11-2023

MetaFormer baselines for vision

Weihao YU

Chenyang SI

Pan ZHOU
Singapore Management University, panzhou@smu.edu.sg

Mi LUO

Yichen ZHOU

See next page for additional authors

Citation

YU, Weihao; SI, Chenyang; ZHOU, Pan; LUO, Mi; ZHOU, Yichen; FENG, Jiashi; YAN, Shuicheng; and WANG, Xinchao. MetaFormer baselines for vision. (2023). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 46, (2), 896-912.
Available at: https://ink.library.smu.edu.sg/sis_research/9054

MetaFormer Baselines for Vision

Weihao Yu, Chenyang Si, Pan Zhou, Mi Luo, Yichen Zhou, Jiashi Feng, Shuicheng Yan, *Fellow, IEEE,* and Xinchao Wang, *Senior Member, IEEE*

Abstract—MetaFormer, the abstracted architecture of Transformer, has been found to play a significant role in achieving competitive performance. In this paper, we further explore the capacity of MetaFormer, again, by migrating our focus away from the token mixer design: we introduce several baseline models under MetaFormer using the most basic or common mixers, and demonstrate their gratifying performance. We summarize our observations as follows:

(1) **MetaFormer ensures solid lower bound of performance**. By merely adopting identity mapping as the token mixer, the MetaFormer model, termed *IdentityFormer*, achieves >80% accuracy on ImageNet-1K.

(2) **MetaFormer works well with arbitrary token mixers.** When specifying the token mixer as even a random matrix to mix tokens, the resulting model *RandFormer* yields an accuracy of >81%, outperforming IdentityFormer. Rest assured of MetaFormer's results when new token mixers are adopted.

(3) **MetaFormer effortlessly offers state-of-the-art results.** With just conventional token mixers dated back five years ago, the models instantiated from MetaFormer already beat state of the art.

  (a) **ConvFormer outperforms ConvNeXt**. Taking the common depthwise separable convolutions as the token mixer, the model termed *ConvFormer*, which can be regarded as pure CNNs, outperforms the strong CNN model ConvNeXt.

  (b) **CAFormer sets new record on ImageNet-1K**. By simply applying depthwise separable convolutions as token mixer in the bottom stages and vanilla self-attention in the top stages, the resulting model *CAFormer* sets a new record on ImageNet-1K: it achieves an accuracy of 85.5% at 224×224 resolution, under normal supervised training without external data or distillation.

In our expedition to probe MetaFormer, we also find that a new activation, *StarReLU*, reduces 71% FLOPs of activation compared with commonly-used GELU yet achieves better performance. Specifically, StarReLU is a variant of Squared ReLU dedicated to alleviating distribution shift. We expect StarReLU to find great potential in MetaFormer-like models alongside other neural networks. Code and models are available at https://github.com/sail-sg/metaformer.

Index Terms—MetaFormer, Transformer, Neural Networks, Image Classification, Deep Learning.

1 Introduction

In recent years, Transformers [9] have demonstrated unprecedented success in various computer vision tasks [10], [11], [12], [13]. The competence of Transformers has been long attributed to its attention module. As such, many attention-based token mixers [4], [5], [14], [15], [16] have been proposed in the aim to strengthen the Vision Transformers (ViTs) [11]. Nevertheless, some work [17], [18], [19], [20], [21] found that, by replacing the attention module in Transformers with simple operators like spatial MLP [17], [22], [23] or Fourier transform [18], the resultant models still produce encouraging performance.

Along this line, the work [24] abstracts Transformer into a general architecture termed *MetaFormer*, and hypothesizes that it is MetaFormer that plays an essential role for models in achieving competitive performance. To verify this hypothesis, [24] adopts embarrassingly simple operator, pooling, to be the token mixer, and discovers that *PoolFormer* effectively outperforms the delicate ResNet/ViT/MLP-like baselines [1], [2], [4], [11], [17], [22], [25], [26], which confirms the significance of MetaFormer.

- *This work was partially performed when Weihao Yu was a research intern at Sea AI Lab.*
- *Weihao Yu, Mi Luo and Xinchao Wang are with National University of Singapore.*
  Emails: weihaoyu@u.nus.edu, xinchao@nus.edu.sg.
- *Chenyang Si, Pan Zhou, Yichen Zhou, Jiashi Feng, and Shuicheng Yan are with Sea AI Lab. Email: yansc@sea.com.*
- *Corresponding authors: Xinchao Wang and Shuicheng Yan.*

In this paper, we make further steps exploring the boundaries of MetaFormer, through, again, deliberately taking our eyes off the token mixers. Our goal is to push the limits of MetaFomer, based on which we may have a comprehensive picture of its capacity. To this end, we adopt the most basic or common token mixers, and study the performance of the resultant MetaFormer models on the large-scale ImageNet-1K image classification. Specifically, we examine the token mixers being bare operators such as identity mapping or global random mixing, and being the common techniques dated back years ago such as separable convolution [6], [7], [8] and vanilla self-attention [9], as shown in Figure 2. We summarize our key experimental results in Figure 1, alongside our main observations are as follows.

- **MetaFormer secures solid lower bound of performance**. By specifying the token mixer to be the plainest operator, identity mapping, we build a MetaFormer model termed *IdentityFormer* to probe the performance lower bound. This crude model, surprisingly, already achieves gratifying accuracy. For example, with 73M parameters and 11.5G MACs, IdentityFormer attains top-1 accuracy of 80.4% on ImageNet-1K. Results of IdentityFormer demonstrate that MetaFormer is indeed a dependable architecture that ensures a favorable performance, even when the lowest degree of token mixing is involved.

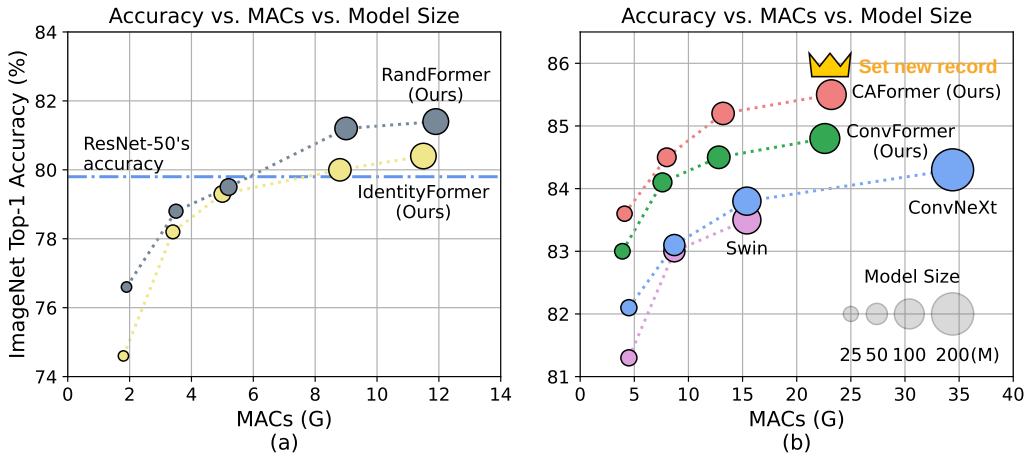

Fig. 1: **Performance of MetaFormer baselines and other state-of-the-art models on ImageNet-1K at 224^2 resolution.** The architectures of our proposed models are shown in Figure 2. (a) IdentityFormer/RandFormer achieve over 80%/81% accuracy, indicating MetaFormer has solid lower bound of performance and works well on arbitrary token mixers. The accuracy of well-trained ResNet-50 [1] is from [2]. (b) Without novel token mixers, pure CNN-based ConvFormer outperforms ConvNeXt [3], while CAFormer sets a new record of 85.5% accuracy on ImageNet-1K at 224^2 resolution under normal supervised training without external data or distillation.

- **MetaFormer works well with arbitrary token mixers.** To explore MetaFormer's universality to token mixers, we further cast the token mixer to be random, with which the message passing between tokens is enabled but largely arbitrary. Specifically, we equip the token mixers with random mixing in the top two stages and preserve the identity mapping in the bottom two stages, to avoid bringing excessive computation cost and frozen parameters. The derived model, termed *RandFormer*, turns out to be efficacious and improves IdentityFormer by 1.0%, yielding an accuracy of 81.4%. This result validates the MetaFormer's universal compatibility with token mixers. As such, please rest assured of MetaFormer's performance when exotic token mixers are introduced.
- **MetaFormer effortlessly offers state-of-the-art performance.** We make further attempts by injecting more informative operators into MetaFormer to probe its performance. Again, without introducing novel token mixers, MetaFormer models equipped with "old-fashioned" token mixers invented years ago, including inverted separable convolutions [6], [7], [8] and vanilla self-attention [9], readily delivers state-of-the-art results. Specifically,
  - **ConvFormer outperforms ConvNeXt.** By instantiating the token mixer as separable depthwise convolutions, the resultant model, termed *ConvFormer*, can be treated as a pure-CNN model without channel or spatial attention [9], [11], [27], [28]. Experiments results showcase that ConvFormer consistently outperforms the strong pure-CNN model ConvNeXt [3].
  - **CAFormer sets new record on ImageNet-1K.** If we are to introduce attention into ConvFormer by even adopting the vanilla self-attention [9], the derived model, termed *CAFormer*, readily yields record-setting performance on ImageNet-1K. Specifically, CAFormer replaces the token mixer of ConvFormer in the top two stages with vanilla self-attention, and hits a new record of 85.5% top-1 accuracy at 224^2 resolution on ImageNet-1K under the normal supervised setting (without extra data or distillation).

These MetaFormer models, with most basic or commonly-used token mixers, readily serve as dependable and competitive baselines for vision applications. When delicate token mixers or advanced training strategies are introduced, we will not be surprised at all to see the performance of MetaFormer-like models hitting new records.

Along our exploration, we also find that a new activation, *StarReLU*, largely reduces the activation FLOPs up to 71%, when compared with the commonly-adopted GELU. StarReLU is a variant of Squared ReLU, but particularly designed for alleviating distribution shifts. In our experiments, specifically, StarReLU outperforms GELU by 0.3%/0.2% accuracy on ConvFormer-S18/CAFormer-S18, respectively. We therefore expect StarReLU to find great potential in MetaFormer-like models alongside other neural networks.

