Mobile-Former: Bridging MobileNet and Transformer

Yinpeng Chen¹ Xiyang Dai¹ Dongdong Chen¹ Mengchen Liu¹ Xiaoyi Dong²
Lu Yuan¹ Zicheng Liu¹

¹ Microsoft ² University of Science and Technology of China
{yiche,xidai,dochen,mengcliu,luyuan,zliu}@microsoft.com, dlight@mail.ustc.edu.cn

Abstract

We present Mobile-Former, a parallel design of MobileNet and transformer with a two-way bridge in between. This structure leverages the advantages of MobileNet at local processing and transformer at global interaction. And the bridge enables bidirectional fusion of local and global features. Different from recent works on vision transformer, the transformer in Mobile-Former contains very few tokens (e.g., 6 or fewer tokens) that are randomly initialized to learn global priors, resulting in low computational cost. Combining with the proposed light-weight cross attention to model the bridge, Mobile-Former is not only computationally efficient, but also has more representation power. It outperforms MobileNetV3 at low FLOP regime from 25M to 500M FLOPs on ImageNet classification. For instance, Mobile-Former achieves 77.9% top-1 accuracy at 294M FLOPs, gaining 1.3% over MobileNetV3 but saving 17% of computations. When transferring to object detection, Mobile-Former outperforms MobileNetV3 by 8.6 AP in RetinaNet framework. Furthermore, we build an efficient end-to-end detector by replacing backbone, encoder and decoder in DETR with Mobile-Former, which outperforms DETR by 1.1 AP but saves 52% of computational cost and 36% of parameters.

1. Introduction

Recently, vision transformer (ViT) [9, 32] demonstrates the advantage of global processing and achieves significant performance boost over CNNs. However, when constraining the computational budget within 1G FLOPs, the gain of ViT diminishes. If we further challenge the computational cost, MobileNet [15, 16, 26] and its extensions [11, 18] still dominate their backyard (e.g. fewer than 300M FLOPs for ImageNet classification) due to their efficiency in local processing filters via decomposition of depthwise and pointwise convolution. This in turn naturally raises a question: How to design efficient networks to effectively encode both local processing and global interaction?

A straightforward idea is to combine convolution and vision transformer. Recent works [10, 39, 40] show the benefit of combining convolution and vision transformer in series, either using convolution at the beginning or intertwining convolution into each transformer block.

In this paper, we shift the design paradigm from series to parallel, and propose a new network that parallelizes MobileNet and transformer with a two-way bridge in between (see Figure 1). We name it Mobile-Former, where Mobile refers to MobileNet and Former (refers to transformer) communicate through a bidirectional bridge, which is modeled by the proposed light-weight cross attention. Best viewed in color.
portantly, Mobile-Former consistently outperforms both ef-

cient CNNs and vision transformers from 25M to 500M FLOPs (see Figure 2), showcasing the usage of transformer at the low FLOP regime where efficient CNNs dominate.

When transferring from image classification to object detection, Mobile-Former significantly outperforms MobileNetV3 as backbone in RetinaNet [20], gaining 8.6 AP (35.8 vs. 27.2 AP) with even less computational cost. In addition, we build an efficient end-to-end detector by using Mobile-Former to replace backbone and encoder/decoder in DETR [1]. Using the same number of object queries (100), it gains 1.1 AP over DETR (43.1 vs. 42.0 AP) but has significantly fewer FLOPs (41G vs. 86G) and smaller model size (26.6M vs. 41.3M).

Finally we note that exploring the optimal network parameters (e.g. width, height) in Mobile-Former is not a goal of this work, rather we demonstrate that the parallel design provides an efficient and effective network architecture.

2. Related Work

Light-weight convolutional neural networks (CNNs): MobileNets [15, 16, 26] efficiently encode local features by stacking depthwise and pointwise convolutions. ShuffleNet [25, 46] uses group convolution and channel shuffle to simplify pointwise convolution. MicroNet [18] presents micro-factorized convolution to handle extremely low FLOPs by lowering node connectivity to enlarge network width. Dynamic operators [3, 4, 17, 43] have been studied to boost performance for MobileNet with negligible computational cost. Other efficient operators include butterfly transform [34], cheap linear transformations in GhostNet [11], and using additions to trade multiplications in AdderNet [2]. In addition, MixConv [29] explores mixing up multiple kernel sizes, and Sandglass [48] flips the structure of inverted residual block. EfficientNet [28, 30] and TinyNet [12] study the compound scaling of depth, width and resolution.

Vision transformers (ViTs): Recently, ViT [9] and its follow-ups [8, 22, 32, 35, 44] achieve impressive performance on multiple vision tasks. The original ViT requires training on large dataset such as JFT-300M. DeiT [32] introduces training strategies on the smaller ImageNet-1K dataset. Later, hierarchical transformers are proposed to handle high resolution images. Swin [22] computes self-attention within shifted local windows and CSWin [8] further improves it by introducing cross-shaped window. T2T-ViT [44] progressively converts an image to tokens by recursively aggregating neighboring tokens. HaloNet [35] improves speed, memory usage and accuracy by two extensions (blocked local attention and attention downsampling).

Combination of CNN and ViT: Recent works [6, 10, 27, 39, 40] show advantages of combining convolution and transformer. BoTNet [27] improves both instance segmentation and object detection by just using self-attention in the last
three blocks of ResNet [13]. ConViT [6] presents a gated positional self-attention for soft convolutional inductive biases. CvT [39] introduces depthwise/pointwise convolution before multi-head attention. LeViT [10] and ViT [40] use convolutional stem to replace the patchify stem and achieve clear improvement. In this paper, we propose a different design that parallelizes MobileNet and transformer with bidirectional cross attention in between. Our approach is efficient and effective, outperforming both efficient CNNs and ViT variants at low FLOP regime.

