Mobile-Former: Bridging MobileNet and Transformer (Supplementary Material)

Yinpeng Chen[1] Xiyang Dai[1] Dongdong Chen[1] Mengchen Liu[1] Xiaoyi Dong[2]
Lu Yuan[1] Zicheng Liu[1]

[1] Microsoft [2] University of Science and Technology of China

{yiche,xidai,dochen,mengcliu,luyuan,zliu}@microsoft.com, dlight@mail.ustc.edu.cn

In this supplementary material, we discuss (a) architecture details of Mobile-Former variants over multiple FLOPs, (b) more experimental results including inference latency and additional ablations, and (c) visualization of the two-way bridge (*Mobile→Former* and *Mobile←Former*).

1. Mobile-Former Architecture

Seven model variants: Table 1 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 [2, 6] : (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-F$^{\downarrow}$ in Table 1) to handle the spatial downsampling. M-F$^{\downarrow}$ has a slightly different *Mobile* sub-block

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	6×192			6×192			6×192			6×192			4×128			3×128		
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			
2	M-F$^{\downarrow}$	144	40	M-F$^{\downarrow}$	96	24	M-F$^{\downarrow}$	72	20	M-F$^{\downarrow}$	72	16	M-F$^{\downarrow}$	72	16	bneck-lite$^{\downarrow}$	24	12
	M-F	120	40	M-F	96	24	M-F	60	20	M-F	48	16				M-F	36	12
3	M-F$^{\downarrow}$	240	72	M-F$^{\downarrow}$	144	48	M-F$^{\downarrow}$	120	40	M-F$^{\downarrow}$	96	32	M-F$^{\downarrow}$	96	32	M-F$^{\downarrow}$	72	24
	M-F	216	72	M-F	192	48	M-F	160	40	M-F	96	32	M-F	96	32	M-F	72	24
4	M-F$^{\downarrow}$	432	128	M-F$^{\downarrow}$	288	96	M-F$^{\downarrow}$	240	80	M-F$^{\downarrow}$	192	64	M-F$^{\downarrow}$	192	64	M-F$^{\downarrow}$	144	48
	M-F	512	128	M-F	384	96	M-F	320	80	M-F	256	64	M-F	256	64	M-F	192	48
	M-F	768	176	M-F	576	128	M-F	480	112	M-F	384	88	M-F	384	88	M-F	288	64
	M-F	1056	176	M-F	768	128	M-F	672	112	M-F	528	88						
5	M-F$^{\downarrow}$	1056	240	M-F$^{\downarrow}$	768	192	M-F$^{\downarrow}$	672	160	M-F$^{\downarrow}$	528	128	M-F$^{\downarrow}$	528	128	M-F$^{\downarrow}$	384	96
	M-F	1440	240	M-F	1152	192	M-F	960	160	M-F	768	128	M-F	768	128	M-F	576	96
	M-F	1440	240	M-F	1152	192	M-F	960	160	M-F	768	128	conv 1×1	–	768	conv 1×1	–	576
	conv 1×1	–	1440	conv 1×1	–	1152	conv 1×1	–	960	conv 1×1	–	768						
pool concat	–	–	1632	–	–	1344	–	–	1152	–	–	960	–	–	896	–	–	704
FC1	–	–	1920	–	–	1920	–	–	1600	–	–	1280	–	–	1280	–	–	1024
FC2	–	–	1000	–	–	1000	–	–	1000	–	–	1000	–	–	1000	–	–	1000

Table 1. **Specification of Mobile-Former models**. "bneck-lite" denotes the lite bottleneck block [3]. "bneck-lite$^{\downarrow}$" denotes the downsample variant of lite bottleneck, in which the depthwise convolution has stride 2. "M-F" denotes the Mobile-Former block and "M-F$^{\downarrow}$" 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 2. **Hyper-parameters** of seven Mobile-Former models for ImageNet [1] classification.

