Separable Self-attention for Mobile Vision Transformers

Sachin Mehta
Apple

Mohammad Rastegari
Apple

Abstract

Mobile vision transformers (MobileViT) can achieve state-of-the-art performance across several mobile vision tasks, including classification and detection. Though these models have fewer parameters, they have high latency as compared to convolutional neural network-based models. The main efficiency bottleneck in MobileViT is the multi-headed self-attention (MHA) in transformers, which requires $O(k^2)$ time complexity with respect to the number of tokens (or patches) k. Moreover, MHA requires costly operations (e.g., batch-wise matrix multiplication) for computing self-attention, impacting latency on resource-constrained devices. This paper introduces a *separable self-attention* method with linear complexity, i.e. $O(k)$. A simple yet effective characteristic of the proposed method is that it uses element-wise operations for computing self-attention, making it a good choice for resource-constrained devices. The improved model, `MobileViTv2`, is state-of-the-art on several mobile vision tasks, including ImageNet object classification and MS-COCO object detection. With about three million parameters, `MobileViTv2` achieves a top-1 accuracy of 75.6% on the ImageNet dataset, outperforming MobileViT by about 1% while running $3.2\times$ faster on a mobile device. Our source code is available at: `https://github.com/apple/ml-cvnets`

1 Introduction

Vision transformers (ViTs) [1] have become ubiquitous for a wide variety of visual recognition tasks [2, 3], including mobile vision tasks [4]. At the heart of the ViT-based models, including mobile vision transformers, is the transformer block [5]. The main efficiency bottleneck in ViT-based models, especially for inference on resource-constrained devices, is the multi-headed self-attention (MHA). MHA allows the tokens (or patches) to interact with each other, and is a key for learning global representations. However, the complexity of self-attention in transformer block is $O(k^2)$, i.e., it is quadratic with respect to the number of tokens (or patches) k. Besides this, computationally expensive operations (e.g., batch-wise matrix multiplication; see Fig. 1) are required to compute attention matrix in MHA. This, in particular, is concerning for deploying ViT-based models on resource-constrained devices, as these devices have reduced computational capabilities, restrictive memory constraints, and a limited power budget. Therefore, this paper seeks to answer this question: *can self-attention in transformer block be optimized for resource-constrained devices?*

Several methods [e.g., 7–10] have been proposed for optimizing the self-attention operation in transformers (not necessarily for ViTs). Among these, a widely studied approach in sequence modeling tasks is to introduce sparsity in self-attention layers, wherein each token attends to a subset of tokens in an input sequence [7, 9]. Though these approaches reduces the time complexity from $O(k^2)$ to $O(k\sqrt{k})$ or $O(k \log k)$, the cost is a performance drop. Another popular approach for approximating self-attention is via low-rank approximation. Linformer [10] decomposes the self-attention operation into multiple smaller self-attention operations via linear projections, and reduces the complexity of self-attention from $O(k^2)$ to $O(k)$. However, Linformer still uses costly

Preprint. Under review.

