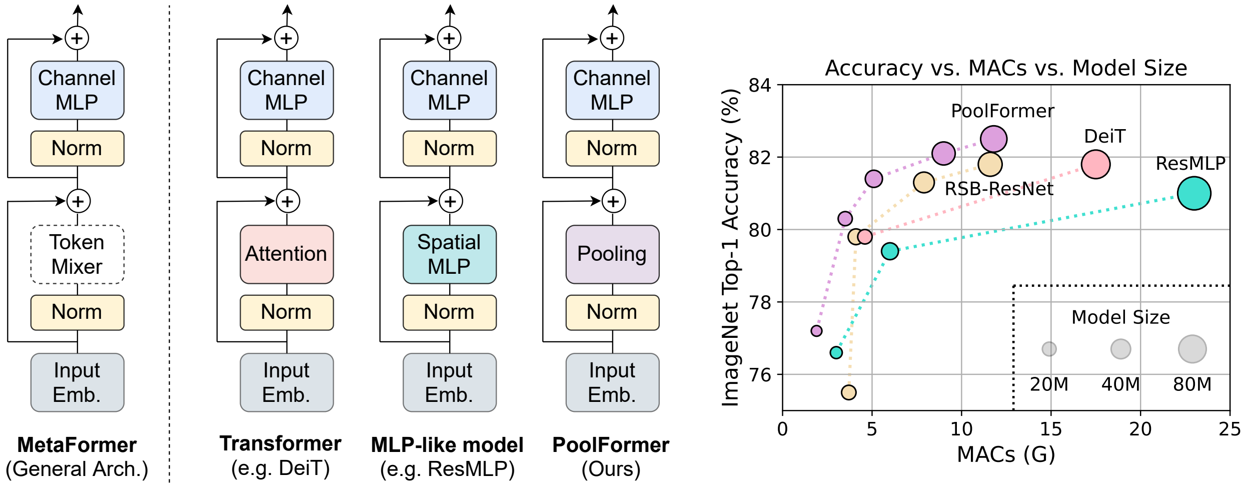
This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

# MetaFormer is Actually What You Need for Vision

Weihao Yu[1,2*] Mi Luo[1] Pan Zhou[1] Chenyang Si[1] Yichen Zhou[1,2]
Xinchao Wang[2] Jiashi Feng[1] Shuicheng Yan[1]
[1]Sea AI Lab [2]National University of Singapore
weihaoyu6@gmail.com {luomi,zhoupan,sicy,zhouyc,fengjs,yansc}@sea.com xinchao@nus.edu.sg

Code: https://github.com/sail-sg/poolformer



(a) (b)

Figure 1. **MetaFormer and performance of MetaFormer-based models on ImageNet-1K validation set.** As shown in (a), we present *MetaFormer* as a general architecture abstracted from transformers [54] by not specifying the token mixer. When using attention/spatial MLP as the token mixer, MetaFormer is instantiated as transformer/MLP-like models. We argue that the competence of transformer/MLP-like models primarily stems from the general architecture MetaFormer instead of the equipped specific token mixers. To demonstrate this, we exploit an embarrassingly simple non-parametric operator, *pooling*, to conduct extremely basic token mixing. Surprisingly, the resulted model *PoolFormer* consistently outperforms the well-tuned vision transformer [17] baseline (DeiT [51]) and MLP-like [49] baseline (ResMLP [50]) as shown in (b), which well supports that MetaFormer is actually what we need to achieve competitive performance. RSB-ResNet in (b) means the results are from "ResNet Strikes Back" [57] where ResNet [23] are trained with improved training procedure for 300 epochs.

## Abstract

*Transformers have shown great potential in computer vision tasks. A common belief is their attention-based token mixer module contributes most to their competence. However, recent works show the attention-based module in transformers can be replaced by spatial MLPs and the resulted models still perform quite well. Based on this observation, we hypothesize that the general architecture of the transformers, instead of the specific token mixer module, is more essential to the model's performance. To verify this, we deliberately replace the attention module in transformers with an embarrassingly simple* spatial pooling *operator to conduct only basic token mixing. Surprisingly, we observe that the derived model, termed as PoolFormer, achieves competitive performance on multiple computer vision tasks. For example, on ImageNet-1K, PoolFormer achieves 82.1% top-1 accuracy, surpassing well-tuned vision transformer/MLP-like baselines DeiT-B/ResMLP-B24 by 0.3%/1.1% accuracy with 35%/52% fewer parameters and 49%/61% fewer MACs. The effectiveness of PoolFormer verifies our hypothesis and urges us to initiate the concept of "MetaFormer", a general architecture abstracted from transformers without specifying the token mixer. Based on the extensive experiments, we argue that MetaFormer is the key player in achieving superior results for recent transformer and MLP-like models on vision tasks. This work calls for more future research dedicated to improving MetaFormer instead of focusing on the token mixer modules. Additionally, our proposed PoolFormer could serve as a starting baseline for future MetaFormer architecture design.*

[*] Work done during an internship at Sea AI Lab.

# 1. Introduction

Transformers have gained much interest and success in the computer vision field [3, 8, 43, 53]. Since the seminal work of vision transformer (ViT) [17] that adapts pure transformers to image classification tasks, many follow-up models are developed to make further improvements and achieve promising performance in various computer vision tasks [35, 51, 61].

The transformer encoder, as shown in Figure 1(a), consists of two components. One is the attention module for mixing information among tokens and we term it as *token mixer*. The other component contains the remaining modules, such as channel MLPs and residual connections. By regarding the attention module as a specific token mixer, we further abstract the overall transformer into a general architecture *MetaFormer* where the token mixer is not specified, as shown in Figure 1(a).

The success of transformers has been long attributed to the attention-based token mixer [54]. Based on this common belief, many variants of the attention modules [13, 21, 55, 66] have been developed to improve the vision transformer. However, a very recent work [49] replaces the attention module completely with spatial MLPs as token mixers, and finds the derived MLP-like model can readily attain competitive performance on image classification benchmarks. The follow-up works [25, 34, 50] further improve MLP-like models by data-efficient training and specific MLP module design, gradually narrowing the performance gap to ViT and challenging the dominance of attention as token mixers.

