Vision Permutator: A Permutable MLP-Like Architecture for Visual Recognition

Qibin Hou Zihang Jiang Li Yuan Ming-Ming Cheng Shuicheng Yan Jiashi Feng

Abstract—In this paper, we present Vision Permutator, a conceptually simple and data efficient MLP-like architecture for visual recognition. By realizing the importance of the positional information carried by 2D feature representations, unlike recent MLP-like models that encode the spatial information along the flattened spatial dimensions, Vision Permutator separately encodes the feature representations along the height and width dimensions with linear projections. This allows Vision Permutator to capture long-range dependencies and meanwhile avoid the attention building process in transformers. The outputs are then aggregated in a mutually complementing manner to form expressive representations. We show that our Vision Permutators are formidable competitors to convolutional neural networks (CNNs) and vision transformers. Without the dependence on spatial convolutions or attention mechanisms, Vision Permutator achieves 81.5% top-1 accuracy on ImageNet without extra large-scale training data (e.g., ImageNet-22k) using only 25M learnable parameters, which is much better than most CNNs and vision transformers under the same model size constraint. When scaling up to 88M, it attains 83.2% top-1 accuracy, greatly improving the performance of recent state-of-the-art MLP-like networks for visual recognition. We hope this work could encourage research on rethinking the way of encoding spatial information and facilitate the development of MLP-like models. PyTorch/MindSpore/Jittor code is available at https://github.com/Andrew-Qibin/VisionPermutator.

Index Terms—Vision permutator, permutator, image classification, multi-layer perceptron, deep neural network

1 Introduction

Recent studies [1], [2] have shown that pure multi-layer perceptron based networks perform well in ImageNet classification [3]. Compared to convolutional neural networks (CNNs) and vision transformers that employ spatial convolutions or self-attention layers to encode spatial information, MLP-like networks (a.k.a., MLPs) make use of pure fully-connected layers (or called 1×1 convolutions) and hence are more efficient in both training and inference [1]. However, the good performance of MLPs in image classification largely benefits from training on large-scale datasets (e.g., ImageNet-22K and JFT-300M). Without the support of sufficiently large amount of training data, their performance still lags largely behind CNNs [4], [5], [6] and vision transformers [7], [8], [9].

In this work, we are interested in exploiting the potential of MLPs with using merely the ImageNet-1k data for training and target data-efficient MLPs. To this end, we propose the *Vision Permutator* architecture. Specially, Vision Permutator innovates the existing MLP architectures by presenting a new layer structure that can more effectively encode spatial information based on the basic matrix multiplication routine. Unlike current MLP-like models, such as Mixer [1] and ResMLP [2], that encode spatial information by flattening the spatial dimensions first and then conducting linear projection along the spatial dimension (i.e., operating on tokens with shape "tokens×channels"), leading to the loss of positional information carried by 2D feature representations, Vision Permutator maintains the original spatial dimensions of the input tokens and separately encodes spatial information along the height and width dimensions to preserve positional information. This makes our Vision Permutator quite different from the existing MLP-like models.

To be specific, our Vision Permutator begins with a similar tokenization operation to vision transformers, which uniformly splits the input image into small patches and then maps them to token embeddings with linear projections, as depicted in Figure 1. The resulting token embeddings with shape "height×width×channels" are then fed into a sequence of Permutator blocks, each of which consists of a Permute-MLP for spatial information encoding and a Channel-MLP for channel information mixing. The Permute-MLP layer, as depicted in Figure 2, consists of three independent branches, each of which encodes features along a specific dimension, *i.e.,* the height, width or channel dimension. Compared to existing MLP-like models that mix the two spatial dimensions into one, our Vision Permutator separately processes the token representations along these dimensions, resulting in more discriminative token representations, which we will demonstrate essential for visual recognition in our experiment section.

Experiments show that our Vision Permutator can largely improve the classification performance of existing MLP-like models. Taking the small-sized Vision Permutator with 25M parameters as an example, it attains 81.5% top-1 accuracy on ImageNet without any extra training data. The result is better than most of the classic CNN-based models, such as ResNet-50d (79.5%), SE-ResNeXt-50 (79.9%), and the strong ResNeSt-50 (81.1%). Scaling up the model to 55M and 88M, we can further improve the results and achieve 82.7%

- *Q. Hou and M.M. Cheng are with School of Computer Science, Nankai University, Tianjin, China. (andrewhoux@gmail.com, cmm@nankai.edu.cn)*
- *Z. Jiang is with Department of Electrical and Computer Engineering, National University of Singapore, Singapore. (jzh0103@gmail.com)*
- *L. Yuan is with School of Electronic and Computer Engineering, Peking University, China. (ylustcnus@gmail.com)*
- *S. Yan and J. Feng are with Sea AI Lab, Singapore. ({yansc, fengjs}@sea.com)*

