on Swin-style, named as **SpiralMLP-T**, **SpiralMLP-S**, **SpiralMLP-B**.

SpiralMLP variants in PVT-style are detailed in Tab. 8. Each stage contains multiple Spiral Blocks with uniform configurations, where the parameters $S$, $E$, $C$, and $L$ denote the shift size, expansion ratio, channel dimension, and the number of layers in each stage, respectively.

SpiralMLP variants in Swin-style are shown in Tab. 9 and Fig. 7. The input image dimension is $224 \times 224$. The process 'concat $n \times n$' refers to the concatenation of features from $n \times n$ neighboring features in a patch. This operation effectively downsamples the feature map by a factor of $n$. The notation '96-d' represents a linear layer whose output dimension is 96. In the term [(3,2), 96] indicates that $A_{\max} = 3$, $k = 2$, $C_{\text{out}} = 96$.

|  | Output Size | Layer Name | PVT-Style |  |  |  |  |
|---|---|---|---|---|---|---|---|
|  |  |  | B1 | B2 | B3 | B4 | B5 |
| Stage1 | $\frac{H}{4} \times \frac{W}{4}$ | Overlapping Patch Embedding | $S_1 = 4$ |  |  |  |  |
|  |  |  | $C_1 = 64$ |  |  |  | $C_1 = 96$ |
|  |  | SpiralMLP Block | $E_1 = 4$ $L_1 = 2$ | $E_1 = 4$ $L_1 = 2$ | $E_1 = 4$ $L_1 = 3$ | $E_1 = 4$ $L_1 = 3$ | $E_1 = 4$ $L_1 = 3$ |
| Stage2 | $\frac{H}{8} \times \frac{W}{8}$ | Overlapping Patch Embedding | $S_2 = 2$ |  |  |  |  |
|  |  |  | $C_2 = 128$ |  |  |  | $C_2 = 192$ |
|  |  | SpiralMLP Block | $E_2 = 4$ $L_2 = 2$ | $E_2 = 4$ $L_2 = 3$ | $E_2 = 4$ $L_2 = 4$ | $E_2 = 4$ $L_2 = 8$ | $E_2 = 4$ $L_2 = 4$ |
| Stage3 | $\frac{H}{16} \times \frac{W}{16}$ | Overlapping Patch Embedding | $S_3 = 2$ |  |  |  |  |
|  |  |  | $C_3 = 320$ |  |  |  | $C_3 = 384$ |
|  |  | SpiralMLP Block | $E_3 = 4$ $L_3 = 4$ | $E_3 = 4$ $L_3 = 10$ | $E_3 = 4$ $L_3 = 18$ | $E_3 = 4$ $L_3 = 27$ | $E_3 = 4$ $L_3 = 24$ |
| Stage4 | $\frac{H}{32} \times \frac{W}{32}$ | Overlapping Patch Embedding | $S_4 = 2$ |  |  |  |  |
|  |  |  | $C_4 = 512$ |  |  |  | $C_4 = 768$ |
|  |  | SpiralMLP Block | $E_4 = 4$ $L_4 = 2$ | $E_4 = 4$ $L_4 = 3$ | $E_4 = 4$ $L_4 = 3$ | $E_4 = 4$ $L_4 = 3$ | $E_4 = 4$ $L_4 = 3$ |
| Parameters (M) |  |  | 14 | 24 | 34 | 46 | 68 |
| FLOPs (G) |  |  | 2.0 | 3.6 | 5.6 | 8.2 | 11.3 |
| Accuracy Top-1 (%) |  |  | 79.8 | 81.9 | 83.4 | 83.8 | 84.0 |

Table 8. Configurations of SpiralMLP variants in PVT-style.

### C.2. Experimental Setup for Image Classification

We train our model on ImageNet-1k [50], which contains about 1.2M images. The accuracy report is standard Top-1 accuracy on the validation set containing roughly 50k images, evenly distributed among 1000 categories. The code of our implementation is inspired by CycleMLP as well as DeiT and is written in Pytorch. Our augmentation policy includes RandAugment [10], Mixup [80], Cutmix [78], random erasing [81] and stochastic depth [24]. The optimizer used is AdamW [44] with learning rate of $5 \times$ 10e-4 with momentum of 0.9 and weight decay of $5 \times$ 10e-2.

