
VISION PERMUTATOR: A PERMUTABLE MLP-LIKE ARCHITECTURE FOR VISUAL RECOGNITION

Qibin Hou, Zihang Jiang, Li Yuan

Department of Electrical and Computer Engineering
National University of Singapore
Singapore
{andrewhou, jzh0103, ylustcnus}@gmail.com

Ming-Ming Cheng

School of Computer Science
Nankai University
Tianjin, China
cmm@nankai.edu.cn

Shuicheng Yan

Sea AI Labs
Singapore
yansc@sea.com

Jiashi Feng

ECE, NUS & Sea AI Labs
Singapore
elefjia@nus.edu.sg

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 along one spatial direction and meanwhile preserve precise positional information along the other direction. The resulting position-sensitive outputs are then aggregated in a mutually complementing manner to form expressive representations of the objects of interest. 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. We hope this work could encourage research on rethinking the way of encoding spatial information and facilitate the development of MLP-like models. Code is available at <https://github.com/Andrew-Qibin/VisionPermutator>.

1 INTRODUCTION

Recent studies (Tolstikhin et al., 2021; Touvron et al., 2021a) have shown that pure multi-layer perceptron based networks perform well in ImageNet classification (Deng et al., 2009). 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 (Tolstikhin et al., 2021). 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 (Tan & Le, 2019; Brock et al., 2021; Zhang et al., 2020) and vision transformers (Jiang et al., 2021; Touvron et al., 2021b; Liu et al., 2021b).

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

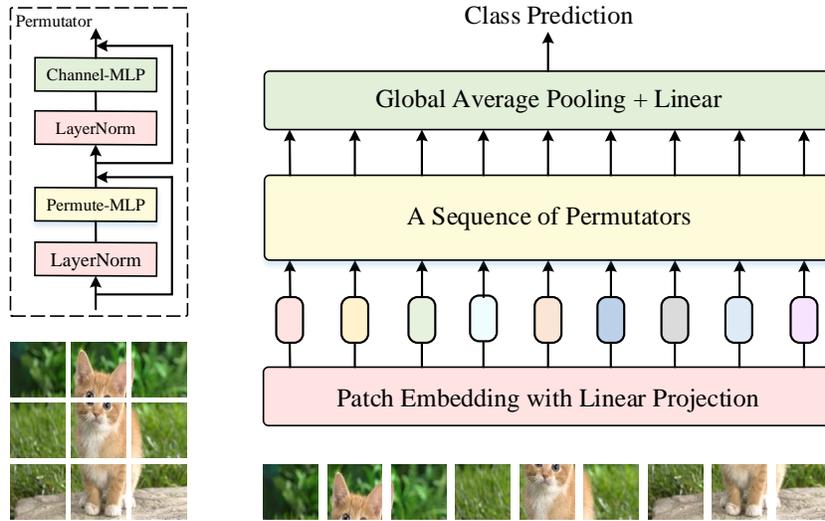


Figure 1: Basic architecture of the proposed Vision Permutator. The evenly divided image patches are tokenized with linear projection first and then fed into a sequence of Permutators for feature encoding. A global average pooling layer followed by a fully-connected layer is finally used to predict the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

multiplication routine. Unlike current MLP-like models, such as Mixer (Tolstikhin et al., 2021) and ResMLP (Touvron et al., 2021a), 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 \times 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 encode spatial information along the height and width dimensions to preserve positional information.

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 \times width \times 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 tokens with direction-specific information, which has been demonstrated essential for visual recognition (Hou et al., 2021; Wang et al., 2020).

Experiments show that our Vision Permutator can largely improve the classification performance of existing MLP-like models. Taking the small-sized Vision Permutator as an example, it attains 81.5% top-1 accuracy on ImageNet without any extra training data. Scaling up the model to 55M and 88M, we can further achieve 82.7% and 83.2% accuracy, respectively.

2 RELATED WORK

Modern deep neural networks for image classification can be mainly categorized into three different classes: 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 (Krizhevsky et al., 2012) and VGGNet (Simonyan & Zisserman, 2014), mostly adopt structures with a stack of spatial convolutions (with kernel size ≥ 3) and