Some recent approaches [31, 38, 39, 44] explore other types of token mixers within the MetaFormer architecture, and have demonstrated encouraging performance. For example, [31] replaces attention with Fourier Transform and still achieves around 97% of the accuracy of vanilla transformers. Taking all these results together, it seems as long as a model adopts MetaFormer as the general architecture, promising results could be attained. We thus hypothesize *compared with specific token mixers, MetaFormer is more essential for the model to achieve competitive performance*.

To verify this hypothesis, we apply an extremely simple non-parametric operator, *pooling*, as the token mixer to conduct only basic token mixing. Astonishingly, this derived model, termed *PoolFormer*, achieves competitive performance, and even consistently outperforms well-tuned transformer and MLP-like models, including DeiT [51] and ResMLP [50], as shown in Figure 1(b). More specifically, PoolFormer-M36 achieves 82.1% top-1 accuracy on ImageNet-1K classification benchmark, surpassing well-tuned vision transformer/MLP-like baselines DeiT-B/ResMLP-B24 by 0.3%/1.1% accuracy with 35%/52% fewer parameters and 49%/61% fewer MACs. These results demonstrate that MetaFormer, even with a naive token mixer, can still deliver promising performance. We thus argue that MetaFormer is our *de facto* need for vision models which is more essential to achieve competitive performance rather than specific token mixers. Note that it does not mean the token mixer is insignificant. MetaFormer still has this abstracted component. It means token mixer is not limited to a specific type, e.g. attention.

The contributions of our paper are two-fold. Firstly, we abstract transformers into a general architecture MetaFormer, and empirically demonstrate that the success of transformer/MLP-like models is largely attributed to the MetaFormer architecture. Specifically, by only employing a simple non-parametric operator, pooling, as an extremely weak token mixer for MetaFormer, we build a simple model named PoolFormer and find it can still achieve highly competitive performance. We hope our findings inspire more future research dedicated to improving MetaFormer instead of focusing on the token mixer modules. Secondly, we evaluate the proposed PoolFormer on multiple vision tasks including image classification [14], object detection [33], instance segmentation [33], and semantic segmentation [65], and find it achieves competitive performance compared with the SOTA models using sophistic design of token mixers. The PoolFormer can readily serve as a good starting baseline for future MetaFormer architecture design.

# 2. Related work

Transformers are first proposed by [54] for translation tasks and then rapidly become popular in various NLP tasks. In language pre-training tasks, transformers are trained on large-scale unlabeled text corpus and achieve amazing performance [2, 15]. Inspired by the success of transformers in NLP, many researchers apply attention mechanism and transformers to vision tasks [3, 8, 43, 53]. Notably, Chen *et al.* introduce iGPT [6] where the transformer is trained to auto-regressively predict pixels on images for self-supervised learning. Dosovitskiy *et al.* propose vision transformer (ViT) with hard patch embedding as input [17]. They show that on supervised image classification tasks, a ViT pre-trained on a large propriety dataset (JFT dataset with 300 million images) can achieve excellent performance. DeiT [51] and T2T-ViT [61] further demonstrate that the ViT pre-trained on only ImageNet-1K ($\sim$ 1.3 million images) from scratch can achieve promising performance. A lot of works have been focusing on improving the token mixing approach of transformers by shifted windows [35], relative position encoding [59], refining attention map [66], or incorporating convolution [12, 20, 58], *etc.* In addition to attention-like token mixers, [49, 50] surprisingly find that merely adopting MLPs as token mixers can still achieve competitive performance. This discovery challenges the dominance of attention-based token mixers and triggers a heated discussion in the research community

**Algorithm 1** Pooling for PoolFormer, PyTorch-like Code

```
import torch.nn as nn

class Pooling(nn.Module):
   def __init__(self, pool_size=3):
      super().__init__()
      self.pool = nn.AvgPool2d(
         pool_size, stride=1,
         padding=pool_size//2,
         count_include_pad=False,
      )
   def forward(self, x):
      """
      [B, C, H, W] = x.shape
      Subtraction of the input itself is added
      since the block already has a
      residual connection.
      """
      return self.pool(x) - x
```

about which token mixer is better [7, 25]. However, the target of this work is neither to be engaged in this debate nor to design new complicated token mixers to achieve new state of the art. Instead, we examine a fundamental question: What is truly responsible for the success of the transformers and their variants? Our answer is the general architecture *i.e.*, MetaFormer. We simply utilize pooling as basic token mixers to probe the power of MetaFormer.

Contemporarily, some works contribute to answering the same question. Dong *et al.* prove that without residual connections or MLPs, the output converges doubly exponentially to a rank-1 matrix [16]. Raghu *et al.* [42] compare the feature difference between ViT and CNNs, finding that self-attention enables early aggregation of global information while residual connections strongly propagate features from lower to higher layers. Park *et al.* [41] shows that multi-head self-attentions improve accuracy and generalization by flattening the loss landscapes. Unfortunately, they do not abstract transformers into a general architecture and study them from the aspect of general framework.

# 3. Method

## 3.1. MetaFormer

We present the core concept "MetaFormer" for this work at first. As shown in Figure 1, abstracted from transformers [54], MetaFormer is a general architecture where the token mixer is not specified while the other components are kept the same as transformers. The input $I$ is first processed by input embedding, such as patch embedding for ViTs [17],

$$X = \mathrm{InputEmb}(I), \tag{1}$$

where $X \in \mathbb{R}^{N \times C}$ denotes the embedding tokens with sequence length $N$ and embedding dimension $C$.

