Vision Permutator：一种用于视觉识别的可互换的类 MLP 架构

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

Abstract—在本篇文章中，我们提出了一个概念上简单且数据高效（Data Efficient）的类 MLP 结构，Vision Permutator，用以视觉识别。与最近的类 MLP 模型沿扁平化的空间维度编码空间信息不同，我们意识到二维特征表示所携带的位置信息的重要性，因此 Vision Permutator 将沿高度和宽度两个维度分别用线性投影单独地编码特征表示。这允许 Vision Permutator 捕捉长远的依赖，而不需要使用 Transformer 的注意力机制。然后，这些输出以相互补充的方式汇总，形成更加具有表达力的特征表示。 我们认为，我们提出的 Vision Permutator 是卷积神经网络（CNN）和视觉 Transformer 的强力竞争对手。在不依赖空间卷积或注意力机制的情况下，Vision Permutator 在 ImageNet 上的 top-1 准确率达到了 81.5%，且其不需要额外的大规模训练数据（例如， ImageNet-22k）。此外，Vision Permutator 的可学习参数量仅有 25M，且其性能比大多数具有相同大小的 CNN 和视觉 Transformer 模型要优秀得多。当模型的参数量扩展到 88M 时，它的 Top-1 准确率将达到 83.2%，大大提高了 SOTA MLP 类模型在视觉识别方面的性能。我们希望这项工作能够鼓励人们重新思考空间信息的编码方式，并促进类 MLP 模型的发展。PyTorch/MindSpore/Jittor 等版本的代码发布于https://github.com/Andrew-Qibin/VisionPermutator.

Index Terms—Vision Permutator，Permutator，图像分类，多层感知机，深度神经网络

1 引言

最近的研究 [2]，[3] 表明纯粹的基于网络的多层感知机在 ImageNet 分类任务 [4] 中表现优异。与利用空间卷积或自注意力机制编码空间信息的卷积神经网络（CNN）和视觉 Transformer 相比，类 MLP 模型（又名 MLP）仅利用纯粹的全连接层（或称 1×1 卷积），因此其能更高效地进行训练和推理 [2]。但是，MLP 在图像分类中的优异性能受益于其在大规模训练集上的训练（如，ImageNet-22K 和 JFT-300M 数据集）。没有了足够大规模的数据集的支持，他们的表现依然远落后于 CNN [5]–[7] 和视觉 Transformer [8]–[10]。

在这项工作中，我们希望仅使用 ImageNet-1k 数据集进行训练，构建数据高效的 MLP 模型，以发挥 MLP 的潜力。为此，我们提出了 Vision Permutator 结构。特别指出的是，Vision Permutator 通过提出一种全新的层结构，创新了现有的 MLP 结构，其可以基于基本的矩阵乘法更有效地编码空间信息。目前的一些类 MLP 模型，如 Mixer [2] 和 ResMLP [3]，它们首先将空间维度扁平化，然后沿空间维度进行线性投影 (即，对形状为 “Token×Channel” 的 Token 进行操作)，以此编码空间信息，这导致了二维特征表示所携带的位置信息的丢失。与上述模型不同的是，Vision Permutator 保持输入 Token 的原始空间尺寸，并沿高度和宽度尺寸分别编码空间信息以保留位置信息。这使我们的 Vision Permutator 与现有的类 MLP 模型有很大的不同。

具体来说，如图 1所示，我们的 Vision Permutator 从一个类似于视觉 Transformer 的 Tokenization 操作开始，它将输入图像均匀地分割成小的 Patch，然后通过线性投影将它们映射为 Token Embedding。由上述步骤生成的形状为 “Height×Width×Channel” 的 Token Embedding 被送入一系列 Permutator 块中，每一个模块都包含一个用于空间信息编码的 Permute-MLP 和一个用于融合通道信息的 Channel-MLP。如图 2所示，Permute-MLP 层由三个独立的分支组成，每个分支沿着一个特定的维度编码特征，即，高度、宽度或通道维度。和现有的将两个空间纬度混合成一个维度的类 MLP 模型相比，我们的 Vision Permutator 沿着这些维度分别处理 Token 表征，从而生成更具辨别力的 Token 表征。我们将在实验部分证明这一操作对于视觉识别至关重要。

实验表明，我们的 Vision Permutator 可以在很大程度上提高现有的类 MLP 模型在分类任务上的性能。以参数量为 25M 的小规模 Vision Permutator 为例，在没有任何额外训练数据的情况下，它在 ImageNet 上的 Top-1 准确率达到

