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](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], [23], [25], [26], 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 MetaFormer, 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.

- MetaFormer naturally 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.

- 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.

- 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.

- 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.

- 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](https://github.com/sail-sg/metaformer).
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, pure CNN-based ConvFormer outperforms ConvNeXt [3], while CAFormer sets a new record of 85.5% accuracy on ImageNet-1K at 224 × 224 resolution under normal supervised training without external data or distillation.

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 × 224 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]:

\[ X = \text{InputEmbedding}(I). \]  

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

\[ X' = X + \text{TokenMixer}(\text{Norm}_1(X)), \]  \hspace{1cm} (2)

\[ X'' = X' + \sigma(\text{Norm}_2(X')W_1)W_2, \]  \hspace{1cm} (3)
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 \( \text{Norm}_1(\cdot) \) and \( \text{Norm}_2(\cdot) \) are normalizations [29], [30]; \( \text{TokenMixer}(\cdot) \) 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,

\[
\text{IdentityMapping}(X) = X.
\]  
(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,

\[
\text{RandomMixing}(X) = W_R X,
\]  
(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...
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 [33] and the overall frameworks are shown in Figure 2 [32].

### 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],

\[
\text{Convolutions}(X) = \text{Conv}_{pw2}(\text{Conv}_{dw}(\sigma(\text{Conv}_{pw1}(X)))) ,
\]

where \(\text{Conv}_{pw1}()\) and \(\text{Conv}_{pw2}()\) are pointwise convolutions, \(\text{Conv}_{dw}()\) is the depthwise convolution, and \(\sigma()\) 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, 2] (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 3. 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

\[
\text{ReLU}(x) = \max(0, x),
\]

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,

\[
\text{GELU}(x) = x\Phi(x)
\]

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

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

\[
\text{SquaredReLU}(x) = x\text{ReLU}(x) = \text{ReLU}(x)^2.
\]

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 2.4. 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:

\[
\text{E (ReLU}(x)^2) = 0.5, \quad \text{Var (ReLU}(x)^2) = 1.25.
\]

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

\[
\text{StarReLU}(x) = \frac{\text{ReLU}(x)^2 - \text{E (ReLU}(x)^2)}{\sqrt{\text{Var (ReLU}(x)^2)}}
\]

\[
= \frac{(\text{ReLU}(x)^2 - 0.5)}{\sqrt{1.25}}
\]

\[
\approx 0.8944 \cdot \text{ReLU}(x)^2 - 0.4472.
\]

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

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

---

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

```python
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(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 = x * torch.einsum("mn, bmc -> bmc", self.random_matrix, x)
        return x
```

---

**Table 2** for model configurations of different sizes.

---

**Figure 2** for model configurations of different sizes.

---

**Figure 3** for model configurations of different sizes.
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     | (64, 128, 320, 512), | (2, 2, 6, 2) |              |
| S24     | (64, 128, 320, 512), | (4, 4, 12, 4) |              |
| S36     | (64, 128, 320, 512), | (6, 6, 18, 6) |              |
| M36     | (96, 192, 384, 768), | (6, 6, 18, 6) |              |
| M48     | (96, 192, 384, 768), | (8, 8, 24, 8) |              |

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     | (64, 128, 320, 512), | (3, 3, 9, 3)     |
| S36     | (64, 128, 320, 512), | (3, 12, 18, 3)   |
| M36     | (96, 192, 384, 764), | (3, 12, 18, 3)   |
| B36     | (128, 256, 512, 768), | (3, 12, 18, 3)   |


df 

where \( x \) and \( b \) 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 as intuitively this variant 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 \( \sim \)14M images of 21841 classes, and then fine-tune the pretrained model on ImageNet-1K for evaluation. Our implementation is based on PyTorch library [61] and Timm codebase [62], and the experiments are run on TPU.

Training and fine-tuning on ImageNet-1K. We mainly follow the hyper-parameters of DeiT [25]. Specifically, models
1.9 76.6
[0x0]3.5 78.8
[0x0]5.2 79.5
[48x71]224
data set, we pre-train ConvFormer and CAFormer on ImageNet-1K.
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\times 384 resolution, we fine-tune the models trained at 224^2 resolution for 30 epochs with Exponential Moving Average (EMA) [74].
See the appendix for more details of hyper-parameters.

**Robustness evaluation.** Following ConvNeXt [2], 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 \textasciitilde 73M parameters and \textasciitilde 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.