Then, embedding tokens are fed to repeated MetaFormer blocks, each of which includes two residual sub-blocks. Specifically, the first sub-block mainly contains a token mixer to communicate information among tokens and this sub-block can be expressed as

$$Y = \mathrm{TokenMixer}(\mathrm{Norm}(X)) + X, \tag{2}$$

where $\mathrm{Norm}(\cdot)$ denotes the normalization such as Layer Normalization [1] or Batch Normalization [27]; $\mathrm{TokenMixer}(\cdot)$ means a module mainly working for mixing token information. It is implemented by various attention mechanism in recent vision transformer models [17, 61, 66] or spatial MLP in MLP-like models [49, 50]. Note that the main function of the token mixer is to propagate token information although some token mixers can also mix channels, like attention.

The second sub-block primarily consists of a two-layered MLP with non-linear activation,

$$Z = \sigma(\mathrm{Norm}(Y)W_1)W_2 + Y, \tag{3}$$

where $W_1 \in \mathbb{R}^{C \times rC}$ and $W_2 \in \mathbb{R}^{rC \times C}$ are learnable parameters with MLP expansion ratio $r$; $\sigma(\cdot)$ is a non-linear activation function, such as GELU [24] or ReLU [40].

**Instantiations of MetaFormer.** MetaFormer describes a general architecture with which different models can be obtained immediately by specifying the concrete design of the token mixers. As shown in Figure 1(a), if the token mixer is specified as attention or spatial MLP, MetaFormer then becomes a transformer or MLP-like model respectively.

## 3.2. PoolFormer

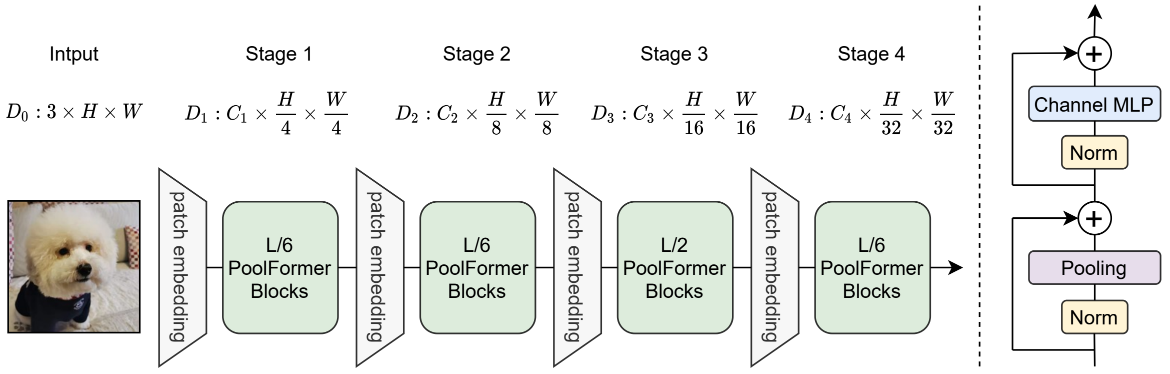
From the introduction of transformers [54], lots of works attach much importance to the attention and focus on designing various attention-based token mixer components. In contrast, these works pay little attention to the general architecture, *i.e.*, the MetaFormer.

In this work, we argue that this MetaFormer general architecture contributes mostly to the success of the recent transformer and MLP-like models. To demonstrate it, we deliberately employ an embarrassingly simple operator, pooling, as the token mixer. This operator has no learnable parameters and it just makes each token averagely aggregate its nearby token features.

Since this work is targeted at vision tasks, we assume the input is in channel-first data format, *i.e.*, $T \in \mathbb{R}^{C \times H \times W}$. The pooling operator can be expressed as

$$T'_{:,i,j} = \frac{1}{K \times K} \sum_{p,q=1}^{K} T_{:,i+p-\frac{K+1}{2},i+q-\frac{K+1}{2}} - T_{:,i,j}, \tag{4}$$

where $K$ is the pooling size. Since the MetaFormer block already has a residual connection, subtraction of the input itself is added in Equation (4). The PyTorch-like code of the pooling is shown in Algorithm 1.


Intput
Stage 1
Stage 2
Stage 3
Stage 4
$D_0 : 3 \times H \times W$
$D_1 : C_1 \times \frac{H}{4} \times \frac{W}{4}$
$D_2 : C_2 \times \frac{H}{8} \times \frac{W}{8}$
$D_3 : C_3 \times \frac{H}{16} \times \frac{W}{16}$
$D_4 : C_4 \times \frac{H}{32} \times \frac{W}{32}$
patch embedding
L/6 PoolFormer Blocks
patch embedding
L/6 PoolFormer Blocks
patch embedding
L/2 PoolFormer Blocks
patch embedding
L/6 PoolFormer Blocks
Channel MLP
Norm
Pooling
Norm

(a) Overal framework with L PoolFormer blocks

(b) PoolFormer block

Figure 2. **(a) The overall framework of PoolFormer.** Similar to [23, 35, 55], PoolFormer adopts hierarchical architecture with 4 stages. For a model with L PoolFormer blocks, stage [1, 2, 3, 4] have [L/6, L/6, L/2, L/6] blocks, respectively. The feature dimension $D_i$ of stage $i$ is shown in the figure. **(b) The architecture of PoolFormer block.** Compared with transformer block, it replaces attention with extremely simple non-parametric operator, pooling, to conduct only basic token mixing.

As well known, self-attention and spatial MLP has computational complexity quadratic to the number of tokens to mix. Even worse, spatial MLPs bring much more parameters when handling longer sequences. As a result, self-attention and spatial MLPs usually can only process hundreds of tokens. In contrast, the pooling needs a computational complexity linear to the sequence length without any learnable parameters. Thus, we take advantage of pooling by adopting a hierarchical structure similar to traditional CNNs [23, 30, 47] and recent hierarchical transformer variants [35, 55]. Figure 2 shows the overall framework of PoolFormer. Specifically, PoolFormer has 4 stages with $\frac{H}{4} \times \frac{W}{4}$, $\frac{H}{8} \times \frac{W}{8}$, $\frac{H}{16} \times \frac{W}{16}$, and $\frac{H}{32} \times \frac{W}{32}$ tokens respectively, where $H$ and $W$ represent the width and height of the input image. There are two groups of embedding size: 1) small-sized models with embedding dimensions of 64, 128, 320, and 512 responding to the four stages; 2) medium-sized models with embedding dimensions 96, 192, 384, and 768. Assuming there are $L$ PoolFormer blocks in total, stages 1, 2, 3, and 4 will contain $L/6$, $L/6$, $L/2$, and $L/6$ PoolFormer blocks respectively. The MLP expansion ratio is set as 4. According to the above simple model scaling rule, we obtain 5 different model sizes of PoolFormer and their hyper-parameters are shown in Table 1.