- 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)

- 本文是 [1] 的中译版。

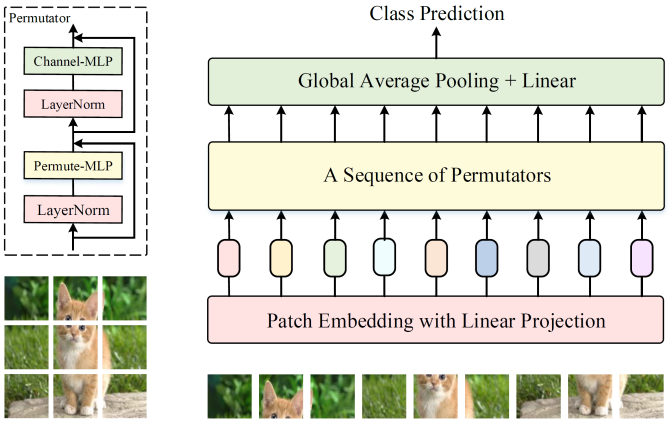

Figure 1. **所提出的** Vision Permutator **的基本结构。均匀分割的** Patch **首先用线性投影法进行** Tokenization，**然后输入一连串的** Permutator **进行特征编码。最后通过全局平均池化层和全连接层来预测类别。**

了 81.5%。该结果好于大多数经典的基于 CNN 的模型，例如 ResNet-50d（79.5%），SE-ResNeXt-50 （79.9%）和 Strong ResNeSt-50（81.1%）。 当模型参数量达到 55M 和 88M 时，结果将被进一步提高，其在 ImageNet 数据集上的 Top-1 准确率分别达到了 82.7% 和 83.2%。

2 相关工作

目前，用于图像分类的深度神经网络模型有三个突出的系列：卷积神经网络（CNN）、视觉 Transformer（ViT）和基于多层感知器的模型（MLP）。在下文中，我们将简要介绍每种类型网络的发展趋势，并说明所提出的 Vision Permutator 与以前的工作的区别。

CNN 作为多年来计算机视觉领域事实上的标准网络，人们已经对其进行了深入的研究。早期的 CNN 模型，例如 AlexNet [11] 和 VGGNet [12]，大多采用由空间卷积（Kernel 大小为 ≥ 3）和池化操作组成的结构。之后的 ResNet 和它的变体 [13]–[15] 在 CNN 中引入 Skip Connection 和具有 Bottleneck 结构的模块，使训练非常深的网络成为可能。Inception [16], [17] 利用带有多套专门滤波器（Filter）的并行路径，翻新了传统的积木式结构的设计模式。注意力机制 [18]–[24] 改善了卷积操作在捕捉局部特征方面的缺点。我们的工作也可以被看作是一个特殊的 CNN。与以往的 CNN 不同的是，我们的 Vision Permutator 仅由 1×1 卷积组成，但其依然可以编码全局信息。

我们的工作也与视觉 Transformer [25] 存在联系。和采用局部卷积去编码空间信息的 CNN 不同的是，视觉 Transformer 利用自注意机制来捕捉全局信息，是最近图像分类领域的主流研究方向。从那时起，出现了大量的基于 Transformer 的分类模型，旨在通过引入局部性（Locality）[10], [26]–[30]，或扩展深度 [9], [31]，或定制强大的优化策略 [8] 来推进原始视觉 Transformer 的发展。此外，也有一些旨在改进自注意机制的工作。例如，Ho 等人 [24], [32] 通过将多维的自注意分解为多个一维自注意，利用轴向（Axial）注意力来处理表征。虽然这种方法降低了计算成本，但其仍然依赖于自注意机制。与上述方法不同，我们的 Vision Permutator 消除了对自注意机制的依赖，因此更加高效。

最近，一些工作 [2], [3], [33], [34] 使用纯粹的类 MLP 模型进行图像分类。为了使用 MLP 编码丰富的空间信息，这些方法通常将空间维度扁平化，并将三维（高度、宽度和通道）Token 表征作为一个二维的表格输入。不同的是，我们的 Vision Permutator 针对三维特征表示进行操作，并沿高度和宽度两个维度分别编码空间信息。我们将在实验部分展示所提出的 Vision Permutator 相对于现有的类 MLP 模型的优势。