2 Method

2.1 Recap the concept of MetaFormer

The concept MetaFormer [24] is a general architecture instead of a specific model, which is abstracted from Transformer [9] by not specifying token mixer. Specifically, the input is first embedded as a sequence of features (or called tokens) [9], [11]:

$$X = \text{InputEmbedding}(I). \tag{1}$$

Then the token sequence $X \in \mathbb{R}^{N \times C}$ with length N and channel dimension C is fed into repeated MetaFormer blocks, one of which can be expressed as

$$X' = X + \text{TokenMixer}\left(\text{Norm}_1(X)\right), \tag{2}$$

$$X'' = X' + \sigma\left(\text{Norm}_2(X')W_1\right)W_2, \tag{3}$$

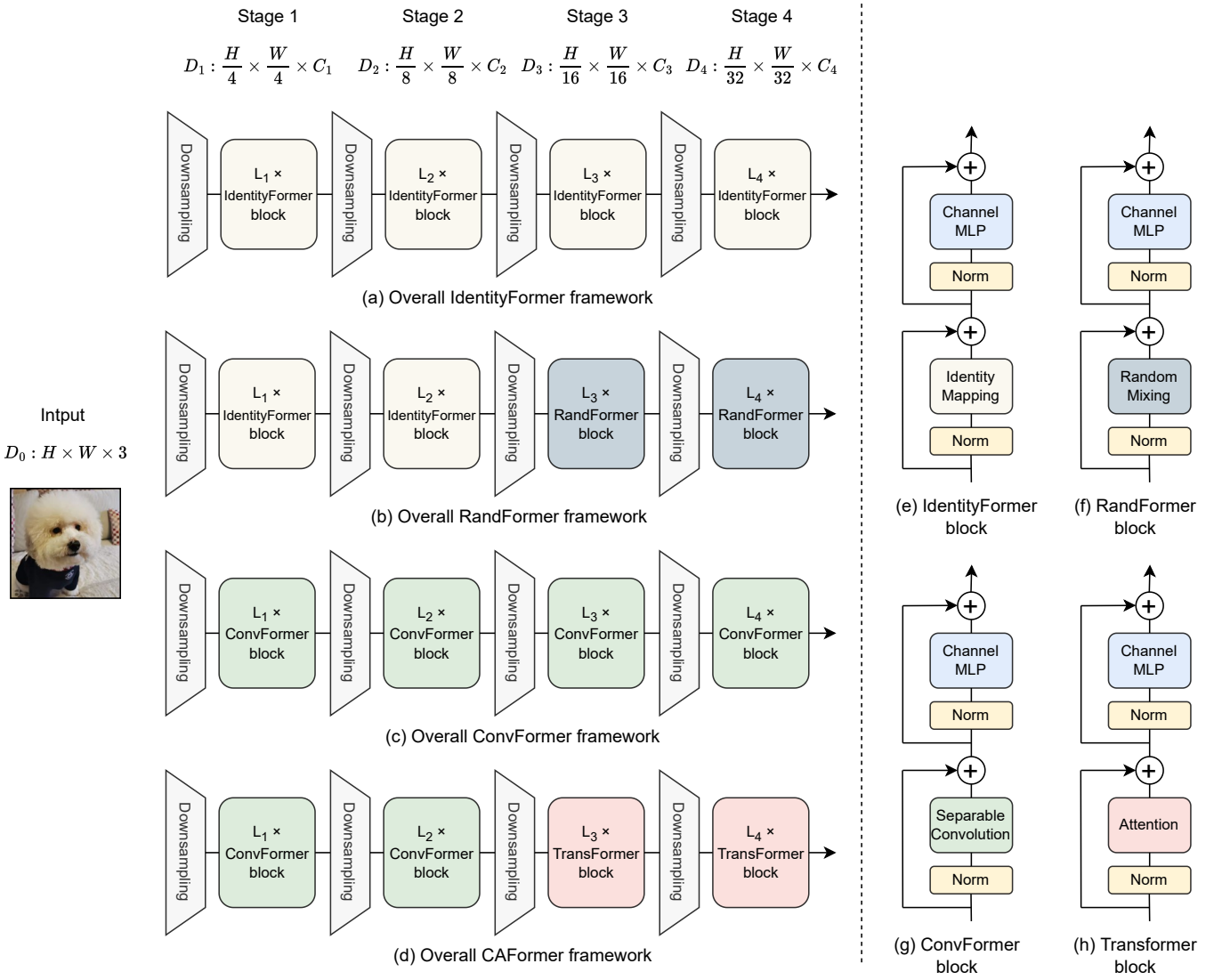

Fig. 2: **(a-d) Overall frameworks of IdentityFormer, RandFormer, ConvFormer and CAFormer.** Similar to [1], [4], [5], the models adopt hierarchical architecture of 4 stages, and stage i has L_i blocks with feature dimension D_i. Each downsampling module is implemented by a layer of convolution. The first downsampling has kernel size of 7 and stride of 4, while the last three ones have kernel size of 3 and stride of 2. **(e-h) Architectures of IdentityFormer, RandFormer, ConvFormer and Transformer blocks**, which have token mixer of identity mapping, global random mixing (Equation 5), separable depthwise convolutions [6], [7], [8] (Equation 6) or vanilla self-attention [9], respectively.

where $\mathrm{Norm}_1(\cdot)$ and $\mathrm{Norm}_2(\cdot)$ are normalizations [29], [30]; $\mathrm{TokenMixer}(\cdot)$ means token mixer mainly for propagating information among tokens; $\sigma(\cdot)$ denotes activation function; W_1 and W_2 are learnable parameters in channel MLP. By specifying token mixers as concrete modules, MetaFormer is then instantiated into specific models.

2.2 IdentityFormer and RandFormer

Following [24], we would like to instantiate token mixer as basic operators, to further probe the capacity of MetaFormer. The first one we considered is the identity mapping,

$$\mathrm{IdentityMapping}(X) = X. \tag{4}$$

Identity mapping does not conduct any token mixing, so actually, it can not be regarded as token mixer. For convenience, we still treat it as one type of token mixer to compare with other ones.

Another basic token mixer we utilize is global random mixing,

$$\mathrm{RandomMixing}(X) = XW_R, \tag{5}$$

where $X \in \mathbb{R}^{N \times C}$ is the input with sequence length N and channel dimension C, and $W_R \in \mathbb{R}^{N \times N}$ is a matrix that are frozen after random initialization. This token mixer will bring extra frozen parameters and computation cost quadratic to the token number, so it is not suitable for large token number. The PyTorch-like code of the identity mapping and random mixing are shown in Algorithm 1.

To build the overall framework, we simply follow the 4-stage model [1], [31] configurations of PoolFormer [24] to build models of different sizes. Specifically, we specify

Algorithm 1 Token mixers of identity mapping and random mixing, PyTorch-like Code

```
import torch
import torch.nn as nn

# Identity mapping
from torch.nn import Identity

# Random mixing
class RandomMixing(nn.Module):
    def __init__(self, num_tokens=196):
        super().__init__()
        self.random_matrix = nn.parameter.Parameter(
            data=torch.softmax(torch.rand(num_tokens,
                num_tokens), dim=-1),
            requires_grad=False)

    def forward(self, x):
        B, H, W, C = x.shape
        x = x.reshape(B, H*W, C)
        x = torch.einsum('mn, bnc -> bmc', self.
             random_matrix, x)
        x = x.reshape(B, H, W, C)
        return x
```

token mixer as identity mapping in all four stages and name the derived model IdentityFormer. To build RandFormer, considering that the token mixer of random mixing will bring much extra frozen parameters and computation cost for long token length, we thus remain identity mapping in the first two stages but set global random mixing as token mixer in the last two stages.

To compare IdentityFormer/RandFormer with PoolFormer [24] fairly, we also apply the techniques mentioned above to PoolFormer and name the new model PoolFormerV2. The model configurations are shown in Table 1 and the overall frameworks are shown in Figure 2.

2.3 ConvFormer and CAFormer

The above section utilizes basic token mixers to probe the lower bound of performance and model universality in terms of token mixers. In this section, without designing novel token mixers, we just specify the token mixer as commonly-used operators to probe the model potential for achieving state-of-the-art performance. The first token mixer we choose is depthwise separable convolution [6], [7]. Specifically, we follow the inverted separable convolution module in MobileNetV2 [8],

$$\text{Convolutions}(\text{X}) = \text{Conv}_{\text{pw2}}(\text{Conv}_{\text{dw}}(\sigma(\text{Conv}_{\text{pw1}}(X)))), \tag{6}$$

where $\text{Conv}_{\text{pw1}}(\cdot)$ and $\text{Conv}_{\text{pw2}}(\cdot)$ are pointwise convolutions, $\text{Conv}_{\text{dw}}(\cdot)$ is the depthwise convolution, and $\sigma(\cdot)$ means the non-linear activation function. In practice, we set the kernel size as 7 following [3] and the expansion ratio as 2. We instantiate the MetaFormer as *ConvFormer* by specifying the token mixers as the above separable convolutions. ConvFormer also adopts 4-stage framework [1], [4], [5] (Figure 2) and the model configurations of different sizes are shown in the Table 2.

Besides convolutions, another common token mixer is vanilla self-attention [9] used in Transformer. This global operator is expected to have better ability to capture long-range dependency. However, since the computational complexity of self-attention is quadratic to the number of tokens, it will be cumbersome to adopt vanilla self-attention in the first two stages that have many tokens. As a comparison, convolution is a local operator with computational complexity linear to token length. Inspired by [11], [12], [24], [32], we adopt 4-stage framework and specify token mixer as convolutions in the first two stages and attention in the last two stages to build *CAFormer*, as shown in Figure 2. See Table 2 for model configurations of different sizes.

2.4 Techniques to improve MetaFormer

This paper does not introduce complicated token mixers. Instead, we introduce a new activation StarReLU and other two modifications [33], [34], [35] to improve MetaFormer.