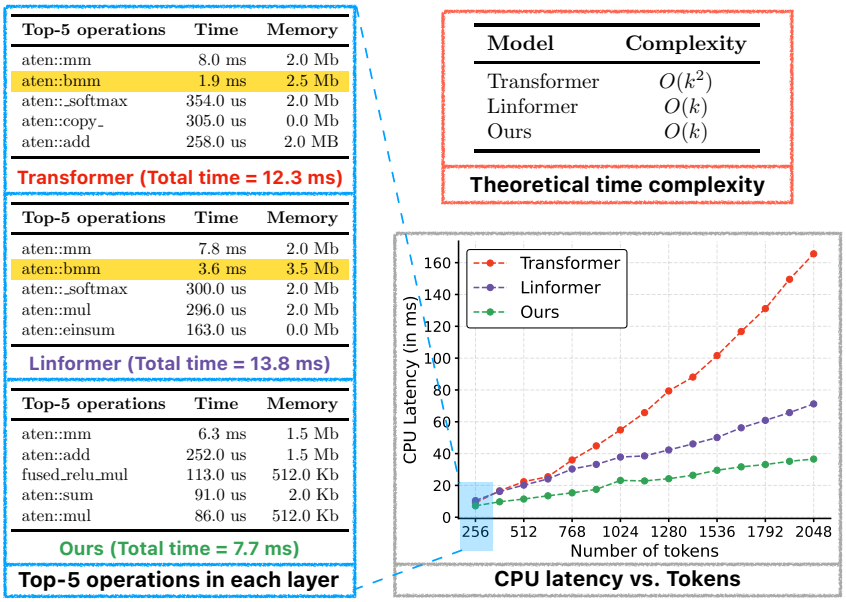
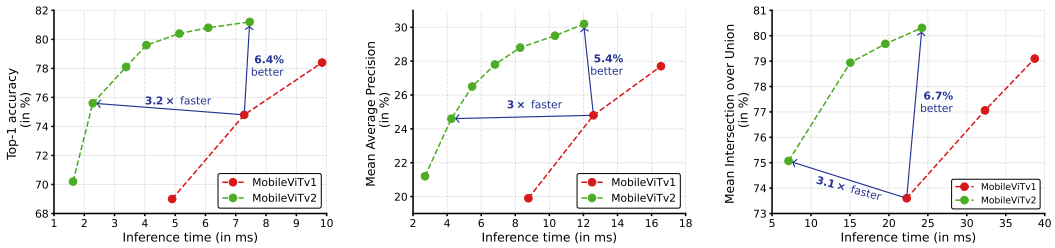


Figure 1: **Comparison between different attention units.** Transformer and Linformer use costly operations (batch-wise matrix multiplication) for computing self-attention. Such operations are a bottleneck for efficient inference on resource-constrained devices. The proposed method does not use such operations, thus accelerating inference on resource-constrained devices. **Left** compares top-5 operations (sorted by CPU time) in a single layer of different attention units for $k = 256$ tokens. **Top Right** compares complexity of different attention units. **Bottom Right** compares the latency of different attention units as a function of the number of tokens k. These results are computed on a single CPU core machine with a 2.4 GHz 8-Core Intel Core i9 processor, $d = 512$ (token dimensionality), $h = 8$ (number of heads; for Transformer and Linformer), and $p = 256$ (projected tokens in Linformer) using a publicly available profiler in PyTorch [6].



(a) ImageNet-1k classification (b) MS-COCO object detection (c) PASCAL VOC segmentation

Figure 2: `MobileViTv2` **models are faster and better than MobileViTv1 models [4] across different tasks.** `MobileViTv2` models are constructed by replacing multi-headed self-attention in MobileViTv1 with the proposed *separable self-attention* (Section 3.2). Here, inference time is measured on an iPhone12 for an input resolution of 256×256, 512×512, and 320×320 for classification, segmentation, and detection respectively.

operations (e.g., batch-wise matrix multiplication; Fig. 1) for learning global representations in MHA, which may hinder the deployment of these models on resource-constrained devices.

This paper introduces a novel method, *separable self-attention*, with $O(k)$ complexity for addressing the bottlenecks in MHA in transformers. For efficient inference, the proposed self-attention method also replaces the computationally expensive operations (e.g., batch-wise matrix multiplication) in MHA with element-wise operations (e.g., summation and multiplication). Experimental results on standard vision datasets and tasks demonstrates the effectiveness of the proposed method (Fig. 2).

2 Related work

Improving self-attention Improving the efficiency of MHA in transformers is an active area of research. The first line of research introduces locality to address the computational bottleneck in MHA [e.g., 7, 9, 11, 12]. Instead of attending to all k tokens, these methods use predefined patterns to limit the receptive field of self-attention from all k tokens to a subset of tokens, reducing the time complexity from $O(k^2)$ to $O(k\sqrt{k})$ or $O(k \log k)$. However, such methods suffer from large performance degradation with moderate training/inference speed-up over the standard MHA in transformers. To improve the efficiency of MHA, the second line of research uses similarity measures to group tokens [8, 13, 14]. For instance, Reformer [8] uses locality-sensitive hashing to group the tokens and reduces the theoretical self-attention cost from $O(k^2)$ to $O(k \log k)$. However, the efficiency gains over standard MHA are noticeable only for large sequences ($k > 2048$) [8]. Because $k < 1024$ in ViTs, these approaches are not suitable for ViTs. The third line of research improves the efficiency of MHA via low-rank approximation [10, 15]. The main idea is to approximate the self-attention matrix with a low-rank matrix, reducing the computational cost from $O(k^2)$ to $O(k)$. Even though these methods speed-up the self-attention operation significantly, they still use expensive operations for computing attention, which may hinder the deployment of these models on resource-constrained devices (Fig. 1).