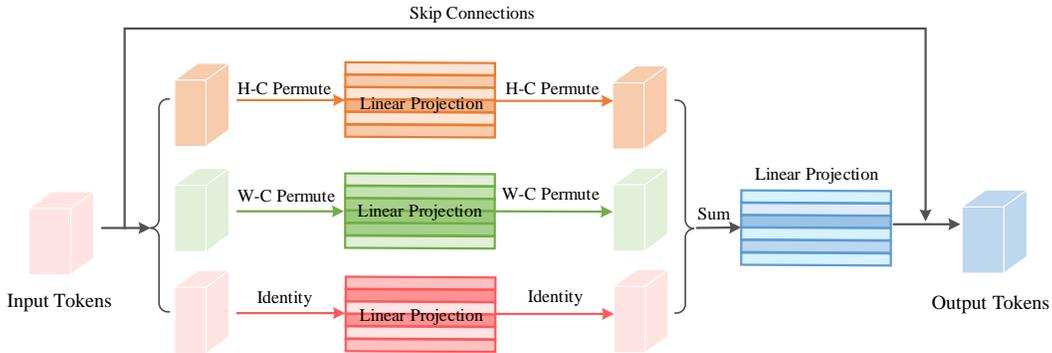


Figure 2: Basic structure of the proposed Permute-MLP layer. The proposed Permute-MLP layer contains three branches that are responsible for encoding features along the height, width, and channel dimensions, respectively. The outputs from the three branches are then combined using element-wise addition,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

pooling operations. Later, ResNets and their variants (He et al., 2016; Xie et al., 2017; Zagoruyko & Komodakis, 2016) introduce skip connection and building blocks with bottleneck structure into CNNs, enabling training very deep networks possible. Inceptions (Szegedy et al., 2015; 2016) renovate the design of traditional building block structure and utilize multiple parallel paths of sets of specialized filters. Attention mechanisms (Hu et al., 2018; 2019; Wang et al., 2018; Bello et al., 2019; Liu et al., 2020; Chen et al., 2018) break through the limitations of convolutions in capturing local features and further promote the development of CNNs. 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 (Dosovitskiy et al., 2020). 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 (Zhou et al., 2021b; Vaswani et al., 2021; Wu et al., 2021; Liu et al., 2021b; Han et al., 2021; Yuan et al., 2021), or scaling the depth (Zhou et al., 2021a; Touvron et al., 2021b), or tailoring powerful optimization strategies (Jiang et al., 2021). 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 (Tolstikhin et al., 2021; Touvron et al., 2021a; Liu et al., 2021a; Guo et al., 2021) 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 (Tolstikhin et al., 2021). 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.

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)

##### initialization #####
proj_h = nn.Linear(C, C) # Encoding spatial information along the height dimension
proj_w = nn.Linear(C, C) # Encoding spatial information along the width dimension
proj_c = nn.Linear(C, C) # Encoding channel information
proj = nn.Linear(C, C) # For information fusion

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

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

3.1 PERMUTATOR

A diagrammatic illustration of the proposed Permutator block can be found at 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 (Vaswani et al., 2017) that comprises two fully-connected layers with a GELU activation in the middle. For spatial information encoding, unlike the recent Mixer (Tolstikhin et al., 2021) that conducts linear projection along the spatial dimension with respect to all the tokens, we propose to separately processing 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:

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

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

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 (Dosovitskiy et al., 2020; Jiang et al., 2021; Touvron et al., 2020) and Mixer (Tolstikhin et al., 2021) that receive an input of two dimensions (“tokens \times 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 the either 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 head-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, 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 $[\mathbf{X}_{H_1}, \mathbf{X}_{H_2}, \dots, \mathbf{X}_{H_S}]$, satisfying $C = N * S^1$. In case where the patch size is set to 14×14 , the value of N is identical to 16 and $\mathbf{X}_{H_i} \in \mathbb{R}^{H \times W \times N}$, ($i \in \{1, \dots, S\}$). We then perform a height-channel permutation operation² with respect to each segment \mathbf{X}_{H_i} , yielding $[\mathbf{X}_{H_1}^\top, \mathbf{X}_{H_2}^\top, \dots, \mathbf{X}_{H_S}^\top]$, which are then concatenated along the channel dimension as the output 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

¹In our case, N is identical to H or W .

²Transpose the first (Height) dimension and the third (Channel) 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
#Tokens	14×14	16×16	32×32	32×32	32×32
#Permutators	-	-	4	7	9
Patch size	-	-	2×2	2×2	2×2
Hidden size	336	384	384	512	512
#Tokens	14×14	16×16	16×16	16×16	16×16
#Permutators	18	18	14	17	27
Number of layers	18	18	18	24	36
MLP Ratio	3	3	3	3	3
Stoch. Dep.	0.1	0.1	0.1	0.2	0.3
Parameters (M)	23M	30M	25M	55M	88M

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:

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

where $\text{FC}(\cdot)$ 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 recalibrating the importance of different branches and present Weighted Permute-MLP. This can be easily implemented by exploiting the split attention (Zhang et al., 2020). 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.

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 16.

Our ‘ViP-Small/7’, ‘ViP-Medium/7,’ and ‘ViP-Large/7’ have two stages, each of which starts with a patch embedding module. For these models, we add a few Permutators targeting at encoding fine-level token representations which we found beneficial to the model performance. In our experiment section, we will show the advantage of encoding fine-level token representations.