### C.3. Experimental Setup for Object detection and Instance Segmentation

For object detection and instance segmentation experiments, we train our model on COCO [38] containing 118k training images together with 5k validation images. We employ the mmdetection toolbox [5] and use RetinaNet [37] and Mask R-CNN [19] as the framworks with SpiralMLP variants as backbones. The weights are initialized with the pretrained weights from ImageNet-1k and additional added layers are initialized using Xavier [16] initialization. The optimizer is AdamW [44] with a learning rate of $1 \times$ 10e-4. The images are resized to 800 for the shorter side and a maximum limit of 1333 pixels for height and width of the image. The model is trained on 4 A100 GPUs with the batch size of 32 for 12 epochs.

### C.4. Experimental Setup for Semantic Segmentation

The semantic segmentation is conducted on ADE20K [82], which consists of 20k training images and 2k validation images. The frameworks used for this purpose are Semantic FPN [26] and UperNet [70], employing SpiralMLP with ImageNet-1k pretrained weights as the backbones. For optimization, the AdamW [44] optimizer is chosen. In the case of Mask R-CNN, the optimizer is set with an initial learning rate of 0.0001 and a weight decay of 0.05. For Semantic FPN, the same initial learning rate of 0.0001 is used, but the weight decay is lower, at 0.0001. The training process is carried out on 4 A100 GPUs. A batch size of 32 is maintained throughout, and the model undergoes training for 12 epochs.

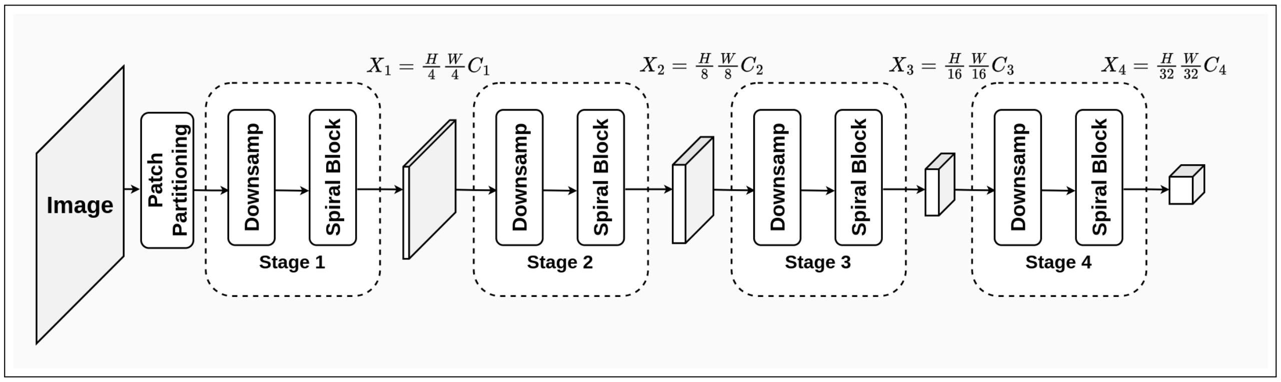
## D. Merge Head

### D.1. How does the Merge Head Work?

In Eq. (7), Merge Head employs a trainable weights $a$ to determine the contribution of $X^{\text{self}} \in \mathbb{R}^{H \times W \times C_{\text{out}}}$, $X^{\text{cross}} \in \mathbb{R}^{H \times W \times C_{\text{out}}}$, from Channel FC and Spiral FC, respectively. For a clearer understanding, we present the tensor shape of each step in Merge Head, as shown in Tab. 10. Rather than merely employing a straightforward addition, the merge head innovatively takes into account the inputs themselves to formulate the trainable weights $a$. This approach allows for a more dynamic and input-responsive

| | Output Size | SpiralMLP-T | SpiralMLP-S | SpiralMLP-B |
|---|---|---|---|---|
| Stage1 | $4 \times (56 \times 56)$ | concat $4 \times 4$, 64-d, LN | concat $4 \times 4$, 96-d, LN | concat $4 \times 4$, 96-d, LN |
| | | [(3,2), 64]$\times$2 | [(3,2), 96]$\times$3 | [(3,2), 96]$\times$3 |
| Stage2 | $8 \times (28 \times 28)$ | concat $2 \times 2$, 128-d, LN | concat $2 \times 2$, 192-d, LN | concat $2 \times 2$, 192-d, LN |
| | | [(3,2), 128]$\times$2 | [(3,2), 192]$\times$4 | [(3,2), 192]$\times$4 |
| Stage3 | $16 \times (14 \times 14)$ | concat $2 \times 2$, 320-d, LN | concat $2 \times 2$, 384-d, LN | concat $2 \times 2$, 384-d, LN |
| | | [(3,2), 320]$\times$6 | [(3,2), 384]$\times$18 | [(3,2), 384]$\times$24 |
| Stage4 | $32 \times (7 \times 7)$ | concat $2 \times 2$, 512-d, LN | concat $2 \times 2$, 768-d, LN | concat $2 \times 2$, 768-d, LN |
| | | [(3,2), 512]$\times$2 | [(3,2), 768]$\times$3 | [(3,2), 768]$\times$3 |
| Parameters (M) | | 15 | 56 | 67 |
| FLOPs (G) | | 2.3 | 9.1 | 11.0 |
| Accuracy Top-1 (%) | | 79.6 | 83.3 | 83.6 |

Table 9. Configurations of SpiralMLP variants in Swin-style.