In summary, existing methods for improving MHA are limited in their reduction of inference time and memory consumption, especially for resource-constrained devices. This work introduces a separable self-attention method that is fast and memory-efficient (see Fig. 1), which is desirable for resource-constrained devices.

Improving transformer-based models There has been significant work on improving the efficiency of transformers [3, 4, 16–18]. The majority of these approaches reduce the number of tokens in the transformer block using different methods, including down-sampling [19, 18] and pyramidal structure [3, 20, 4]. Because the proposed separable self-attention module is a drop-in replacement to MHA, it can be easily integrated with any transformer-based model to further improve its efficiency.

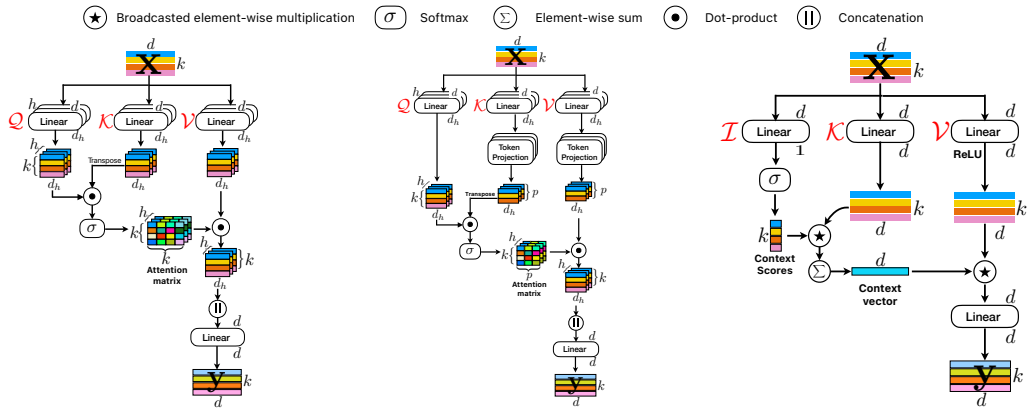
Other methods Transformer-based models performance can be improved using different methods, including mixed-precision training [21], efficient optimizers [22, 23], and knowledge distillation [2]. These methods are orthogonal to our work, and by default, we use mixed-precision during training.

3 `MobileViTv2`

MobileViT [4] is a hybrid network that combines the strengths of CNNs and ViTs. MobileViT views transformers as convolutions, which allows it to leverage the merits of both convolutions (e.g., inductive biases) and transformers (e.g., long-range dependencies) to build a light-weight network for mobile devices. Though MobileViT networks have significantly fewer parameters and deliver better performance as compared to light-weight CNNs (e.g., MobileNets [24, 25]), they have high latency. The main efficiency bottleneck in MobileViT is the multi-headed self-attention (MHA; Fig. 3a).

MHA uses scaled dot-product attention to capture the contextual relationships between k tokens (or patches). However, MHA is expensive as it has $O(k^2)$ time complexity. This quadratic cost is a bottleneck for transformers with a large number of tokens k (Fig. 1). Moreover, MHA uses computationally- and memory-intensive operations (e.g., batch-wise matrix multiplication and softmax for computing attention matrix; Fig. 1); which could be a bottleneck on resource-constrained devices. To address the limitations of MHA for efficient inference on resource-constrained devices, this paper introduces *separable self-attention* with linear complexity (Fig. 3c).