### 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. The details of hyper-parameters are shown in the appendix.

### Table 3: Performance on ImageNet-1K of RSB-ResNet and MetaFormer models with basic tokens of identity mapping, random maxing and pooling.

| 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      | 81.2      |
| 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 (%) |
|------------------------------|------------|----------|-----------|
| Pre-trained ViT [25]         | 22         | 4.6      | 79.5      |
| 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.
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 | Testing at resolution ↑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-T [14] | ✓ | Attn | 22 | 4.8 | 81.5 | 17.1 | 83.3 |
| Swin-T [3] | ✓ | 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 [52] | ✓ | 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 [3] | ✓ | 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 [52] | ✓ | 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-21 [14] | ✓ | Attn | 64 | 13.8 | 82.3 | - | - |
| Swin-B [3] | ✓ | 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 [52] | ✓ | 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 [52] | ✓ | 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 $224^2$, ConvFormer-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 strong 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 [3] with input size of $480^2$ 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 5. 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 [3], ConvNeXt-L [3], showing the promising scaling capability 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 CAFormer-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.
### 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, 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, the performance is already satisfied. For ConvFormer-18, it can not match that of GELU. 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/CAFormer-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 and can be conveniently applied for different models and initialization. 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 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 | Testing at resolution @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 $1.7 \times$ 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 (%) |
|------------------------|--------------------------------|-----------|
|                        | Baseline                       | 83.0      |
| Activation types       |                                |           |
| StarReLU → RelU         | 36                             | 82.1 (0.9)|
| StarReLU → Squared RelU | 41                             | 82.6 (0.4)|
| StarReLU → GELU         | 38                             | 82.7 (0.3)|
| StarReLU variants       |                                |           |
| α · (ReLU(x))^2 + β     |                                | 82.6 (0.4)|
| → α · (ReLU(x))^2       |                                | 83.0 (0.0)|
| → (ReLU(x))^2 + β       |                                | 83.0 (0.0)|
| → 1/√T25 · (ReLU(x))^2  |                                | 82.6 (0.4)|
| − 0.5/√T25 · (ReLU(x))^2|                                | 83.0 (0.0)|
| Branch output scaling   |                                |           |
| ResScale → None         |                                | 82.8 (0.2)|
| ResScale → LayerScale   | 46                             | 82.8 (0.2)|
| ResScale → BranchScale  |                                | 82.9 (0.1)|
| Biases in each block    | Disable biases of Norm, FC and Conv | 83.0 (0.0)|
|                        | Enable biases                   | 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 | APm (50) | APm (70) | APs (50) | APs (70) | AP | mIoU (%) |
|----------|----------|-----|----------|----------|----------|----------|----|----------|
| Swin-T   | 267      | 19.0 | 46.0     | 50.3     | 41.6     | 65.1     | 44.9|          |
| ConvNeXt-T | 262  | 22.1 | 46.2     | 67.9     | 50.8     | 65.0     | 44.9|          |
| ConvFormer-S18 | 251        | 18.3 | 47.7     | 69.6     | 52.3     | 66.3     | 45.9|          |
| CAFormer-S18 | 254        | 18.0 | 48.6     | 70.5     | 53.4     | 67.5     | 47.4|          |
| MaxViT-T | 475*     | -    | 52.1     | 71.9     | 56.8     | 44.6     | 69.1 | 48.4     |
| MaxViT-T | 595*     | -    | 53.1     | 72.5     | 58.1     | 45.4     | 69.8 | 49.5     |
| MaxViT-T | 856*     | -    | 53.4     | 72.9     | 58.1     | 45.7     | 70.3 | 50.0     |
| Swin-T   | 745*     | -    | 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     | 68.2 | 48.8     |
| Swin-S   | 838      | 7.8  | 51.9     | 70.7     | 56.3     | 45.0     | 68.2 | 48.8     |
| ConvNeX-T | 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   | 60       | 945      | 21.3 | 45.8     |
| ConvNeXt-T | 60   | 939      | 21.3 | 46.7     |
| ConvFormer-S18 | 54  | 925      | 23.7 | 48.6     |
| CAFormer-S18 | 54  | 1024     | 21.4 | 48.9     |
| Swin-S   | 81       | 1038     | 14.7 | 49.5     |
| ConvNeXt-S | 82    | 1027     | 15.7 | 49.6     |
| ConvFormer-S36 | 67  | 1003     | 11.9 | 50.7     |
| CAFormer-S36 | 67  | 1197     | 10.8 | 50.8     |
| Swin-B   | 121      | 1188     | 14.6 | 49.7     |
| ConvNeXt-B | 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 × 1/M 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) | 143,067 (CUDA optimized) |
| StarReLU    | 20,273          |
|              | 35,257 (1.7x)   |