Stage	**E2E-MF 508M**	**E2E-MF 294M**	**E2E-MF 214M**	**E2E-MF 151M**
query	100×256	100×256	100×256	100×256
$\frac{1}{32}$	projection †M-F ×5	projection †M-F ×6	projection †M-F ×5	projection †M-F ×3
$\frac{1}{16}$	up-conv M-F ×2	up-conv M-F ×3	up-conv M-F ×2	up-conv M-F ×2
$\frac{1}{8}$	up-conv M-F ×2	–	–	–

Table 3. **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 depthwise and a pointwise convolution. "M-F ×2" refers to stacking two Mobile-Former blocks. In the detection head, we use lite bottleneck [3] in *Mobile* sub-block to reduce the computational cost. At the lowest resolution $\frac{1}{32}$, multi-head attention is added into *Mobile*, which is denoted as †M-F.

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 2 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 [8].

Head model variants in end-to-end object detection: Table 3 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

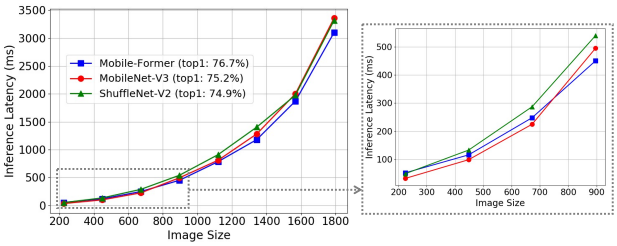

Figure 1. **Inference latency** over different image sizes. The latency is measured on an Intel(R) Xeon(R) CPU E5-2650 v3 (2.3GHz), following the common settings (single-thread with batch size 1) in [2, 6]. Compared to MobileNetV3 [2] and ShuffleNetV2 [5], Mobile-Former is slower on small images, but has faster inference on larger images. Best viewed in color.

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 [3] in *Mobile*. Moreover, we find that the performance can be further improved at a small additional cost by adding multi-head attention in *Mobile* sub-block at the lowest scale ($\frac{1}{32}$) of the head (denoted as †M-F). It is especially helpful for detecting large objects.

2. More Experimental Results

Inference latency (CPU): Figure 1 compares Mobile-Former-214M with MobileNetV3 Large [2] and ShuffleNetV2 2× [5] on inference latency. Mobile-Former is more accurate than the two baselines (76.7% vs. 75.2% vs. 74.9% top-1 on ImageNet). 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 [2,6]. The comparison is performed on multiple image sizes due to the resolution variation across tasks (e.g. classification, detection). Mobile-Former is slower at low resolution (224×224). As the image resolution increases, the gap shrinks until resolution 750×750, after which Mobile-Former has faster inference.

This is because *Former* and embedding projections in *Mobile*→*Former* and *Mobile*←*Former* are resolution independent, and their PyTorch implementations are not as efficient as convolution. Thus, the overhead is relative large when image is small, but becomes negligible as image size grows. The runtime performance of Mobile-Former can be further improved by optimizing the implementation of these

Kernel Size in Mobile	#Param	MAdds	Top-1	Top-5
3×3	11.4M	294M	77.8	93.7
5×5	11.5M	332M	77.9	93.9

Table 4. **Ablation of the kernel size in the depthwise convolution** (in *Mobile* sub-block). The evaluation is performed on ImageNet [1] classification. Mobile-Former-294M is used.

MHA in *Mobile* at scale $\frac{1}{32}$	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	MAdds (G)	#Params (M)
	42.5	61.0	46.0	23.2	46.3	58.7	**36.0**	**23.7**
✓	**43.3**	**61.8**	**46.8**	**24.6**	**47.0**	**60.4**	41.4	26.3

Table 5. **Ablation of multi-head attention (MHA)** in *Mobile* at resolution $\frac{1}{32}$ of the detection head. The evaluation is performed on COCO [4] 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.

components. We will investigate this in the future work.

Inference latency (GPU): We report GPU performance of Mobile-Former-214M, and compare it with MobileNetV3 Large [2] and ShuffleNetV2 2× [5]. Their top-1 accuracy on ImageNet are 76.7%, 75.2%, 74.9% respectively. Our Mobile-Former achieves 29.8 FPS on 2K resolution, which is slower than MobileNetV3 (34.7 FPS) and ShuffleNetV2 (39.8 FPS). This is because *Former* and two-way cross attention are *neither* GPU efficient *nor* implemented in parallel to *Mobile* in PyTorch. This could be improved by implementing *Mobile* and *Former* in parallel with proper allocation of GPU cores.

