MetaFormer Baselines for Vision

Weihao Yu Chenyang Si Pan Zhou Mi Luo Yichen Zhou
Jiashi Feng Shuicheng Yan Xinchao Wang
https://github.com/sail-sg/metaformer

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 $\times$ 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.

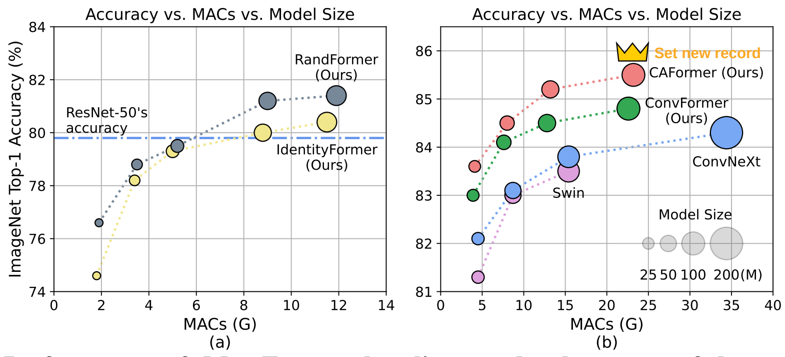

Figure 1: **Performance of MetaFormer baselines and other state-of-the-art models on ImageNet-1K at** 224^2 **resolution.** The architectures of our proposed models are shown in Figure 2. (a) IdentityFormer/RandFormer achieve over 80%/81% accuracy, indicating MetaFormer has solid lower bound of performance and works well on arbitrary token mixers. The accuracy of well-trained ResNet-50 [24] is from [76]. (b) Without novel token mixers, pure CNN-based ConvFormer outperforms ConvNeXt [46], 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.

- This work was partially performed when W. Yu was a research intern at Sea AI Lab.
- W. Yu, M. Luo and X. Wang are with National University of Singapore. Emails: weihaoyu@u.nus.edu, xinchao@nus.edu.sg.
- C. Si, P. Zhou, Y. Zhou, J. Feng, and S. Yan are with Sea AI Lab. Email: yansc@sea.com.
- Corresponding authors: X. Wang and S. Yan.

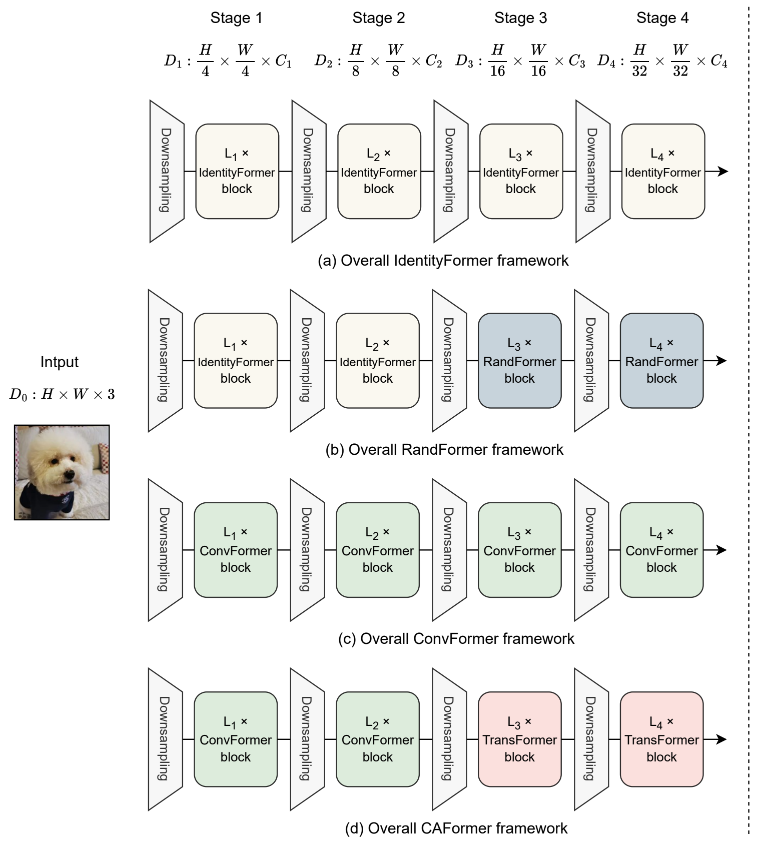
Stage 1
Stage 2
Stage 3
Stage 4
$D_1 : \frac{H}{4} \times \frac{W}{4} \times C_1$
$D_2 : \frac{H}{8} \times \frac{W}{8} \times C_2$
$D_3 : \frac{H}{16} \times \frac{W}{16} \times C_3$
$D_4 : \frac{H}{32} \times \frac{W}{32} \times C_4$
Intput
$D_0 : H \times W \times 3$
Downsampling
L_1 × IdentityFormer block
L_2 × IdentityFormer block
L_3 × IdentityFormer block
L_4 × IdentityFormer block
(a) Overall IdentityFormer framework
L_3 × RandFormer block
L_4 × RandFormer block
(b) Overall RandFormer framework
L_1 × ConvFormer block
L_2 × ConvFormer block
L_3 × ConvFormer block
L_4 × ConvFormer block
(c) Overall ConvFormer framework
L_3 × TransFormer block
L_4 × TransFormer block
(d) Overall CAFormer framework
Channel MLP
Norm
Identity Mapping
Random Mixing
Separable Convolution
Attention
(e) IdentityFormer block
(f) RandFormer block
(g) ConvFormer block
(h) Transformer block