2.4.1 StarReLU

In vanilla Transformer [9], ReLU [36] is chosen as the activation function that can be expressed as

$$\text{ReLU}(x) = \max(0, x), \tag{7}$$

where x denotes any one neural unit of the input. This activation costs 1 FLOP for each unit. Later, GPT [37] uses GELU [38] as activation and then many subsequent Transformer models (*e.g.*, BERT [39], GPT-3 [40] and ViT [11]) employ this activation by default. GELU can be approximated as,

$$\text{GELU}(x) = x\Phi(x) \tag{8}$$

$$\approx 0.5 \times x(1 + \tanh(\sqrt{2/\pi}(x + 0.044715 \times x^3))), \tag{9}$$

where $\Phi(\cdot)$ is the Cumulative Distribution Function for Gaussian Distribution (CDFGD). Although revealing better performance than ReLU [24], [41], GELU approximately brings 14 FLOPs [1], much larger then ReLU's 1 FLOP of cost. To simplify GELU, [41] finds that CDFGD can be replaced by ReLU,

$$\text{SquaredReLU}(x) = x\text{ReLU}(x) = (\text{ReLU}(x))^2. \tag{10}$$

This activation is called Squared ReLU [41], only costing 2 FLOPs for each input unit. Despite the simplicity of Squared ReLU, we find its performance can not match that of GELU for some models on image classification task as shown in Section 3.4. We hypothesize the worse performance may be resulted from the distribution shift of the output [43]. Assuming input x follows normal distribution with mean 0 and variance 1, *i.e.*$x \sim N(0, 1)$, we have:

$$\text{E}\left((\text{ReLU}(x))^2\right) = 0.5, \quad \text{Var}\left((\text{ReLU}(x))^2\right) = 1.25. \tag{11}$$

See the appendix for the derivation process of Equation 11. Therefore the distribution shift can be solved by

$$\text{StarReLU}(x) = \frac{(\text{ReLU}(x))^2 - \text{E}\left((\text{ReLU}(x))^2\right)}{\sqrt{\text{Var}\left((\text{ReLU}(x))^2\right)}} \tag{12}$$

$$= \frac{(\text{ReLU}(x))^2 - 0.5}{\sqrt{1.25}} \tag{13}$$

$$\approx 0.8944 \cdot (\text{ReLU}(x))^2 - 0.4472. \tag{14}$$

We name the above activation *StarReLU* as multiplications (*) is heavily used. However, the assumption of standard normal distribution for input is strong [43]. To make the

1. tanh is counted 6 FLOPs for simplicity [42].

TABLE 1: **Model configurations of IdentityFormer, RandFormer and PoolFormerV2.** "C", "L" and "T" means channel number, block number and token mixer type, respectively. "Id", "Rand" and "Pool" denotes token mixer of identity mapping, random mixing and pooling, respectively. The contents in the tuples represent the configurations in the four stages of the models.

Model		IdentityFormer	RandFormer	PoolFormerV2
Size	S12	$C = (64, 128, 320, 512)$,	$L = (2, 2, 6, 2)$	
	S24	$C = (64, 128, 320, 512)$,	$L = (4, 4, 12, 4)$	
	S36	$C = (64, 128, 320, 512)$,	$L = (6, 6, 18, 6)$	
	M36	$C = (96, 192, 384, 768)$,	$L = (6, 6, 18, 6)$	
	M48	$C = (96, 192, 384, 768)$,	$L = (8, 8, 24, 8)$	
Token Mixer		$T = (\text{Id}, \text{Id}, \text{Id}, \text{Id})$	$T = (\text{Id}, \text{Id}, \text{Rand}, \text{Rand})$	$T = (\text{Pool}, \text{Pool}, \text{Pool}, \text{Pool})$
Classifier Head		Global average pooling, Norm, FC		

TABLE 2: **Model configurations of ConvFormer and CAFormer.** "C", "L" and "T" means channel number, block number and token mixer type. "Conv" and "Attn" denotes token mixer of separable convolution and vanilla self-attention, respectively. The contents in the tuples represent the configurations in the four stages of the models.

Model		ConvFormer	CAFormer
Size	S18	$C = (64, 128, 320, 512)$,	$L = (3, 3, 9, 3)$
	S36	$C = (64, 128, 320, 512)$,	$L = (3, 12, 18, 3)$
	M36	$C = (96, 192, 384, 576)$,	$L = (3, 12, 18, 3)$
	B36	$C = (128, 256, 512, 768)$,	$L = (3, 12, 18, 3)$
Token Mixer		$T = (\text{Conv}, \text{Conv}, \text{Conv}, \text{Conv})$	$T = (\text{Conv}, \text{Conv}, \text{Attn}, \text{Attn})$
Classifier Head		Global average pooling, Norm, MLP	

activation adaptable to different situations, like different models or initialization, scale and bias can be set to be learnable [44], [45]. We uniformly re-write the activation as

$$\text{StarReLU}(x) = s \cdot (\text{ReLU}(x))^2 + b, \tag{15}$$

where $s \in \mathbb{R}$ and $b \in \mathbb{R}$ are scalars of scale and bias respectively, which are shared for all channels and can be set to be constant or learnable to attain different StarReLU variants. StarReLU only costs 4 FLOPs (or 3 FLOPs with only s or b), much fewer than GELU's 14 FLOPs but achieving better performance as shown in Section 3.4. For convenience, *we utilize StarReLU with learnable scale and bias as default activation in this paper* as intuitively this variant can more widely adapt to different situations [44], [45]. We leave the study of StarReLU variant selection for different situations in the future.

2.4.2 Other modifications

Scaling branch output. To scale up Transformer model size from depth, [46] proposes *LayerScale* that multiplies layer output by a learnable vector:

$$X' = X + \lambda_l \odot \mathcal{F}(\text{Norm}(X)), \tag{16}$$

where $X \in \mathbb{R}^{N \times C}$ denotes the input features with sequence length N and channel dimension C, $\text{Norm}(\cdot)$ is the normalization, $\mathcal{F}(\cdot)$ means the token mixer or channel MLP module, $\lambda_l \in \mathbb{R}^C$ represents the learnable LayerScale parameters initialized by a small value like 1e-5, and $\odot$ means element multiplication. Similar to LayerScale, [33], [47], [48] attempt to stabilize architectures by scaling the residual branch (*ResScale* [33]):

$$X' = \lambda_r \odot X + \mathcal{F}(\text{Norm}(X)), \tag{17}$$

where $\lambda_r \in \mathbb{R}^C$ denotes learnable parameters initialized as 1. Apparently, we can merge the above two techniques into *BranchScale* by scaling all branches:

$$X' = \lambda_r \odot X + \lambda_l \odot \mathcal{F}(\text{Norm}(X)). \tag{18}$$

Among these three scaling techniques, we find ResScale performs best according to our experiments in Section 3.4. Thus, *we adopt ResScale [33] by default in this paper.*

Disabling biases. Following [34], [35], we disable the biases of fully-connected layers, convolutions (if have) and normalization in the MetaFormer blocks, finding it does not hurt performance and even can bring slight improvement for specific models as shown in the ablation study. For simplicity, *we disable biases in MetaFormer blocks by default.*

3 Experiments

3.1 Image Classification

3.1.1 Setup

ImageNet-1K [59] image classification is utilized to benchmark these baseline models. ImageNet-1K is one of the most widely-used datasets in computer vision which contains about 1.3M images of 1K classes on training set, and 50K images on validation set. For ConvFormer-B36 and CAFormer-B36, we also conduct pre-training on ImageNet-21K [59], [60], a much larger dataset containing ∼14M images of 21841 classes, and then fine-tune the pretrained model on ImageNet-1K for evaluation. Our implementation is based on PyTorch library [61] and `Timm` codebase [62] and the experiments are run on TPUs.

Training and fine-tuning on ImageNet-1K. We mainly follow the hyper-parameters of DeiT [25]. Specifically, models

TABLE 3: **Performance on ImageNet-1K of RSB-ResNet and MetaFormer models with basic tokens of identity mapping, random maxing and pooling.** The underlined numbers mean the numbers of parameters that are frozen after random initialization.

Model	Params (M)	MACs (G)	Top-1 (%)
RSB-ResNet-18 [1], [2]	11.7	1.8	70.6
IdentityFormer-S12	11.9	1.8	74.6
RandFormer-S12	11.9 + 0.2	1.9	76.6
PoolFormerV2-S12 [24]	11.9	1.8	78.0
RSB-ResNet-34 [1], [2]	21.8	3.7	75.5
IdentityFormer-S24	21.3	3.4	78.2
RandFormer-S24	21.3 + 0.5	3.5	78.8
PoolFormerV2-S24 [24]	21.3	3.4	80.7
RSB-ResNet-50 [1], [2]	25.6	4.1	79.8
IdentityFormer-S36	30.8	5.0	79.3
RandFormer-S36	30.8 + 0.7	5.2	79.5
PoolFormerV2-S36 [24]	30.8	5.0	81.6
RSB-ResNet-101 [1], [2]	44.5	7.9	81.3
IdentityFormer-M36	56.1	8.8	80.0
RandFormer-M36	56.1 + 0.7	9.0	81.2
PoolFormerV2-M36 [24]	56.1	8.8	82.2
RSB-ResNet-152 [1], [2]	60.2	11.6	81.8
IdentityFormer-M48	73.3	11.5	80.4
RandFormer-M48	73.3 + 0.9	11.9	81.4
PoolFormerV2-M48 [24]	73.3	11.5	82.6

TABLE 4: **Comparison among ViT (DeiT), isotropic IdentityFormer and isotropic IdentityFormer with stem of 4 convolutional layers** with stride of 2 and kernel size of 7^2, 3^2, 3^2, and 3^2 respectively.

Model	Params (M)	MACs (G)	Top-1 (%)
DeiT-S [25]	22	4.6	79.8
IdentityFormer-S (*iso.*)	22	4.2	68.2
IdentityFormer-S (*iso.*, conv stem)	23	4.6	75.4

are trained for 300 epochs at 224^2 resolution. Data augmentation and regularization techniques include RandAugment [63], Mixup [64], CutMix [65], Random Erasing [66], weight decay, Label Smoothing [67] and Stochastic Depth [68]. We do not use repeated augmentation [69], [70] and LayerScale [46], but use ResScale [33] for the last two stages. We adopt AdamW [71], [72] optimizer with batch size of 4096 for most models except CAFormer since we found it suffers a slight performance drop compared with that with batch size of 1024. The problem may be caused by the large batch size, so we use a large-batch-size-friendly optimizer LAMB [73] for CAFormer. For 384^2 resolution, we fine-tune the models trained at 224^2 resolution for 30 epochs with Exponential Moving Average (EMA) [74]. The details of hyper-parameters are shown in the appendix.

Pre-training on ImageNet-21K and fine-tuning on ImageNet-1K. To probe the scaling capacity with a larger dataset, we pre-train ConvFormer and CAFormer on ImageNet-21K for 90 epochs at the resolution of 224^2. Then the pre-trained models are fine-tuned on ImageNet-1K at the resolution of 224^2 and 384^2 for 30 epochs with EMA [74]. See the appendix for more details of hyper-parameters.

Robustness evaluation. Following ConvNeXt [3], we also directly evaluate our ImageNet models on several robustness benchmarks, *i.e.* ImageNet-C [75], ImageNet-A [76], ImageNet-R [77] and ImageNet-Sketch [78]. Note that we do not adopt additional fine-tuning or any specialized modules. Mean corruption error (mCE) is reported for ImageNet-C and top-1 accuracy is for all other datasets.