$X_1 = \frac{H}{4}\frac{W}{4}C_1$
$X_2 = \frac{H}{8}\frac{W}{8}C_2$
$X_3 = \frac{H}{16}\frac{W}{16}C_3$
$X_4 = \frac{H}{32}\frac{W}{32}C_4$
Image
Patch Partitioning
Downsamp
Spiral Block
Stage 1
Stage 2
Stage 3
Stage 4

Figure 7. The architecture of SpiralMLP in Swin-style.

| Step | Tensor Shape |
|---|---|
| $(X^{\text{self}}, X^{\text{cross}})$ | $(\mathbb{R}^{H \times W \times C_{\text{out}}}, \mathbb{R}^{H \times W \times C_{\text{out}}})$ |
| $\mathcal{F}(\cdot)$ | $\mathbb{R}^{H \cdot W \times C_{\text{out}}}$ |
| Average | $\mathbb{R}^{1 \times C_{\text{out}}}$ |
| $W^{\text{merge}}$ | $\mathbb{R}^{2 \times C_{\text{out}}}$ |

Table 10. Tensor shape of each step in Merge Head.

weight adjustment, enhancing the effectiveness of the merging process.

### D.2. Complexity of Merge Head.

Two inputs are both of shape $\mathbb{R}^{H \times W \times C}$. The addition has $O(HW)$. While the reshaping operation $\mathcal{F}(\cdot)$ is implemented by `torch.flatten()` with $O(1)$. The average calculation is implemented by `torch.mean()` with $O(HW)$. The multiplication involves $W^{\text{merge}} \in \mathbb{R}^{2,1}$ contributes $O(1)$. Therefore, the total complexity of this process is linear $O(HW)$.

## E. Why Spiral FC works?

Eq. (21) indicates that SpiralMLP, along with other criss-cross MLPs, can be effectively implemented using a specialized convolution layer. Currently, deformable convolution [11] emerges as the optimal method for this purpose.

In the given scenario as shown in Fig. 8, points are identified as $(o_x, o_y)$, $(\mu_x, \mu_y)$, $(\nu_x, \nu_y)$, and $(\mu_x, \nu_y)$, all located on feature map $X$. Specifically, when $(o_x, o_y)$ is considered the reference point, the points $(\mu_x, \mu_y)$ and $(\nu_x, \nu_y)$ are categorized into set $P$. This set includes pairs that are either horizontally or vertically aligned with the reference point $(o_x, o_y)$. The point $(\mu_x, \nu_y)$ falls into set $Q$, which comprises pairs that can be located at any position within the feature map $X$. Then the set $P$ and set $Q$ are defined as:

$$P = \{(\mu_x, \mu_y) \in \mathbb{R}^2 | (\mu_x = o_x \vee \mu_y = o_y)\} \tag{26}$$
$$Q = \{(\mu_x, \nu_y) \in \mathbb{R}^2\} \tag{27}$$

We further denote the $Y_C \in \mathbb{R}^{H \times W \times C}$ as the output of the convolution layer, while $X \in \mathbb{R}^{H \times W \times C}$ is still the feature map, this is similar to Eq. (21):