attention-based token mixers [4], [6], [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], [3], [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], [59], [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 224². 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      | 1353               |
| 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:

\[
E(x^2) = \text{Var}(x) = 1
\]  \hspace{1cm} (19)

\[
E((\text{ReLU}(x))^2) = \frac{1}{2} E(x^2) = 0.5
\]  \hspace{1cm} (20)

\[
E(x^4) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} z^4 e^{-\frac{z^2}{2}} dz
\]

\[
= -\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} z^3 \left( \exp \left( -\frac{z^2}{2} \right) \right) dz
\]

\[
= \left( -\frac{3}{\sqrt{2\pi}} \right) \left( \exp \left( -\frac{z^2}{2} \right) \right) \bigg|_{-\infty}^{+\infty}
\]

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

\[
= 0 + 3E(x^2) = 3
\]  \hspace{1cm} (25)

\[
E((\text{ReLU}(x))^4) = \frac{1}{2} E(x^4) = 1.5
\]  \hspace{1cm} (26)

\[
\text{Var}((\text{ReLU}(x))^2) = E((\text{ReLU}(x))^4) - (E((\text{ReLU}(x))^2))^2
\]

\[
= 1.5 - 0.5^2 = 1.25
\]  \hspace{1cm} (28)

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

APPENDIX B

CODE OF SEPARABLE CONVOLUTION

Algorithm 7 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           | 224²           |            |              |
| 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       |                |            |              |

### TABLE 14: Hyper-parameters of ConvFormer and CAFormer trained on ImageNet-1K and finetuned at larger resolution of 384².

| Hyper-parameter      | ConvFormer | CAFormer |
|----------------------|------------|----------|
| Model size           | S18/S36/M36/B36 | Train | Finetune |
| Epochs               | 300        | 30       | 300       | 30       |
| Resolution           | 224²       | 384²     | 224²      | 384²     |
| 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. 10012–10022.

[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, L. 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 224² and 384².

| Hyper-parameter      | ConvFormer Pretrain | Finetune | CAFormer Pretrain | Finetune |
|----------------------|----------------------|----------|--------------------|----------|
| Model size           | S18/S36/M36/B36      |          |                    |          |
| Epochs               | 90                   | 30       | 90                 | 30       |
| Resolution           | 224²                 | 224²/384²| 224²               | 224²/384²|
| Batch size           | 4096                 | 1024     | 4096               | 1024     |
| Optimizer            | AdamW                | AdamW    | LAMB               | LAMB     |
| Learning rate        | 1e-3                 | 5e-3     | 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 | 0/0.0/0.1/0.2 | 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

```python
import torch
import torch.nn as nn

# Separable convolution
class SepConv(nn.Module):
    """Inverted separable convolution from MobileNetV2""
    def __init__(self, dim, kernel_size=1, padding=0, expansion_ratio=2, act1=nn.Identity, act2=nn.Identity):
        super().__init__()
        channels = dim
        self.pwconv1 = nn.Linear(channels, dim, bias=True)  # pointwise conv implemented by FC
        self.dwconv = nn.Conv2d(channels, channels, kernel_size=kernel_size, padding=padding, groups=channels, bias=True)  # depthwise conv
        self.act1 = act1()
        self.act2 = act2()
        self.pwconv2 = nn.Linear(channels, dim, bias=True)  # pointwise conv implemented by FC

    def forward(self, x):
        x = self.pwconv1(x)
        x = tf.identity(x)
        x = tf.permute(x, (0, 3, 1, 2)) # [B, D, H, W] -> [B, D, H, W]
        x = self.dwconv(x)
        x = tf.permute(x, (0, 3, 1, 2)) # [B, D, H, W] -> [B, D, H, W]
        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.

[17] 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.

[18] 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.

[19] Y. Yao, 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.

[20] 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,” in Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 10 183–10 192.

[21] Y. Li, X. Jiang, Y. Lu, X. Jin, Q. Hou, and J. Feng, “Refiner: Refining self-attention for vision transformers,” arXiv preprint arXiv:2106.03714, 2021.

[22] 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.

[23] Y. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ortonano, “Fnet: Mixing tokens with fourier transforms,” arXiv preprint arXiv:2105.03824, 2021.

[24] 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.

[25] 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.

[26] 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,” in Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 10 183–10 192.

[27] Y. Li, X. Jiang, Y. Lu, X. Jin, Q. Hou, and J. Feng, “Refiner: Refining self-attention for vision transformers,” arXiv preprint arXiv:2106.03714, 2021.

[28] 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.

[29] Y. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ortonano, “Fnet: Mixing tokens with fourier transforms,” arXiv preprint arXiv:2105.03824, 2021.