3 Vision Permutator

图 1展示了 Vision Permutator 的基本结构。我们的网络将大小为 224×224 作为输入，并均匀地将其分割为一连串的 Patch（14×14 or 7×7）。此后，我们将和 [2] 一样使用一个共享的线性层，将所有的块映射为线性 Embedding（或称为 Token）。接下来，我们将所有的 Token 输入一系列的 Permutator，以编码空间和通道信息。最后，沿空间维度对前面生成的 Token 进行平均，然后由全连接层进行类别预测。在下文中，我们将详细地介绍所提出的 Permutator 模块和模型的相关设置。

3.1 Permutator

图1的左上角是一个 Permutator 块的示意图。可以看出，忽略 LayerNorm 和 Skip Connection，我们的 Permutator 由两个组件组成：Permute-MLP and Channel-MLP，它们分别负责编码空间信息和通道信息。Channel-MLP 模块与 Transformer [35] 中的前馈层有类似的结构，其有两个全连接层，两个全连接层中间有一个 GELU 激活函数。对于空间信息的编码，与最近的 Mixer [2] 沿空间维度对所有 Token 进行线性投影不同，我们分别沿高度和宽度两个维度对 Token 进行处理。在数学上，给定一个 C 维的 TokenX $\in \mathbb{R}^{H \times W \times C}$，Permutator 的公式如下：

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

$$\mathrm{Z} = \text{MLP}(\text{LN}(\mathrm{Y})) + \mathrm{Y}, \tag{2}$$

其中，LN 指的是 LayerNorm。输出 Z 直到最后一个 Permutator 块之前都将作为为下一个 Permutator 块的输入。

Permute-MLP: 图2为 Permute-MLP 的示意图。与 Transformer [8], [25], [36] 和 Mixer [2] 的二维输入（“Token×Channel’，’ 即，$HW \times C$）不同的是，Permute-MLP 的输入为三维的 Token 表征。如图 2所示，我们的 Permute-MLP 由三个分支组成，每个分支都分别负责沿高度纬度、宽度维度或者通道维度编码信息。通道信息的编码较为简单，我

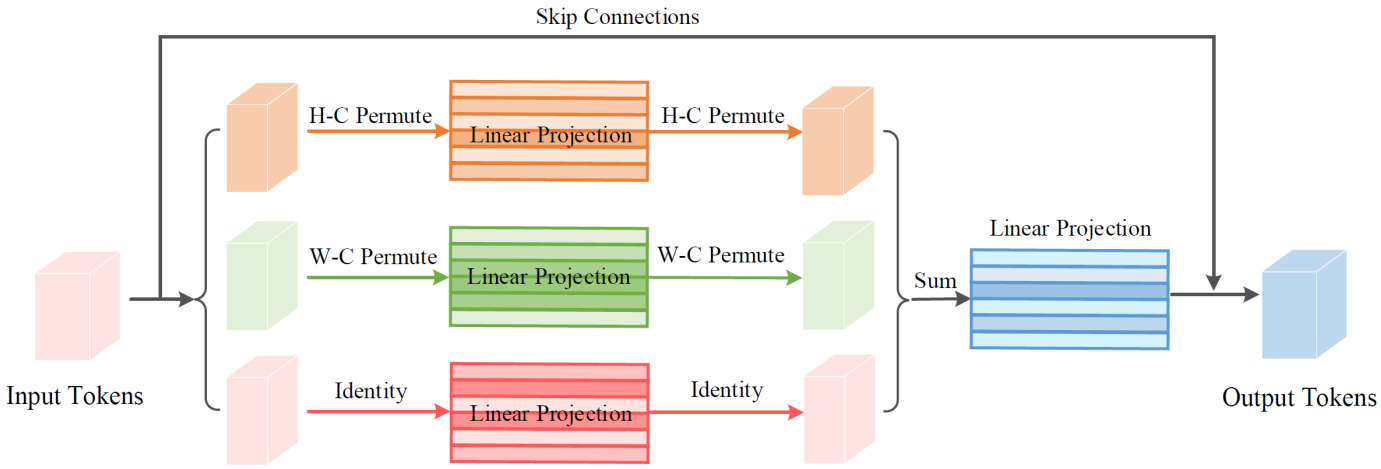

Figure 2. **所提出的** Permute-MLP **层的基本结构**。Permute-MLP **层包含三个分支，分别负责沿高度、宽度和通道维度对特征进行编码。然后，这三个分支的输出通过元素级加法进行组合，接着使用一个全连接层进行特征融合**。

Table 1
不同的 Vision Permutator **模型的配置。我们根据不同的模型尺寸，列出了三种不同的模型（小、中、大）。符号** "Small/16" **指在初始** Patch Embedding **模块中，** Patch **大小为** 16×16 **的模型**。