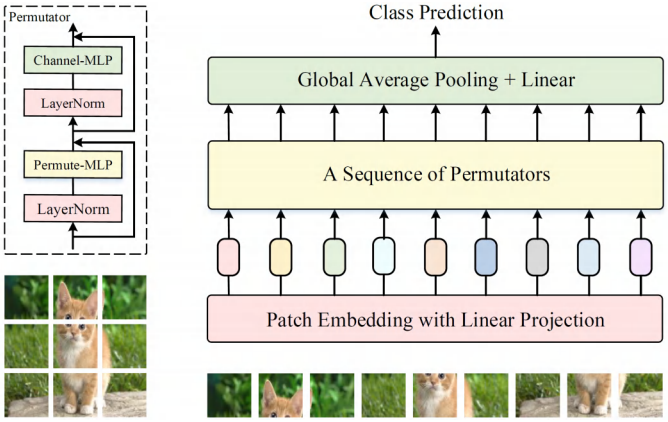

Fig. 1: Basic architecture of the proposed Vision Permutator. The evenly divided image patches are tokenized with linear projection first and then fed into a sequence of Permutators for feature encoding. A global average pooling layer followed by a fully-connected layer is finally used to predict the class.

and 83.2% top-1 accuracy on ImageNet, respectively.

2 Related Work

Modern deep neural networks for image classification have three prominent families: convolutional neural networks (CNNs), vision transformers (ViTs), and multi-layer perceptron based models (MLPs). In the following, we will briefly describe the development trend of each type of networks and state the differences of the proposed Vision Permutator from previous work.

CNNs, as the de-facto standard networks in computer vision for years, have been deeply studied. Early CNN models, such as AlexNet [10] and VGGNet [11], mostly adopt structures with a stack of spatial convolutions (with kernel size ≥ 3) and pooling operations. Later, ResNets and their variants [12], [13], [14] introduce skip connection and building blocks with bottleneck structure into CNNs, enabling training very deep networks possible. Inceptions [15], [16] renovate the design of traditional building block structure and utilize multiple parallel paths of sets of specialized filters. Attention mechanisms [17], [18], [19], [20], [21], [22], [23] break through the limitations of convolutions in capturing local features. Our work can also be regarded as a special CNN. Different from previous CNNs that globally aggregate the locally captured features with spatial convolutions, our Vision Permutator is composed of pure 1×1 convolutions but can encode global information.

Our work is also related to vision transformers [24]. Unlike CNNs that exploit local convolutions to encode spatial information, vision transformers takes advantage of the self-attention mechanism to capture global information and have been the prevailing research direction in image classification recently. Since then, a great number of transformer-based classification models appear, aiming at advancing the original vision transformer by either introducing locality [9], [25], [26], [27], [28], [29], or scaling the depth [8], [30], or tailoring powerful optimization strategies [7]. In addition, there are also some works aiming at improving the self-attention mechanism. For example, Ho et al. and Wang et al. [23], [31] leverages axial attention to process representations by factorizing multidimensional self-attention into multiple 1D self-attentions. Such an approach reduces the computational cost but it still relies on self-attention. Different from the aforementioned methods, our Vision Permutator eliminates the dependence on self-attention and hence is more efficient.

Very recently, there are also some work [1], [2], [32], [33] targeting at developing pure MLP-like models for ImageNet classification. To encode rich spatial information with MLPs, these methods flatten the spatial dimensions and treat the three-dimensional (height, width, and channel) token representations as a two-dimensional input table. Differently, our Vision Permutator operates on three-dimensional feature representations and encodes spatial information separately along the height and width dimensions. We will show the advantages of the proposed Vision Permutator over existing MLP-like models in our experiment section.

3 Vision Permutator

The basic architecture of the proposed Vision Permutator can be found in Figure 1. Our network takes an image of size 224×224 as input and uniformly splits it into a sequence of image patches (14×14 or 7×7). All the patches are then mapped into linear embeddings (or called tokens) using a shared linear layer as [1]. We next feed all the tokens into a sequence of Permutators to encode both spatial and channel information. The resulting tokens are finally averaged along the spatial dimensions, followed by a fully-connected layer for class prediction. In the following, we will detail the proposed Permutator block and the network settings.

3.1 Permutator

A diagrammatic illustration of the proposed Permutator block can be found at the top-left corner of Figure 1. As can be seen, regardless of the LayerNorms and the skip connections, our Permutator consists of two components: Permute-MLP and Channel-MLP, which are responsible for encoding spatial information and channel information, respectively. The Channel-MLP module shares a similar structure to the feed forward layer in Transformers [34] that comprises two fully-connected layers with a GELU activation in the middle. For spatial information encoding, unlike the recent Mixer [1] that conducts linear projection along the spatial dimension with respect to all the tokens, we propose to separately process the tokens along the height and width dimensions. Mathematically, given an input C-dim tokens $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, the formulation of Permutator can be written as follows:

$$\mathbf{Y} = \text{Permute-MLP}(\text{LN}(\mathbf{X})) + \mathbf{X}, \quad (1)$$

$$\mathbf{Z} = \text{MLP}(\text{LN}(\mathbf{Y})) + \mathbf{Y}, \quad (2)$$

where, LN refers to LayerNorm. The output $\mathbf{Z}$ will serve as the input to the next Permutator block until the last one.