3.1.2 Results of Models with basic token mixers

Table 3 shows the performance of models with basic token mixers on ImageNet-1K. Surprisingly, with bare identity mapping as token mixer, IdentityFormer already performs very well, especially for small model sizes. For example, IdentityFormer-S12/S24 outperforms RSB-ResNet-18/34 [1], [2] by 4.0%/2.7%, respectively. We further scale up the model size of IdentityFormer to see what accuracy it can achieve. By scaling up model size to ∼73M parameters and ∼12G MACs, IdentityFormer-M48 can achieve accuracy of 80.4%. Without considering the comparability of model size, this accuracy already surpasses 79.8% of RSB-ResNet-50. The results of IdentityFormer indicate that MetaFormer ensures solid lower bound of performance. That is to say, if you adopt MetaFormer as general framework to develop your own models, the accuracy will not be below 80% with similar parameter numbers and MACs of IdentityFormer-M48.

To see whether the amazing performance of IdentityFormer is from hierarchical structure, we follow ViT-S (DeiT-S) to build isotropic IdentityFormer, and the results are shown in Table 4. IdentityFormer-S (*iso.*) can achieve 68.2% accuracy and IdentityFormer-S (*iso.*) with the stem of 4 convolutional layers can even obtain an accuracy of 75.4%. These results show that isotropic IdentityFormer also works well, demonstrating the performance of IdentityFormer is not from hierarchical structure. An important factor for model performance is the receptive field of the stem or downsampling layers, based on the large improvement of IdentityFormer-S (*iso.*, conv stem) over IdentityFormer-S (*iso.*).

Another surprising finding is that by replacing token mixer of IdentityFormer with random mixing in the top two stages, RandFormer can consistently improve IdentityFormer. For example, RandFormer-S12/M48 obtains accuracy of 76.6%/81.4%, surpassing IdentityFormer-S12/M48 by 2.0%/1.0%, respectively. For medium and large model sizes, RandFormer can also achieve accuracy comparable to RSB-ResNet, like RandFormer-M36's 81.2% *vs.*RSB-ResNet-101's 81.3%. The promising performance of RandFormer, especially its consistent improvement over IdentityFormer, demonstrates MetaFormer can work well with arbitrary token mixers and validates MetaFormer's universal compatibility with token mixers. Therefore, rest assured of MetaFormer's performance when exotic token mixers are equipped.

Compared with PoolFormerV2 [24] with basic token mixer of pooling, neither of IdentityFormer nor RandFormer can match its performance. The worse performance of IdentityFormer makes sense as identity mapping does not conduct any token mixing. The performance gap between

TABLE 5: **Performance of models trained on ImageNet-1K at the resolution of 224^2 and finetuned at 384^2.** Model highlighted with gray background are proposed in this paper. The column "MetaFormer" denotes whether models adopt MetaFormer architecture (partially). * To the best of our knowledge, the model **sets a new record on ImageNet-1K** with the accuracy of 85.5% at 224^2 resolution under normal supervised setting (without external data or distillation), surpassing the previous best record of 85.3% set by MViTv2-L [49] with 55% fewer parameters and 45% fewer MACs.

Model	MetaFormer	Mixing Type	Params (M)	Testing at resolution			
				@224		↑384	
				MACs (G)	Top-1 (%)	MACs (G)	Top-1 (%)
RSB-ResNet-50 [1], [2]	✗	Conv	26	4.1	79.8	-	-
RegNetY-4G [2], [50]	✗	Conv	21	4.0	81.3	-	-
ConvNeXt-T [3]	✗	Conv	29	4.5	82.1	-	-
VAN-B2 [51]	✓	Conv	27	5.0	82.8	-	-
ConvFormer-S18	✓	Conv	27	3.9	**83.0**	11.6	84.4
DeiT-S [25]	✓	Attn	22	4.6	79.8	-	-
T2T-ViT-14 [14]	✓	Attn	22	4.8	81.5	17.1	83.3
Swin-T [5]	✓	Attn	29	4.5	81.3	-	-
CSWin-T [52]	✓	Attn	23	4.3	82.7	-	-
MViTv2-T [49]	✓	Attn	24	4.7	82.3	-	-
Dual-ViT-S [53]	✓	Attn	25	4.8	83.4	-	-
CoAtNet-0 [32]	✓	Conv + Attn	25	4.2	81.6	13.4	83.9
UniFormer-S [54]	✓	Conv + Attn	22	3.6	82.9	-	-
iFormer-S [55]	✓	Conv + Attn	20	4.8	83.4	16.1	84.6
CAFormer-S18	✓	Conv + Attn	26	4.1	**83.6**	13.4	85.0
RSB-ResNet-101 [1], [2]	✗	Conv	45	7.9	81.3	-	-
RegNetY-8G [2], [50]	✗	Conv	39	8.0	82.1	-	-
ConvNeXt-S [3]	✗	Conv	50	8.7	83.1	-	-
VAN-B3 [51]	✓	Conv	45	9.0	83.9	-	-
ConvFormer-S36	✓	Conv	40	7.6	**84.1**	22.4	85.4
T2T-ViT-19 [14]	✓	Attn	39	8.5	81.9	-	-
Swin-S [5]	✓	Attn	50	8.7	83.0	-	-
CSWin-S [52]	✓	Attn	35	6.9	83.6	-	-
MViTv2-S [49]	✓	Attn	35	7.0	83.6	-	-
CoAtNet-1 [32]	✓	Conv + Attn	42	8.4	83.3	27.4	85.1
UniFormer-B [54]	✓	Conv + Attn	50	8.3	83.9	-	-
CAFormer-S36	✓	Conv + Attn	39	8.0	**84.5**	26.0	85.7
RSB-ResNet-152 [1], [2]	✗	Conv	60	11.6	81.8	-	-
RegNetY-16G [2], [50]	✗	Conv	84	15.9	82.2	-	-
ConvNeXt-B [3]	✗	Conv	89	15.4	83.8	45.0	85.1
RepLKNet-31B [56]	✓	Conv	79	15.3	83.5	45.1	84.8
VAN-B4 [51]	✓	Conv	60	12.2	84.2	-	-
SLaK-B [57]	✓	Conv	95	17.1	84.0	50.3	85.5
ConvFormer-M36	✓	Conv	57	12.8	**84.5**	37.7	85.6
DeiT-B [25]	✓	Attn	86	17.5	81.8	55.4	83.1
T2T-ViT-24 [14]	✓	Attn	64	13.8	82.3	-	-
Swin-B [5]	✓	Attn	88	15.4	83.5	47.1	84.5
CSwin-B [52]	✓	Attn	78	15.0	84.2	47.0	85.4
MViTv2-B [49]	✓	Attn	52	10.2	84.4	36.7	85.6
CoAtNet-2 [32]	✓	Conv + Attn	75	15.7	84.1	49.8	85.7
MaxViT-S [58]	✓	Conv + Attn	69	11.7	84.5	36.1	85.2
iFormer-L [55]	✓	Conv + Attn	87	14.0	84.8	45.3	85.8
CAFormer-M36	✓	Conv + Attn	56	13.2	**85.2**	42.0	86.2
RegNetY-32G [2], [50]	✗	Conv	145	32.3	82.4	-	-
ConvNeXt-L [3]	✗	Conv	198	34.4	84.3	101.0	85.5
ConvFormer-B36	✓	Conv	100	22.6	**84.8**	66.5	85.7
MViTv2-L [49]	✓	Attn	218	42.1	85.3	140.2	86.3
CoAtNet-3 [32]	✓	Conv + Attn	168	34.7	84.5	107.4	85.8
MaxViT-B [58]	✓	Conv + Attn	120	23.4	85.0	74.2	86.3
CAFormer-B36	✓	Conv + Attn	99	23.2	**85.5***	72.2	86.4

RandFormer and PoolFormerV2 may result from the local inductive bias of pooling.

3.1.3 Results of models with commonly-used token mixers

We build ConvFormer by specifying the token mixer in MetaFormer as separable convolutions [6], [7] used in MobileNetV2 [8]. Meanwhile, CAFormer is built with token mixers of separable convolutions in the bottom two stages and vanilla self-attention in the top two stages. The results of models trained on ImageNet-1K are shown in Table 5.

ConvFormer actually can be regarded as pure CNN-based model without any attention mechanism [9], [11], [27], [28]. It can be observed that ConvFormer outperforms strong CNN model ConvNeXt [3] significantly. For example, at the resolution of 224^2, ConvFomer-B36 (100M parameters and 22.6G MACs) surpasses ConvNeXt-B (198M parameters and 34.4G MACs) by 0.5% top-1 accuracy while only requiring 51% parameters and 66% MACs. Compared with another stong CNN model EfficientNetV2-L [79] with input size of 480^2 (120M parameters, 53.0G MACs, 85.7% top-1 accuracy), ConvFormer-B36 with input size of 384^2 can match its accuracy.

Also, ConvFormer outperforms various strong attention-based or hybrid models. For instance, ConvFormer-M36 outperforms Swin-B [5]/CoAtNet-2 [32] by 1.0%/0.4% with 35%/24% fewer parameters and 17%/18% fewer MACs.

Besides ConvFormer, CAFormer achieves more remarkable performance. Although CAFormer is just built by equipping token mixers of separable convolutions [6], [7], [8] in bottom stages and vanilla self-attention [9] in top stages, it already consistently outperforms other models in different sizes, as clearly shown in Figure 3. Remarkably, to the best of our knowledge, CAFormer **sets new record on ImageNet-1K** with top-1 accuracy of 85.5% at 224^2 resolution under normal supervised setting (without external data or distillation models).

When pre-trained on ImageNet-21K (Table 7), the model performance is further improved. For instance, the performance of ConvFormer-B36 and CAFormer-B36 surges to 87.0% and 87.4%, with 2.2% and 1.9% accuracy improvement compared with the results of ImageNet-1K training only. Both models keep superior to Swin-L [5]/ConvNeXt-L [3], showing the promising scaling capacity with a larger pre-training dataset. For example, ConvFormer-B36 outperforms ConvNeXt-L by 0.2% with 49% fewer parameters and 34% fewer MACs. Compared with another strong model EfficientNetV2-XL (input size of 480^2, 94.0G MACs, 208M parameters, 87.3% top-1 accuracy), ConvFormer-B36 and CAForemer-B36 surpass it by 0.3% and 0.8% with only half of the parameters and 71%/77% MACs, respectively.

Just equipped with "old-fashioned" token mixers, ConvFormer and CAFormer instantiated from MetaFormer already can achieve remarkable performance, especially CAFormer sets a new record on ImageNet-1K. These results demonstrate MetaFormer can offer high potential for achieving state-of-the-art performance. When advanced token mixers or training strategies are introduced, we will not be surprised to see the performance of MetaFormer-like models setting new records. We expect ConvFormer and CAFormer as well as IdentityFormer and RandFormer to be dependable baselines for future neural architecture design.