Ablation of the kernel size in *Mobile*: We perform an ablation on the kernel size of the depthwise convolution in *Mobile*, to validate the contribution of *Former* and bridge on global interaction. Table 4 shows that the gain of increasing kernel size (from 3×3 to 5×5) is negligible. We believe this is because *Former* and the bridge enlarge the reception field for *Mobile* via fusing global features. Therefore, using larger kernel size is not necessary in Mobile-Former.

Ablation of multi-head attention in *Mobile* at resolution $\frac{1}{32}$ of the detection head: Table 5 shows the effect of using multi-head attention (MHA) in the five blocks at the lowest resolution $\frac{1}{32}$ (†M-F in Table 3). Without MHA, a solid performance (42.5 AP) is achieved at low FLOPs (36.0G). Adding MHA gains 0.8 AP with 15% additional computational cost. It is especially helpful for detecting large objects (58.7→60.4 AP_L).

3. 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 2, 3, and 4. The ImageNet pretrained Mobile-Former-294M is used, which includes six global tokens and eleven Mobile-Former blocks. We observe three interesting pat-

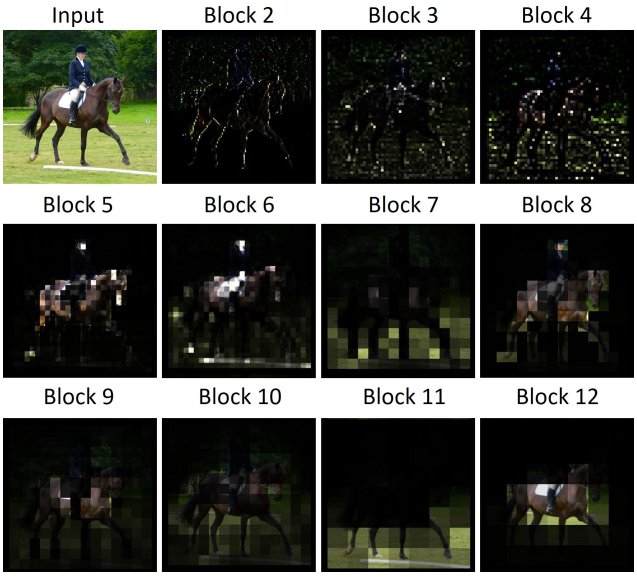

Figure 2. **Cross attention over the entire featuremap for the first token in *Mobile*→*Former*** across all Mobile-Former blocks. Attention is normalized over pixels, showing the focused region. The focused region changes from low to high levels. The token starts paying more attention to edges/corners 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.

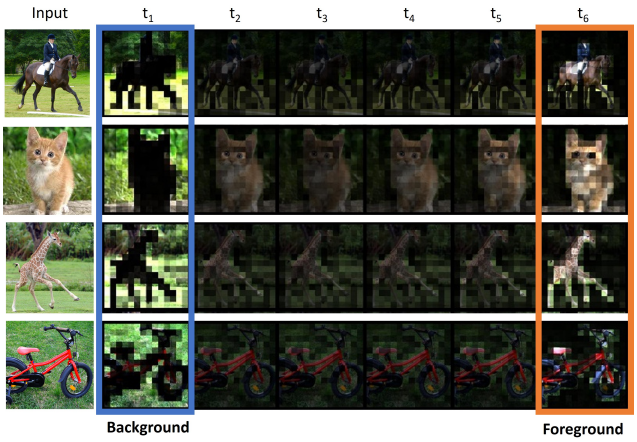

Figure 3. **Cross attention in *Mobile*←*Former* separates foreground and background at middle layers**. Attention is normalized over tokens showing the contribution of different tokens at each pixel. Block 8 is chosen where background pixels pay more attention to the first token and foreground pixels pay more attention to the last token. Best viewed in color.

Figure 4. **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.

terns as follows:

Patten 1 – global tokens shift focus over levels: The focused regions of global tokens change progressively from low to high levels. Figure 2 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 3 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 4, 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 [7, 9, 10]. The full visualization of two-way cross attention for all blocks is shown in Figure 5.