Permute-MLP: The visual illustration of the proposed Permute-MLP can be found in Figure 2. Unlike vision transformers [7], [24], [35] and Mixer [1] that receive an

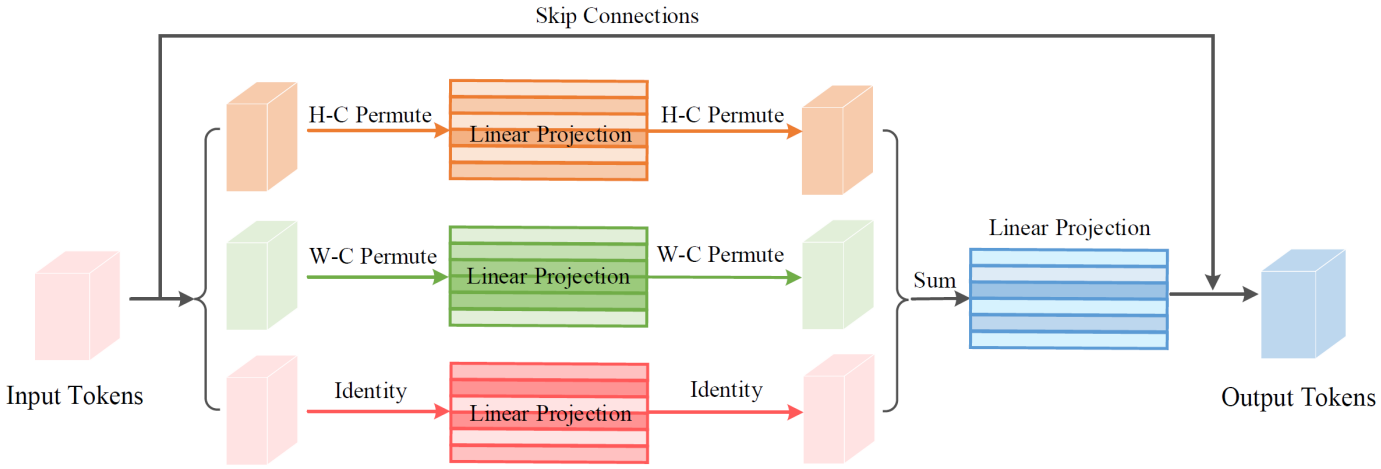

Fig. 2: Basic structure of the proposed Permute-MLP layer. The proposed Permute-MLP layer contains three branches that are responsible for encoding features along the height, width, and channel dimensions, respectively. The outputs from the three branches are then combined using element-wise addition, followed by a fully-connected layer for feature fusion.

Algorithm 1: Code for Permute-MLP (PyTorch-like)

```
# H: height, W: width, C: channel, S: number of segments
# x: input tensor of shape (H, W, C)

def init():
    proj_h = nn.Linear(C, C)    # height dimension
    proj_w = nn.Linear(C, C)    # width dimension
    proj_c = nn.Linear(C, C)    # channel information
    proj   = nn.Linear(C, C)    # fusion

###################### forward ########################
def permute_mlp(x):
    N = C // S
    x_h = x.reshape(H, W, N, S)
    x_h = x_h.permute(2, 1, 0, 3).reshape(N, W, H*S)
    x_h = self.proj_h(x_h).reshape(N, W, H, S)
    x_h = x_h.permute(2, 1, 0, 3).reshape(H, W, C)

    x_w = x.reshape(H, W, N, S)
    x_w = x_w.permute(0, 2, 1, 3).reshape(H, N, W*S)
    x_w = self.proj_w(x_w).reshape(H, N, W, S)
    x_w = x_w.permute(0, 2, 1, 3).reshape(H, W, C)

    x_c = self.proj_c(x)

    x = x_h + x_w + x_c
    x = self.proj(x)
    return x
```

input of two dimensions ("tokens×channels," *i.e.*, $HW \times C$), Permute-MLP accepts 3-dimensional token representations. As shown in Figure 2, our Permute-MLP consists of three branches, each of which is in charge of encoding information along either the height, or width, or channel dimension. The channel information encoding is simple as we only need a fully-connected layer with weights $\mathbf{W}_C \in \mathbb{R}^{C\times C}$ to perform a linear projection with respect to the input $\mathbf{X}$, yielding $\mathbf{X}_C$. In the following, we will describe how to encode spatial information by introducing a segment-wise permutation operation between dimensions.

Suppose the hidden dimension C is 384 and the input image is with resolution 224×224. To encode the spatial information along the height dimension (H), we first conduct a height-channel permutation operation. Given the input $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$, we first split it into S segments along the channel dimension, yielding $[\mathbf{X}_{H_1}, \mathbf{X}_{H_2}, \cdots, \mathbf{X}_{H_S}]$, satisfying $C = N^1 \times S$. In case where the patch size is set to 14×14, the value of N is identical to $224/14 = 16$ and $\mathbf{X}_{H_i} \in \mathbb{R}^{H\times W\times N}, (i \in \{1, \cdots, S\})$. We then perform a height-channel permutation operation[2] with respect to each segment $\mathbf{X}_{H_i}$, yielding $[\mathbf{X}_{H_1}^\top, \mathbf{X}_{H_2}^\top, \cdots, \mathbf{X}_{H_S}^\top]$, which are then concatenated along the channel dimension as the output of the permutation operation. Next, a fully-connected layer with weight $\mathbf{W}_H \in \mathbb{R}^{C\times C}$ is connected to mix the height information. To recover the original dimensional information to $\mathbf{X}$, we only need to perform the height-channel permutation operation once again, outputting $\mathbf{X}_H$. Similarly, in the second branch, we conduct the same operations as above to permute the width dimension and the channel dimension for $\mathbf{X}$ and yield $\mathbf{X}_W$. Finally, we feed the summation of all the token representations from the three branches into a new fully-connected layer to attain the output of the Permute-MLP layer, which can be formulated as follows:

$$\hat{\mathbf{X}} = \mathrm{FC}(\mathbf{X}_H + \mathbf{X}_W + \mathbf{X}_C), \tag{3}$$

where $\mathrm{FC}(\cdot)$ denotes a fully-connected layer with weight $\mathbf{W}_P \in \mathbb{R}^{C\times C}$. A PyTorch-like pseudo code can be found in Alg. 1.