The main idea of our *separable self-attention* approach, shown in Fig. 4b, is to compute context scores with respect to a latent token L. These scores are then used to re-weight the input tokens and produce a context vector, which encodes the global information. Because the self-attention is computed with respect to a latent token, the proposed method can reduce the complexity of self-attention in the transformer by a factor k. A simple yet effective characteristic of the proposed method is that it uses element-wise operations (e.g., summation and multiplication) for its implementation, making it a good choice for resource-constrained devices. We call the proposed attention method separable self-attention because it allows us to encode global information by replacing the quadratic MHA



(a) MHA in Transformers [5] (b) MHA in Linformer [10] (c) Separable self-attention (ours)

Figure 3: **Different self-attention units.** **(a)** is a standard multi-headed self-attention (MHA) in transformers. **(b)** extends MHA in (a) by introducing token projection layers, which project k tokens to a pre-defined number of tokens p, thus reducing the complexity from $O(k^2)$ to $O(k)$. However, it still uses costly operations (e.g., batch-wise matrix multiplication) for computing self-attention, impacting latency on resource-constrained devices (Fig. 1). **(c)** is the proposed separable self-attention layer that is linear in complexity, i.e., $O(k)$, and uses element-wise operations for faster inference.

with two separate linear computations. The improved model, `MobileViTv2`, is obtained by replacing MHA with separable self-attention in MobileViT.

In the rest of this section, we first briefly describe MHA (Section 3.1), and then elaborate on the details of separable self-attention (Section 3.2) and `MobileViTv2` architecture (Section 3.3).

3.1 Overview of multi-headed self-attention

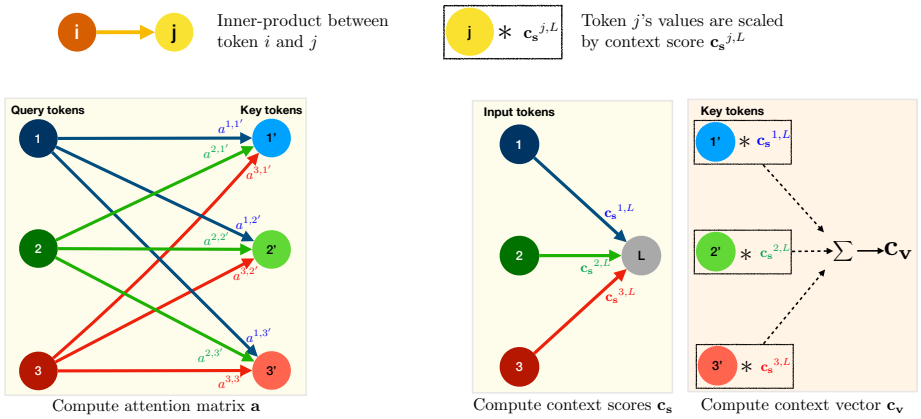
MHA (Fig. 3a) allows transformer to encode inter-token relationships. Specifically, MHA takes an input $\mathbf{x} \in \mathbb{R}^{k \times d}$ comprising of k d-dimensional token (or patch) embeddings. The input $\mathbf{x}$ is then fed to three branches, namely query $\mathcal{Q}$, key $\mathcal{K}$, and value $\mathcal{V}$. Each branch ($\mathcal{Q}$, $\mathcal{K}$, and $\mathcal{V}$) is comprised of h linear layers (or heads), which enables the transformer to learn multiple views of the input. The dot-product between the output of linear layers in $\mathcal{Q}$ and $\mathcal{K}$ is then computed simultaneously for all h heads, and is followed by a softmax operation σ to produce an attention (or context-mapping) matrix $\mathbf{a} \in \mathbb{R}^{k \times k \times h}$. Another dot-product is then computed between $\mathbf{a}$ and the output of linear layers in $\mathcal{V}$ to produce weighted sum output $\mathbf{y_w} \in \mathbb{R}^{k \times d_h \times h}$, where $d_h = \frac{d}{h}$ is the head dimension . The outputs of h heads are concatenated to produce a tensor with k d-dimensional tokens, which is then fed to another linear layer with weights $\mathbf{W_O} \in \mathbb{R}^{d \times d}$ to produce the output of MHA $\mathbf{y} \in \mathbb{R}^{k \times d}$. Mathematically, this operation can be described as:

$$\mathbf{y} = \text{Concat}\left(\langle \underbrace{\sigma\left(\langle \mathbf{x}\mathbf{W_Q}^0, \mathbf{x}\mathbf{W_K}^0 \rangle \right)}_{\mathbf{a}^0 \in \mathbb{R}^{k \times k}}, \mathbf{x}\mathbf{W_V}^0 \rangle, \cdots, \langle \underbrace{\sigma\left(\langle \mathbf{x}\mathbf{W_Q}^h, \mathbf{x}\mathbf{W_K}^h \rangle \right)}_{\mathbf{a}^h \in \mathbb{R}^{k \times k}}, \mathbf{x}\mathbf{W_V}^h \rangle \right) \mathbf{W_O} \tag{1}$$