3. Our Method: Mobile-Former

In this section, we first overview the design of Mobile-Former (see Figure 1), and then discuss details within a Mobile-Former block (see Figure 3). Finally, we show the network specification and variants with different FLOPs.

3.1. Overview

Parallel structure: Mobile-Former parallelizes MobileNet and transformer, and connects them by bidirectional cross attention (see Figure 1). Mobile (refers to MobileNet) takes an image as input ($X \in \mathbb{R}^{HW \times 3}$) and applies inverted bottleneck blocks [26] to extract local features. Former (refers to transformer) takes learnable parameters (or tokens) as in-tleneck blocks [26] to extract local features. Former splits for the query is kept at $M \leq 6$ in this paper), each represents a global prior of the image. This results in much less computational cost.

Low cost two-way bridge: Mobile and Former communicate through a two-way bridge where local and global features are fused bidirectionally. The two directions are denoted as Mobile→Former and Mobile←Former, respectively. We propose a light-weight cross attention to model them, in which the projections ($W$) are randomly initialized. Different from vision transformer (ViT) [9], where tokens project the local image patch linearly, Former has significantly fewer tokens ($M \leq 6$ in this paper), each represents a global prior of the image. This results in much less computational cost.

\begin{align}
A_{X \rightarrow Z} &= \left[\text{Attn}(\tilde{x}_i, W_{i, K}^O, \tilde{z}_i, W_{i, V}^O)\right]_{i=1:h} W^O, \tag{1}
\end{align}

where the local feature $X$ and global tokens $Z$ are split into $h$ heads as $X = [\tilde{x}_1 \cdots \tilde{x}_h], Z = [\tilde{z}_1 \cdots \tilde{z}_h]$ for multi-head attention. The split for the $i$th head $\tilde{z}_i \in \mathbb{R}^{M \times d}$ is different to the $i$th token $z_i \in \mathbb{R}^d$. $W_{i, K}^O$ is the query projection matrix for the $i$th head. $W^O$ is used to combine multiple heads together. $\text{Attn}(Q, K, V)$ is the standard attention function [36] over query $Q$, key $K$, and value $V$.

3.2. Mobile-Former Block

Mobile-Former consists of stacked Mobile-Former blocks (see Figure 1). Each block has four pillars: a Mobile sub-block, a Former sub-block, and two-way cross attention Mobile→Former and Mobile←Former (shown in Figure 3).

Input and output: Mobile-Former block has two inputs: (a) local feature map $X \in \mathbb{R}^{HW \times C}$, which has $C$ channels over height $H$ and width $W$, and (b) global tokens $Z \in \mathbb{R}^{M \times d}$, where $M$ and $d$ are the number and dimension of tokens, respectively. Note that $M$ and $d$ are identical across all blocks. Mobile-Former block outputs the updated local feature map $X'$ and global tokens $Z'$, which are used as input for the next block.
**Mobile sub-block:** As shown in Figure 3, Mobile sub-block takes the feature map $X$ as input and its output is taken as the input for Mobile$\rightarrow$Former. It is slightly different to the inverted bottleneck block in [26] by replacing ReLU with dynamic ReLU [4] as the activation function. Different from the original dynamic ReLU, in which the parameters are generated by applying two MLP layers on the average pooled feature, we save the average pooling by applying the two MLP layers ($\theta$ in Figure 3) on the first global token output $z_1$ from Former. Note that the kernel size of depthwise convolution is 3×3 for all blocks.

**Former sub-block:** Former sub-block is a standard transformer block including a multi-head attention (MHA) and a feed-forward network (FFN). Expansion ratio 2 (instead of 4) is used in FFN. We follow [36] to use post layer normalization. Former is processed between Mobile$\rightarrow$Former and Mobile$\leftarrow$Former (see Figure 3).

**Mobile$\rightarrow$Former:** The proposed light-weight cross attention (Equation 1) is used to fuse local features $X$ to global tokens $Z$. Compared to the standard attention, the projection matrices for key $W^K$ and value $W^V$ (on the local features $X$) are removed to save computations (see Figure 3).

**Mobile$\leftarrow$Former:** Here, the cross attention (Equation 2) is on the opposite direction to Mobile$\rightarrow$Former. It fuses global tokens to local features. The local features are the query and global tokens are the key and value. Therefore, we keep the projection matrices for the key $W^K$ and value $W^V$, but remove the projection matrix for the query $W^Q$ to save computations, as shown in Figure 3.

**Computational complexity:** The four pillars of a Mobile-Former block have different computational costs. Given an input feature map with size $HW \times C$, and $M$ global tokens with $d$ dimensions, Mobile consumes the most computations $O(HWC^2)$. Former and the two-way bridge have light weight, consuming less than 20% of the total computational cost. Specifically, Former has complexity $O(M^2d + Md^2)$ for self attention and FFN. Mobile$\rightarrow$Former and Mobile$\leftarrow$Former share the complexity $O(MHW + MdC)$ for cross attention.

### 3.3. Network Specification

**Architecture:** Table 1 shows a Mobile-Former architecture with 294M FLOPs for image size 224×224, which stacks 11 Mobile-Former blocks at different input resolutions. All blocks have six global tokens with dimension 192. It starts with a 3×3 convolution as stem and a lite bottleneck block [18] at stage 1, which expands and then squeezes the number of channels by stacking a 3×3 depthwise and a pointwise convolution. Stage 2–5 consists of Mobile-Former blocks. Each stage handles downsampling by a downsampling variant denoted as Mobile-Former$^+$ (see appendix A for details). The classification head applies average pooling on the local features, concatenates with the first global token, and then passes through two fully connected layers with h-swish [15] in between.

**Mobile-Former variants:** Mobile-Former has seven models with different computational costs from 26M to 508M FLOPs. They share the similar architecture, but have different width and height. We follow [40] to refer our models by their FLOPs, e.g. Mobile-Former-294M, Mobile-Former-96M. The details of network architecture for these models are listed in appendix A (see Table 11).