3.1.4 Robustness of models with commonly-used token mixers

The robustness results of ConvFormer, CAFormer and other SOTA models are shown in Table 6. Compared with Swin [5] and ConvNeXt [3], ConvFormer exhibits better or competitive performance. For example, for models trained on ImageNet-1K, ConvFormer-S18 obtains 25.3% and 48.7% on ImageNet-A [76] and ImageNet-R [77], outperforming ConvNeXt-T by 1.1% and 1.5%, respectively. CAFormer attains more impressive performance: It not only consistently outperforms Swin and ConvNeXt, but also surpasses SOTA robust method FAN [80]. For instance, for models trained on ImageNet-1K, CAFormer-M36/CAFormer-B36 obtain 45.6% and 48.5% on ImageNet-A [76], surpassing FAN [80] by 8.4%/11.3%, respectively.

3.2 Object detection and instance segmentation

3.2.1 Setup

We evaluate ConvFormer and CAFormer on COCO [83] which contains 118K training images and 5K validation images. Following Swin [5] and ConvNeXt [3], we adopt ConvFormer and CAFormer pretrained on ImageNet-1K as the backbones for Mask R-CNN [84] and Cascade Mask R-CNN [85]. Due to the large image resolution, we find adopting CAFormer will result in out-of-memory. To solve it, we limit attention of CAFormer in sliding window [86], [87]. We also adopt AdamW multi-scale and $3\times$ schedule training setting, following Swin and ConvNeXt.

3.2.2 Results

Table 9 shows the results of ConvFormer, CAFormer, and other two strong backbones Swin and ConvNeXt for COCO object detection and instance segmentation. The models with backbones of ConvFormer and CAFormer consistently outperform Swin and ConvNeXt. For example, Cascade Mask R-CNN with CAFormer-S18 as backbone largely surpasses that with Swin-T/ConvNeXt-T, *i.e.*, 52.3 *vs*.50.4/50.4 for box AP, and 45.2 *vs*.43.7/43.7 for mask AP.

3.3 Semantic segmentation

3.3.1 Setup

We also evaluate ConvFormer and CAFormer on the ADE20K [88] for semantic segmentation task. ADE20K consists of 20K/2K images on the training/validation set, including 150 semantic categories. Following Swin and ConvNeXt, we equip ConvFormer and CAFormer as backbones for UperNet [89]. All models are trained with AdamW optimizer [71], [72] and batch size of 16 for 160K iterations.

3.3.2 Results

Table 10 shows the results of UperNet with different backbones. ConvFormer and CAFormer as backbones obtain better performance compared with Swin and ConvNeXt. For instance, the model with CAFormer-S18 obtains 48.9 mIoU, higher than those with Swin-T/ConvNeXt-T by 3.1/2.2.

TABLE 6: **Model evaluation of robustness.** Model highlighted with gray background are proposed in this paper. Note that we do not adopt additional fine-tuning or any specialized modules.

Model	Params (M)	Testing at resolution											
		@224						↑384					
		MACs (G)	Clean (%)	C (↓) (mCE)	A (%)	R (%)	SK (%)	MACs (G)	Clean (%)	C (↓) (mCE)	A (%)	R (%)	SK (%)
Trained on ImageNet-1K													
Swin-T [5]	29	4.5	81.3	62.0	21.6	41.3	29.1	-	-	-	-	-	-
RVT-S* [81]	23	4.7	81.9	49.4	25.7	47.7	34.7	-	-	-	-	-	-
ConvNeXt-T [3]	29	4.5	82.1	53.2	24.2	47.2	33.8	-	-	-	-	-	-
FAN-S [80]	28	5.3	82.5	47.7	29.1	50.4	-	-	-	-	-	-	-
ConvFormer-S18	27	3.9	83.0	51.7	25.3	48.7	35.2	11.6	84.4	51.0	42.0	50.7	36.2
CAFormer-S18	26	4.1	83.6	47.4	33.5	48.7	36.6	13.4	85.0	46.1	48.9	51.3	37.7
Swin-S [5]	50	8.7	83.0	52.7	32.3	45.1	32.4	-	-	-	-	-	-
ConvNeXt-S [3]	50	8.7	83.1	51.2	31.2	49.5	37.1	-	-	-	-	-	-
FAN-B [80]	54	10.4	83.6	44.4	35.4	51.8	-	-	-	-	-	-	-
ConvFormer-S36	40	7.6	84.1	47.1	33.2	50.8	38.4	22.4	85.4	47.7	49.9	51.9	37.8
CAFormer-S36	39	8.0	84.5	44.7	40.9	51.7	39.5	26.0	85.7	42.7	57.1	54.5	41.7
Swin-B [5]	88	15.4	83.5	54.4	35.8	46.6	32.4	47.1	84.5	49.4	45.3	47.0	32.9
RVT-B* [81]	92	17.7	82.6	46.8	28.5	48.7	36.0	-	-	-	-	-	-
ConvNeXt-B [3]	89	15.4	83.8	46.8	36.7	51.3	38.2	45.0	85.1	48.6	47.6	52.2	38.5
FAN-L [80]	81	15.8	83.9	43.3	37.2	**53.1**	-	-	-	-	-	-	-
Robust-ResNet [82]	-	17.3	81.6	**34.9**	-	51.1	38.1	-	-	-	-	-	-
ConvFormer-M36	57	12.8	84.5	46.5	37.6	51.0	39.2	37.7	85.6	48.4	53.5	52.2	38.5
CAFormer-M36	56	13.2	85.2	42.6	45.6	51.7	39.6	42.0	86.2	**41.7**	60.2	**55.0**	41.5
ConvNeXt-L [3]	198	34.4	84.3	46.6	41.1	53.4	40.1	101.0	85.5	46.8	52.5	53.6	39.9
ConvFormer-B36	100	22.6	84.8	46.3	40.1	51.1	39.5	66.5	85.7	48.1	55.3	52.2	38.9
CAFormer-B36	99	23.2	**85.5**	42.6	**48.5**	**53.9**	**42.5**	72.2	**86.4**	42.8	**61.9**	**55.0**	**42.5**
Pretrained on ImageNet-21K													
ConvNeXt-T [3]	29	4.5	82.9	52.3	36.6	51.0	38.5	13.1	84.1	51.5	45.8	51.3	38.9
ConvFormer-S18	27	3.9	83.7	47.5	33.4	53.4	40.3	11.6	85.0	47.2	50.1	55.0	41.6
CAFormer-S18	26	4.1	84.1	44.8	43.3	54.1	41.2	13.4	85.4	43.3	58.3	55.9	42.0
ConvNeXt-S [3]	50	8.7	84.6	45.6	45.1	57.3	43.6	25.5	85.8	44.2	57.0	59.1	45.8
ConvFormer-S36	40	7.6	85.4	41.0	47.3	58.9	46.9	22.4	86.4	41.3	62.9	59.9	47.1
CAFormer-S36	39	8.0	85.8	38.5	55.5	60.7	46.7	26.0	86.9	36.8	70.6	63	48.5
Swin-B [5]	88	15.4	85.2	42.0	51.7	59.3	45.5	47.1	86.4	37.8	65.3	63.0	48.5
ConvNeXt-B [3]	89	15.4	85.8	41.9	54.8	61.8	49.8	45.0	86.8	43.1	62.3	64.9	51.6
ConvFormer-M36	57	12.8	86.1	38.4	56.1	60.9	49.1	37.7	86.9	39.0	68.5	61.8	49.1
CAFormer-M36	56	13.2	86.6	35.2	60.9	63.4	49.7	42.0	87.5	33.9	73.9	65.3	51.0
Swin-L [5]	197	34.5	86.3	38.0	61.2	63.7	49.0	103.9	87.3	34.5	70.7	66.0	50.4
ConvNeXt-L [3]	198	34.4	86.8	38.3	60.5	63.9	49.9	101.0	87.5	40.2	65.5	66.7	52.8
ConvFormer-B36	100	22.6	87.0	35.0	63.3	65.3	52.7	66.5	87.6	35.8	73.5	66.5	52.9
CAFormer-B36	99	23.2	**87.4**	**31.8**	**69.4**	**68.3**	**52.8**	72.2	**88.1**	**30.8**	**79.5**	**70.4**	**54.5**

3.4 Ablation

This paper does not design novel token mixers but utilizes three techniques to MetaFormer. Therefore, we conduct ablation study for them, respectively. ConvFormer-S18 and CAFormer-S18 on ImageNet-1K are taken as the baselines. The results are shown in Table 8. When the StarReLU is replaced with ReLU [36], the performance of ConvFormer-S18/CAFormer significantly drops from 83.0%/83.6% to 82.1%/82.9%, respectively. When the activation is Squared ReLU [41], the performance is already satisfied. But for ConvFormer-18, it can not match that of GELU [38]. As for the StarReLU, it not only can reduce 71% activation FLOPs compared with GELU, but also achieves better performance with 0.3%/0.2% accuracy improvement for ConvFormer-S18/CAForemr-S18, respectively. This result expresses the promising application potential of StarReLU in MetaFormer-like models and other neural networks. We further observe the performance of different StarReLU variants on ConvFormer-S18. We adopt StarReLU with learnable scale and bias by default because it does not need to meet the assumption of standard normal distribution for input [43] and can be conveniently applied for different models and initialization [44], [45]. But for specific model ConvFormer-18, StarReLU with learnable or frozen bias is enough since it can already match the accuracy of the default StarReLU version. We leave the study of StarReLU variant selection in the future.