Figure 2: **(a-d) Overall frameworks of IdentityFormer, RandFormer, ConvFormer and CAFormer.** Similar to [24, 74, 45], 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 15), separable depthwise convolutions [9, 48, 60] (Equation 16) or vanilla self-attention [73], respectively.

1 Introduction

In recent years, Transformers [73] have demonstrated unprecedented success in various computer vision tasks [5, 17, 4, 88]. The competence of Transformers has been long attributed to its attention module. As such, many attention-based token mixers [84, 21, 74, 45, 90] have been proposed in the aim to strengthen the Vision Transformers (ViTs) [17]. Nevertheless, some work [67, 36, 87, 22, 57] found that, by replacing the attention module in Transformers with simple operators like spatial MLP [67, 68, 66] or Fourier transform [36], the resultant models still produce encouraging performance.

Along this line, the work [83] 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, [83] adopts embarrassingly simple operator, pooling, to be the token mixer, and discovers that *PoolFormer* effectively outperforms the delicate ResNet/ViT/MLP-like baselines [24, 76, 17, 69, 74, 67, 68, 42], which confirms the significance of MetaFormer.

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 [9, 48, 60] and vanilla self-attention [73], as shown in Figure 2. We summarize our key experimental results in Figure 1, and 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.
- **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 [9, 48, 60] and vanilla self-attention [73], 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 [28, 77, 73, 17]. Experiments results showcase that ConvFormer consistently outperforms the strong pure-CNN model ConvNeXt [46].
  - **CAFormer sets new record on ImageNet-1K.** If we are to introduce attention into ConvFormer by even adopting the vanilla self-attention [73], 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 [83] is a general architecture instead of a specific model, which is abstracted from Transformer [73] by not specifying token mixer. Specifically, the input is first embedded as a sequence of features (or called tokens) [73, 17]:

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

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

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

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

where $\mathrm{Norm}_1(\cdot)$ and $\mathrm{Norm}_2(\cdot)$ are normalizations [31, 1]; $\mathrm{TokenMixer}(\cdot)$ 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 Techniques to improve MetaFormer

This paper does not introduce complicated token mixers. Instead, we introduce a new activation StarReLU and other two modifications [61, 55, 10] to improve MetaFormer.

2.2.1 StarReLU

In vanilla Transformer [73], ReLU [49] is chosen as the activation function that can be expressed as

$$\mathrm{ReLU}(x) = \max(0, x), \tag{4}$$

where x denotes any one neural unit of the input. This activation costs 1 FLOP for each unit. Later, GPT [53] uses GELU [25] as activation and then many subsequent Transformer models (*e.g.*, BERT [14], GPT-3 [3] and ViT [17]) employ this activation by default. GELU can be approximated as,

$$\mathrm{GELU}(x) = x\Phi(x) \approx 0.5 \times x(1 + \tanh(\sqrt{2/\pi}(x + 0.044715 \times x^3))), \tag{5}$$

where $\Phi(\cdot)$ is the Cumulative Distribution Function for Gaussian Distribution (CDFGD). Although revealing better performance than ReLU [63, 83], GELU approximately brings 14 FLOPs [1], much larger then ReLU's 1 FLOP of cost. To simplify GELU, [63] finds that CDFGD can be replaced by ReLU,