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

们只需要一个权重为 $\mathrm{W}_C \in \mathbb{R}^{C\times C}$ 的全连接层对输入 X 进行线性投影，得到 X_C。在下文中，我们将描述如何通过引入维度之间的分段式（Segment-Wise）互换操作（Permutation）来编码空间信息。

假设隐藏维度 C 为 384，且输入图像的分辨率为 224×224。为了沿着高度维度（H）编码空间信息，我们首先在高度通道进行一个互换操作。给定输入 $\mathrm{X} \in \mathbb{R}^{H\times W\times C}$，我们首先将其沿着通道维度分为 S 个 Segment，得到 $[\mathrm{X}_{H_1}, \mathrm{X}_{H_2}, \cdots, \mathrm{X}_{H_S}]$，满足 $C = N^{1} \times S$. 在 Patch 的大小被设置为 14×14 的情况下，N 的值等同于 $224/14 = 16$，且 $\mathrm{X}_{H_i} \in \mathbb{R}^{H\times W\times N}, (i \in \{1, \cdots, S\})$。我们接下来对每个 Segment X_{H_i} 进行一个高度-通道互换操作[2]，得到 $[\mathrm{X}_{H_1}^{\top}, \mathrm{X}_{H_2}^{\top}, \cdots, \mathrm{X}_{H_S}^{\top}]$，然后沿通道维度连接起来，以此作为互换操作的输出。接着，一个权重为 $\mathrm{W}_H \in \mathbb{R}^{C\times C}$ 的全连接层被用于融合高度信息。为了恢复 X 的原始纬度信息，我们只需要再进行一次高度通道互换操作，输出 X_H。同样，在第二分支中，我们进行与上述相同的操作，对 X 的宽度维度和通道维度进行互换，得到 X_W。最后，我们将三个分支的 Token 表征的和送入一个新的全连接层，以获得 Permute-MLP 层的输出，可表述为：

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

其中 $\mathrm{FC}(\cdot)$ 表示 $\mathrm{W}_P \in \mathbb{R}^{C\times C}$ 的全连接层。算法 1为类 PyTorch 的伪代码。

加权 Permute-MLP:

在式 3中，我们使用元素级加法简单地混合三个分支的输出。在这里，我们通过重新校准不同分支的重要性来进一步改进 Permute-MLP，并提出了加权 Permute-MLP。这可以

1. 这里，N 等同于 H。
2. 交换第一维（**高度**）和第三维（**通道**）：$(H, W, C) \rightarrow (C, W, H)$。

Algorithm 1 Permute-MLP 的伪代码 (PyTorch-like)

```
# H: 高度, W: 宽度, C: 通道, S: 段数
# x: 大小为 (H, W, C) 的输入张量
def init():
    proj_h = nn.Linear(C, C)   # 高度维度
    proj_w = nn.Linear(C, C)   # 宽度维度
    proj_c = nn.Linear(C, C)   # 通道信息
    proj   = nn.Linear(C, C)   # 融合

############ 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
```

通过将分离注意力（Split Attention）机制 [7] 轻松实现。不同的是，分离注意机制被用于 X_H, X_W 和 X_C，而不是由分组卷积产生的一组张量。之后，我们将在 Permutator 中默认使用加权的 Permute-MLP。

与卷积网络和 Transformer 之间的联系:

与 Mixer-MLP 相似，我们的 Vision Permutator 主要由 MLP 组成。但是，从模型实现的角度，我们确实是用了卷积来进行 Patch 的嵌入操作，这也可以被视为一个下采样（Downsampling）的操作。我们的工作可以被视为 ConvNet 和 MLP 的混合体，但所使用的大多数算子是全连接层。Transformer 也只由 MLP 组成，但它们依靠自注意力机制来建立成对 Token 之间的配对关系。我们的 Permutator 没有明确地模拟成对 Token 的相似性，因此和 Transformer 有着很大不同。

3.2 Vision Permutator 的各种配置

在表 1中，我们总结了 Vision Permutator 的各种配置。我们列出了三个不同版本的 Vision Permutator（ViP），根据其模型大小分别表示为 'ViP-Small'，'ViP-Medium' 和 'ViP-Large'。'ViP-Small/14' 表示在初始 Patch Embedding 中 Patch 大小为 14×14 的小尺寸模型 在 'ViP-Small/16' 和 'ViP-Small/14' 模型中，只有一个 Patch Embedding 模块，之后是一连串的 Permutator。 每个模型均有 18 个 Permutator。