| Stage | #Tokens | Layer Specification | | PoolFormer | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S12 | S24 | S36 | M36 | M48 |
| 1 | $\frac{H}{4} \times \frac{W}{4}$ | Patch Embedding | Patch Size | $7 \times 7$, stride 4 | | | | |
| | | | Embed. Dim. | 64 | | | 96 | |
| | | PoolFormer Block | Pooling Size | $3 \times 3$, stride 1 | | | | |
| | | | MLP Ratio | 4 | | | | |
| | | | # Block | 2 | 4 | 6 | 6 | 8 |
| 2 | $\frac{H}{8} \times \frac{W}{8}$ | Patch Embedding | Patch Size | $3 \times 3$, stride 2 | | | | |
| | | | Embed. Dim. | 128 | | | 192 | |
| | | PoolFormer Block | Pooling Size | $3 \times 3$, stride 1 | | | | |
| | | | MLP Ratio | 4 | | | | |
| | | | # Block | 2 | 4 | 6 | 6 | 8 |
| 3 | $\frac{H}{16} \times \frac{W}{16}$ | Patch Embedding | Patch Size | $3 \times 3$, stride 2 | | | | |
| | | | Embed. Dim. | 320 | | | 384 | |
| | | PoolFormer Block | Pooling Size | $3 \times 3$, stride 1 | | | | |
| | | | MLP Ratio | 4 | | | | |
| | | | # Block | 6 | 12 | 18 | 18 | 24 |
| 4 | $\frac{H}{32} \times \frac{W}{32}$ | Patch Embedding | Patch Size | $3 \times 3$, stride 2 | | | | |
| | | | Embed. Dim. | 512 | | | 768 | |
| | | PoolFormer Block | Pooling Size | $3 \times 3$, stride 1 | | | | |
| | | | MLP Ratio | 4 | | | | |
| | | | # Block | 2 | 4 | 6 | 6 | 8 |
| Parameters (M) | | | | 11.9 | 21.4 | 30.8 | 56.1 | 73.4 |
| MACs (G) | | | | 1.9 | 3.5 | 5.1 | 9.0 | 11.8 |

Table 1. **Configurations of different PoolFormer models.** There are two groups of embedding dimensions, *i.e.*, small size with [64, 128, 320, 512] dimensions and medium size with [96, 196, 384, 768]. Notation "S24" means the model is in small size of embedding dimensions with 24 PoolFormer blocks in total.

# 4. Experiments

## 4.1. Image classification

**Setup.** ImageNet-1K [14] is one of the most widely used datasets in computer vision. It contains about 1.3M training images and 50K validation images, covering common 1K classes. Our training scheme mainly follows [51] and [52]. Specifically, MixUp [63], CutMix [62], CutOut [64] and RandAugment [11] are used for data augmentation. The models are trained for 300 epochs using AdamW optimizer [28, 36] with weight decay $0.05$ and peak learning rate $\text{lr} = 1e^{-3} \cdot \text{batch size}/1024$ (batch size 4096 and learning rate $4e^{-3}$ are used in this paper). The number of warmup epochs is 5 and cosine schedule is used to decay the learning rate. Label Smoothing [48] is set as 0.1. Dropout is disabled but stochastic depth [26] and LayerScale [52] are

| General Arch. | Token Mixer | Outcome Model | Image Size | Params (M) | MACs (G) | Top-1 (%) |
|---|---|---|---|---|---|---|
| Convolutional Neural Netowrks | — | ▼ RSB-ResNet-18 [57] | 224 | 12 | 1.8 | 70.6 |
| | | ▼ RSB-ResNet-34 [57] | 224 | 22 | 3.7 | 75.5 |
| | | ▼ RSB-ResNet-50 [57] | 224 | 26 | 4.1 | 79.8 |
| | | ▼ RSB-ResNet-101 [57] | 224 | 45 | 7.9 | 81.3 |
| | | ▼ RSB-ResNet-152 [57] | 224 | 60 | 11.6 | 81.8 |
| MetaFormer | Attention | ▲ ViT-B/16* [17] | 224 | 86 | 17.6 | 79.7 |
| | | ▲ ViT-L/16* [17] | 224 | 307 | 63.6 | 76.1 |
| | | ▲ DeiT-S [51] | 224 | 22 | 4.6 | 79.8 |
| | | ▲ DeiT-B [51] | 224 | 86 | 17.5 | 81.8 |
| | | ▲ PVT-Tiny [55] | 224 | 13 | 1.9 | 75.1 |
| | | ▲ PVT-Small [55] | 224 | 25 | 3.8 | 79.8 |
| | | ▲ PVT-Medium [55] | 224 | 44 | 6.7 | 81.2 |
| | | ▲ PVT-Large [55] | 224 | 61 | 9.8 | 81.7 |
| | Spatial MLP | ▶ MLP-Mixer-B/16 [49] | 224 | 59 | 12.7 | 76.4 |
| | | ▶ ResMLP-S12 [50] | 224 | 15 | 3.0 | 76.6 |
| | | ▶ ResMLP-S24 [50] | 224 | 30 | 6.0 | 79.4 |
| | | ▶ ResMLP-B24 [50] | 224 | 116 | 23.0 | 81.0 |
| | | ▶ Swin-Mixer-T/D24 [35] | 256 | 20 | 4.0 | 79.4 |
| | | ▶ Swin-Mixer-T/D6 [35] | 256 | 23 | 4.0 | 79.7 |
| | | ▶ Swin-Mixer-B/D24 [35] | 224 | 61 | 10.4 | 81.3 |
| | | ▶ gMLP-S [34] | 224 | 20 | 4.5 | 79.6 |
| | | ▶ gMLP-B [34] | 224 | 73 | 15.8 | 81.6 |
| | Pooling | ● PoolFormer-S12 | 224 | 12 | 1.9 | 77.2 |
| | | ● PoolFormer-S24 | 224 | 21 | 3.5 | 80.3 |
| | | ● PoolFormer-S36 | 224 | 31 | 5.1 | 81.4 |
| | | ● PoolFormer-M36 | 224 | 56 | 9.0 | 82.1 |
| | | ● PoolFormer-M48 | 224 | 73 | 11.8 | 82.5 |