where $\mathbf{W_Q}^i \in \mathbb{R}^{d \times d_h}$, $\mathbf{W_K}^i \in \mathbb{R}^{d \times d_h}$, and $\mathbf{W_V}^i \in \mathbb{R}^{d \times d_h}$ are the weights of the i-th linear layer (or head) in $\mathcal{Q}$, $\mathcal{K}$, and $\mathcal{V}$ branches respectively. The symbol $\langle \cdot, \cdot \rangle$ denotes the dot-product operation.

3.2 Separable self-attention

The structure of separable self-attention is inspired by MHA. Similar to MHA, the input $\mathbf{x}$ is processed using three branches, i.e., input $\mathcal{I}$, key $\mathcal{K}$, and value $\mathcal{V}$. The input branch $\mathcal{I}$ maps each d-dimensional token in $\mathbf{x}$ to a scalar using a linear layer with weights $\mathbf{W_I} \in \mathbb{R}^d$. The weights $\mathbf{W_I}$ serves as the latent node L in Fig. 4b. This linear projection is an inner-product operation and computes the distance between latent token L and $\mathbf{x}$, resulting in a k-dimensional vector. A softmax operation is then applied to this k-dimensional vector to produce context scores $\mathbf{c_s} \in \mathbb{R}^k$. Unlike transformers that compute the attention (or context) score for each token with respect to all k tokens, the proposed



(a) Self-attention in transformers (b) Proposed separable self-attention method

Figure 4: **Example illustrating the interaction between tokens to learn global representations in different attention layers.** In (a), each query token computes the distance with all key tokens via dot-product. These distances are then normalized using softmax to produce an attention matrix $\mathbf{a}$, which encodes contextual relationships. In (b), the inner product between input tokens and latent token L is computed. The resultant vector is normalized using softmax to produce context scores $\mathbf{c_s}$. These context scores are used to weight key tokens and produce a context vector $\mathbf{c_v}$, which encodes contextual information.

method only computes the context score with respect to a latent token L. This reduces the cost of computing attention (or context) scores from $O(k^2)$ to $O(k)$.

The context scores $\mathbf{c_s}$ are used to compute a context vector $\mathbf{c_v}$. Specifically, the input $\mathbf{x}$ is linearly projected to a d-dimensional space using key branch $\mathcal{K}$ with weights $\mathbf{W_K} \in \mathbb{R}^{d \times d}$ to produce an output $\mathbf{x_K} \in \mathbb{R}^{k \times d}$. The context vector $\mathbf{c_v} \in \mathbb{R}^d$ is then computed as a weighted sum of $\mathbf{x_K}$ as:

$$\mathbf{c_v} = \sum_{i=1}^{k} \mathbf{c_s}(i)\mathbf{x_K}(i) \tag{2}$$

The context vector $\mathbf{c_v}$ is analogous to the attention matrix $\mathbf{a}$ in Eq. (1) in a sense that it also encodes the information from all tokens in the input $\mathbf{x}$, but is cheap to compute.

The contextual information encoded in $\mathbf{c_v}$ is shared with all tokens in $\mathbf{x}$. To do so, the input $\mathbf{x}$ is linearly projected to a d-dimensional space using a value branch $\mathcal{V}$ with weights $\mathbf{W_V} \in \mathbb{R}^{d \times d}$, followed by a ReLU activation to produce an output $\mathbf{x_V} \in \mathbb{R}^{k \times d}$. The contextual information in $\mathbf{c_v}$ is then propagated to $\mathbf{x_V}$ via broadcasted element-wise multiplication operation. The resultant output is then fed to another linear layer with weights $\mathbf{W_O} \in \mathbb{R}^{d \times d}$ to produce the final output $\mathbf{y} \in \mathbb{R}^{k \times d}$. Mathematically, separable self-attention can be defined as:

$$\mathbf{y} = \left(\underbrace{\sum \left(\overbrace{\sigma\left(\mathbf{x}\mathbf{W_I}\right)}^{\mathbf{c_s} \in \mathbb{R}^k} * \mathbf{x}\mathbf{W_K} \right)}_{\mathbf{c_v} \in \mathbb{R}^d} * \text{ReLU}\left(\mathbf{x}\mathbf{W_V}\right) \right) \mathbf{W_O} \tag{3}$$

where $*$ and $\sum$ are broadcastable element-wise multiplication and summation operations, respectively.