### 4. Efficient End-to-End Object Detection

Mobile-Former can be easily used for object detection in both backbone and head, providing an efficient end-to-end detector. Using the same number of object queries (100), it outperforms DETR [1], but uses much lower FLOPs.

** Backbone–Head architecture:** We use Mobile-Former blocks in both backbone and head (see Figure 4), which have separate tokens. The backbone has six global tokens while the head has 100 object queries generated similarly to DETR [1]. Different from DETR that has a single scale ($\frac{1}{12}$ or $\frac{1}{10}$) in the head, Mobile-Former head employs multiscales ($\frac{1}{12}$, $\frac{1}{10}$, $\frac{1}{8}$) in low FLOPs due to its computational efficiency. The upsampling is achieved via bilinear interpolation followed by adding the feature output from the backbone (with the same resolution). All object queries progressively refine their representation across scales from coarse to fine, saving the manual process in FPN [19] to allocate objects across scales by size. We follow DETR to use prediction FFNs and auxiliary losses in the head during the training. The head is trained from scratch and the backbone

| Stage | Input | Operator | exp size | #out | Stride |
|-------|-------|----------|----------|------|--------|
| stem  | 224×3 | conv2d, 3×3 | – | 16 | 2 |
| 1     | 112×16 | bneck-lite | 32 | 16 | 1 |
| 2     | 112×16 | Mobile-Former$^+$ | 96 | 24 | 2 |
| 3     | 128×24 | Mobile-Former$^+$ | 144 | 48 | 2 |
| 4     | 28×48  | Mobile-Former$^+$ | 288 | 96 | 2 |
| 5     | 14×128 | Mobile-Former$^+$ | 384 | 96 | 1 |
| 6     | 14×128 | Mobile-Former$^+$ | 576 | 128 | 1 |
| 7     | 14×128 | Mobile-Former$^+$ | 768 | 128 | 1 |

** Table 1. Specification for Mobile-Former-294M.** “bneck-lite” denotes the lite bottleneck block [18]. “Mobile-Former$^+$” denotes the variant of downsample block.
All object queries pass through multiple resolutions (bone has 6 global tokens while the head has 100 object queries. and head use Mobile-Former blocks (see Figure 1, 3). The back-
Mobile-Former for object detection
Figure 4.
spatial-shared
dynamic ReLU in the backbone from
Spatial-aware dynamic ReLU in backbone:
We extend
head structure is listed in appendix A (see Table 13).
forms DETR by 1.1 AP (43.1 vs. 42.0 AP). The details of
nine Mobile-Former blocks in the head is 41.4G FLOPs,
MF-508M that uses Mobile-Former-508M as backbone and
detector is computationally efficient. The total cost of E2E-
is pretrained on ImageNet. Our end-to-end Mobile-Former
detector is computationaly efficient. The total cost of E2E-
5. Experimental Results
We evaluate the proposed Mobile-Former on both ImageNet classification [7], and COCO object detection [21].
5.1. ImageNet Classification
Image classification experiments are conducted on ImageNet [7] that has 1000 classes, including 1,281,167 images for training and 50,000 images for validation.
Training setup: The image resolution is 224×224. All models are trained from scratch using AdamW [23] optimizer for 450 epochs with cosine learning rate decay. A batch size of 1024 is used. Data augmentation includes Mixup [45], auto-augmentation [5], and random erasing [47]. Different combinations of initial learning rate, weight decay and dropout are used for models with different complexities, which are listed in appendix A (see Table 12).
Comparison with efficient CNNs: Table 2 shows the comparison between Mobile-Former and classic efficient CNNs: (a) MobileNetV3 [15], (b) EfficientNet [28], and (c) ShuffleNetV2 [25] and its extension WeightNet [24]. The comparison covers the FLOP range from 26M to 508M, organized in seven groups based on similar FLOPs. Mobile-Former consistently outperforms efficient CNNs with even less computational cost except the group around 150M FLOPs, where Mobile-Former costs slightly more FLOPs than ShuffleNet/WeightNet (151M vs. 138M/141M), but achieves significantly higher top-1 accuracy (75.2% vs. 69.1%/72.4%). This demonstrates that our parallel design improves the representation capability efficiently.
Comparison with ViTs: In Table 3, we compare Mobile-Former with multiple variants (DeiT [32], T2T-ViT [44], PVT [37], ConViT [6], CoaT [41], ViT-D [40], Swin [22]) of vision transformer. All variants use image resolution 224×224 and are trained without distillation from a teacher
choose ViT variants that use image resolution 224 and are evaluated on ImageNet [7] classification. Here, we evaluate on ImageNet [7] classification. Mobile-Former-294M is used.

Table 2. Comparing Mobile-Former with efficient CNNs evaluated on ImageNet [7] classification.