Weighted Permute-MLP: In Eqn. 3, we simply fuse the outputs from all three branches with element-wise addition. Here, we further improve the above Permute-MLP by re-calibrating the importance of different branches and present Weighted Permute-MLP. This can be easily implemented by exploiting the split attention [6]. What is different is that the split attention is applied to $\mathbf{X}_H$, $\mathbf{X}_W$, and $\mathbf{X}_C$ instead of a group of tensors generated by a grouped convolution. In the following, we use the weighted Permute-MLP in Permutator by default.

Relation to ConvNets and Transformers: Similar to MixerMLP, our Vision Permutator is mostly composed of MLPs. However, from the angle of implementation, we do use convolutions for patch embedding, which can also be viewed

1. Here, N is identical to H.
2. Transpose the first (**H**eight) dimension and the third (**C**hannel) dimension: $(H, W, C) \rightarrow (C, W, H)$.

TABLE 1: Configurations of different Vision Permutator models. We present three different models (Small, Medium, and Large) according to the different model sizes. Notation "Small/16" means the model with patch size 16×16 in the starting patch embedding module.

Specification	ViP-Small/16	ViP-Small/14	ViP-Small/7	ViP-Medium/7	ViP-Large/7
Patch size	16×16	14×14	7×7	7×7	7×7
Hidden size	-	-	192	256	256
Number of Tokens	14×14	16×16	32×32	32×32	32×32
Number of Permutators	-	-	4	7	9
Downsampling Rate	-	-	2×2	2×2	2×2
Hidden size	336	384	384	512	512
Number of Tokens	14×14	16×16	16×16	16×16	16×16
Number of Permutators	18	18	14	17	27
Number of layers	18	18	18	24	36
MLP Expansion Ratio	3	3	3	3	3
Stochastic Depth Rate	0.1	0.1	0.1	0.2	0.3
Parameters (M)	23M	30M	25M	55M	88M

as a type of downsampling operation. Our work can be regarded as a hybrid between ConvNet and MLP, but the majority of the operator used are fully-connected layers. Transformers also only consist of MLPs but they rely on self-attention to build pairwise relationships between pairs of tokens. Our Permutator does not model the similarities between pairs of tokens explicitly and hence is quite different from transformers.

3.2 Various Configurations of Vision Permutator

We summarize various configurations of the proposed Vision Permutator in Table 1. We present three different versions of Vision Permutator (ViP), denoted as 'ViP-Small', 'ViP-Medium', and 'ViP-Large' respectively, according to their model size. Notation 'ViP-Small/14' denotes the small-sized model with patch size 14×14 in the starting patch embedding module. In 'ViP-Small/16' and 'ViP-Small/14', there is only one patch embedding module, which is then followed by a sequence of Permutators. The total number of Permutators for them are 18.

Our 'ViP-Small/7,' 'ViP-Medium/7,' and 'ViP-Large/7' have two stages. The first stage starts with a patch embedding module. A few Permutators are added targeting at encoding fine-level token representations which we found beneficial to the model performance. For the second stage, we use a downsampling operation at the beginning to map the token representations to a lower level. For all the models, we set the MLP expansion ratio to 3 following T2T-ViT [29]. We found using an expansion ratio of 4 yields nearly no improvement but brings in more computations.

4 Experiments

We report of the results of our proposed Vision Permutator on the widely-used ImageNet-1k [3] dataset. The code is implemented based on `PyTorch` [36] and the `timm` [37] toolbox. Note that in training, we do not use any extra training data.

4.1 Experiment Setup

We adopt the AdamW optimizer [38] with a linear learning rate scaling strategy $lr = 10^{-3} \times \frac{batch\_size}{1024}$ and 5×10^{-2} weight decay rate to optimize all the models as suggested by previous work [7], [35]. The batch size is set to 2048 which we found works better than 1024 in our Vision Permutator. Stochastic Depth [39] is used. Detailed drop rates can be found in Table 1. We train our models on the ImageNet dataset for 300 epochs. For data augmentation methods, we use Random Erasing [40], RandAug [41], MixUp [42], and CutMix [43]. Note that we do not use positional encoding in our Vision Permutator as we found it hurts the performance. Training small-sized Vision Permutator models requires a machine node with 8 NVIDIA V100 GPUs (32G memory). Two nodes are needed for medium-sized and large-sized Vision Permutator models.