Comparison with self-attention methods Fig. 1 compares the proposed method with Transformer and Linformer. Because time complexity of self-attention methods do not account for the cost of operations that are used to implement these methods, some of the operations may become bottleneck on resource-constrained devices. For holistic understanding, module-level latency on a single CPU core with varying k is also measured in addition to theoretical metrics. The proposed separable self-attention is fast and efficient as compared to MHA in Transformer and Linformer.

Besides these module-level results, when we replaced the MHA in the transformer with the proposed self-separable attention in the MobileViT architecture, we observe $3\times$ improvement in inference

Table 1: **Effect of different self-attention methods** on the performance of MobileViT [4] on the ImageNet-1k dataset. Here, all models have similar number of parameters and FLOPs, and latency is measured on iPhone12.

Attention unit	Latency ↓	Top-1 ↑
Self-attention in Transformer (Fig. 3a; [5])	9.9 ms	**78.4**
Self-attention in Linformer (Fig. 3b; [10])	10.2 ms	78.2
Separable self-attention (Ours; Fig. 3c)	**3.4** ms	78.1

speed with similar performance on the ImageNet-1k dataset (Table 1). These results show the efficacy of the proposed separable self-attention at the architecture-level. Note that self-attention in Transformer and Linformer yields similar results for MobileViT. This is because the number of tokens k in MobileViT is fewer ($k \leq 1024$) as compared to language models, where Linformer is significantly faster than the transformer.

Relationship with additive addition The proposed approach resembles the attention mechanism of Bahdanau et al. [26], which also encodes the global information by taking a weighted-sum of LSTM outputs at each time step. Unlike [26], where input tokens interact via recurrence, the input tokens in the proposed method interact only with a latent token.

3.3 `MobileViTv2` architecture

To demonstrate the effectiveness of the proposed separable self-attention on resource-constrained devices, we integrate separable self-attention with a recent ViT-based model, MobileViT [4]. MobileViT is a light-weight, mobile-friendly hybrid network that delivers significantly better performance than other competitive CNN-based, transformer-based, or hybrid models, including MobileNets [27, 24, 25]. To avoid ambiguity, we refer to MobileViT as MobileViTv1 in the rest of the paper.

Specifically, we replace MHA in the transformer block in the MobileViTv1 with the proposed separable self-attention method. We call the resultant architecture `MobileViTv2`. We also do not use the skip-connection and fusion block in the MobileViT block (Fig. 1b in [4]) as it improves the performance marginally (Fig. 12 in [4]). Furthermore, to create `MobileViTv2` models at different complexities, we uniformly scale the width of `MobileViTv2` network using a width multiplier $\alpha \in \{0.5, 2.0\}$. This is in contrast to MobileViTv1 which trains three specific architectures (XXS, XS, and S) for mobile devices. More details about `MobileViTv2`'s architecture are given in Appendix A.

4 Experimental results

4.1 Object classification on the ImageNet dataset

Training on ImageNet-1k from scratch We train `MobileViTv2` for 300 epochs with an effective batch size of 1024 images (128 images per GPU $\times$ 8 GPUs) using AdamW [28] on the ImageNet-1k dataset [29] with 1.28 million and 50 thousand training and validation images respectively. We linearly increase the learning rate from 10^{-6} to 0.002 for the first 20k iterations. After that, the learning rate is decayed using a cosine annealing policy [30]. To reduce stochastic noise during training, we use exponential moving average (EMA) [31] as we find it helps larger models. We implement our models using `CVNets` [4, 32], and use their provided scripts for data processing, training, and evaluation.