我们的 'ViP-Small/7'，'ViP-Medium/7' 和 'ViP-Large/7' 有两个阶段。第一个阶段从 Patch Embedding 模块开始。我们增加了几个 Permutator，以此对细粒度的 Token 表征进行编码，我们发现这对模型的性能有好处。在第二阶段中，我们在开始时使用了一个下采样操作，将 Token 表征映射到较低水平。对于所有的模型，我们依据 T2T-ViT 模型 [30] 将 MLP 的扩展率（Expansion Ratio）设为 3。我们发现扩展率设为 4 时，模型性能几乎没有改善，但增加了更多的计算代价。

4 实验

我们在广泛使用的 ImageNet-1k [4] 数据集上对我们的 Vision Permutator 进行了实验，并列出了实验结果。实验代码是基于 PyTorch [37] 和 timm [38] 工具箱实现的。注意，在训练中，我们不使用任何额外的训练数据。

4.1 实验环境及设置

根据先前工作 [8], [36] 的建议，我们使用 AdamW 优化器 [39]，其通过线性学习率扩展策略 $lr = 10^{-3} \times \frac{batch\_size}{1024}$ 和 5×10^{-2} 的权重衰减率来优化实验中的所有模型。在我们的 Vision Permutator 中，Batch Size 设为 2048，通过实验我们发现这比将 Batch Size 设为 1024 效果更好。在实验中，我们也使用了随机深度（Stochastic Depth） [40]。表 1展示了具体的下降率（Drop Rate）。我们在 ImageNet 数据集上进行了 300 个 epoch 的训练. 我们使用 Random Erasing [41], RandAug [42]，MixUp [43] 和 CutMix [44] 作为数据增广的方法。请注意，我们的 Vision Permutator 中不使用位置编码，因为我们发现它降低了模型的性能。训练小型的 Vision Permutator 模型需要一个带有 8 个 NVIDIA V100 GPU（32G memory）的机器节点。对于中型和大型的 Vision Permutator 模型，则需要两个机器节点。

4.2 ImageNet 上的主要结果

在本小节中，我们将我们提出的 Vision Permutator 与之前在 ImageNet [4]、ImageNet Real [45] 和 ImageNet-V2 [46] 上的基于 CNN、基于 Transformer 和类 MLP 的模型进行比较。在表 2中，我们首先将我们提出的 Vision Permutator 与最近的类 MLP 模型进行对比。'Train size' 和 'Test size' 分别表示训练时采用的分辨率和测试时采用的分辨率。可以看到，我们的 ViP-Small/7 模型虽然只有 25M 的参数，但其 top-1 精度达到了 81.5%。这个结果已经优于大多数现有的类 MLP 模型，并且可以与最好的 gMLP-B [33] 模型相媲美，而后者有 73M 的参数，远远超过我们。当模型参数量达到 55M 时，使我们的 ViP-Medium/7 达到 82.7% 的准确率。如表 2所示，优于所有其他的类 MLP 模型。当模型参数量进一步达到 88M 时，可以达到更好的结果：83.2%。在 ImageNet Real 和 ImageNet-V2 数据集上也可以观察到类似的改进，这反映出我们的方法与其他模型相比可以更好地防止过拟合现象的发生。

我们认为，导致我们的 Vision Permutator 相比于其他模型有所改进的主要因素是第 3节中所描述针对空间信息的

Table 2
和最近的类 MLP 模型在 [4]，ImageNet Real [45] 和 ImageNet-V2 [46] 数据集上 Top-1 准确率的比较结果。所有模型都是在没有外部数据的情况下训练的。在相同的计算资源和参数量的约束下，我们的模型始终优于其他具有相似吞吐量（Throughput）的方法。按照 [3]，吞吐量是在一台带有 V100 GPU（32GB）的机器上测量的，Batch Size 为 32。† 我们发现我们的训练策略比论文中报告的训练策略效果更好。

Networks	Parameters	FLOPs	Throughput	Train size	Test size	ImageNet	ImageNet Real	ImageNetV2
EAMLP-14 [34]	30M	-	711 img/s	224	224	78.9	-	-
gMLP-S [33]	20M	4.5B	-	224	224	79.6	-	-
ResMLP-S24 [3]	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 [34]	55M	-	464 img/s	224	224	79.4	-	-
Mixer-B/16 [2]†	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 [33]	73M	15.8B	-	224	224	81.6	-	-
ResMLP-B24 [3]	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
和经典的 CNN 模型与视觉 Transformer 模型在 [4]，ImageNet Real [45] 和 ImageNet-V2 [46] 数据集上 Top-1 准确率的比较结果。所有模型都是在没有外部数据的情况下训练的。在相同的计算资源和参数量的约束下，我们的模型与一些强大的基于 CNN 和基于 Transformer 的模型相比具有竞争力。