| Model                  | Input #Params MAdds | Top-1 |
|------------------------|---------------------|-------|
| MobileNetV3 Small 1.0× [15] | 224× | 160² | 2.5M | 57M | 67.5 |
| Mobile-Former-26M      | 224× | 3.2M | 26M | 64.0 |
| MobileNetV3 Small 1.0× [15] | 224× | 74.5 | 76.6 |
| Mobile-Former-52M      | 224× | 3.5M | 52M | 68.7 |
| MobileNetV3 1.0× [15]  | 224× | 5.4M | 112M | 77.6 |
| Mobile-Former-96M      | 224× | 4.6M | 96M | 72.8 |
| ShuffleNetV2 1.0× [25] | 224× | 2.2M | 138M | 69.1 |
| ShuffleNetV2 1.0×+WeightNet 4x [24] | 224× | 5.1M | 141M | 72.4 |
| MobileNetV3 0.75× [15] | 224× | 4.0M | 155M | 73.3 |
| Mobile-Former-151M     | 224× | 7.6M | 151M | 75.2 |
| MobileNetV3 1.0× [15]  | 224× | 5.4M | 217M | 75.2 |
| Mobile-Former-214M     | 224× | 9.4M | 214M | 76.7 |
| ShuffleNetV2 1.5× [25] | 224× | 3.5M | 299M | 72.6 |
| ShuffleNetV2 1.5×+WeightNet 4× [24] | 224× | 9.6M | 307M | 75.0 |
| MobileNetV3 1.25× [15] | 224× | 7.5M | 356M | 76.6 |
| EfficientNet-B0 [28]   | 224× | 5.3M | 390M | 77.1 |
| Mobile-Former-294M     | 224× | 11.4M | 294M | 77.9 |
| ShuffleNetV2 2× [25]   | 224× | 5.5M | 557M | 74.5 |
| ShuffleNetV2 2×+WeightNet 4× [24] | 224× | 18.1M | 573M | 76.5 |
| Mobile-Former-508M     | 224× | 14.0M | 508M | 79.3 |

Table 3. Comparing Mobile-Former with vision transformer variants evaluated on ImageNet [7] classification. Here, we choose ViT variants that use image resolution 224×224 and are trained without distillation from a teacher network. We group ViT models based on FLOPs (using 1.5G as threshold) and rank them based on top-1 accuracy. † indicates our implementation.

| Model                  | Input #Params MAdds | Top-1 |
|------------------------|---------------------|-------|
| T2T-ViT-7 [44]         | 224² | 4.3M | 1.2G | 71.7 |
| DeiT-Tiny [32]         | 224² | 5.7M | 1.2G | 72.2 |
| ConViT-Tiny [6]        | 224² | 6.0M | 1.0G | 73.4 |
| ConViT-Ti [42]         | 224² | 5.8M | 0.8G | 74.9 |
| ViT-C [40]             | 224² | 4.6M | 1.1G | 75.3 |
| ConViT-S [42]          | 224² | 10.1M | 1.5G | 76.5 |
| Swin-1G [22]†          | 224² | 7.3M | 1.0G | 77.3 |
| Mobile-Former-294M     | 224² | 11.4M | 294M | 77.9 |
| PVT-Tiny [37]          | 224² | 13.2M | 1.9G | 75.1 |
| T2T-ViT-12 [44]        | 224² | 6.9M | 2.2G | 76.5 |
| ConViT-Lite Tiny [41]  | 224² | 5.7M | 1.6G | 76.6 |
| ConViT-Lite+ [6]       | 224² | 10.0M | 2G | 76.7 |
| DeiT-2G [32]†         | 224² | 9.5M | 2.0G | 77.6 |
| ConViT-Lite Mini [41]  | 224² | 11.0M | 2.0G | 78.9 |
| BoT-S1-50 [27]         | 224² | 20.8M | 4.3G | 79.1 |
| Swin-2G [22]†          | 224² | 12.8M | 2.0G | 79.2 |
| Mobile-Former-508M     | 224² | 14.0M | 508M | 79.3 |

Table 4. Ablation ofFormer+bridge and dynamic ReLU evaluated on ImageNet classification. Mobile-Former-294M is used.

| Model                  | #Params MAdds | Top-1 | Top-5 |
|------------------------|---------------|-------|-------|
| Mobile (using ReLU)    | 6.1M | 259M | 74.2 | 91.8 |
| + Former and Bridge    | 10.1M | 290M | 76.8 (±2.6) | 93.2 (±1.4) |
| + DY-ReLU in Mobile    | 11.4M | 294M | 77.8 (±1.0) | 93.7 (±0.5) |

fewer FLOPs (294M vs. 305M) than LeViT.

**Accuracy–FLOP plot:** Figure 2 compares Mobile-Former with more CNN models (e.g. GhostNet [11]) and vision transformer variants (e.g. Swin [22] and DeiT [32]) in one plot. We implement Swin and DeiT from 100M to 2G FLOPs, by carefully reducing network width and height. Mobile-Former clearly outperforms both CNNs and ViT variants, showcasing the parallel design to integrate MobileNet and transformer. Although vision transformers are inferior to efficient CNNs by a large margin, our work demonstrates that the transformer can also contribute to the low FLOP regime with proper architecture design.

5.2. Ablations

In this subsection, we show Mobile-Former is effective and efficient via several ablations performed on ImageNet classification. Here, Mobile-Former-294M is used and all models are trained for 300 epochs. Moreover, we summarize interesting observations on the visualization of the two-way bridge between Mobile and Former.

**Mobile-Former is effective:** Mobile-Former is more effective than MobileNet as it encodes global interaction via Former, resulting in more accurate prediction. As shown in Table 4, adding Former and bridge (Mobile→Former and Mobile→Former) only costs 12% of the computational cost, but gains 2.6% top-1 accuracy over the baseline that uses Mobile alone. In addition, using dynamic ReLU [4] in Mobile sub-block (see Figure 3) gains additional 1.0% top-1 accuracy. Note that the parameters in dynamic ReLU [4] in Mobile are used to generate the first global token. This validates our parallel design in Mobile-Former. We also find that increasing kernel size (3×3 → 5×5) of the depthwise convolution in Mobile only introduces negligible gain (see Table 14 in appendix B), as the reception field of Mobile is enlarged by fusing global features from Former.

**Mobile-Former is efficient:** Mobile-Former is not only effective in encoding both local processing and global interaction, but achieves this efficiently. Ablations below show that Former only requires a few global tokens with low dimension. In addition, the efficient parallel design of Mobile-Former is stable when removing FFN in Former or replacing multi-head attention with position mixing MLP [31].