Table 2. **Performance of different types of models on ImageNet-1K classification.** All these models are only trained on the ImageNet-1K training set and the accuracy on the validation set is reported. RSB-ResNet means the results are from "ResNet Strikes Back" [57] where ResNet [23] is trained with improved training procedure for 300 epochs. * denotes results of ViT trained with extra regularization from [49].

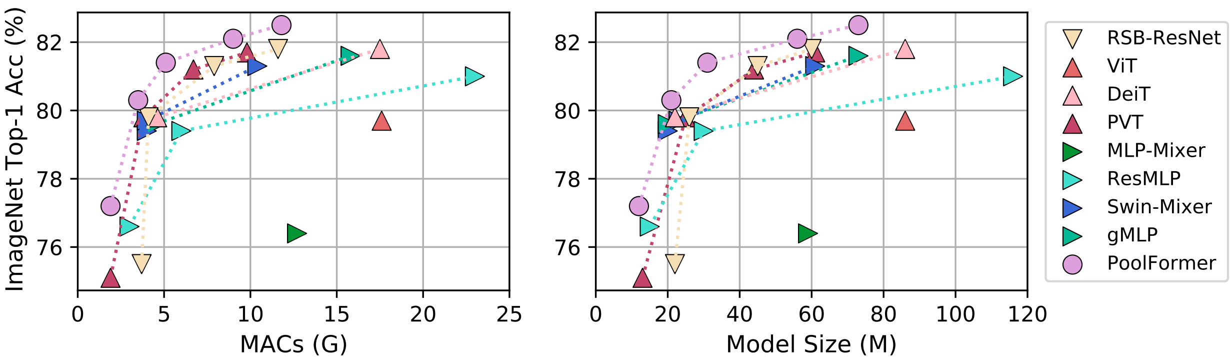


Figure 3. **ImageNet-1K validation accuracy *vs*. MACs/Model Size.** RSB-ResNet means the results are from "ResNet Strikes Back" [57] where ResNet [23] is trained with improved training procedure for 300 epochs.

used to help train deep models. We modified Layer Normalization [1] to compute the mean and variance along token and channel dimensions compared to only channel dimension in vanilla Layer Normalization. Modified Layer Normalization (MLN) can be implemented for channel-first data format with GroupNorm API in PyTorch by specifying the group number as 1. MLN is preferred by PoolFormer as shown in Section 4.4. See the appendix for more details on hyper-parameters. Our implementation is based on the `Timm` codebase [56] and the experiments are run on TPUs.

**Results.** Table 2 shows the performance of PoolFormers on ImageNet classification. Qualitative results are shown in the appendix. Surprisingly, despite the simple pooling token mixer, PoolFormers can still achieve highly competitive performance compared with CNNs and other MetaFormer-like models. For example, PoolFormer-S24 reaches the top-1 accuracy of more than 80 while only requiring 21M parameters and 3.5G MACs. Comparatively, the well-established ViT baseline DeiT-S [51], attains slightly worse accuracy of 79.8 but requires 31% more MACs (4.6G). To obtain similar accuracy, MLP-like model ResMLP-S24 [50] needs 43% more parameters (30M) as well as 71% more computation (6.0G) while only 79.4 accuracy is attained. Even compared with more improved ViT and MLP-like variants [34, 55], PoolFormer still shows better performance. Specifically, the pyramid transformer PVT-Medium obtains 81.2 top-1 accuracy with 44M parameters and 6.7G MACs while PoolFormer-S36 reaches 81.4 with 30% fewer parameters (31M) and 24% fewer MACs (5.1G) than those of PVT-Medium.

Besides, compared with RSB-ResNet ("ResNet Strikes Back") [57] where ResNet [23] is trained with improved training procedure for the same 300 epochs, PoolFormer still performs better. With $\sim$ 22M parameters/3.7G MACs, RSB-ResNet-34 [57] gets 75.5 accuracy while PoolFormer-S24 can obtain 80.3. Since the local spatial modeling ability of the pooling layer is much worse than the neural convolution layer, the competitive performance of PoolFormer can only be attributed to its general architecture MetaFormer.

With the pooling operator, each token evenly aggregates the features from its nearby tokens. Thus it is an extremely basic token mixing operation. However, the experiment results show that even with this embarrassingly simple token mixer, MetaFormer still obtains highly competitive performance. Figure 3 clearly shows that PoolFormer surpasses other models with fewer MACs and parameters. This finding conveys that the general architecture MetaFormer is actually what we need when designing vision models. By adopting MetaFormer, it is guaranteed that the derived models would have the potential to achieve reasonable performance.

### 4.2. Object detection and instance segmentation

**Setup.** We evaluate PoolFormer on the challenging COCO benchmark [33] that includes 118K training images (`train2017`) and 5K validation images (`val2017`). The models are trained on training set and the performance on validation set is reported. PoolFormer is employed as the backbone for two standard detectors, *i.e.*, RetinaNet [32] and Mask R-CNN [22]. ImageNet pre-trained weights are utilized to initialize the backbones and Xavier [19] to initialize the added layers. AdamW [28,36] is adopted for training with an initial learning rate of $1 \times 10^{-4}$ and batch size of 16. Following [22, 32], we employ 1× training schedule, *i.e.*, training the detection models for 12 epochs. The training images are resized into shorter side of 800 pixels and longer side of no more than 1,333 pixels. For testing, the shorter side of the images is also resized to 800 pixels. The implementation is based on the `mmdetection` [4] codebase and the experiments are run on 8 NVIDIA A100 GPUs.