$$\mathrm{SquaredReLU}(x) = x\mathrm{ReLU}(x) = (\mathrm{ReLU}(x))^2. \tag{6}$$

This activation is called Squared ReLU [63], 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 [34]. Assuming input x follows normal distribution with mean 0 and variance 1, *i.e.*$x \sim N(0, 1)$, we have:

$$\mathrm{E}\left((\mathrm{ReLU}(x))^2\right) = 0.5, \qquad \mathrm{Var}\left((\mathrm{ReLU}(x))^2\right) = 1.25, \tag{7}$$

where $\mathrm{E}(\cdot)$ and $\mathrm{Var}(\cdot)$ denote expectation and variance, respectively. See Appendix A.1 for the derivation process of Equation 7. Therefore the distribution shift can be solved by

$$\mathrm{StarReLU}(x) = \frac{(\mathrm{ReLU}(x))^2 - \mathrm{E}\left((\mathrm{ReLU}(x))^2\right)}{\sqrt{\mathrm{Var}\left((\mathrm{ReLU}(x))^2\right)}} = \frac{(\mathrm{ReLU}(x))^2 - 0.5}{\sqrt{1.25}} \tag{8}$$

$$\approx 0.8944 \cdot (\mathrm{ReLU}(x))^2 - 0.4472. \tag{9}$$

We name the above activation *StarReLU* as multiplications (*) is heavily used. However, the assumption of standard normal distribution for input is strong [34]. To make the activation adaptable to different situations, like different models or initialization, scale and bias can be set to be learnable [23, 8]. We uniformly re-write the activation as

$$\mathrm{StarReLU}(x) = s \cdot (\mathrm{ReLU}(x))^2 + b, \tag{10}$$

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 [23, 8]. We leave the study of StarReLU variant selection for different situations in the future.

[1]tanh is counted 6 FLOPs for simplicity [72].