**Number of tokens in Former:** Table 5 shows the ImageNet classification results for using different number of global tokens. Mobile-Former achieves higher accuracy but uses 3~4 times less computational cost. This is because that Mobile-Former uses significantly fewer tokens to model global interaction and leverages MobileNet to extract local features efficiently. Note that our Mobile-Former (trained in 450 epochs without distillation) even outperforms LeViT [10] which leverages the distillation of a teacher network and much longer training (1000 epochs). Our method achieves higher top-1 accuracy (77.9% vs. 76.6%) but uses fewer FLOPs (294M vs. 305M) than LeViT.
tokens in Former. The token dimension is 192. Interestingly, even a single global token achieves a good performance (77.1% top-1 accuracy). Additional improvement (0.5% and 0.7% top-1 accuracy) is achieved when using 3 and 6 tokens. The improvement stops when more than 6 tokens are used. Such compactness of global tokens is a key contributor to the efficiency of Mobile-Former.

Token dimension: Table 6 shows the results for different token lengths (or dimensions). Here, six global tokens are used in Former. The accuracy improves from 76.8% to 77.8% when token dimension increases from 64 to 192, but converges when higher dimension is used. This further supports the efficiency of Former. With six tokens of dimension 192, the total computational cost of Former and the bridge only consumes 12% of the overall budget (35M/294M).

FFN in Former: As shown in Table 7, removing FFN introduces a small drop in top-1 accuracy (−0.3%). Compared to the important role of FFN in the original vision transformer, FFN has limited contribution in Mobile-Former. We believe this is because FFN is not the only module for channel fusion in Mobile-Former. The 1 × 1 convolution in Mobile helps the channel fusion of local features, while the projection matrix W'O in Mobile→Former (see Equation 1) contributes to the fusion between local and global features.

Multi-head attention (MHA) vs. position-mixing MLP: Table 7 shows the results of replacing multi-head attention (MHA) with token/position mixing MLP [31] in both Former and bridge (Mobile→Former and Mobile←Former).

The top-1 accuracy drops from 77.8% to 77.3%. The implementation of MLP is more efficient by a single matrix multiplication, but it is static (i.e. not adaptive to different input images).

Mobile-Former is explainable: We observe three interesting patterns in the two-way bridge (Mobile→Former and Mobile←Former). First, from low to high levels, the global tokens change their focus from edges/corners, to foreground/background, and finally on the most discriminative region. Second, the cross attention has more diversity across tokens at lower levels than high levels. Thirdly, the separation between foreground and background is found at middle layers of Mobile←Former. The detailed visualization is shown in appendix C.

5.3. Object Detection

Object detection experiments are conducted on COCO 2017 [21], which contains 118K training and 5K validation images. We evaluate Mobile-Former in two detection frameworks: (a) comparing with other backbone networks in RetinaNet [20] framework that has dense proposals, and (b) end-to-end comparison with DETR [1] where sparse proposals are used.

RetinaNet training setup: We follow the standard settings of RetinaNet [20] and replace the backbone with our Mobile-Former to generate multi-scale feature maps. All models are trained for 12 epochs (1×) from ImageNet pretrained weights.

DETR training setup: All Mobile-Former models are trained for 300 epochs on 8 GPUs with 2 images per GPU. AdamW optimizer is used with initial learning rate 1e-5 for the backbone and 1e-4 for the head. The learning rate drops by a factor of 10 after 200 epochs. The weight decay is 1e-4 and dropout rate is 0.1. BatchNorm layers in the ImageNet pretrained Mobile-Former backbone are frozen. The head includes 100 object queries with 256 channels.
This paper presents Mobile-Former, a new parallel design of MobileNet and Transformer with two-way bridge in between to communicate. It leverages the efficiency of MobileNet in local processing and the advantage of Transformer in encoding global interaction. This design is not only effective to boost accuracy, but also efficient to save computations, but results in a 1.2 AP drop. Then using multi-scale Mobile-Former head to replace DETR’s encoder/decoder gains another 0.9 AP (40.5 → 41.4). The detection of small and medium objects is improved, while a slight degradation is found in large objects. Finally, adapting position embedding provides additional 1.7 AP gain. The three proposed components are complementary, gaining 3.7 AP (39.4 → 43.1) in total.

Efficient and effective backbone in RetinaNet: In Table 8, we compare Mobile-Former with both CNNs (ResNet [13], MobileNetV3 [15], ShuffleNetV2 [25]) and vision transformers (PVT [37] and ConT [42]). Mobile-Former significantly outperforms MobileNetV3 and ShuffleNetV2 by 8.3+ AP under similar computational cost. Compared to ResNet and transformer variants, Mobile-Former achieves higher AP with significantly less FLOPs in the backbone. Specifically, Mobile-Former-508M only takes 9.8G FLOPs in backbone but achieves 38.0 AP, outperforming ResNet-50, PVT-Tiny, and ConT-M which consume 7 times more computation (65G to 84G FLOPs) in the backbone. This showcases Mobile-Former as an effective and efficient backbone in object detection.

Efficient and effective end-to-end detector: Table 9 compares end-to-end Mobile-Former detectors (denoted by prefix E2E-MF) with DETR [1]. All models use 100 object queries. Our E2E-MF-508M gains 1.1 AP over DETR but consumes fewer FLOPs (48%), fewer model parameters (64%) and fewer training epochs (300 vs. 500). It is just a little behind DETR-DC5 that has four times more FLOPs. The other three Mobile-Former variants achieve 40.5, 39.3 and 37.2 AP with 24.1G, 17.8G and 12.7G FLOPs respectively, providing more compact end-to-end detectors.