Network	Parameters	FLOPs	Train size	Test size	ImageNet	ImageNet Real	ImageNetV2
NFNet-F6 + SAM [6]	438M	377B	448	576	86.5	89.2	75.8
CaiT-M48 [9]	356M	330B	224	448	86.5	90.2	76.9
VOLO-D5 [47]	296M	304B	224	448	87.0	90.6	77.8
ResNet-50d [13], [48]	25.6M	4.3B	224	224	79.5	-	-
SE-ResNeXt-50 [14], [18]	27.6M	4.3B	224	224	79.9	85.3	68.7
RegNet-4GF [49]	21M	4.0B	224	224	80.0	-	-
ResNeSt-50 [7]	27.5M	5.4B	224	224	81.1	-	-
DeiT-S [36]	22M	4.6B	224	224	79.8	85.7	68.5
T2T-ViT-14 [30]	22M	5.2B	224	224	81.5	86.8	69.9
Swin-T [10]	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 [13], [48]	44.6M	7.9B	224	224	80.4	85.8	69.0
SE-ResNeXt-101 [14], [18]	49.0M	8.0B	224	224	80.9	86.0	70.0
ResNeSt-101 [7]	48.3M	10.2B	256	256	82.9	87.3	72.6
DeepViT [31]	55M	12.5B	224	224	83.1	-	-
ViP-Medium/7	55M	16.3B	224	224	82.7	87.4	72.2
RegNet-16GF [49]	83.6M	15.9B	224	224	82.9	88.1	72.4
DeiT-B [36]	86M	17.5B	224	224	81.8	86.7	71.5
T2T-ViT-24 [30]	64M	13.8B	224	224	82.3	-	-
TNT-B [29]	66M	14.1B	224	224	82.8	-	-
ViP-Large/7	88M	24.3B	224	224	83.2	87.6	72.7

编码方式。与表 2中列出的流行的类 MLP 模型不同，我们沿着高度和宽度的维度分别编码 Token 表征。此外，我们的 Vision Permutator 不仅编码了粗粒度的 Token 表征（16×16 的 Token），而且还编码了细粒度的特征（32×32 的 Token），这点在视觉 Transformer [47] 中被证明是很重要的。我们将在下一小节中详细介绍这一点。

在表 3中，我们将 Vision Permutator 与经典的基于 CNN 的模型和基于 Transformer 的模型进行比较。与经典的 CNN 模型相比，如 ResNets [13]，SE-ResNeXt [14], [18] 和 RegNet [49]，我们的 Vision Permutator 在相近模型尺寸的约束下

得到了更好的结果。以 ViP-Small/7 模型为例，其在 ImageNet 数据集上的 Top-1 准确率为 81.5%，这甚至比 ResNeSt-50 (81.5% v.s. 81.1%) 更好。与基于 Transformer 的模型相比，如 DeiT [36]，T2T-ViT [30] 和 Swin Transformers [10]，我们模型的结果也更好。这一现象表明，类 MLP 模型是基于 CNN 和 Transformer 的模型的有力竞争者。然而，我们的 Vision Permutator 与基于 CNN 和 Transformer 的 SOTA 模型，如 NFNet [6]、CaiT [9] 和 VOLO [47]，之间仍有很大差距。我们相信，与视觉 Transformer 一样，类 MLP 的模型还有很大的改进空间。

4.3 方法分析

在本小节中，我们进行了一系列关于细粒度信息编码、模型规模、数据增广和 Permutator 的消融实验。我们把 ViP-Small/14 模型作为 Baseline。