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
```

2.2.2 Other modifications

Scaling branch output. To scale up Transformer model size from depth, [70] proposes *LayerScale* that multiplies layer output by a learnable vector:

$$X' = X + \lambda_l \odot \mathcal{F}(\mathrm{Norm}(X)), \tag{11}$$

where $X \in \mathbb{R}^{N \times C}$ denotes the input features with sequence length N and channel dimension C, $\mathrm{Norm}(\cdot)$ 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, [91, 43, 61] attempt to stabilize architectures by scaling the residual branch (*ResScale* [61]):

$$X' = \lambda_r \odot X + \mathcal{F}(\mathrm{Norm}(X)), \tag{12}$$

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:

$$X' = \lambda_r \odot X + \lambda_l \odot \mathcal{F}(\mathrm{Norm}(X)). \tag{13}$$

Among these three scaling techniques, we find ResScale performs best according to our experiments in Section 3.4. Thus, *we adopt ResScale [61] by default in this paper.*

Disabling biases. Following [55, 10], 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.*

2.3 IdentityFormer and RandFormer

Following [83], 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,

$$\mathrm{IdentityMapping}(X) = X. \tag{14}$$

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,

$$\mathrm{RandomMixing}(X) = XW_R, \tag{15}$$

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

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		$T = (\mathrm{Id}, \mathrm{Id}, \mathrm{Id}, \mathrm{Id})$	$T = (\mathrm{Id}, \mathrm{Id}, \mathrm{Rand}, \mathrm{Rand})$	$T = (\mathrm{Pool}, \mathrm{Pool}, \mathrm{Pool}, \mathrm{Pool})$
Classifier Head		Global average pooling, Norm, FC		

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		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		$T = (\mathrm{Conv}, \mathrm{Conv}, \mathrm{Conv}, \mathrm{Conv})$	$T = (\mathrm{Conv}, \mathrm{Conv}, \mathrm{Attn}, \mathrm{Attn})$
Classifier Head		Global average pooling, Norm, MLP	

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.

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 [35, 24] configurations of PoolFormer [83] to build models of different sizes. Specifically, we specify 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 [83] 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.

2.4 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 [9, 48]. Specifically, we follow the inverted separable convolution module in MobileNetV2 [60],

$$\mathrm{Convolutions}(\mathrm{X}) = \mathrm{Conv}_{\mathrm{pw2}}(\mathrm{Conv}_{\mathrm{dw}}(\sigma(\mathrm{Conv}_{\mathrm{pw1}}(X)))), \tag{16}$$

where $\mathrm{Conv}_{\mathrm{pw1}}(\cdot)$ and $\mathrm{Conv}_{\mathrm{pw2}}(\cdot)$ are pointwise convolutions, $\mathrm{Conv}_{\mathrm{dw}}(\cdot)$ is the depthwise convolution, and $\sigma(\cdot)$ means the non-linear activation function. The PyTorch-like code of this module is shown in Algorithm 2 in the appendix. In practice, we set the kernel size as 7 following [46] 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 [24, 74, 45] (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 [73] 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

Model	Params (M)	MACs (G)	Top-1 (%)
RSB-ResNet-18 [24, 76]	11.7	1.8	70.6
IdentityFormer-S12	11.9	1.8	74.6
RandFormer-S12	11.9 + $\underline{0.2}$	1.9	76.6
PoolFormerV2-S12 [83]	11.9	1.8	78.0
RSB-ResNet-34 [24, 76]	21.8	3.7	75.5
IdentityFormer-S24	21.3	3.4	78.2
RandFormer-S24	21.3 + $\underline{0.5}$	3.5	78.8
PoolFormerV2-S24 [83]	21.3	3.4	80.7
RSB-ResNet-50 [24, 76]	25.6	4.1	79.8
IdentityFormer-S36	30.8	5.0	79.3
RandFormer-S36	30.8 + $\underline{0.7}$	5.2	79.5
PoolFormerV2-S36 [83]	30.8	5.0	81.6
RSB-ResNet-101 [24, 76]	44.5	7.9	81.3
IdentityFormer-M36	56.1	8.8	80.0
RandFormer-M36	56.1 + $\underline{0.7}$	9.0	81.2
PoolFormerV2-M36 [83]	56.1	8.8	82.2
RSB-ResNet-152 [24, 76]	60.2	11.6	81.8
IdentityFormer-M48	73.3	11.5	80.4
RandFormer-M48	73.3 + $\underline{0.9}$	11.9	81.4
PoolFormerV2-M48 [83]	73.3	11.5	82.6

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.

comparison, convolution is a local operator with computational complexity linear to token length. Inspired by [4, 17, 12, 83], 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.

3 Experiments

3.1 Setup

ImageNet-1K [13] 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 [13, 59], a much larger dataset containing $\sim$14M images of 21841 classes, and then fine-tune the pretrained model on ImageNet-1K for evaluation. Our implementation is based on PyTorch library [50] and `Timm` codebase [75] and the experiments are run on TPUs.