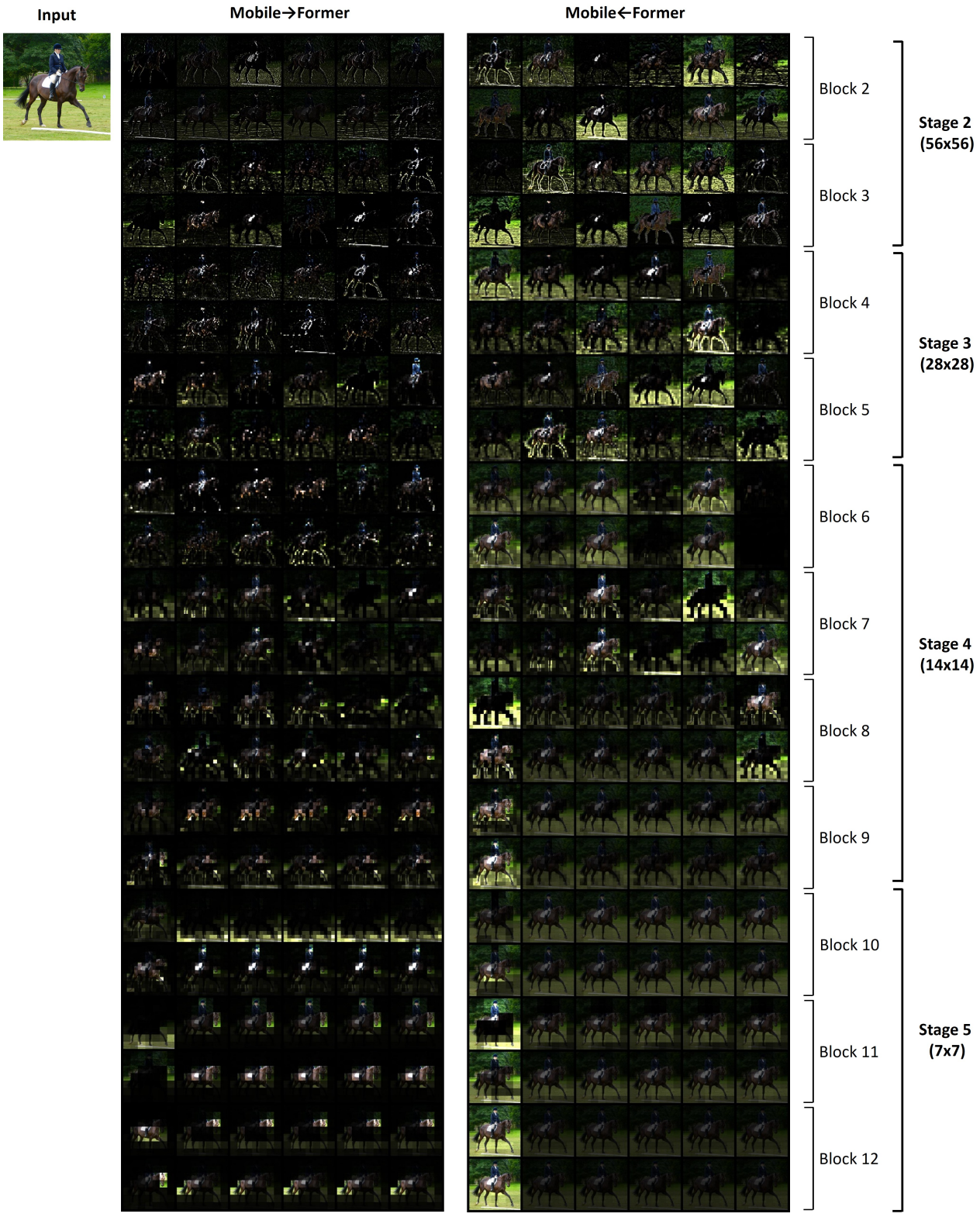

Figure 5. **Visualization of the two-way cross attention**: *Mobile→Former* and *Mobile←Former*. Mobile-Former-294M is used, which includes six tokens (each corresponds to a column) and eleven Mobile-Former blocks (block 2–12) across four stages. Each block has two attention heads that are visualized 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.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2, 3

[2] 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. 1, 2, 3

[3] Yunsheng 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. 1, 2

[4] 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. 3

[5] 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, 3

[6] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 1, 2

[7] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers, 2021. 4

[8] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 2

[9] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer, 2021. 4

[10] Daquan Zhou, Yujun Shi, Bingyi Kang, Weihao Yu, Zihang Jiang, Yuan Li, Xiaojie Jin, Qibin Hou, and Jiashi Feng. Refiner: Refining self-attention for vision transformers, 2021. 4