细粒度 Token 表征编码的重要性: 我们认为编码更细粒度的 Token 表征对类 MLP 模型很重要。我们以两种方式证明这一论点：I) 调整初始 Patch Embedding 层中的 Patch 大小，并保持 Backbone 部分不变; II) 将每个 Patch 的大小减半，并引入几个 Permutator 来编码细粒度的 Token 表征。表 4总结了 ViP-Small/16、ViP-Small/14 和 ViP-Small/7 的性能。和 ViP-Small/16 相比，ViP-Small/14 有较小的初始图像块尺寸和更多的 Token 输入至 Permutator。根据结果，ViP-Small/14 比 ViP-Small/16 效果更好 (80.5% v.s. 79.8%)。尽管在 ViP-Small/14 中使用了更多的 Token 和更多的参数量，但效率 (吞吐量) 并没有什么变化这表明我们可以适当地使用较小的初始 Patch 尺寸来提高模型的性能。我们将初始 Patch 大小进一步从 14×14 减少至 7×7。与 ViP-Small/14 相比，ViP-Small/7 采用 4 个 Permutator 来编码细粒度 Token 表征 (32×32 的 Token)。根据表 4，这种轻微的修改可以在很大程度上提高性能并减少模型的参数量。Top-1 准确率将从 80.5% 提升至 81.5%。这表明，对细粒度的 Token 表征进行编码确实有助于提高我们的模型性能，但缺点是模型的效率会有所下降。

模型规模的作用: 扩大深度神经网络的模型规模始终是提高模型性能的有效途径。在这里，我们通过增加层数和隐藏维度，展示了模型规模对 Vision Permutator 的影响。表 5列出了三个不同版本的 Vision Permutator：ViP-Small/7, ViP-Medium/7, and ViP-Large/7。我们可以看到，增加层数和隐藏维度可以使我们的 Vision Permutator 产生更好的结果。ViP-Medium/7 可以将 ViP-Small/7 的准确率提高至 82.7%，性能增益超过 1%。进一步增加模型的大小会使准确率进一步提高至到 83.2%。

数据增广的作用: 数据增广已被证明是提升深度学习中模型性能的高效方法 [8], [36], [48]。以下是四种常用的数据增广方法：Random Augmentation [42]，Random Erasing [41]，MixUp [43] 和 CutMix [44]。这里，我们展示了上述各种方法对模型性能的影响。表 6为实验结果。在没有任何数据增广的情况下，我们的 ViP-Small/14 模型的 Top-1 准确率达到了 75.3%。使用 Random Augmentation 模型的性能将提升至 77.7% (+2.4%)。增加 Random Erasing 使结果提升至 78.0% (+2.7%)。增加 MixUp 是 Top-1 准确率达到了 80.2%(+4.9%)。使用 CutMix 将进一步将准确率提升至 80.6% (+5.3%)。这些实验表明，在训练 CNN [48] 和视觉 Transformer [8], [36] 时，数据增广是极为重要。

从 Mixer 到 Vision Permutator: 原始的 Mixer [2] 通过扁平化所有的 Token 来编码空间信息。为了显示单独编码空间信息的优势，我们将图 2中的前两个分支替换成 Mixer-MLP 中的 token-mixing MLP 部分。表 7列出了实验的结果。如表 7第三行所示，新模型的性能为 78.9%，比通过我们的训练策略训练的 Mixer-B/16 略好，但比我们的 ViP-Small/14 差 (78.9 这些实验表明，分别对高度和宽度信息进行编码对视觉识别更有用。

对 Permutator 的消融实验: 在这里，我们证明了沿高度和宽度维度分别编码空间信息的重要性，并且展示了加权的 Permutator 如何帮助模型提高性能。在表 7中，我们总结了 Permutator 不同设置下的结果。关于每个设置的详细说明可以在表格的说明中找到。我们可以看到，放弃对高度信息或宽度信息编码都会导致更差的性能 (80.2% v.s. 72.8% or 72.7%)。这表明对高度和宽度信息的编码都很重要。此外，我们还可以观察到，用加权的 Permute-MLP 替代默认的 Permute-MLP，可以进一步地将模型的准确率从 80.2% 提升至 80.6%。

如图 2所示，我们分别沿着高度和宽度两个维度编码 Token 表征 (前两个分支)。嵌入空间信息的另一种方式是将前两个分支合并为一个，即沿着两个空间维度依次处理表征，这与轴向注意力机制 [24] 类似。然而，我们根据经验发现，这种按顺序编码空间信息的方式不如我们的方法有效。实验表明，沿着水平和垂直维度对空间信息进行顺序编码会降低模型的性能。如表 7所示，这种操作将模型的分类性能从 80.2% 降低至 79.8% (-0.4%)。

迁移学习: 为了测试迁移学习的能力，我们尝试使用 ViP-Small/7 在 CIFAR10，CIFAR100 和 iNaturalist 2021 [50] 数据集上进行实验。我们使用和 T2T-ViT 一样的设置。表 8列出了实验结果。我们可以看到，所提出的 ViP-Small/7 在 CIFAR10、CIFAR100 和 iNaturalist 2021 数据集上取得了与最近流行的 T2T-ViT 相同甚至更好的结果。这表明，我们的 Vision Permutator 在小型分类数据集上的迁移学习中表现良好。