Training and fine-tuning on ImageNet-1K. We mainly follow the hyper-parameters of DeiT [69]. Specifically, models are trained for 300 epochs at 224^2 resolution. Data augmentation and regularization techniques include RandAugment [11], Mixup [86], CutMix [85], Random Erasing [89], weight decay, Label Smoothing [64] and Stochastic Depth [30]. We do not use repeated augmentation [2, 26] and LayerScale [70], but use ResScale [61] for the last two stages. We adopt AdamW [33, 47] optimizer with batch size of 4096 for most models except CAFormer since we found it is not stable to train CAFormer under this setting. The problem may be caused by the large batch size, so we use a large-batch-size-friendly optimizer LAMB [82] for CAFormer. For 384^2 resolution, we fine-tune the models trained at 224^2 resolution for 30 epochs with Exponential Moving Average (EMA) [52]. The details of hyper-parameters are shown in Table 7 and 8 of 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

Model	Meta-Former	Mixing Type	Params (M)	@224 MACs (G)	@224 Top-1 (%)	↑384 MACs (G)	↑384 Top-1 (%)
RSB-ResNet-50 [24, 76]	✗	Conv	26	4.1	79.8	-	-
RegNetY-4G [54, 76]	✗	Conv	21	4.0	81.3	-	-
ConvNeXt-T [46]	✗	Conv	29	4.5	82.1	-	-
VAN-B2 [19]	✓	Conv	27	5.0	82.8	-	-
ConvFormer-S18	✓	Conv	27	3.9	**83.0**	11.6	84.4
DeiT-S [69]	✓	Attn	22	4.6	79.8	-	-
T2T-ViT-14 [84]	✓	Attn	22	4.8	81.5	17.1	83.3
Swin-T [45]	✓	Attn	29	4.5	81.3	-	-
CSWin-T [16]	✓	Attn	23	4.3	82.7	-	-
MViTv2-T [40]	✓	Attn	24	4.7	82.3	-	-
CoAtNet-0 [12]	✓	Conv + Attn	25	4.2	81.6	13.4	83.9
UniFormer-S [37]	✓	Conv + Attn	22	3.6	82.9	-	-
iFormer-S [62]	✓	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 [24, 76]	✗	Conv	45	7.9	81.3	-	-
RegNetY-8G [54, 76]	✗	Conv	39	8.0	82.1	-	-
ConvNeXt-S [46]	✗	Conv	50	8.7	83.1	-	-
VAN-B3 [19]	✓	Conv	45	9.0	83.9	-	-
ConvFormer-S36	✓	Conv	40	7.6	**84.1**	22.4	85.4
T2T-ViT-19 [84]	✓	Attn	39	8.5	81.9	-	-
Swin-S [45]	✓	Attn	50	8.7	83.0	-	-
CSWin-S [16]	✓	Attn	35	6.9	83.6	-	-
MViTv2-S [40]	✓	Attn	35	7.0	83.6	-	-
CoAtNet-1 [12]	✓	Conv + Attn	42	8.4	83.3	27.4	85.1
UniFormer-B [37]	✓	Conv + Attn	50	8.3	83.9	-	-
CAFormer-S36	✓	Conv + Attn	39	8.0	**84.5**	26.0	85.7
RSB-ResNet-152 [24, 76]	✗	Conv	60	11.6	81.8	-	-
RegNetY-16G [54, 76]	✗	Conv	84	15.9	82.2	-	-
ConvNeXt-B [46]	✗	Conv	89	15.4	83.8	45.0	85.1
RepLKNet-31B [15]	✓	Conv	79	15.3	83.5	45.1	84.8
VAN-B4 [19]	✓	Conv	60	12.2	84.2	-	-
ConvFormer-M36	✓	Conv	57	12.8	**84.5**	37.7	85.6
DeiT-B [69]	✓	Attn	86	17.5	81.8	55.4	83.1
T2T-ViT-24 [84]	✓	Attn	64	13.8	82.3	-	-
Swin-B [45]	✓	Attn	88	15.4	83.5	47.1	84.5
CSwin-B [16]	✓	Attn	78	15.0	84.2	47.0	85.4
MViTv2-B [40]	✓	Attn	52	10.2	84.4	36.7	85.6
CoAtNet-2 [12]	✓	Conv + Attn	75	15.7	84.1	49.8	85.7
MaxViT-S [71]	✓	Conv + Attn	69	11.7	84.5	36.1	85.2
iFormer-L [62]	✓	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 [54, 76]	✗	Conv	145	32.3	82.4	-	-
ConvNeXt-L [46]	✗	Conv	198	34.4	84.3	101.0	85.5
ConvFormer-B36	✓	Conv	100	22.6	**84.8**	66.5	85.7
MViTv2-L [40]	✓	Attn	218	42.1	85.3	140.2	86.3
CoAtNet-3 [12]	✓	Conv + Attn	168	34.7	84.5	107.4	85.8
MaxViT-B [71]	✓	Conv + Attn	120	23.4	85.0	74.2	86.3
CAFormer-B36	✓	Conv + Attn	99	23.2	**85.5***	72.2	86.4

Table 4: **Performance of models trained on ImageNet-1K at the resolution of** 224^2 **and fine-tuned at** 384^2**.** Model highlighted with blue 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 [40] with 55% fewer parameters and 45% fewer MACs.

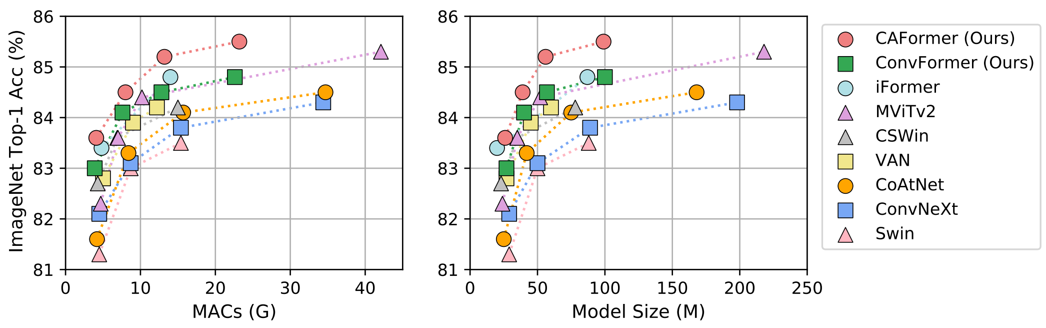

Figure 3: **ImageNet-1K validation accuracy *vs.*MACs/Model Size at the resolution of 224^2.** Models with token (feature) mixing based on convolution, attention or hybrid are presented by □, △ or ○ respectively.

of 224^2 and 384^2 for 30 epochs with EMA [52]. See Table 9 in the appendix for more details of hyper-parameters.

3.2 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 [24, 76] 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.

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

3.3 Results of models with commonly-used token mixers

We build ConvFormer by specifying the token mixer in MetaFormer as separable convolutions [9, 48] used in MobileNetV2 [60]. 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 4.

ConvFormer actually can be regarded as pure CNN-based model without any attention mechanism [28, 77, 73, 17]. It can be observed that ConvFormer outperforms strong CNN model ConvNeXt [46] 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. Also, ConvFormer outperforms

Model	Meta-Former	Mixing Type	Params (M)	Testing at resolution			
				@224		↑384	
				MACs (G)	Top-1 (%)	MACs (G)	Top-1 (%)
ConvNeXt-T [46]	✗	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 [46]	✗	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 [46]	✗	Conv	89	15.4	85.8	45.1	86.8
ConvFormer-M36	✓	Conv	57	12.8	**86.1**	37.7	86.9
Swin-B [45]	✓	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 [46]	✗	Conv	198	34.4	86.8	101.0	87.5
ConvFormer-B36	✓	Conv	100	22.6	**87.0**	66.5	87.6
Swin-L [45]	✓	Attn	197	34.5	86.3	103.9	87.3
CAFormer-B36	✓	Conv + Attn	99	23.2	**87.4**	72.2	88.1

Table 5: **Performance of models pre-trained on ImageNet-21K and fine-tuned on ImageNet-1K for evaluation.**

various strong attention-based or hybrid models. For instance, ConvFormer-M36 outperforms Swin-B [45]/CoAtNet-2 [12] 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 [9, 48, 60] in bottom stages and vanilla self-attention [73] 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, 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 [45]/ConvNeXt-L [46], 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.