4.2 Main Results on ImageNet

In this subsection, we compare our proposed Vision Permutator with previous CNN-based, Transformer-based, and MLP-like models on ImageNet [3], ImageNet Real [44], and ImageNet-V2 [45]. We first compare our proposed Vision Permutator with recent MLP-like models in Table 2. The 'Train size' and 'Test size' refer to the training resolution and test resolution, respectively. As can be seen, our ViP-Small/7 model with only 25M parameters achieves top-1 accuracy of 81.5%. This result is already better than most of the existing MLP-like models and comparable to the best one gMLP-B [32] which has 73M parameters, far more than ours. Scaling up the model to 55M allows our ViP-Medium/7 to attain 82.7% accuracy, which is better than all other MLP-like models as shown in Table 2. Further increasing the model size to 88M leads to a better result 83.2%. Similar improvement can also be observed on ImageNet Real and ImageNet-V2, reflecting that our method can better prohibit overfitting compared to other models.

We argue that the main factor leading to the improvement for our Vision Permutator is the way of encoding spatial information as described in Sec. 3. Different from concurrent popular MLP-like models listed in Table 2, we separately encode the token representations along the height and width dimensions. In addition, our Vision Permutator encodes not only coarse-level token representations (with 16×16 tokens) but also features at fine-level (with 32×32

TABLE 2: Top-1 accuracy comparison with the recent MLP-like models on ImageNet [3], ImageNet Real [44], and ImageNet-V2 [45]. All models are trained without external data. With the same computation and parameter constraint, our model consistently outperforms other methods but has similar throughput. Following [2], the throughput is measured on a single machine with V100 GPU (32GB) with batch size set to 32. † Implementation with our training recipe, which we found works better than the one reported in the paper.

Networks	Parameters	FLOPs	Throughput	Train size	Test size	ImageNet	ImageNet Real	ImageNetV2
EAMLP-14 [33]	30M	-	711 img/s	224	224	78.9	-	-
gMLP-S [32]	20M	4.5B	-	224	224	79.6	-	-
ResMLP-S24 [2]	30M	6.0B	715 img/s	224	224	79.4	85.3	67.9
ViP-Small/14 (ours)	30M	6.9B	789 img/s	224	224	80.5	86.1	69.6
ViP-Small/7 (ours)	25M	6.9B	719 img/s	224	224	81.5	86.9	70.9
EAMLP-19 [33]	55M	-	464 img/s	224	224	79.4	-	-
Mixer-B/16 [1]†	59M	11.6B	-	224	224	78.5	-	-
ViP-Medium/7 (ours)	55M	16.3B	418 img/s	224	224	82.7	87.4	72.2
gMLP-B [32]	73M	15.8B	-	224	224	81.6	-	-
ResMLP-B24 [2]	116M	23.0B	231 img/s	224	224	81.0	86.1	69.0
ViP-Large/7 (ours)	88M	24.3B	298 img/s	224	224	83.2	87.6	72.7

TABLE 3: Top-1 accuracy comparison with classic CNNs and Vision Transformers on ImageNet [3], ImageNet Real [44], and ImageNet-V2 [45]. All models are trained without external data. With the same computation and parameter constraint, our models are competitive to some powerful CNN-based and transformer-based counterparts.

Network	Parameters	FLOPs	Train size	Test size	ImageNet	ImageNet Real	ImageNetV2
NFNet-F6 + SAM [5]	438M	377B	448	576	86.5	89.2	75.8
CaiT-M48 [8]	356M	330B	224	448	86.5	90.2	76.9
VOLO-D5 [46]	296M	304B	224	448	87.0	90.6	77.8
ResNet-50d [12], [47]	25.6M	4.3B	224	224	79.5	-	-
SE-ResNeXt-50 [13], [17]	27.6M	4.3B	224	224	79.9	85.3	68.7
RegNet-4GF [48]	21M	4.0B	224	224	80.0	-	-
ResNeSt-50 [6]	27.5M	5.4B	224	224	81.1	-	-
DeiT-S [35]	22M	4.6B	224	224	79.8	85.7	68.5
T2T-ViT-14 [29]	22M	5.2B	224	224	81.5	86.8	69.9
Swin-T [9]	29M	4.5B	224	224	81.3	86.7	69.5
ViP-Small/7	25M	6.9B	224	224	81.5	86.9	70.9
ResNet-101d [12], [47]	44.6M	7.9B	224	224	80.4	85.8	69.0
SE-ResNeXt-101 [13], [17]	49.0M	8.0B	224	224	80.9	86.0	70.0
ResNeSt-101 [6]	48.3M	10.2B	256	256	82.9	87.3	72.6
DeepViT [30]	55M	12.5B	224	224	83.1	-	-
ViP-Medium/7	55M	16.3B	224	224	82.7	87.4	72.2
RegNet-16GF [48]	83.6M	15.9B	224	224	82.9	88.1	72.4
DeiT-B [35]	86M	17.5B	224	224	81.8	86.7	71.5
T2T-ViT-24 [29]	64M	13.8B	224	224	82.3	-	-
TNT-B [28]	66M	14.1B	224	224	82.8	-	-
ViP-Large/7	88M	24.3B	224	224	83.2	87.6	72.7

tokens), which has been demonstrated important in vision transformers [46]. We will detail this in next subsection.