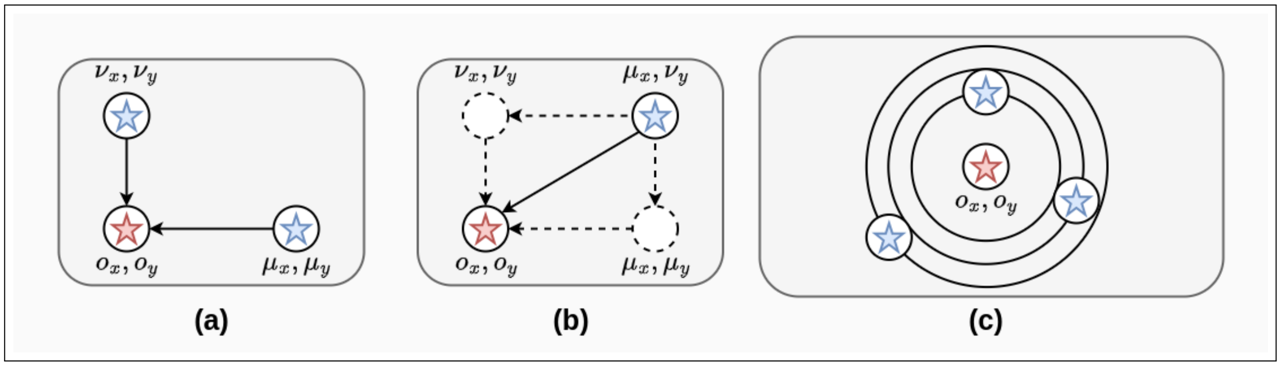


Figure 8. Spiral FC covers a more comprehensive receptive field.

$$Y_C = \sum_{c=0}^{C_{\text{in}}} X_{i+\bar{\phi}_i(c), j+\bar{\phi}_j(c), c} W_{c,:} + b \tag{28}$$

where, $\bar{\phi}_i(\cdot), \bar{\phi}_j(\cdot)$ denote the universal offset functions.

We take two architectures with criss-cross offset functions as examples. When using CyelcMLP [6] as an example, the offset function is updated into:

$$\phi_i^{\text{cycle}}(c) = (c \mod S_H) - 1 \tag{29}$$

$$\phi_j^{\text{cycle}}(c) = (\lfloor \frac{c}{S_H} \rfloor \mod S_W) - 1 \tag{30}$$

where, $S_H, S_W$ are the predefined step size.

When using ASMLP [36], the offset function is updated into:

$$\phi_i^{\text{as}}(c) = \lfloor \frac{c}{C/s} \rfloor - \lfloor \frac{s}{2} \rfloor \cdot d \tag{31}$$

$$\phi_j^{\text{as}}(c) = \lfloor \frac{c}{C/s} \rfloor - \lfloor \frac{s}{2} \rfloor \cdot d \tag{32}$$

where, $s$ is the shift size and $d$ is the dilation rate.

In Fig. 8 (a), points $(\mu_x, \mu_y)$ and $(\nu_x, \nu_y)$ from set $P$ are reachable by the offset functions $\phi^{\text{cycle}}(\cdot)$, and $\phi^{\text{as}}(\cdot)$ due to their placement along the horizontal or vertical axes. However, as illustrated in Fig. 8 (b), points that do not lie on these axes pose a challenge for the criss-cross methodology, which inherently lacks the capability to capture such spatial information effectively.

To address this limitation, Fig. 8 (c) suggests the adoption of a predefined multi-helix offset approach. This method, while effective, still offers room for optimization to achieve model efficiency in terms of size and computational speed. A viable solution is the refinement of the multi-helix offset into a spiral-like offset function. This adjustment not only enables the model to recognize points from set $Q$—those not aligned with the horizontal or vertical axes—but also maintains a compact model architecture and ensures rapid processing speeds.

| Model | Params(M) | Throughput (/sec) | Wall-Clock (s) |
|---|---|---|---|
| Spiral-B1 | 14 | 425 | 22.8 |
| Spiral-B2 | 24 | 253 | 30.5 |
| Spiral-B3 | 34 | 143 | 52.2 |
| Spiral-B4 | 46 | 94 | 72.5 |
| Spiral-B5 | 68 | 89 | 79.2 |
| Cycle-B1 [6] | 15 | 347 | 23.8 |
| Cycle-B2 | 27 | 207 | 36.0 |
| Cycle-B3 | 38 | 117 | 60.4 |
| Cycle-B4 | 52 | 79 | 84.0 |
| Cycle-B5 | 76 | 75 | 92.0 |
| ResNet-152 [20] | 60 | 152 | 45.7 |
| Vit-B/16 [14] | 86 | 102 | 75.2 |
| Deit-B/16 [61] | 86 | 97 | 78.3 |

Table 11. Throughput and wall-clock test.

## F. Latency & Throughput Analysis

As shown in Tab. 11, throughput testing is conducted on a single NVIDIA A100 GPU, with a batch size of 32 and image resolution of 3x224x224; wall-clock testing is conducted on ImageNet-1k with a single NVIDIA A100 GPU, with a batch size of 64, the number of total batches to 100 and image resolution of 3x224x224. Furthermore, we evaluate the latency speeds on one A100 across various architectures for input resolutions of $224^2$, $384^2$, and $512^2$. These results are detailed in Tab. 12.