4 EXPERIMENTS

We report of the results of our proposed Vision Permutator on the widely-used ImageNet-1k (Deng et al., 2009) dataset. The code is implemented based on PyTorch (Paszke et al., 2019) and the timm (Wightman, 2019) toolbox. Note that in training, we do not use any extra training data.

Table 2: Top-1 accuracy comparison with recent MLP-like models on ImageNet (Deng et al., 2009). All models are trained without external data. With the same computation and parameter constraint, our model consistently outperforms other methods. Following (Touvron et al., 2021a), 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	Throughput	Train size	Test size	Top-1 Acc. (%)
EAMLP-14 (Guo et al., 2021)	30M	711 img/s	224	224	78.9
gMLP-S (Liu et al., 2021a)	20M	-	224	224	79.6
ResMLP-S24 (Touvron et al., 2021a)	30M	715 img/s	224	224	79.4
ViP-Small/14 (ours)	30M	789 img/s	224	224	80.5
ViP-Small/7 (ours)	25M	719 img/s	224	224	81.5
EAMLP-19 (Guo et al., 2021)	55M	464 img/s	224	224	79.4
Mixer-B/16 (Tolstikhin et al., 2021)†	59M	-	224	224	78.5
ViP-Medium/7 (ours)	55M	418 img/s	224	224	82.7
gMLP-B (Liu et al., 2021a)	73M	-	224	224	81.6
ResMLP-B24 (Touvron et al., 2021a)	116M	231 img/s	224	224	81.0
ViP-Large/7 (ours)	88M	298 img/s	224	224	83.2

4.1 EXPERIMENT SETUP

We adopt the AdamW optimizer (Loshchilov & Hutter, 2017) with a linear learning rate scaling strategy $lr = 10^{-3} \times \frac{\text{batch_size}}{1024}$ and 5×10^{-2} weight decay rate to optimize all the models as suggested by previous work (Touvron et al., 2020; Jiang et al., 2021). The batch size is set to 2048 which we found works better than 1024 in our Vision Permutator. Stochastic Depth (Huang et al., 2016) 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 CutOut (Zhong et al., 2020), RandAug (Cubuk et al., 2020), MixUp (Zhang et al., 2017), and CutMix (Yun et al., 2019). 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. 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. Our ViP-Small/7 model with 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 (Liu et al., 2021a) with 73M parameters. 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%.

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 encoding the token representations along the height and width dimensions, generating position-sensitive outputs that are crucial for locating and identifying objects of interest (Hou et al., 2021; Wang et al., 2020). 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 tokens). 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 (He et al., 2016), SE-ResNeXt (Xie et al., 2017; Hu et al., 2018), and RegNet (Radosavovic et al., 2020), our Vision Permutator with similar model size constraint receives better results. Taking the ViP-Small/7 model as an example, the performance 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 (Touvron et al., 2020), T2T-ViT (Yuan et al., 2021), and Swin Transformers (Liu et al., 2021b), our results are also better. However, there is still a large gap between our Vision Permutator and recent state-of-the-art CNN- and transformer-based models, such as

Table 3: Top-1 accuracy comparison with classic CNNs and Vision Transformers on ImageNet (Deng et al., 2009). 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	Train size	Test size	Top-1 Acc. (%)
ResNet-50d (He et al., 2016; 2019)	25.6M	224	224	79.5
SE-ResNeXt-50 (Xie et al., 2017; Hu et al., 2018)	27.6M	224	224	79.9
RegNet-6.4GF (Radosavovic et al., 2020)	30.6M	224	224	79.9
ResNeSt-50 (Zhang et al., 2020)	27.5M	224	224	81.1
DeiT-S (Touvron et al., 2020)	22M	224	224	79.9
T2T-ViT-14 (Yuan et al., 2021)	22M	224	224	81.5
Swin-T (Liu et al., 2021b)	29M	224	224	81.3
ViP-Small/7	25M	224	224	81.5
ResNet-101d (He et al., 2016; 2019)	44.6M	224	224	80.4
SE-ResNeXt-101 (Xie et al., 2017; Hu et al., 2018)	49.0M	224	224	80.9
RegNet-12GF (He et al., 2016; 2019)	51.8M	224	224	80.3
ResNeSt-101 (Zhang et al., 2020)	48.3M	256	256	82.9
DeepViT (Zhou et al., 2021a)	55M	224	224	83.1
ViP-Medium/7	55M	224	224	82.7
RegNet-16GF (Radosavovic et al., 2020)	83.6M	224	224	80.4
DeiT-B (Touvron et al., 2020)	86M	224	224	81.8
T2T-ViT-24 (Yuan et al., 2021)	64M	224	224	82.3
TNT-B (Han et al., 2021)	66M	224	224	82.8
ViP-Large/7	88M	224	224	83.2

NFNet (Brock et al., 2021) (86.5%), LV-ViT (Jiang et al., 2021) (86.4%), and CaiT (Touvron et al., 2021b) (86.5%). 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 ABLATION 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.