**Results.** Equipped with RetinaNet for object detection, PoolFormer-based models consistently outperform their comparable ResNet counterparts as shown in Table 3. For instance, PoolFormer-S12 achieves 36.2 AP, largely surpassing that of ResNet-18 (31.8 AP). Similar results are observed for those models based on Mask R-CNN on object detection and instance segmentation. For example, PoolFormer-S12 largely surpasses ResNet-18 (bounding box AP 37.3 *vs*. 34.0, and mask AP 34.6 *vs*. 31.2). Overall, for COCO object detection and instance segmentation, PoolForemrs achieve competitive performance, consistently outperforming those counterparts of ResNet.

### 4.3. Semantic segmentation

**Setup.** ADE20K [65], a challenging scene parsing benchmark, is selected to evaluate the models for semantic segmentation. The dataset includes 20K and 2K images in the training and validation set, respectively, covering 150 fine-grained semantic categories. PoolFormers are evaluated as backbones equipped with Semantic FPN [29]. ImageNet-1K trained checkpoints are used to initialize the backbones while Xavier [19] is utilized to initialize other newly added layers. Common practices [5, 29] train models for 80K iterations with a batch size of 16. To speed up training, we double the batch size to 32 and decrease the iteration number to 40K. The AdamW [28,36] is employed with an initial learning rate of $2 \times 10^{-4}$ that will decay in the polynomial decay schedule with a power of 0.9. Images are resized and cropped into 512 × 512 for training and are resized to shorter side of 512 pixels for testing. Our implementation is based on the `mmsegmentation` [10] codebase and the experiments are conducted on 8 NVIDIA A100 GPUs.

**Results.** Table 4 shows the ADE20K semantic segmentation performance of different backbones using FPN [29]. PoolFormer-based models consistently outperform the models with backbones of CNN-based ResNet [23] and ResNeXt [60] as well as transformer-based PVT. For instance, PoolFormer-12 achieves mIoU of 37.1, 4.3 and 1.5 better than ResNet-18 and PVT-Tiny, respectively.

These results demonstrate that our PoorFormer which serves as backbone can attain competitive performance on semantic segmentation although it only utilizes pooling for basically communicating information among tokens. This further indicates the great potential of MetaFormer and supports our claim that MetaFormer is actually what we need.

### 4.4. Ablation studies

The experiments of ablation studies are conducted on ImageNet-1K [14]. Table 5 reports the ablation study of PoolFormer. We discuss the ablation below according to the following aspects.

| Backbone | RetinaNet 1× | | | | | | | Mask R-CNN 1× | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Params (M) | AP | AP$_{50}$ | AP$_{75}$ | AP$_S$ | AP$_M$ | AP$_L$ | Params (M) | AP$^{\mathrm{b}}$ | AP$^{\mathrm{b}}_{50}$ | AP$^{\mathrm{b}}_{75}$ | AP$^{\mathrm{m}}$ | AP$^{\mathrm{m}}_{50}$ | AP$^{\mathrm{m}}_{75}$ |
| ▼ ResNet-18 [23] | 21.3 | 31.8 | 49.6 | 33.6 | 16.3 | 34.3 | 43.2 | 31.2 | 34.0 | 54.0 | 36.7 | 31.2 | 51.0 | 32.7 |
| ● PoolFormer-S12 | 21.7 | 36.2 | 56.2 | 38.2 | 20.8 | 39.1 | 48.0 | 31.6 | 37.3 | 59.0 | 40.1 | 34.6 | 55.8 | 36.9 |
| ▼ ResNet-50 [23] | 37.7 | 36.3 | 55.3 | 38.6 | 19.3 | 40.0 | 48.8 | 44.2 | 38.0 | 58.6 | 41.4 | 34.4 | 55.1 | 36.7 |
| ● PoolFormer-S24 | 31.1 | 38.9 | 59.7 | 41.3 | 23.3 | 42.1 | 51.8 | 41.0 | 40.1 | 62.2 | 43.4 | 37.0 | 59.1 | 39.6 |
| ▼ ResNet-101 [23] | 56.7 | 38.5 | 57.8 | 41.2 | 21.4 | 42.6 | 51.1 | 63.2 | 40.4 | 61.1 | 44.2 | 36.4 | 57.7 | 38.8 |
| ● PoolFormer-S36 | 40.6 | 39.5 | 60.5 | 41.8 | 22.5 | 42.9 | 52.4 | 50.5 | 41.0 | 63.1 | 44.8 | 37.7 | 60.1 | 40.0 |

Table 3. **Performance of object detection using RetinaNet, and object detection and instance segmentation using Mask R-CNN on COCO `val2017` [33].** 1× training schedule (*i.e.*12 epochs) is used for training detection models. $AP^b$ and $AP^m$ represent bounding box AP and mask AP, respectively.

| Backbone | Semantic FPN | |
|---|---|---|
| | Params (M) | mIoU (%) |
| ▼ ResNet-18 [23] | 15.5 | 32.9 |
| ▲ PVT-Tiny [55] | 17.0 | 35.7 |
| ● PoolFormer-S12 | 15.7 | 37.2 |
| ▼ ResNet-50 [23] | 28.5 | 36.7 |
| ▲ PVT-Small [55] | 28.2 | 39.8 |
| ● PoolFormer-S24 | 23.2 | 40.3 |
| ▼ ResNet-101 [23] | 47.5 | 38.8 |
| ▼ ResNeXt-101-32x4d [60] | 47.1 | 39.7 |
| ▲ PVT-Medium [55] | 48.0 | 41.6 |
| ● PoolFormer-S36 | 34.6 | 42.0 |
| ▲ PVT-Large [55] | 65.1 | 42.1 |
| ● PoolFormer-M36 | 59.8 | 42.4 |
| ▼ ResNeXt-101-64x4d [60] | 86.4 | 40.2 |
| ● PoolFormer-M48 | 77.1 | 42.7 |

Table 4. **Performance of Semantic segmentation on ADE20K [65] validation set.** All models are equipped with Semantic FPN [29].