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.

3.4 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. When the StarReLU is replaced with ReLU [49], 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 [63], the performance is already satisfied. But for ConvFormer-18, it can not match that of GELU [25]. 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

Ablation	Variant	Top-1 (%)	
		ConvFormer-S18	CAFormer-S18
–	Baseline	83.0	83.6
Activation types	StarReLU $\rightarrow$ ReLU [49]	82.1 (-0.9)	82.9 (-0.7)
	StarReLU $\rightarrow$ Squared ReLU [63]	82.6 (-0.4)	83.4 (-0.2)
	StarReLU $\rightarrow$ GELU [25]	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 [61] $\rightarrow$ None	82.8 (-0.2)	83.2 (-0.4)
	ResScale [61] $\rightarrow$ LayerScale [70]	82.8 (-0.0)	83.0 (-0.6)
	ResScale [61] $\rightarrow$ BranchScale	82.9 (-0.1)	83.3 (-0.3)
Biases in each block	Disable biases of Norm, FC and Conv $\rightarrow$ Enable biases	83.0 (-0.0)	83.5 (-0.1)

Table 6: **Ablation for ConvFormer-S18/CAFormer-S18 on ImageNet-1K.** * α and β denote learnable scalars shared for all channels.

of standard normal distribution for input [34] and can be conveniently applied for different models and initialization [23, 8]. 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 [61] performs best among the branch output scaling techniques; disabling biases [55, 10] 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.

4 Related work

Transformer, since being introduced in [73], has become a popular backbone for various tasks in NLP [53, 3, 44, 79, 55], computer vision [5, 17, 69, 84, 74, 45] and other domains [18, 51, 39, 32, 29, 38, 7]. In computer vision, iGPT [5] and ViT [17] 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 attention-based token mixers [84, 74, 45]. However, it is shown in MLP-Mixer [67] and FNet [36] that, by replacing attention in Transformer with spatial MLP [66] and Fourier transform, the resulting models still deliver competitive results. Along this line, [83] 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, [83] 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 [24, 76, 17, 69, 74, 67, 42, 68]. The power of MetaFormer can also be verified by the recent models adopting MetaFormer as the general architecture but with different attention-based [40, 58, 81], MLP-based [67, 68, 27, 6, 41], convolution-based [15, 19, 80, 56, 78], hybrid [62, 12, 37, 71] or other types of [65, 20] 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.

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 achieve better performance but also greatly reduces FLOPs of the activation function when compared with GELU. We expect MetaFormer to find its even boarder domain of vision applications in the future work, and cheerfully invite readers to try out the proposed MetaFormer baselines.

Acknowledgement

Weihao Yu and Xinchao Wang would like to thank TRC program and GCP 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), and by the National Research Foundation Singapore under its AI Singapore Programme (Award Number: AISG2-RP-2021-023).

A Appendix

A.1 Expectation and variance of Squared ReLU

Assuming the input x of Squared ReLU [63] (Equation 6) follows normal distribution with mean 0 and variance 1, *i.e.*$x \sim N(0,1)$, we have:

$$\begin{aligned}
\mathrm{E}(x^2) &= \mathrm{Var}(x) = 1 && (17)\\
\mathrm{E}\left((\mathrm{ReLU}(x))^2\right) &= \frac{1}{2}\mathrm{E}(x^2) = 0.5 && (18)\\
\mathrm{E}(x^4) &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} z^4 \exp\left(-\frac{z^2}{2}\right) dz && (19)\\
&= -\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} z^3 d\left(\exp\left(-\frac{z^2}{2}\right)\right) && (20)\\
&= \left(-z^3 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right)\right)\Bigg|_{-\infty}^{+\infty} + 3\int_{-\infty}^{+\infty} z^2 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) dz && (21)\\
&= 0 + 3\mathrm{E}(x^2) = 3 && (22)\\
\mathrm{E}\left((\mathrm{ReLU}(x))^4\right) &= \frac{1}{2}\mathrm{E}(x^4) = 1.5 && (23)\\
\mathrm{Var}\left((\mathrm{ReLU}(x))^2\right) &= \mathrm{E}\left((\mathrm{ReLU}(x))^4\right) - \left(\mathrm{E}\left((\mathrm{ReLU}(x))^2\right)\right)^2 = 1.5 - 0.5^2 = 1.25 && (24)
\end{aligned}$$

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

A.2 Code of separable convolution

Algorithm 2 shows the PyTorch-like code of inverted separable convolution from MobileNetV2 [60].

Algorithm 2 Token mixers 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=True):
        super().__init__()
        med_channels = int(expansion_ratio * dim)
        self.pwconv1 = nn.Linear(dim, med_channels, bias=bias) # pointwise conv implemented by FC
        self.act1 = preconv_act()
        self.dwconv = nn.Conv2d(med_channels, med_channels, kernel_size=kernel_size,
            padding=padding, groups=med_channels, bias=bias) # depthwise conv
        self.act2 = act_layer()
        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
```