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.

Models	Initial Patch Size	Fine Tokens	Layers	Parameters	Throughput	Top-1 Acc. (%)
ViP-Small/16	16×16	No	18	23M	803 img/s	79.8
ViP-Small/14	14×14	No	18	30M	789 img/s	80.6
ViP-Small/7	7×7	Yes	18	25M	719 img/s	81.5

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 token 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.

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.	Fine Tokens	Parameters	Throughput	Top-1 Acc. (%)
ViP-Small/7	18	384	Yes	25M	719 img/s	81.5
ViP-Medium/7	24	512	Yes	55M	418 img/s	82.7
ViP-Large/7	36	512	Yes	88M	298 img/s	83.2

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 (He et al., 2019; Touvron et al., 2020; Jiang et al., 2021). Four commonly-used data augmentation methods should be Random Augmentation (Cubuk et al., 2020), CutOut (Zhong et al., 2020), MixUp (Zhang et al., 2017), and CutMix (Yun et al., 2019). 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 CutOut 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 (He et al., 2019) and vision transformers (Touvron et al., 2020; Jiang et al., 2021).

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 (Cubuk et al., 2020), CutOut (Zhong et al., 2020), MixUp (Zhang et al., 2017), and CutMix (Yun et al., 2019). We can see that all 4 methods contribute to the model performance.

Data augmentation methods in training	Layers	Parameters	Top-1 Acc. (%)
Baseline (ViP-Small/14)	16	30M	75.3
+ Random Augmentation (Cubuk et al., 2020)	18	30M	77.7 (+2.4)
+ CutOut (Zhong et al., 2020)	18	30M	78.0 (+2.7)
+ MixUp (Zhang et al., 2017)	18	30M	80.2 (+4.9)
+ CutMix (Yun et al., 2019)	18	30M	80.6 (+5.3)

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

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.

Model Specification	Layers	Parameters	Throughput	Top-1 Acc. (%)
ViP-Small/14 w/o Height Information	18	29M	844 img/s	72.8 (-7.8)
ViP-Small/14 w/o Width Information	18	29M	843 img/s	72.7 (-7.9)
ViP-Small/14 w/ Permute-MLP	18	29M	847 img/s	80.2 (-0.4)
ViP-Small/14 w/ Weighted Permute-MLP	18	30M	789 img/s	80.6

CONCLUSIONS AND FUTURE WORK

In this paper, we present a novel MLP-like network architecture for visual recognition, termed Vision Permutator. We demonstrate that separately encoding the height and width information can largely improve the model performance compared to recent 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 down-stream 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.

REFERENCES

- Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3286–3295, 2019.
- Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.
- Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. A²-nets: Double attention networks. In *Advances in neural information processing systems*, pp. 352–361, 2018.
- Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc 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*, pp. 702–703, 2020.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu, and Shi-Min Hu. Beyond self-attention: External attention using two linear layers for visual tasks. *arXiv preprint arXiv:2105.02358*, 2021.
- Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

-
- 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*, pp. 770–778, 2016.
- Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 558–567, 2019.
- Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. In *Computer Vision and Pattern Recognition*, 2021.
- Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3464–3473, 2019.
- Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.
- Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.
- Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Xiaojie Jin, Anran Wang, and Jiashi Feng. Token labeling: Training a 85.4% top-1 accuracy vision transformer with 56m parameters on imagenet. *arXiv preprint arXiv:2104.10858*, 2021.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
- Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021a.
- Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Changhu Wang, and Jiashi Feng. Improving convolutional networks with self-calibrated convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10096–10105, 2020.
- 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. *arXiv preprint arXiv:2103.14030*, 2021b.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.
- Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428–10436, 2020.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
- Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.
- Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

-
- Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.
- Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.
- Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021a.
- Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021b.
- 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*, 30:5998–6008, 2017.
- Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. *arXiv preprint arXiv:2103.12731*, 2021.
- Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pp. 108–126. Springer, 2020.
- Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018.
- Ross Wightman. Pytorch image models. <https://github.com/rwightman/pytorch-image-models>, 2019.
- Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.
- Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.
- Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.
- Sangdoon Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.
- Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.
- Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.
- Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.
- Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13001–13008, 2020.

Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021a.

Daquan Zhou, Yujun Shi, Bingyi Kang, Weihao Yu, Zihang Jiang, Yuan Li, Xiaojie Jin, Qibin Hou, and Jiashi Feng. Refiner: Refining self-attention for vision transformers. *arXiv preprint arXiv:2106.03714*, 2021b.