For other techniques, we find ResScale [33] performs best

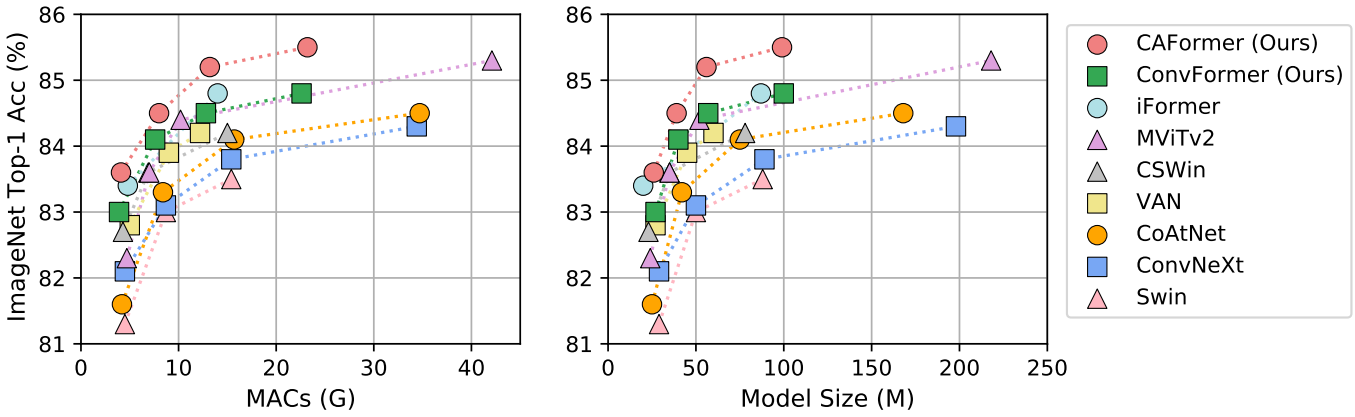

Fig. 3: **ImageNet-1K validation accuracy *vs*.MACs/Model Size at the resolution of 224^2.** Models with token (feature) mixing based on convolution, attention or hybrid are presented by □, △ or ○ respectively.

TABLE 7: **Performance of models pre-trained on ImageNet-21K and fine-tuned on ImageNet-1K for evaluation.** Model highlighted with gray background are proposed in this paper.

Model	MetaFormer	Mixing Type	Params (M)	Testing at resolution			
				@224		↑384	
				MACs (G)	Top-1 (%)	MACs (G)	Top-1 (%)
ConvNeXt-T [3]	✗	Conv	29	4.5	82.9	13.1	84.1
ConvFormer-S18	✓	Conv	27	3.9	**83.7**	11.6	85.0
CAFormer-S18	✓	Conv + Attn	26	4.1	**84.1**	13.4	85.4
ConvNeXt-S [3]	✗	Conv	50	8.7	84.6	25.5	85.8
ConvFormer-S36	✓	Conv	40	7.6	**85.4**	22.4	86.4
CAFormer-S36	✓	Conv + Attn	39	8.0	**85.8**	26.0	86.9
ConvNeXt-B [3]	✗	Conv	89	15.4	85.8	45.1	86.8
ConvFormer-M36	✓	Conv	57	12.8	**86.1**	37.7	86.9
Swin-B [5]	✓	Attn	88	15.4	85.2	47.1	86.4
CAFormer-M36	✓	Conv + Attn	56	13.2	**86.6**	42.0	87.5
ConvNeXt-L [3]	✗	Conv	198	34.4	86.8	101.0	87.5
ConvFormer-B36	✓	Conv	100	22.6	**87.0**	66.5	87.6
Swin-L [5]	✓	Attn	197	34.5	86.3	103.9	87.3
CAFormer-B36	✓	Conv + Attn	99	23.2	**87.4**	72.2	88.1

among the branch output scaling techniques; disabling biases [34], [35] in each block does not affect the performance for ConvFormer-S18 and can bring improvement of 0.1% for CAFormer-S18. We thus employ ResScale and disabling biases of each block by default.

3.5 Benchmark speed

We first benchmark the speed of the proposed StarReLU and the commonly-used GELU [38] on NVIDIA A100 GPU that is shown in Table 11. It can be seen that compared with GELU (Equation 9), StarReLU enjoys significant speedup, with 1.7× speedup on NVIDIA A100 GPU. We also note that StarReLU is slower than GELU (PyTorch API) because the current implementation of StarReLU is not CUDA-optimized. Once optimized, we expect a further speedup for StarReLU, likely a significant one.

Then we further benchmark the ConvFormer, CAFormer, and other strong models (Swin [5] and ConvNeXt [3]). For fair comparison, we replace StarReLU in ConvFormer and CAFormer with GELU. The results are shown in Table 12. We can see that for similar model sizes and MACs, ConvNeXt has the highest throughput. This is because ConvNeXt block has only one residual connection while MetaFormer block has two. However, ConvFormer and CAFormer obtain higher accuracy among these models and also achieve relatively higher throughputs than Swin and MaxViT, resulting in better trade-off between accuracy and throughput, as shown in Figure 4.

4 Related Work

Transformer, since being introduced in [9], has become a popular backbone for various tasks in NLP [34], [37], [40], [90], [91], computer vision [4], [5], [10], [11], [14], [25] and other domains [92], [93], [94], [95], [96], [97], [98]. In computer vision, iGPT [10] and ViT [11] introduce pure Transformer for self-supervised learning and supervised learning, attracting great attention in the research community to further improve Transformers. The success of Transformers had long been attributed to the attention module, and thus many research endeavors have been focused on improving

TABLE 8: **Ablation for ConvFormer-S18/CAFormer-S18 on ImageNet-1K.** * α and β denote learnable scalars shared for all channels.

Ablation	Variant	Top-1 (%) ConvFormer-S18	Top-1 (%) CAFormer-S18
–	Baseline	83.0	83.6
Activation types	StarReLU → ReLU [36]	82.1 (-0.9)	82.9 (-0.7)
	StarReLU → Squared ReLU [41]	82.6 (-0.4)	83.4 (-0.2)
	StarReLU → GELU [38]	82.7 (-0.3)	83.4 (-0.2)
StarReLU variants	$\alpha \cdot (\mathrm{ReLU}(x))^2 + \beta$*		
	$\rightarrow \alpha \cdot (\mathrm{ReLU}(x))^2$	82.6 (-0.4)	83.6 (-0.0)
	$\rightarrow (\mathrm{ReLU}(x))^2 + \beta$	83.0 (-0.0)	83.5 (-0.1)
	$\rightarrow 1/\sqrt{1.25} \cdot (\mathrm{ReLU}(x))^2 - 0.5/\sqrt{1.25}$	83.0 (-0.0)	83.5 (-0.1)
	$\rightarrow 1/\sqrt{1.25} \cdot (\mathrm{ReLU}(x))^2$	82.6 (-0.4)	83.4 (-0.2)
	$\rightarrow (\mathrm{ReLU}(x))^2 - 0.5$	83.0 (-0.0)	83.3 (-0.3)
Branch output scaling	ResScale [33] → None	82.8 (-0.2)	83.2 (-0.4)
	ResScale [33] → LayerScale [46]	82.8 (-0.2)	83.0 (-0.6)
	ResScale [33] → BranchScale	82.9 (-0.1)	83.3 (-0.3)
Biases in each block	Disable biases of Norm, FC and Conv → Enable biases	83.0 (-0.0)	83.5 (-0.1)

TABLE 9: **Performance of object detection and instance segmentation on COCO** with Mask R-CNN and Cascade Mask R-CNN. The MACs are measured with input size of 800×1333 (* except 896×896 of MaxViT). The FPS are measured on NVIDIA V100 GPU.

Backbone	MACs (G)	FPS	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Mask R-CNN 3× schedule								
Swin-T	267	19.0	46.0	68.1	50.3	41.6	65.1	44.9
ConvNeXt-T	262	22.1	46.2	67.9	50.8	41.7	65.0	44.9
ConvFormer-S18	251	18.3	47.7	69.6	52.3	42.6	66.3	45.9
CAFormer-S18	254	18.0	**48.6**	70.5	53.4	**43.7**	67.5	47.4
Cascade Mask R-CNN 3× schedule								
MaxViT-T	475*	-	52.1	71.9	56.8	44.6	69.1	48.4
MaxViT-S	595*	-	53.1	72.5	58.1	45.4	69.8	49.5
MaxViT-B	856*	-	53.4	72.9	58.1	45.7	70.3	50.0
Swin-T	745	8.5	50.4	69.2	54.7	43.7	66.6	47.3
ConvNeXt-T	741	9.4	50.4	69.1	54.8	43.7	66.5	47.3
ConvFormer-S18	729	8.7	51.5	70.7	55.8	44.6	67.8	48.2
CAFormer-S18	733	8.7	**52.3**	71.3	56.9	**45.2**	68.6	48.8
Swin-S	838	7.8	51.9	70.7	56.3	45.0	68.2	48.8
ConvNeXt-S	827	8.6	51.9	70.8	56.5	45.0	68.4	49.1
ConvFormer-S36	805	7.4	52.5	71.1	57.0	45.2	68.6	48.8
CAFormer-S36	811	7.1	**53.2**	72.1	57.7	**46.0**	69.5	49.8
Swin-B	982	7.7	51.9	70.5	56.4	45.0	68.1	48.9
ConvNeXt-B	964	8.2	52.7	71.3	57.2	45.6	68.9	49.5
ConvFormer-M36	912	6.7	53.0	71.4	57.4	45.7	69.2	49.5
CAFormer-M36	920	6.4	**53.8**	72.5	58.3	**46.5**	70.1	50.7

TABLE 10: **Performance of Semantic segmentation with UperNet [89] on ADE20K [88] validation set.** Images are cropped to 512×512 for training. The MACs are measured with input size of 512×2048. The FPS are measured on NVIDIA V100 GPU.

Backbone	UperNet Params (M)	MACs (G)	FPS	mIoU (%)
Swin-T [5]	60	945	21.3	45.8
ConvNeXt-T [3]	60	939	21.3	46.7
ConvFormer-S18	54	925	23.7	48.6
CAFormer-S18	54	1024	21.4	**48.9**
Swin-S [5]	81	1038	14.7	49.5
ConvNeXt-S [3]	82	1027	15.7	49.6
ConvFormer-S36	67	1003	11.9	50.7
CAFormer-S36	67	1197	10.8	**50.8**
Swin-B [5]	121	1188	14.6	49.7
ConvNeXt-B [3]	122	1170	15.0	49.9
ConvFormer-M36	85	1113	11.5	51.3
CAFormer-M36	84	1346	9.8	**51.7**

TABLE 11: **Benchmarking speed of activations.** We benchmark the speed by 10K runs with input shape of $1 \times 1M$ in PyTorch [61] on an NVIDIA A100 GPU. Note that it is unfair to directly compare GELU (PyTorch API) and StarReLU because the former is CUDA optimized while the last is not.

Activation	Speed (runs/s)
GELU (PyTorch API)	145,067 (CUDA optimized)
GELU (Equation 9)	20,273
StarReLU	35,257 (1.7x)

attention-based token mixers [4], [5], [14]. However, it is shown in MLP-Mixer [17] and FNet [18] that, by replacing attention in Transformer with spatial MLP [23] and Fourier transform, the resulting models still deliver competitive results. Along this line, [24] abstracts the Transformer into a general architecture termed MetaFormer, and meanwhile proposes the hypothesis that, it is the MetaFormer that really plays a critical role in achieving promising performance. To this end, [24] specifies the token mixer to be as embarrassingly simple as pooling, and observes that the resultant model PoolFormer surpasses the well-tuned ResNet/ViT/MLP-like baselines [1], [2], [4], [11], [17], [22], [25], [26]. The power of MetaFormer can also be verified by the recent models adopting MetaFormer as the general architecture but with different attention-based [49], [99], [100], MLP-based [17], [22], [101], [102], [103], convolution-based [51], [56], [104], [105], [106], hybrid [32], [54], [55], [58] or other types of [107], [108] token mixers. Unlike these works, we do not attempt to introduce novel token mixers, but merely specify token mixers as the most basic or commonly-used operators to probe the capacity of MetaFormer.