A.3 Hyper-parameters

The hyper-parameters of IdentityFormer, RandFormer and PoolFormerV2 trained on ImageNet-1K [13] are shown in Table 7, while those of ConvFormer and CAFormer are shown in Table 8 for training on ImageNet-1K and Table 9 for pre-training on ImageNet-1K and fine-tuning 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 7: Hyper-parameters of IdentityFormer, RandFormer, PoolFormerV2 trained on ImageNet-1K.

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

Table 8: Hyper-parameters of ConvFormer and CAFormer trained on ImageNet-1K and finetuned at larger resolution of 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-3
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

Table 9: 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.

References

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

[2] Maxim Berman, Hervé Jégou, Andrea Vedaldi, Iasonas Kokkinos, and Matthijs Douze. Multigrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*, 2019.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] 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.

[5] 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.

[6] 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.

[7] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. 2019.

[8] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. In *European Conference on Computer Vision*, pages 351–367. Springer, 2020.

[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.

[10] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[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.

[12] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977, 2021.

[13] 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.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang 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*, pages 11963–11975, 2022.

[16] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining 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*, pages 12124–12134, 2022.

[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. *arXiv preprint arXiv:2010.11929*, 2020.

[18] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[19] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022.

[20] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. Vision gnn: An image is worth graph of nodes. *arXiv preprint arXiv:2206.00272*, 2022.

[21] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34, 2021.