位置编码: Mixer 的工作表明，没有必要在类 MLP 模型中使用位置 Embedding。在这里，我们进行了相关实验调查了它是否有利于我们的 Vision Permutator。我们试图在第一个 Permutator 之前使用位置 Embedding，并使其可被学习，同

Table 4
细粒度 Token 表征编码的作用。'Initial Patch Size' 表示初始 Patch Embedding 模块的 Patch 大小。'Fine Token' 表示编码细粒度 Token 表征的模型。如表 1所规定的，大的 Patch 的尺寸意味着输入 Permutator 的 Token 数量会变少。我们可以看到，在改变初始 Patch 大小时，模型的速度效率并没有发生太大的变化。我们也列出了基于两个 Batch Size（32 和 128）的吞吐量。

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
模型规模的作用。我们通过增加模型大小（包括层数、隐藏维度）来扩展模型规模。'Hidden Dim'。指第二阶段的隐藏维度，它在第一阶段被减半。显然，增加模型规模可以持续改善模型性能。

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

Table 6
对数据增广方法的消融实验。我们在基于 CNN 和 Transformer 的模型中针对常用的四种数据增广方法进行了实验，这些方法包括 Random Augmentation [42]，Random Erasing [41]，MixUp [43] 和 CutMix [44]。我们可以看到，所有 4 种方法都对提升模型性能有所贡献。

Data augmentation methods	Layers	Params	Top-1 Acc. (%)
Baseline (ViP-Small/14)	18	30M	75.3
+ Random Aug. [42]	18	30M	77.7 (+2.4)
+ Random Erasing [41]	18	30M	78.0 (+2.7)
+ MixUp [43]	18	30M	80.2 (+4.9)
+ CutMix [44]	18	30M	80.6 (+5.3)

时我们也尝试增加相对位置 Embedding [51]。实验表明，这两种方法都使模型的准确率下降了约 0.3%。这是因为我们的方法在单独处理空间特征时已经嵌入了位置信息。因此，在我们的 Vision Permutator 中不需要位置 Embedding。

5 总结与展望

这本文中，我们提出了一种用于视觉识别的新型的类 MLP 网络结构，称为 Vision Permutator。我们认为，与将两个空间维度视为一体的类 MLP 模型相比，分别对两个空间信息进行编码可以在很大程度上提高模型性能。我们的实验也充分支持了这一点。尽管我们的 Vision Permutator 与目前流行的类 MLP 模型相比有很大的改进，但我们提出的 Permutator 仍然有一个明显缺点，即空间维度上的缩放问题，这在其他类 MLP 模型中也存在。由于全连接层中参数的形状是固定的，因此无法处理具有任意尺寸的输入图像。这使得类 MLP 模型很难被应用于具有各种尺寸的输入图像的下游任务中。

考虑到并行化的高效性，我们未来的工作将持续放在类 MLP 模型的开发上。具体来说，我们将继续研究如何克服类 MLP 模型的局限性，即无法处理具有任意形状的输入图像，以及它们在下游任务中的应用，如物体检测和语义分割。

致谢

这项工作得到了中国国家重点研发计划（批准号：2018AAA0100400）的支持。Ming-Ming Cheng 得到了 CAAI-Huawei Open Fund 的支持。

References

[1] Q. Hou, Z. Jiang, L. Yuan, M.-M. Cheng, S. Yan, and J. Feng, "Vision permutator: A permutable mlp-like architecture for visual recognition," 2021.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

Table 7
对 Vision Permutator 的消融实验。'ViP-Small/14 w/o Height' 是指 ViP-Small/14 模型的高度信息编码部分被通道编码取代（图 2中的底部分支）。类似的含义适用于 'ViP-Small/14 w/o Width'。'ViP-Small/14 w/ Permute-MLP' 指的是默认的 Permute-MLP。'ViP-Small/14 w/ Cascaded Permute-MLP' 指的是以级联的方式沿两个空间维度对空间信息进行编码。

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. → 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
对下游数据集（CIFAR10，CIFAR100 和 iNaturalist 2021 [50]）进行预训练的 ViP-S7 的微调（FineTuning）结果。和 [30] 一样，我们对所有的模型进行了 60 个 epoch 的微调。

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

[8] 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.

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

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

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

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

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

[40] 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.

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.