TABLE 12: **Inference throughputs of different models.** Models are trained and tested on resolution of 224^2. We benchmark the throughputs on an NVIDIA A100 GPU with batch size of 128 and TF32.

Models	Params (M)	MACs (G)	Top-1 (%)	Throughput (img/s)
Swin-T	29	4.5	81.3	1768
ConvNeXt-T	29	4.5	82.1	2413
ConvFormer-S18	27	3.9	83.0	2213
CAFormer-S18	26	4.1	83.6	2093
Swin-S	50	8.7	83.0	1131
ConvNeXt-S	50	8.7	83.1	1535
MaxViT-T	31	5.6	83.6	904
ConvFormer-S36	40	7.6	84.1	1205
CAFormer-S36	39	8.0	84.5	1138
Swin-B	88	15.4	83.5	843
ConvNeXt-B	89	15.4	83.8	1122
MaxViT-S	69	11.7	84.5	616
ConvFormer-M36	57	12.8	84.5	899
CAFormer-M36	56	13.2	85.2	852
ConvNeXt-L	198	34.4	84.3	681
MaxViT-B	120	23.4	85.0	345
ConvFormer-B36	100	22.6	84.8	677
CAFormer-B36	99	23.2	85.5	644

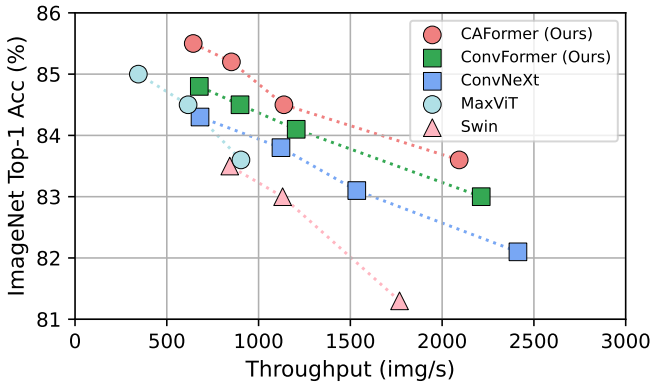

Fig. 4: **Trade-off between accuracy and inference throughput.** The throughputs are measured on an NVIDIA A100 GPU with batch size of 128 and TF32.

5 Conclusion

In this paper, we make our exploration to study the capacity of MetaFormer, the abstracted architecture of Transformer. We take our eyes off the token-mixer design, and merely rely on the most basic or "old-fashioned" token mixers dated back years ago to build the MetaFormer models, namely IdentityFormer, RandFormer, ConvFormer, and CAFormer. The former two models, built upon identify mapping and randomized mixing, demonstrate the solid lower bound of MetaFormer and its universality to token mixers; the latter two models, built upon conventional separable convolutions and vanilla self-attention, readily offer recording-setting results. In our investigation, we also discover that a new activation, StarReLU, not only achieves better performance but also greatly reduces FLOPs of the activation function when compared with GELU. We expect MetaFormer to find its even broader domain of vision applications in future work, and cheerfully invite readers to try out the proposed MetaFormer baselines.

Acknowledgments

This project is supported by the Advanced Research and Technology Innovation Centre (ARTIC), the National University of Singapore (project number: A-0005947-21-00, project reference: ECT-RP2), the Singapore Ministry of Education Academic Research Fund Tier 1 (WBS: A-0009440-01-00), and the National Research Foundation Singapore under its AI Singapore Programme (Award Number: AISG2-RP-2021-023). Weihao Yu and Xinchao Wang would like to thank TRC program and GCP research credits for the support of partial computational resources. We would like to thank Fredo Guan (independent researcher) and Ross Wightman (Hugging Face) for merging MetaFormer code into pytorch-image-models codebase.

Appendix A
Expectation and Variance of Squared ReLU

Assuming the input x of Squared ReLU [41] follows normal distribution with mean 0 and variance 1, *i.e.*$x \sim N(0,1)$, we have:

$$\mathrm{E}(x^2) = \mathrm{Var}(x) = 1 \tag{19}$$

$$\mathrm{E}\left((\mathrm{ReLU}(x))^2\right) = \frac{1}{2}\mathrm{E}(x^2) = 0.5 \tag{20}$$

$$\mathrm{E}(x^4) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} z^4 \exp\left(-\frac{z^2}{2}\right) dz \tag{21}$$

$$= -\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} z^3 d\left(\exp\left(-\frac{z^2}{2}\right)\right) \tag{22}$$

$$= \left(-z^3\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{z^2}{2}\right)\right)\Bigg|_{-\infty}^{+\infty} + \tag{23}$$

$$3\int_{-\infty}^{+\infty} z^2 \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{z^2}{2}\right) dz \tag{24}$$

$$= 0 + 3\mathrm{E}(x^2) = 3 \tag{25}$$

$$\mathrm{E}\left((\mathrm{ReLU}(x))^4\right) = \frac{1}{2}\mathrm{E}(x^4) = 1.5 \tag{26}$$

$$\mathrm{Var}\left((\mathrm{ReLU}(x))^2\right) = \mathrm{E}\left((\mathrm{ReLU}(x))^4\right) - \left(\mathrm{E}\left((\mathrm{ReLU}(x))^2\right)\right)^2 \tag{27}$$

$$= 1.5 - 0.5^2 = 1.25 \tag{28}$$

where $\mathrm{E}(\cdot)$ and $\mathrm{Var}(\cdot)$ denote expectation and variance, respectively. Thus, we can obtain the expectation $\mathrm{E}\left((\mathrm{ReLU}(x))^2\right) = 0.5$ and variance $\mathrm{Var}\left((\mathrm{ReLU}(x))^2\right) = 1.25$.

Appendix B
Code of Separable Convolution

Algorithm 2 shows the PyTorch-like code of inverted separable convolution from MobileNetV2 [8].

Appendix C
Hyper-parameters

The hyper-parameters of IdentityFormer, RandFormer and PoolFormerV2 trained on ImageNet-1K [59] are shown in Table 13, while those of ConvFormer and CAFormer are shown in Table 14 for training on ImageNet-1K and Table 15 for pre-training on ImageNet-1K and fine-tuning on ImageNet-1K.

TABLE 13: Hyper-parameters of IdentityFormer, RandFormer, PoolFormerV2 trained on ImageNet-1K.

Hyper-parameter	IdentityFormer	RandFormer	PoolFormerV2
Model size	S12/S24/S36/M36/M48		
Epochs	300		
Resolution	224^2		
Batch size	4096		
Optimizer	AdamW		
Learning rate	4e-3		
Learning rate decay	Cosine		
Warmup epochs	5		
Weight decay	0.05		
Rand Augment	9/0.5		
Mixup	0.8		
Cutmix	1.0		
Erasing prob	0.25		
Peak stochastic depth rate	0.1/0.1/0.2/0.3/0.4	0.1/0.1/0.2/0.3/0.3	0.1/0.1/0.2/0.3/0.4
Label smoothing	0.1		

TABLE 14: Hyper-parameters of ConvFormer and CAFormer trained on ImageNet-1K and finetuned at larger resolution of 384^2.

Hyper-parameter	ConvFormer		CAFormer	
	Train	Finetune	Train	Finetune
Model size	S18/S36/M36/B36			
Epochs	300	30	300	30
Resolution	224^2	384^2	224^2	384^2
Batch size	4096	1024	4096	1024
Optimizer	AdamW	AdamW	LAMB	LAMB
Learning rate	4e-3	5e-5	8e-3	1e-4
Learning rate decay	Cosine	None	Cosine	None
Warmup epochs	20	None	20	None
Weight decay	0.05			
Rand Augment	9/0.5			
Mixup	0.8	None	0.8	None
Cutmix	1.0	None	1.0	None
Erasing prob	0.25			
Peak stochastic depth rate	0.2/0.3/0.4/0.6	0.3/0.5/0.8/0.8	0.15/0.3/0.4/0.6	0.3/0.5/0.7/0.8
MLP head dropout rate	0/0/0/0	0.4/0.4/0.5/0.5	0/0.4/0.4/0.5	0.4/0.4/0.4/0.5
Label smoothing	0.1			
EMA decay rate	None	0.9999	None	0.9999

REFERENCES

[1] 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.

[2] R. Wightman, H. Touvron, and H. Jégou, "Resnet strikes back: An improved training procedure in timm," *arXiv preprint arXiv:2110.00476*, 2021.

[3] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *arXiv preprint arXiv:2201.03545*, 2022.

[4] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 568–578.

[5] 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," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[6] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[7] F. Mamalet and C. Garcia, "Simplifying convnets for fast learning," in *International Conference on Artificial Neural Networks*. Springer, 2012, pp. 58–65.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[9] 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, 2017.

[10] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1691–1703.

[11] 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.

[12] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[13] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr *et al.*, "Rethinking semantic segmentation

TABLE 15: Hyper-parameters of ConvFormer and CAFormer pre-trained on ImageNet-21K and fine-tuned on ImageNet-1K at resolution of 224^2 and 384^2.