**Token mixers.** Compared with transformers, the main change made by PoolFormer is using simple pooling as a token mixer. We first conduct ablation for this operator by directly replacing pooling with identity mapping. Surprisingly, MetaFormer with identity mapping can still achieve 74.3% top-1 accuracy, supporting the claim that MetaFormer is actually what we need to guarantee reasonable performance.

Then the pooling is replaced with global random matrix $W_R \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ for each block. The matrix is initialized with random values from a uniform distribution on the interval [0, 1), and then Softmax is utilized to normalize each row. After random initialization, the matrix parameters are frozen and it conducts token mixing by $X' = W_R X$ where $X \in \mathbb{R}^{\mathbb{N}\times\mathbb{C}}$ are the input token features with the token length of $N$ and channel dimension of $C$. The token mixer of random matrix introduces extra 21M frozen parameters for the S12 model since the token lengths are extremely large at the first stage. Even with such random token mixing method, the model can still achieve reasonable performance of 75.8% accuracy, 1.5% higher than that of identity mapping. It shows that MetaFormer can still work well even with random token mixing, not to say with other well-designed token mixers.

Further, pooling is replaced with Depthwise Convolution [9, 37] that has learnable parameters for spatial modeling. Not surprisingly, the derived model still achieve highly competitive performance with top-1 accuracy of 78.1%, 0.9% higher than PoolFormer-S12 due to its better local spatial modeling ability. Until now, we have specified multiple token mixers in Metaformer, and all resulted models keep promising results, well supporting the claim that MetaFormer is the key to guaranteeing models' competitiveness. Due to the simplicity of pooling, it is mainly utilized as a tool to demonstrate MetaFormer.

We test the effects of pooling size on PoolFormer. We observe similar performance when pooling sizes are 3, 5, and 7. However, when the pooling size increases to 9, there is an obvious performance drop of 0.5%. Thus, we adopt the default pooing size of 3 for PoolFormer.

**Normalization.** We modify Layer Normalization [1] into Modified Layer Normalization (MLN) that computes the mean and variance along token and channel dimensions compared with only channel dimension in vanilla Layer Normalization. The shape of learnable affine parameters of MLN keeps the same as that of Layer Normalization, *i.e.*, $\mathbb{R}^{\mathbb{C}}$. MLN can be implemented with GroupNorm API in PyTorch by setting the group number as 1. See the appendix for details. We find PoolFormer prefers MLN with 0.7% or 0.8% higher than Layer Normalization or Batch Normalization. Thus, MLN is set as default for PoolFormer. When removing normalization, the model can not be trained to converge well, and its performance dramatically drops to only 46.1%.

**Activation.** We change GELU [24] to ReLU [40] or SiLU [18]. When ReLU is adopted for activation, an obvious performance drop of 0.8 % is observed. For SiLU, its performance is almost the same as that of GELU. Thus, we still adopt GELU as default activation.

**Other components.** Besides token mixer and normalization discussed above, residual connection [23] and channel MLP [45, 46] are two other important components in MetaFormer. Without residual connection or channel MLP,

| Ablation | Variant | Params (M) | MACs (G) | Top-1 (%) |
|---|---|---|---|---|
| Baseline | None (PoolFormer-S12) | 11.9 | 1.9 | 77.2 |
| Token mixers | Pooling → Identity mapping | 11.9 | 1.9 | 74.3 |
| | Pooling → Global random matrix$^*$ (extra 21M frozen parameters) | 11.9 | 3.3 | 75.8 |
| | Pooling → Depthwise Convolution [9, 37] | 11.9 | 1.9 | 78.1 |
| | Pooling size 3 → 5 | 11.9 | 1.9 | 77.2 |
| | Pooling size 3 → 7 | 11.9 | 1.9 | 77.1 |
| | Pooling size 3 → 9 | 11.9 | 1.9 | 76.8 |
| Normalization | Modified Layer Normalization$^\dagger$ → Layer Normalization [1] | 11.9 | 1.9 | 76.5 |
| | Modified Layer Normalization$^\dagger$ → Batch Normalization [27] | 11.9 | 1.9 | 76.4 |
| | Modified Layer Normalization$^\dagger$ → None | 11.9 | 1.9 | 46.1 |
| Activation | GELU [24] → ReLU [40] | 11.9 | 1.9 | 76.4 |
| | GELU → SiLU [18] | 11.9 | 1.9 | 77.2 |
| Other components | Residual connection [24] → None | 11.9 | 1.9 | 0.1 |
| | Channel MLP → None | 2.5 | 0.3 | 5.7 |
| Hybrid Stages | [Pool, Pool, Pool, Pool] → [Pool, Pool, Pool, Attention] | 14.0 | 2.0 | 78.3 |
| | [Pool, Pool, Pool, Pool] → [Pool, Pool, Attention, Attention] | 16.5 | 2.6 | 81.0 |
| | [Pool, Pool, Pool, Pool] → [Pool, Pool, Pool, SpatialFC] | 11.9 | 1.9 | 77.5 |
| | [Pool, Pool, Pool, Pool] → [Pool, Pool, SpatialFC, SpatialFC] | 12.2 | 1.9 | 77.9 |

Table 5. **Ablation for PoolFormer on ImageNet-1K classification benchmark.** PoolFormer-S12 is utilized as the baseline to conduct ablation study. The top-1 accuracy on the validation set is reported. $^*$This token mixer utilizes global random matrix $W_R \in \mathbb{R}^{N \times N}$ (parameters are frozen after random initialization) to conduct token mixing by $X' = W_R X$ where $X \in \mathbb{R}^{N \times C}$ are input tokens with the token length of $N$ and channel dimension of $C$. $^\dagger$Modified Layer Normalization (MLN) computes the mean and variance along token and channel dimensions compared with vanilla Layer Normalization only along channel dimension. MLN can be implemented with GroupNorm API in PyTorch by specifying the group number equal to 1.

the model cannot converge and only achieves the accuracy of 0.1%/5.7%, proving the indispensability of these parts.