Ablations of key components: Table 10 shows the effects of three proposed components in the end-to-end Mobile-Former detector: (a) spatial-aware dynamic ReLU in backbone, (b) multi-scale Mobile-Former head, and (c) adapting position embedding. E2E-MF-508M is used and all models are trained for 300 epochs. Compared to DETR trained in 300 epochs, replacing ResNet-50 backbone with MF-508M saves computations, but results in 1.2 AP drop. Adding spatial-aware dynamic ReLU gains 1.1 AP (39.4 → 40.5). Then using multi-scale Mobile-Former head to replace DETR’s encoder/decoder gains another 0.9 AP (40.5 → 41.4). The detection of small and medium objects is improved, while a slight degradation is found in large objects. Finally, adapting position embedding provides additional 1.7 AP gain. The three proposed components are
computational cost. It outperforms both efficient CNNs and vision transformer variants with a clear margin on image classification and object detection in the low FLOP regime. Furthermore, we build an end-to-end Mobile-Former detector that outperforms DETR but consumes significantly less computations and parameters. We hope Mobile-Former encourage new design of efficient CNNs and transformers.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In ECCV, 2020.

[2] Hanming Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[3] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[4] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. In ECCV, 2020.

[5] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

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

[7] 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.

[8] Xiaoai 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. arXiv preprint arXiv:2107.00652, 2021.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhou, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021.

[10] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet’s clothing for faster inference. arXiv preprint arXiv:2210.04136, 2021.

[11] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[12] Kai Han, Yunhe Wang, Qilin Zhang, Wei Zhang, Chunjing Xu, and Tong Zhang. Model rubiks cube: Twisting resolution, depth and width for tinynets. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 19353–19364. Curran Associates, Inc., 2020.

[13] 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.

[14] Geoffrey E. Hinton. How to represent part-whole hierarchies in a neural network. CoRR, abs/2102.12627, 2021.

[15] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[18] Yunsong Li, Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Lu Yuan, Zicheng Liu, Lei Zhang, and Nuno Vasconcelos. Micronet: Improving image recognition with extremely low flops. In International Conference on Computer Vision, 2021.

[19] T. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 939–947, July 2017.

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017.

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

[22] 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. arXiv preprint arXiv:2103.14030, 2021.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In International Conference on Learning Representations, 2019.

[24] Ningning Ma, X. Zhang, J. Huang, and J. Sun. Weightnet: Revisiting the design space of weight networks. volume abs/2007.11823, 2020.
Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In The European Conference on Computer Vision (ECCV), September 2018. 2, 5, 6, 7, 8

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chi Chen. Mobilenet-v2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4510–4520, 2018. 1, 2, 3, 4, 11, 12

Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 16519–16529, June 2021. 2, 6

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In ICLR, pages 6105–6114, Long Beach, California, USA, 09–15 Jun 2019. 2, 5, 6

Mingxing Tan and Quoc V. Le. Mixconv: Mixed depthwise convolutional kernels. In 30th British Machine Vision Conference 2019. 2019. 2

Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020. 2

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021. 6, 7

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jéjou. Training data-efficient image transformers and distillation through attention. arXiv preprint arXiv:2012.12877, 2020. 1, 2, 5, 6

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jéjou. Going deeper with image transformers, 2021. 13

Keivan Alizadeh vahid, Anish Prabhu, Ali Farhadi, and Mohammad Rastegari. Butterfly transform: An efficient fft based neural architecture design. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020. 2

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. 1, 3, 4

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramidal vision transformer: A versatile backbone for dense prediction without convolutions, 2021. 5, 6, 7, 8

Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 11

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing convolutions to vision transformers, 2021. 1, 2, 3

Tete Xiao, Mannat Singh, Eric Minton, Trevor Darrell, Piotr Dollár, and Ross B. Girshick. Early convolutions help transformers see better. CoRR, abs/2106.14881, 2021. 1, 2, 3, 4, 5, 6

Wenjia Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Coscale conv-attentional image transformers, 2021. 5, 6

Haotian Yan, Zhe Li, Weijian Li, Changhu Wang, Ming Wu, and Chuang Zhang. Contnet: Why not use convolution and transformer at the same time? CoRR, abs/2104.13497, 2021. 6, 7, 8

Brandon Yang, Gabriel Bender, Quoc V. Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In NeurIPS, 2019. 2

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. arXiv preprint arXiv:2101.11986, 2021. 2, 5, 6

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In International Conference on Learning Representations, 2018. 5

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018. 2

Zhou Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2020. 5

Daquan Zou, Qi-Bin Hou, Y. Chen, Jiashi Feng, and S. Yan. Rethinking bottleneck structure for efficient mobile network design. In ECCV, August 2020. 2

Daquan Zou, Bingyi Kang, Xiaoje Jia, Linjie Yang, Xiaochen Lian, Zhihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer, 2021. 13

Daquan Zou, Yujun Shi, Bingyi Kang, Weihao Yu, Zhihang Jiang, Yuan Li, Xiaoje Jia, Qibin Hou, and Jiashi Feng. Refiner: Refining self-attention for vision transformers, 2021. 13
A. Mobile-Former Architecture

Seven model variants: Table 11 shows six Mobile-Former models (508M–52M). The smallest model Mobile-Former-26M has similar architecture to Mobile-Former-52M except replacing all 1 × 1 convolutions with group convolution (group=4). They are used either in image classification or as the backbone of object detectors. These models are manually designed without searching for the optimal architecture parameters (e.g. width or depth). We follow the well known rules used in MobileNet [15, 26]: (a) the number of channels increases across stages, and (b) the channel expansion rate starts from three at low levels and increases to six at high levels. For the four bigger models (508M–151M), we use six global tokens with dimension 192 and eleven Mobile-Former blocks. But these four models have different widths. Mobile-Former-96M and Mobile-Former-52M are shallower (with only eight Mobile-Former blocks) to meet the low computational budget.

Downsample Mobile-Former block: Note that stage 2–5 has a downsample variant of Mobile-Former block (denoted as M-Fi in Table 11) to handle the spatial downsampling. M-Fi has a slightly different Mobile sub-block that includes four (instead of three) convolutional layers (depthwise→pointwise→depthwise→pointwise), where the first depthwise convolution layer has stride two. The number of channels expands in each depthwise convolution, and squeezes in the following pointwise convolution. This saves computations as the two costly pointwise convolutions are performed at the lower resolution after downsampling.