Hyper-parameter	ConvFormer		CAFormer	
	Pretrain	Finetune	Pretrain	Finetune
Model size	S18/S36/M36/B36			
Epochs	90	30	90	30
Resolution	224^2	$224^2/384^2$	224^2	$224^2/384^2$
Batch size	4096	1024	4096	1024
Optimizer	AdamW	AdamW	LAMB	LAMB
Learning rate	1e-3	5e-5	2e-3	1e-4
Learning rate decay	Cosine	None	Cosine	None
Warmup epochs	5	None	5	None
Weight decay	0.05			
Rand Augment	9/0.5			
Mixup	0.8	None	0.8	None
Cutmix	1.0	None	1.0	None
Erasing prob	0.25			
Peak stochastic depth rate	0/0/0.1/0.2	0.1/0.1/0.1/0.3	0.1/0.1/0.1/0.3	0.1/0.1/0.1/0.3
MLP head dropout rate	0.2/0.2/0.2/0.3	0.2/0.2/0.2/0.5	0.2/0.2/0.2/0.4	0.2/0.2/0.2/0.5
Label smoothing	0.1			
EMA decay rate	None	0.9999	None	0.9999

Algorithm 2 Token mixer of separable convolution, PyTorch-like Code

```
import torch
import torch.nn as nn

# Separable convolution
class SepConv(nn.Module):
    "Inverted separable convolution from MobileNetV2"
    def __init__(self, dim, kernel_size=7, padding=3,
         expansion_ratio=2, act1=nn.ReLU, act2=nn.
         Identity,
        bias=False):
        super().__init__()
        med_channels = int(expansion_ratio * dim)
        self.pwconv1 = nn.Linear(dim, med_channels, bias=
            bias) # pointwise conv implemented by FC
        self.act1 = act1()
        self.dwconv = nn.Conv2d(med_channels, med_channels
            , kernel_size=kernel_size,
           padding=padding, groups=med_channels, bias=bias
                ) # depthwise conv
        self.act2 = act2()
        self.pwconv2 = nn.Linear(med_channels, dim, bias=
            bias) # pointwise conv implemented by FC

    def forward(self, x):
        # [B, H, W, C] = x.shape
        x = self.pwconv1(x)
        x = self.act1(x)
        x = x.permute(0, 3, 1, 2) # [B, H, W, D] -> [B, D,
             H, W]
        x = self.dwconv(x)
        x = x.permute(0, 2, 3, 1) # [B, D, H, W] -> [B, H,
             W, D]
        x = self.act2(x)
        x = self.pwconv2(x)
        return x
```


from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6881–6890.

[14] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 558–567.

[15] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[16] 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.

[17] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[18] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "Fnet: Mixing tokens with fourier transforms," *arXiv preprint arXiv:2105.03824*, 2021.

[19] Y. Zhao, G. Wang, C. Tang, C. Luo, W. Zeng, and Z.-J. Zha, "A battle of network structures: An empirical study of cnn, transformer, and mlp," *arXiv preprint arXiv:2108.13002*, 2021.

[20] Q. Han, Z. Fan, Q. Dai, L. Sun, M.-M. Cheng, J. Liu, and J. Wang, "On the connection between local attention and dynamic depth-wise convolution," in *International Conference on Learning Representations*, 2021.

[21] Y. Rao, W. Zhao, Z. Zhu, J. Lu, and J. Zhou, "Global filter networks for image classification," *Advances in Neural Information Processing Systems*, vol. 34, pp. 980–993, 2021.

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

[23] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, and C. Zheng, "Synthesizer: Rethinking self-attention for transformer models," in *International conference on machine learning*. PMLR, 2021, pp. 10 183–10 192.

[24] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "Metaformer is actually what you need for vision," *arXiv preprint arXiv:2111.11418*, 2021.

[25] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 347–10 357.

[26] H. Liu, Z. Dai, D. So, and Q. V. Le, "Pay attention to mlps," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9204–9215, 2021.

[27] 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.

[28] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[30] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

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

[32] Z. Dai, H. Liu, Q. V. Le, and M. Tan, "Coatnet: Marrying convolution and attention for all data sizes," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3965–3977, 2021.

[33] S. Shleifer, J. Weston, and M. Ott, "Normformer: Improved transformer pretraining with extra normalization," *arXiv preprint arXiv:2110.09456*, 2021.

[34] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer." *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[35] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[36] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[37] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[38] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[40] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[41] D. R. So, W. Mańke, H. Liu, Z. Dai, N. Shazeer, and Q. V. Le, "Primer: Searching for efficient transformers for language modeling," *arXiv preprint arXiv:2109.08668*, 2021.

[42] S. O. users, "How many flops does tanh need?" https://stackoverflow.com/questions/41251698/how-many-flops-does-tanh-need, 2017, accessed: 2022-03-01.

[43] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *Advances in neural information processing systems*, vol. 30, 2017.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[45] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic relu," in *European Conference on Computer Vision*. Springer, 2020, pp. 351–367.

[46] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 32–42.

[47] C. Zhu, R. Ni, Z. Xu, K. Kong, W. R. Huang, and T. Goldstein, "Gradinit: Learning to initialize neural networks for stable and efficient training," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 410–16 422, 2021.

[48] L. Liu, X. Liu, J. Gao, W. Chen, and J. Han, "Understanding the difficulty of training transformers," *arXiv preprint arXiv:2004.08249*, 2020.

[49] Y. Li, C.-Y. Wu, H. Fan, K. Mangalam, B. Xiong, J. Malik, and C. Feichtenhofer, "Mvitv2: Improved multiscale vision transformers for classification and detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4804–4814.

[50] 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.

[51] M.-H. Guo, C.-Z. Lu, Z.-N. Liu, M.-M. Cheng, and S.-M. Hu, "Visual attention network," *arXiv preprint arXiv:2202.09741*, 2022.

[52] X. Dong, J. Bao, D. Chen, W. Zhang, N. Yu, L. Yuan, D. Chen, and B. Guo, "Cswin transformer: A general vision transformer backbone with cross-shaped windows," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 124–12 134.

[53] T. Yao, Y. Li, Y. Pan, Y. Wang, X.-P. Zhang, and T. Mei, "Dual vision transformer," *arXiv preprint arXiv:2207.04976*, 2022.

[54] K. Li, Y. Wang, P. Gao, G. Song, Y. Liu, H. Li, and Y. Qiao, "Uniformer: Unified transformer for efficient spatiotemporal representation learning," *arXiv preprint arXiv:2201.04676*, 2022.

[55] C. Si, W. Yu, P. Zhou, Y. Zhou, X. Wang, and S. Yan, "Inception transformer," *arXiv preprint arXiv:2205.12956*, 2022.

[56] X. Ding, X. Zhang, J. Han, and G. Ding, "Scaling up your kernels to 31x31: Revisiting large kernel design in cnns," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 963–11 975.

[57] S. Liu, T. Chen, X. Chen, X. Chen, Q. Xiao, B. Wu, T. Kärkkäinen, M. Pechenizkiy, D. C. Mocanu, and Z. Wang, "More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity," in *The Eleventh International Conference on Learning Representations*, 2022.

[58] Z. Tu, H. Talebi, H. Zhang, F. Yang, P. Milanfar, A. Bovik, and Y. Li, "Maxvit: Multi-axis vision transformer," *arXiv preprint arXiv:2204.01697*, 2022.

[59] 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*. Ieee, 2009, pp. 248–255.

[60] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[61] 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," *Advances in neural information processing systems*, vol. 32, 2019.

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

[63] 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.

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

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] M. Berman, H. Jégou, A. Vedaldi, I. Kokkinos, and M. Douze, "Multigrain: a unified image embedding for classes and instances," *arXiv preprint arXiv:1902.05509*, 2019.

[70] E. Hoffer, T. Ben-Nun, I. Hubara, N. Giladi, T. Hoefler, and D. Soudry, "Augment your batch: Improving generalization through instance repetition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8129–8138.

[71] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[73] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large batch optimization for deep learning: Training bert in 76 minutes," *arXiv preprint arXiv:1904.00962*, 2019.

[74] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM journal on control and optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[75] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[76] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song, "Natural adversarial examples," in *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, 2021, pp. 15 262–15 271.

[77] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo *et al.*, "The many faces of robustness: A critical analysis of out-of-distribution generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8340–8349.

[78] H. Wang, S. Ge, Z. Lipton, and E. P. Xing, "Learning robust global representations by penalizing local predictive power," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[79] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[80] D. Zhou, Z. Yu, E. Xie, C. Xiao, A. Anandkumar, J. Feng, and J. M. Alvarez, "Understanding the robustness in vision transformers," in *International Conference on Machine Learning*. PMLR, 2022, pp. 27 378–27 394.

[81] X. Mao, G. Qi, Y. Chen, X. Li, R. Duan, S. Ye, Y. He, and H. Xue, "Towards robust vision transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 042–12 051.

[82] Z. Wang, Y. Bai, Y. Zhou, and C. Xie, "Can cnns be more robust than transformers?" in *The Eleventh International Conference on Learning Representations*, 2023.

[83] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[84] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[85] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154–6162.

[86] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[87] A. Hassani, S. Walton, J. Li, S. Li, and H. Shi, "Neighborhood attention transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6185–6194.

[88] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641.

[89] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 418–434.

[90] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[91] H. Yan, B. Deng, X. Li, and X. Qiu, "Tener: adapting transformer encoder for named entity recognition," *arXiv preprint arXiv:1911.04474*, 2019.

[92] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[93] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *arXiv preprint arXiv:1904.13377*, 2019.

[94] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

[95] J. Kim, M. El-Khamy, and J. Lee, "T-gsa: Transformer with gaussian-weighted self-attention for speech enhancement," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6649–6653.

[96] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[97] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," *arXiv preprint arXiv:1908.03557*, 2019.

[98] Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, "Uniter: Universal image-text representation learning," in *European conference on computer vision*. Springer, 2020, pp. 104–120.

[99] S. Ren, D. Zhou, S. He, J. Feng, and X. Wang, "Shunted self-attention via multi-scale token aggregation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 853–10 862.

[100] J. Yang, C. Li, P. Zhang, X. Dai, B. Xiao, L. Yuan, and J. Gao, "Focal attention for long-range interactions in vision transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30 008–30 022, 2021.

[101] Q. Hou, Z. Jiang, L. Yuan, M.-M. Cheng, S. Yan, and J. Feng, "Vision permutator: A permutable mlp-like architecture for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[102] S. Chen, E. Xie, C. Ge, D. Liang, and P. Luo, "Cyclemlp: A mlp-like architecture for dense prediction," *arXiv preprint arXiv:2107.10224*, 2021.

[103] D. Lian, Z. Yu, X. Sun, and S. Gao, "As-mlp: An axial shifted mlp architecture for vision," *arXiv preprint arXiv:2107.08391*, 2021.

[104] J. Yang, C. Li, and J. Gao, "Focal modulation networks," *arXiv preprint arXiv:2203.11926*, 2022.

[105] Y. Rao, W. Zhao, Y. Tang, J. Zhou, S.-N. Lim, and J. Lu, "Hornet: Efficient high-order spatial interactions with recursive gated convolutions," *arXiv preprint arXiv:2207.14284*, 2022.

[106] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli, "Pay less attention with lightweight and dynamic convolutions," *arXiv preprint arXiv:1901.10430*, 2019.

[107] Y. Tatsunami and M. Taki, "Sequencer: Deep lstm for image classification," *arXiv preprint arXiv:2205.01972*, 2022.

[108] K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision gnn: An image is worth graph of nodes," *arXiv preprint arXiv:2206.00272*, 2022.