**Hybrid stages.** Among token mixers based on pooling, attention, and spatial MLP, the pooling-based one can handle much longer input sequences while attention and spatial MLP are good at capturing global information. Therefore, it is intuitive to stack MetaFormers with pooling in the bottom stages to handle long sequences and use attention or spatial MLP-based mixer in the top stages, considering the sequences have been largely shortened. Thus, we replace the token mixer pooling with attention or spatial FC [1] in the top one or two stages in PoolFormer. From Table 5, the hybrid models perform quite well. The variant with pooling in the bottom two stages and attention in the top two stages delivers highly competitive performance. It achieves 81.0% accuracy with only 16.5M parameters and 2.6G MACs. As a comparison, ResMLP-B24 needs 7.0× parameters (116M) and 8.8× MACs (23.0G) to achieve the same accuracy. These results indicate that combining pooling with other token mixers for MetaFormer may be a promising direction to further improve the performance.

## 5. Conclusion and future work

In this work, we abstracted the attention in transformers as a token mixer, and the overall transformer as a general architecture termed MetaFormer where the token mixer is not specified. Instead of focusing on specific token mixers, we point out that MetaFormer is actually what we need to guarantee achieving reasonable performance. To verify this, we deliberately specify token mixer as extremely simple pooling for MetaFormer. It is found that the derived PoolFormer model can achieve competitive performance on different vision tasks, which well supports that “MetaFormer is actually what you need for vision”.

In the future, we will further evaluate PoolFormer under more different learning settings, such as self-supervised learning and transfer learning. Moreover, it is interesting to see whether PoolFormer still works on NLP tasks to further support the claim “MetaFormer is actually what you need” in the NLP domain. We hope that this work can inspire more future research devoted to improving the fundamental architecture MetaFormer instead of paying too much attention to the token mixer modules.

## Acknowledgement

The authors would like to thank Quanhong Fu at Sea AI Lab for the help to improve the technical writing aspect of this paper. Weihao Yu would like to thank TPU Research Cloud (TRC) program and Google Cloud research credits for the support of partial computational resources. This project is in part supported by NUS Faculty Research Committee Grant (WBS: A-0009440-00-00). Shuicheng Yan and Xinchao Wang are the corresponding authors.

[1] Following [50], we use only one spatial fully connected layer as a token mixer, so we call it FC.

## References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3, 5, 7, 8

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. 2

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 2

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 6

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 6

[6] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020. 2

[7] Shoufa Chen, Enze Xie, Chongjian Ge, Ding Liang, and Ping Luo. Cyclemlp: A mlp-like architecture for dense prediction. *arXiv preprint arXiv:2107.10224*, 2021. 3

[8] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ 2-nets: Double attention networks. *Advances in Neural Information Processing Systems*, 31:352–361, 2018. 2

[9] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 7, 8

[10] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020. 6

[11] 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*, pages 702–703, 2020. 4

[12] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*, 2021. 2

[13] Stéphane D'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2286–2296. PMLR, 18–24 Jul 2021. 2

[14] 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*, pages 248–255. Ieee, 2009. 2, 4, 6

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, 2019. 2

[16] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. *arXiv preprint arXiv:2103.03404*, 2021. 3

[17] 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. In *International Conference on Learning Representations*, 2020. 1, 2, 3, 5

[18] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. 7, 8

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 6

[20] Jianyuan Guo, Kai Han, Han Wu, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Cmt: Convolutional neural networks meet vision transformers. *arXiv preprint arXiv:2107.06263*, 2021. 2

[21] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 2

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 6

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 4, 5, 6, 7

[24] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3, 7, 8

[25] Qibin Hou, Zihang Jiang, Li Yuan, Ming-Ming Cheng, Shuicheng Yan, and Jiashi Feng. Vision permutator: A permutable mlp-like architecture for visual recognition. *arXiv preprint arXiv:2106.12368*, 2021. 2, 3

[26] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 4

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 3, 8

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4, 6

[29] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 6, 7

[30] 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. 4

[31] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021. 2

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 6

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 6, 7

[34] Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021. 2, 5, 6

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021. 2, 4, 5

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. 4, 6

[37] Franck Mamalet and Christophe Garcia. Simplifying convnets for fast learning. In *International Conference on Artificial Neural Networks*, pages 58–65. Springer, 2012. 7, 8

[38] André Martins, António Farinhas, Marcos Treviso, Vlad Niculae, Pedro Aguiar, and Mario Figueiredo. Sparse and continuous attention mechanisms. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20989–21001. Curran Associates, Inc., 2020. 2

[39] Pedro Henrique Martins, Zita Marinho, and André FT Martins. $\infty$-former: Infinite memory transformer. *arXiv preprint arXiv:2109.00301*, 2021. 2

[40] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 3, 7, 8

[41] Namuk Park and Songkuk Kim. How do vision transformers work? In *International Conference on Learning Representations*, 2022. 3

[42] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *arXiv preprint arXiv:2108.08810*, 2021. 3

[43] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[44] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *arXiv preprint arXiv:2107.00645*, 2021. 2

[45] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961. 7

[46] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 7

[47] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 4

[48] 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*, pages 2818–2826, 2016. 4

[49] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021. 1, 2, 3, 5

[50] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021. 1, 2, 3, 5, 6, 8

[51] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 1, 2, 4, 5, 6

[52] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 4

[53] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12894–12904, 2021. 2

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1, 2, 3

[55] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 568–578, October 2021. 2, 4, 5, 6, 7

[56] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 5

[57] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 1, 5, 6

[58] 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. 2

[59] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10033–10041, 2021. 2

[60] 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*, pages 1492–1500, 2017. 6, 7

[61] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 558–567, October 2021. 2, 3

[62] Sangdoo 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*, pages 6023–6032, 2019. 4

[63] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 4

[64] 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, pages 13001–13008, 2020. 4

[65] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 2, 6, 7

[66] 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*, 2021. 2, 3