Training hyper-parameters: Table 12 lists three hyper-parameters (initial learning rate, weight decay and dropout rate) used for training Mobile-Former models in ImageNet classification. Their values increase as the model becomes bigger to prevent overfitting. Our implementation is based on timm framework [38].

Head model variants in end-to-end object detection: Table 13 shows the head structures for four end-to-end Mobile-Former detectors. All share similar structure and have 100 object queries with dimension 256. The largest model (E2E-MF-508M) has the heaviest head with 9 Mobile-Former blocks over three scales, while the other three smaller models have 9, 7, 5 blocks respectively over two scales to save computations.

All models start by projecting the input feature map linearly to 256 channels through a 1×1 convolution. Then multiple Mobile-Former blocks are stacked with upsampling block in between to move upscale. The upsampling block (denoted as “up-conv”) includes three steps: (a) increasing feature resolution by two using bilinear interpolation, (b) adding the feature output from the backbone, and (c) applying a 3 × 3 depthwise and a pointwise convolution. To handle the computational boost due to resolution increasing, we use lite bottleneck [18] in Mobile. Moreover, we find that the performance can be further improved at a small additional cost by adding multi-head attention in Mo-

| Stage | Mobile-Former-508M | Mobile-Former-294M | Mobile-Former-214M | Mobile-Former-151M | Mobile-Former-96M | Mobile-Former-52M |
|-------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
|       | Block #exp #out    | Block #exp #out    | Block #exp #out    | Block #exp #out    | Block #exp #out    | Block #exp #out    |
| token |                     |                     |                     |                     |                     |                     |
| stem  | conv 3×3 – 24       | conv 3×3 – 16       | conv 3×3 – 12       | conv 3×3 – 12       | conv 3×3 – 12       | conv 3×3 – 8       |
| 1     | bneck-lite 48 24    | bneck-lite 32 16    | bneck-lite 24 12    | bneck-lite 24 12    | bneck-lite 24 12    | bneck-lite 24 12    |
| 2     | M-Fi 144 40         | M-Fi 96 24          | M-Fi 72 20          | M-Fi 72 16          | M-Fi 72 16          | M-Fi 72 16          |
| M-F   | 120 40              | M-F 96 24           | M-F 60 20           | M-F 48 16           | M-F 48 16           | M-F 48 16           |
| 3     | M-Fi 240 72         | M-Fi 144 48         | M-Fi 120 40         | M-Fi 96 32          | M-Fi 96 32          | M-Fi 96 32          |
| M-F   | 216 72              | M-F 192 48          | M-F 160 40          | M-F 96 32           | M-F 96 32           | M-F 96 32           |
| 4     | M-Fi 432 128        | M-Fi 288 96         | M-Fi 240 80         | M-Fi 192 64         | M-Fi 192 64         | M-Fi 192 64         |
| M-F   | 512 128             | M-F 384 96          | M-F 320 80          | M-F 256 64          | M-F 256 64          | M-F 256 64          |
| M-F   | 768 176             | M-F 576 128         | M-F 480 112         | M-F 384 88          | M-F 384 88          | M-F 384 88          |
| M-F   | 1056 176            | M-F 768 128         | M-F 672 112         | M-F 528 88          | M-F 528 88          | M-F 528 88          |
| 5     | M-Fi 1056 240       | M-Fi 768 192        | M-Fi 672 160        | M-Fi 528 128        | M-Fi 528 128        | M-Fi 528 128        |
| M-F   | 1440 240            | M-F 1152 192        | M-F 960 160         | M-Fi 768 128        | M-Fi 768 128        | M-Fi 768 128        |
| M-F   | 1440 240            | M-F 1152 192        | M-Fi 960 160        | M-Fi 768 128        | M-Fi 768 128        | M-Fi 768 128        |
| conv 1×1 – 1440 | conv 1×1 – 1152 | conv 1×1 – 960 | conv 1×1 – 768 | conv 1×1 – 768 | conv 1×1 – 768 | conv 1×1 – 768 |
| pool  | concat – – 1632     | concat – – 1344     | – – 1152            | – – 960             | – – 896             | – – 704             |
|      | FC1 – – 1920        | FC1 – – 1920        | – – 1600            | – – 1280            | – – 1280            | – – 1024            |
|      | FC2 – – 1000        | FC2 – – 1000        | – – 1000            | – – 1000            | – – 1000            | – – 1000            |

Table 11. Specification of Mobile-Former models. “bneck-lite” denotes the lite bottleneck block [18]. “bneck-lite+” denotes the downsample variant of lite bottleneck, in which the depthwise convolution has stride 2. “M-F” denotes the Mobile-Former block and “M-Fi” denotes the Mobile-Former block for downsampling. Mobile-Former-26M has a similar architecture to Mobile-Former-52M except replacing all 1 × 1 convolutions with group convolution (group=4).
| Model              | Learing Rate | Weight Decay | Dropout |
|--------------------|--------------|--------------|---------|
| Mobile-Former-26M  | 8e-4         | 0.08         | 0.1     |
| Mobile-Former-52M  | 8e-4         | 0.10         | 0.2     |
| Mobile-Former-96M  | 8e-4         | 0.10         | 0.2     |
| Mobile-Former-151M | 9e-4         | 0.10         | 0.2     |
| Mobile-Former-214M | 9e-4         | 0.15         | 0.2     |
| Mobile-Former-294M | 1e-3         | 0.20         | 0.3     |
| Mobile-Former-508M | 1e-3         | 0.20         | 0.3     |

Table 12. Hyper-parameters of seven Mobile-Former models for ImageNet [7] classification.