In Table 3, we show the comparison with classic CNN-based and transformer-based models. Compared with classic CNNs, like ResNets [12], SE-ResNeXt [13], [17], and RegNet [48], our Vision Permutator with similar model size constraint receives better results. Taking the ViP-Small/7 model as an example, the top-1 accuracy on ImageNet is 81.5%, which is even better than ResNeSt-50 (81.5% *v.s.* 81.1%). Compared to some transformer-based models, such as DeiT [35], T2T-ViT [29], and Swin Transformers [9], our results are also better. This phenomenon indicates that MLP-like models are strong competitors to CNN-based and Transformer-based models. We also found that our Vision Permutator However, there is still a large gap between our Vision Permutator and recent state-of-the-art CNN- and transformer-based models, such as NFNet [5], CaiT [8] and VOLO [46]. We believe there is still a large room for improving MLP-like models, just like what happened in the research field of vision transformers.

4.3 Method Analysis

In this subsection, we conduct a series of ablation experiments on fine-level information encoding, model scaling, data augmentation, and the proposed Permutator. We take the ViP-Small/14 model as baseline.

Importance of Fine-level Token Representation Encoding: We first show that encoding finer-level token representations is important for MLP-like models. We demonstrate this argument in two ways: I) Adjusting the patch size in the initial patch embedding layer and keep the backbone unchanged; II) Halving the patch size for each patch side and introducing a few Permutators to encode fine-level to-

TABLE 4: Role of fine-level token representation encoding. 'Initial Patch Size' denotes the patch size in the starting patch embedding module and 'Fine Tokens' refers to models encoding fine-level token representations. Larger patch size means that the number of tokens fed into Permutators would be lower as specified in Table 1. We can see that the model efficiency in speed does not change too much when changing the initial patch size. We report throughput values based on two batch size settings: 32 and 128.

Models	Patch Size	Fine Tokens	Params	FLOPs	Peak Memory	Throughput (32)	Throughput (128)	Top-1 Acc. (%)
ViP-Small/16	16×16	No	23M	4.0B	240	803 img/s	1110 img/s	79.8
ViP-Small/14	14×14	No	30M	6.9B	300	789 img/s	944img/s	80.6
ViP-Small/7	7×7	Yes	25M	6.9B	342	719 img/s	800 img/s	81.5

TABLE 5: Role of the model scale. We scale the models by increasing the model size (including number of layers, hidden dimension). 'Hidden Dim.' refers to the hidden dimension in the second stage, which is halved in the first stage. Clearly, increasing the model size can consistently improve the model performance.

Models	Layers	Hidden Dim.	Params	FLOPs	Peak Memory	Throughput (32)	Throughput (128)	Top-1 Acc. (%)
ViP-Small/7	18	384	25M	6.9B	342	719 img/s	800 img/s	81.5
ViP-Medium/7	24	512	55M	16.3B	596	418 img/s	452 img/s	82.7
ViP-Large/7	36	512	88M	24.3B	815	298 img/s	322 img/s	83.2

ken representations. Table 4 summaries the performance for ViP-Small/16, ViP-Small/14, and ViP-Small/7. Compared to ViP-Small/16, ViP-Small/14 has smaller initial patch size and more input tokens to the Permutators. According to the results, ViP-Small/14 yields better performance than ViP-Small/16 (80.5% *v.s.* 79.8%). Despite more tokens and more parameters used in ViP-Small/14, the efficiency (throughput) does not change much. This indicates that we can appropriately use smaller initial patch size to improve the model performance.

We further reduce the initial patch size from 14×14 to 7×7. Compared to ViP-Small/14, ViP-Small/7 adopts 4 Permutators to encode fine-level token representations (with 32×32 tokens). As shown in Table 4, such a slight modification can largely boost the performance and reduce the number of learnable parameters. The top-1 accuracy is improved from 80.5% to 81.5%. This demonstrates that encoding fine-level token representations does help in improving our model performance but a disadvantage is that the efficiency goes down a little.

Role of the Model Scale: Scaling up models for deep neural networks is always an effective way to improve model performance. Here, we show the influence of model scaling on the proposed Vision Permutator by increasing the number of layers and hidden dimension. Table 5 lists the results for three different versions of the proposed Vision Permutator: ViP-Small/7, ViP-Medium/7, and ViP-Large/7. We can see that increasing the number of layers and hidden dimension yields better results for our Vision Permutator. The ViP-Medium/7 can raise the performance of ViP-Small/7 to 82.7% with a performance gain of more than 1%. Further increasing the model size results in better performance 83.2%.

Effect of Data Augmentations: Data augmentation has been demonstrated an effective and efficient way to lift the model performance in deep learning [7], [35], [47]. Four commonly-used data augmentation methods should be Random Augmentation [41], Random Erasing [40], MixUp [42], and CutMix [43]. Here, we show how each method influences the model performance. The results have been shown in Table 6. Without any data augmentation, we achieve 75.3% top-1 accuracy for our ViP-Small/14 model. Using Random Augmentation improves the performance to 77.7% (+2.4%). Adding Random Erasing lifts the result to 78.0% (+2.7%). Adding MixUp yields 80.2% top-1 accuracy (+4.9%) and the result is further improved to 80.6% (+5.3%) by using CutMix. These experiments indicate that data augmentation is extremely important in training Vision Permutator as happened in training CNNs [47] and vision transformers [7], [35].