It is evident that SpiralMLP is faster than other MLP-based models when considering the scale of model size. Additionally, the PVT [67] showcases even faster processing speeds, mainly due to its utilization of dot product operations, inherently benefiting from accelerated computation. The models used for evaluation are directly extracted from their official implementations.

## G. Resolution Compatibility

We compare the image classification models resolution compatibility with Top-1 accuracy on ImageNet-1k, with models pre-trained on 224 x 224 images and tested at var-

| Model | $224^2$ | $384^2$ | $512^2$ | Params(M) | Model | $224^2$ | $384^2$ | $512^2$ | Params(M) |
|---|---|---|---|---|---|---|---|---|---|
| **Spiral-B1** | 11.74 | 11.83 | 11.52 | 14 | Cycle-B1 [6] | 12.12 | 12.06 | 12.23 | 15 |
| **Spiral-B2** | 20.81 | 21.11 | 20.29 | 24 | Cycle-B2 | 21.66 | 21.36 | 21.12 | 27 |
| **Spiral-B3** | 32.33 | 32.53 | 31.38 | 34 | Cycle-B3 | 33.23 | 32.89 | 32.29 | 38 |
| **Spiral-B4** | 47.00 | 47.39 | 45.95 | 46 | Cycle-B4 | 47.86 | 48.29 | 46.73 | 52 |
| **Spiral-B5** | 39.22 | 39.34 | 38.34 | 68 | Cycle-B5 | 41.00 | 40.87 | 39.59 | 76 |
| ATM-xT [69] | 15.01 | 15.32 | 14.86 | 15 | Wave-T-dw [58] | 16.91 | 15.83 | 16.01 | 15 |
| ATM-T | 25.83 | 26.66 | 25.43 | 27 | Wave-T | 18.56 | 17.48 | 17.47 | 17 |
| ATM-S | 37.47 | 38.41 | 37.21 | 39 | Wave-S | 31.97 | 30.76 | 30.61 | 31 |
| ATM-B | 55.45 | 55.85 | 53.39 | 52 | Wave-M | 48.98 | 47.43 | 46.94 | 44 |
| ATM-L | 47.24 | 45.98 | 44.78 | 76 | Wave-B | 41.99 | 40.50 | 40.27 | 64 |
| PVT-Tiny [67] | 8.54 | 8.72 | 8.55 | 13 | PVTv2-B1 [68] | 9.08 | 9.04 | 9.12 | 14 |
| PVT-Small | 15.73 | 15.75 | 15.62 | 25 | PVTv2-B2 | 17.14 | 17.45 | 17.21 | 25 |
| PVT-Medium | 26.78 | 27.48 | 26.71 | 44 | PVTv2-B3 | 30.57 | 30.21 | 30.32 | 45 |
| PVT-Large | 39.28 | 39.75 | 38.68 | 61 | PVTv2-B4 | 44.77 | 44.73 | 43.51 | 63 |
| - | - | - | - | - | PVTv2-B5 | 56.64 | 56.88 | 55.19 | 82 |

Table 12. Latency analysis measured in *milliseconds(ms)* on one A100. A single image with differing resolutions serves as the input

| Model | Params(M) | 128 | 224 | 256 | 384 |
|---|---|---|---|---|---|
| Spiral-B5 | 68 | 77.6 | 84.0 | 83.8 | 83.4 |
| Cycle-B [6] | 88 | 77.0 | 83.2 | 83.1 | 82.7 |
| ViT-B/16 [14] | 86 | - | 81.0 | 81.6 | 79.0 |
| ResNet-152 [20] | 60 | 68.7 | 78.3 | 78.4 | 77.4 |
| DeiT-B/16 [61] | 87 | - | 81.7 | - | 82.5 |

Table 13. Resolution compatibility comparison.

ious other resolutions without additional fine-tuning, as shown in Tab. 13.

## H. Spiral in AttentionViz

In the work AttentionViz [75], the authors introduce a visualization tool for analyzing transformer models, particularly focusing on the interaction between keys and queries distributions across various heads and layers. A notable discovery in their research is the identification of a spiral-like pattern. This pattern suggests that the keys and queries in transformers are spatially aligned in a manner resembling a spiral. The phenomenon results from position vectors that are generated using trigonometric functions, mapping onto a helical curve in a high-dimensional space. In linguistics models, this refers to the arrangement of words or parts of words; while in vision transformers, it relates to the organization of pixel patches.

This spiral distribution is reflective of the initial ordering vector given to transformers, illustrating how positional information is embedded within the model. Furthermore, the research reveals that transformer heads displaying a spiral shape or having clusters of queries/keys tend to yield more dispersed search results.