| Stage | E2E-MF 508M | E2E-MF 294M | E2E-MF 214M | E2E-MF 151M |
|-------|-------------|-------------|-------------|-------------|
| query | 100×256     | 100×256     | 100×256     | 100×256     |
| 1/32  | projection  | projection  | projection  | projection  |
|       | ↑M-F ×5    | ↑M-F ×6    | ↑M-F ×5    | ↑M-F ×3    |
| 1/16  | up-conv    | up-conv    | up-conv    | up-conv    |
|       | M-F  ×2    | M-F  ×3    | M-F  ×2    | M-F  ×2    |
| 1/8   | up-conv    |           |           |           |
|       | M-F  ×2    |           |           |           |

Table 13. Specification of head variants in end-to-end Mobile-Former object detectors. 100 object queries with dimension 256 are used. “projection” denotes projecting an input feature map linearly to 256 channels (through a 1×1 convolution), “up-conv” denotes a convolutional block for upsampling that includes bilinear interpolation followed by a 3×3 deconvolution and a pointwise convolution. “M-F  ×2” refers to stacking two Mobile-Former blocks. In the detection head, we use lite bottleneck [18] in Mobile sub-block to reduce the computational cost. At the lowest resolution 1/32, multi-head attention is added into Mobile, which is denoted as ↑M-F.

B. More Experimental Results

Inference latency: Figure 5 compares between Mobile-Former-214M and MobileNetV3 Large [15] on inference latency, as they have similar FLOPs (214M vs. 217M). The latency is measured on an Intel(R) Xeon(R) CPU E5-2650 v3 (2.3GHz), following the common settings (single-thread with batch size 1) in [15, 26]. Mobile-Former-214M is compared with MobileNetV3 Large [15] as they have similar FLOPs (214M vs. 217M). Mobile-Former is slower when image size is small, but has faster inference than MobileNetV3 as image size grows above 750×750. Best viewed in color.

Table 14. Ablation of the kernel size in the depthwise convolution (in Mobile sub-block). The evaluation is performed on ImageNet [7] classification. Mobile-Former-294M is used.

Table 15. Ablation of multi-head attention (MHA) in Mobile at resolution 1/32 of the detection head. The evaluation is performed on COCO [21] object detection. Both models are trained on train2017 for 300 epochs and tested on val2017. E2E-MF-508M is used. MAdds is based on image size 800×1333.
Figure 6. **Cross attention over the entire featuremap for the first token in Mobile→Former across all Mobile-Former blocks.** Attention is normalized over pixels, showing the focused region. The focused region changes from low to high levels. The token starts paying more attention to edges/corners at block 2–4. Then it focuses more on a large region rather than scattered small pieces at block 5–12. The focused region shifts between the foreground (person and horse) and background (grass). Finally, it locks the most discriminative part (horse body and head) for classification. Best viewed in color.

Computational cost. It is especially helpful for detecting large objects (58.7→60.4 AP\textsubscript{L}).

C. Visualization

In order to understand the collaboration between Mobile and Former, we visualize the cross attention on the two-way bridge (i.e. Mobile→Former and Mobile←Former) in Figure 6, 7, and 8. The ImageNet pretrained Mobile-Former-294M is used, which includes six global tokens and eleven Mobile-Former blocks. We observe three interesting patterns as follows:

**Pattern 1 – global tokens shift focus over levels:** The focused regions of global tokens change progressively from low to high levels. Figure 6 shows the cross attention over pixels for the first token in Mobile→Former. This token begins focusing on local features, e.g. edges/corners (at block 2-4). Then it pays more attention to regions with connected pixels. Interestingly, the focused region shifts between foreground (person and horse) and background (grass) across blocks. Finally, it locates the most discriminative region (horse body and head) for classification.

**Pattern 2 – foreground and background are separated in middle layers:** The separation between foreground and background is surprisingly found in Mobile←Former at middle layers (e.g. block 8). Figure 7 shows the cross attention over six tokens for each pixel in the featuremap. Clearly, the foreground and background are separated in the first and last tokens. This shows that global tokens learn meaningful prototypes that cluster pixels with similar semantics.

**Pattern 3 – attention diversity across tokens diminishes:** The attention has more diversity across tokens at low levels than high levels. As shown in Figure 8, each column corresponds to a token, and each row corresponds to a head in the corresponding multi-head cross attention. Note that the attention is normalized over pixels in Mobile→Former (left half), showing the focused region per token. In contrast, the attention in Mobile←Former is normalized over tokens, comparing the contribution of different tokens at each pixel. Clearly, the six tokens at block 3 and 5 have different cross attention patterns in both Mobile→Former and Mobile←Former. Similar attention maps over tokens are clearly observed at block 8. At block 12, the last five tokens share a similar attention pattern. Note that the first token is the classification token fed into the classifier. The similar observation on token diversity has been identified in recent studies on ViT [33, 49, 50]. The full visualization of two-way cross attention for all blocks is shown in Figure 9.
Figure 8. **Visualization of the two-way cross attention:** Mobile→Former and Mobile←Former. Mobile-Former-294M is used, which includes six tokens (each corresponds to a column). Four blocks with different input resolutions are selected and each has two attention heads that are visualized in two rows. Attention in Mobile→Former (left half) is normalized over pixels, showing the focused region per token. Attention in Mobile←Former (right half) is normalized over tokens showing the contribution of different tokens at each pixel. The cross attention has less variation across tokens at high levels than low levels. Specifically, token 2–5 in the last block have very similar cross attention. Best viewed in color.
Figure 9. **Visualization of the two-way cross attention**: Mobile→Former and Mobile←Former. Mobile-Former-294M is used, which includes six tokens (each corresponds to a column) and eleven Mobile-Former blocks (block 2–12) across four stages. Each block has two attention heads that are visualized in two rows. Attention in Mobile→Former (left) is normalized over pixels, showing the focused region per token. Attention in Mobile←Former (right) is normalized over tokens showing the contribution of different tokens at each pixel.