[22] Qi Han, Zejia Fan, Qi Dai, Lei Sun, Ming-Ming Cheng, Jiaying Liu, and Jingdong Wang. On the connection between local attention and dynamic depth-wise convolution. In *International Conference on Learning Representations*, 2021.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[24] 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.

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

[26] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8129–8138, 2020.

[27] Qibin Hou, Zihang Jiang, Li Yuan, Ming-Ming Cheng, Shuicheng Yan, and Jiashi Feng. Vision permutator: A permutable mlp-like architecture for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[28] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[29] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

[30] 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.

[31] 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.

[32] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon 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)*, pages 6649–6653. IEEE, 2020.

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

[34] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

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

[37] Kunchang Li, Yali Wang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatiotemporal representation learning. *arXiv preprint arXiv:2201.04676*, 2022.

[38] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[39] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6706–6713, 2019.

[40] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4804–4814, 2022.

[41] Dongze Lian, Zehao Yu, Xing Sun, and Shenghua Gao. As-mlp: An axial shifted mlp architecture for vision. *arXiv preprint arXiv:2107.08391*, 2021.

[42] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021.

[43] Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020.

[44] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[45] 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*, pages 10012–10022, 2021.

[46] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[47] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

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

[50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[51] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, Sebastian Stüker, and Alexander Waibel. Very deep self-attention networks for end-to-end speech recognition. *arXiv preprint arXiv:1904.13377*, 2019.

[52] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[53] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[54] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[55] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[56] Yongming Rao, Wenliang Zhao, Yansong Tang, Jie Zhou, Ser-Nam Lim, and Jiwen Lu. Hornet: Efficient high-order spatial interactions with recursive gated convolutions. *arXiv preprint arXiv:2207.14284*, 2022.

[57] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *Advances in Neural Information Processing Systems*, 34:980–993, 2021.

[58] Sucheng Ren, Daquan Zhou, Shengfeng He, Jiashi Feng, and Xinchao Wang. Shunted self-attention via multi-scale token aggregation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10853–10862, 2022.

[59] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[60] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[61] Sam Shleifer, Jason Weston, and Myle Ott. Normformer: Improved transformer pretraining with extra normalization. *arXiv preprint arXiv:2110.09456*, 2021.

[62] Chenyang Si, Weihao Yu, Pan Zhou, Yichen Zhou, Xinchao Wang, and Shuicheng Yan. Inception transformer. *arXiv preprint arXiv:2205.12956*, 2022.

[63] David R So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. Primer: Searching for efficient transformers for language modeling. *arXiv preprint arXiv:2109.08668*, 2021.

[64] 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.

[65] Yuki Tatsunami and Masato Taki. Sequencer: Deep lstm for image classification. *arXiv preprint arXiv:2205.01972*, 2022.

[66] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International conference on machine learning*, pages 10183–10192. PMLR, 2021.

[67] Ilya O 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. *Advances in Neural Information Processing Systems*, 34, 2021.

[68] 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.

[69] 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.

[70] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021.

[71] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. *arXiv preprint arXiv:2204.01697*, 2022.

[72] Stack Overflow users. How many flops does tanh need? `https://stackoverflow.com/questions/41251698/how-many-flops-does-tanh-need`, 2017. Accessed: 2022-03-01.

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[74] 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*, pages 568–578, 2021.

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

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

[77] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[78] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

[79] Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. Tener: adapting transformer encoder for named entity recognition. *arXiv preprint arXiv:1911.04474*, 2019.

[80] Jianwei Yang, Chunyuan Li, and Jianfeng Gao. Focal modulation networks. *arXiv preprint arXiv:2203.11926*, 2022.

[81] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal attention for long-range interactions in vision transformers. *Advances in Neural Information Processing Systems*, 34:30008–30022, 2021.

[82] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

[83] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. *arXiv preprint arXiv:2111.11418*, 2021.

[84] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH 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*, pages 558–567, 2021.

[85] 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.

[86] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[87] Yucheng Zhao, Guangting Wang, Chuanxin Tang, Chong Luo, Wenjun Zeng, and Zheng-Jun Zha. A battle of network structures: An empirical study of cnn, transformer, and mlp. *arXiv preprint arXiv:2108.13002*, 2021.

[88] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021.

[89] 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.

[90] 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.

[91] Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W Ronny Huang, and Tom Goldstein. Gradinit: Learning to initialize neural networks for stable and efficient training. *Advances in Neural Information Processing Systems*, 34:16410–16422, 2021.