TABLE 6: Ablation on data augmentation methods. We ablate four widely used data augmentation methods in both CNN- and transformer-based models, including Random Augmentation [41], Random Erasing [40], MixUp [42], and CutMix [43]. We can see that all 4 methods contribute to the model performance.

Data augmentation methods	Layers	Params	Top-1 Acc. (%)
Baseline (ViP-Small/14)	18	30M	75.3
+ Random Aug. [41]	18	30M	77.7 (+2.4)
+ Random Erasing [40]	18	30M	78.0 (+2.7)
+ MixUp [42]	18	30M	80.2 (+4.9)
+ CutMix [43]	18	30M	80.6 (+5.3)

From Mixer to Vision Permutator: The original Mixer [1] encodes spatial information by flattening all the tokens. To show the advantage of separately encoding the spatial information, we replace the top 2 branches in Fig. 2 with the token-mixing MLP in Mixer-MLP. Results have been listed in Table 7. The new model gives a performance of 78.9% as shown in the third row of Table 7, slightly better than Mixer-B/16 with our training recipe but worse than our ViP-Small/14 (78.9% *v.s.* 80.2%). These experiments indicate that separately encoding the height and width information is more useful for visual recognition.

Ablation on Permutator: Here, we demonstrate the importance of encoding spatial information along the height and width dimensions separately and show how weighted Permutator helps. In Table 7, we summarize the results under different Permutator settings. Detailed description

TABLE 7: Ablation on Vision Permutator. 'ViP-Small/14 w/o Height' means a ViP-Small/14 model with the height information encoding part replaced by channel encoding (the bottom branch in Figure 2). A similar meaning holds for 'ViP-Small/14 w/o Width.' 'ViP-Small/14 w/ Permute-MLP' refers to model with the vanilla Permute-MLP. 'ViP-Small/14 w/ Cascaded Permute-MLP' means encoding spatial information along the two spatial dimensions in a cascaded way.

Model Specification	Layers	Hid. Dim.	Params	FLOPs	Peak Mem.	Throughput	Top-1 Acc.
Mixer-B/16 (original)	12	768	59M	11.6B	521	-	76.4 (-4.2)
Mixer-B/16 (w/ our training recipe)	12	768	59M	11.6B	521	-	78.5 (-2.1)
ViP-Small/14 (sep. enc. $\rightarrow$ token-mixing MLP)	18	384	29M	8.3B	296	763 img/s	78.9 (-1.7)
ViP-Small/14 w/o Height Information	18	384	29M	6.9B	288	844 img/s	72.8 (-7.8)
ViP-Small/14 w/o Width Information	18	384	29M	6.9B	288	843 img/s	72.7 (-7.9)
ViP-Small/14 w/ Cascaded Permute-MLP	18	384	29M	6.9B	279	847 img/s	79.8 (-0.8)
ViP-Small/14 w/ Permute-MLP	18	384	29M	6.9B	288	847 img/s	80.2 (-0.4)
ViP-Small/14 w/ Weighted Permute-MLP	18	384	30M	6.9B	300	789 img/s	80.6

TABLE 8: Results of finetuning the pretrained ViP-S7 to downstream datasets: CIFAR10, CIFAR100, and iNaturalist 2021 [49]. We finetune all the models for 60 epochs as in [29].

Models	Params (M)	CIFAR10	CIFAR100	iNaturalist
ViT/S-16	48.6	97.1	87.1	72.5
T2T-ViT-14	21.5	97.5	88.4	73.0
ViP-Small/7	25.0	98.0	88.4	73.8

on each setting can be found in the caption. We can see that discarding either height information encoding or width information encoding leads to worse performance (80.2% *v.s.* 72.8% or 72.7%). This demonstrates that encoding both height and width information is important. In addition, we can also observe that replacing the vanilla Permute-MLP with the weighted Permute-MLP can further improve the performance from 80.2% to 80.6%.

As shown in Figure 2, we encode the token representations along the height and width dimensions (the top two branches), separately. An alternative way to embed spatial information is to merge the top two branches into one, i.e., sequentially processing the representations along the two spatial dimensions as done in axial attention [23]. However, we empirically found that such a sequential way to encode spatial information is less efficient than our approach. Experiments show that encoding spatial information along the horizontal and vertical dimensions sequentially decreases the performance. As listed in Table 7, this operation reduces the classification performance from 80.2% to 79.8% (-0.4%).

Transfer Learning: To test the transfer learning ability, we attempt to run experiments on CIFAR10, CIFAR100, and iNaturalist 2021 [49] using ViP-Small/7. We use the same settings as in T2T-ViT. Table 8 lists the results. We can see that the proposed ViP-Small/7 has achieved equal or even better results to the recently popular T2T-ViT on CIFAR10, CIFAR100, and iNaturalist 2021. This indicates that the proposed Vision Permutator performs well in transfer learning to small classification datasets.

Positional Encoding: The Mixer work has shown that there is no need to use position embeddings in MLP-like models. Here, we investigate whether it benefits our Vision Permutator. We attempt to use position embeddings before the first Permutator and make them learnable and also attempt to add relative position embeddings [50]. Experiments show that both approaches decrease the performance by around 0.3% accuracy. This is because the proposed method has already embedded positional information while separately processing the spatial features. As a result, position embeddings are not needed in our Vision Permutator.

5 Conclusions and Future Work

In this paper, we present a novel MLP-like network architecture for visual recognition, termed Vision Permutator. We show that separately encoding the two spatial information can largely improve the model performance compared to MLP-like models that deem the two spatial dimensions as one. Our experiments also give full support of this.

Despite the large improvement over concurrent popular MLP-like models, a clear downside of the proposed Permutator is the scaling problem in spatial dimensions, which also exists in other MLP-like models. As the shapes of the parameters in fully-connected layers are fixed, it is impossible to process input images with arbitrary shapes. This makes MLP-like models difficult to be used in downstream tasks with various-sized input images.

Our future work will be continuously put on the development of MLP-like models considering the high efficacy in parallelization. Specifically, we will continue to conquer the limitations of MLP-like models in processing input images with arbitrary shapes and their applications in down-stream tasks, such as object detection and semantic segmentation.

Acknowledgements

Ming-Ming Cheng was supported by National Key Research and Development Program of China (Grant No. 2018AAA0100400) and CAAI-Huawei Open Fund.

References

[1] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, D. Keysers, J. Uszkoreit, M. Lucic *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *arXiv preprint arXiv:2105.01601*, 2021.

[2] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, A. Joulin, G. Synnaeve, J. Verbeek, and H. Jégou, "Resmlp: Feedforward networks for image classification with data-efficient training," *arXiv preprint arXiv:2105.03404*, 2021.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009.

[4] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[5] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," *arXiv preprint arXiv:2102.06171*, 2021.

[6] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha *et al.*, "Resnest: Split-attention networks," *arXiv preprint arXiv:2004.08955*, 2020.

[7] Z. Jiang, Q. Hou, L. Yuan, D. Zhou, X. Jin, A. Wang, and J. Feng, "Token labeling: Training a 85.4% top-1 accuracy vision transformer with 56m parameters on imagenet," *arXiv preprint arXiv:2104.10858*, 2021.

[8] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," *arXiv preprint arXiv:2103.17239*, 2021.

[9] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

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

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[14] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[17] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[18] H. Hu, Z. Zhang, Z. Xie, and S. Lin, "Local relation networks for image recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3464–3473.

[19] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.

[20] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention augmented convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3286–3295.

[21] J.-J. Liu, Q. Hou, M.-M. Cheng, C. Wang, and J. Feng, "Improving convolutional networks with self-calibrated convolutions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 096–10 105.

[22] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "Aˆ 2-nets: Double attention networks," in *Advances in neural information processing systems*, 2018, pp. 352–361.

[23] H. Wang, Y. Zhu, B. Green, H. Adam, A. Yuille, and L.-C. Chen, "Axial-deeplab: Stand-alone axial-attention for panoptic segmentation," in *European Conference on Computer Vision*. Springer, 2020, pp. 108–126.

[24] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[25] D. Zhou, Y. Shi, B. Kang, W. Yu, Z. Jiang, Y. Li, X. Jin, Q. Hou, and J. Feng, "Refiner: Refining self-attention for vision transformers," *arXiv preprint arXiv:2106.03714*, 2021.

[26] A. Vaswani, P. Ramachandran, A. Srinivas, N. Parmar, B. Hechtman, and J. Shlens, "Scaling local self-attention for parameter efficient visual backbones," *arXiv preprint arXiv:2103.12731*, 2021.

[27] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," *arXiv preprint arXiv:2103.15808*, 2021.

[28] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *arXiv preprint arXiv:2103.00112*, 2021.

[29] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," *arXiv preprint arXiv:2101.11986*, 2021.

[30] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Q. Hou, and J. Feng, "Deepvit: Towards deeper vision transformer," *arXiv preprint arXiv:2103.11886*, 2021.

[31] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans, "Axial attention in multidimensional transformers," *arXiv preprint arXiv:1912.12180*, 2019.

[32] H. Liu, Z. Dai, D. R. So, and Q. V. Le, "Pay attention to mlps," *arXiv preprint arXiv:2105.08050*, 2021.

[33] M.-H. Guo, Z.-N. Liu, T.-J. Mu, and S.-M. Hu, "Beyond self-attention: External attention using two linear layers for visual tasks," *arXiv preprint arXiv:2105.02358*, 2021.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, pp. 5998–6008, 2017.

[35] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *arXiv preprint arXiv:2012.12877*, 2020.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[37] R. Wightman, "Pytorch image models," https://github.com/rwightman/pytorch-image-models, 2019.

[38] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[39] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European conference on computer vision*. Springer, 2016, pp. 646–661.

[40] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 13 001–13 008.

[41] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.

[42] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[43] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6023–6032.

[44] L. Beyer, O. J. Hénaff, A. Kolesnikov, X. Zhai, and A. v. d. Oord, "Are we done with imagenet?" *arXiv preprint arXiv:2006.07159*, 2020.

[45] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" in *International Conference on Machine Learning*. PMLR, 2019, pp. 5389–5400.

[46] L. Yuan, Q. Hou, Z. Jiang, J. Feng, and S. Yan, "Volo: Vision outlooker for visual recognition," 2021.

[47] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 558–567.

[48] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 428–10 436.

[49] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist species classification and detection dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8769–8778.

[50] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani, "Bottleneck transformers for visual recognition," *arXiv preprint arXiv:2101.11605*, 2021.