Pre-training on ImageNet-21k-P and finetuning on ImageNet-1k We train on the ImageNet-21k (winter'21 release) that contains about 13 million images across 19k classes. Specifically, we follow [33] to pre-process (e.g., remove classes with fewer samples) the dataset and split it into about 11 million and 522 thousand training and validation images spanning over 10,450 classes, respectively. Following [33], we refer to this pre-processed dataset as ImageNet-21k-P. Note that the ImageNet-21k-P validation set does not overlap with the validation and test sets of ImageNet-1k.

We follow [33] for pre-training `MobileViTv2` on ImageNet-21k-P. For faster convergence, we initialize `MobileViTv2` models with ImageNet-1k weights and finetune it on ImageNet-21k-P for 80

Table 2: **Classification performance on the ImageNet-1k validation set**. Here, NS means that we are not able to measure the latency on mobile device as some operations (e.g., cyclic shifts) are not supported on mobile devices. Following [4], latency is measured on iPhone12 with a batch size of 1. Similar to [3, 34], throughput is measured on NVIDIA V100 GPUs with a batch size of 128. The rows are grouped by network parameters.

Row #	Model	Type	Neural search?	Extra data	Image size	# Params ↓	FLOPs ↓	Latency ↓ (in ms)	Throughput ↑ (images/ sec)	Top-1 ↑ (in %)
R1	MobileViT-XXS [4]	Hybrid	✗	None	256^2	**1.3 M**	0.4 G	4.8	4225	69.0
R2	`MobileViTv2-0.5`	Hybrid	✗	None	256^2	1.4 M	0.5 G	**1.6**	**4595**	**70.2**
R3	MobileFormer-52 [35]	Hybrid	✗	None	224^2	3.6 M	**52 M**	7.1	4445	68.7
R4	`MobileViTv2-1.0`	Hybrid	✗	None	256^2	**4.9 M**	1.8 G	3.4	2351	78.1
R5	EfficientNet-b0 [36]	CNN	✓	None	224^2	5.3 M	**422 M**	**1.6**	**4619**	77.1
R6	DeiT-Tiny [2]	Transformer	✗	None	224^2	5.5 M	1.3 G	3.4	4541	72.2
R7	MobileViT-S [4]	Hybrid	✗	None	256^2	5.6 M	2.0 G	3.4	1986	**78.4**
R8	EfficientNet-b2 [36]	CNN	✓	None	288^2	**9.1 M**	1.2 G	**3.8**	**2032**	80.1
R9	`MobileViTv2-1.5`	Hybrid	✗	None	256^2	10.6 M	4.0 G	5.1	1418	**80.4**
R10	MobileFormer-294 [35]	Hybrid	✗	None	224^2	11.8 M	**294 M**	40.7	1402	77.9
R11	`MobileViTv2-2.0`	Hybrid	✗	None	256^2	**18.5 M**	7.5 G	7.5	1105	81.2
R12	Swin-T [3]	Hybrid	✗	None	224^2	28.3 M	**4.5 G**	NS	1390	81.3
R13	ConvNext-T [34]	CNN	✗	None	224^2	28.6 M	**4.5 G**	**3.7**	**1800**	82.1
R14	DeiT-Base [2]	Transformer	✗	None	224^2	86.6 M	17.6 G	13.2	958	81.8
R15	`MobileViTv2-2.0`	Hybrid	✗	ImageNet-21k-P	256^2	**18.5 M**	7.5 G	7.5	1105	82.4
R16	ConvNext-T [34]	CNN	✗	ImageNet-21k	224^2	28.6 M	**4.5 G**	**3.7**	**1800**	**82.9**
R17	`MobileViTv2-2.0`	Hybrid	✗	ImageNet-21k-P	384^2	**18.5 M**	16.1 G	17.0	488	83.4
R18	ConvNext-T [34]	CNN	✗	ImageNet-21k	384^2	28.6 M	**13.1 G**	**8.6**	**645**	**84.1**

epochs with an effective batch size of 4096 images (128 images per GPU x 32 GPUs). We do not use any linear warm-up. Other settings follow ImageNet-1k training.

We finetune ImageNet-21k-P pre-trained models on ImageNet-1k for 50 epochs using SGD with momentum (0.9) and cosine annealing policy with an effective batch size of 256 images (128 images per GPU $\times$ 2 GPUs).

Finetuning at higher resolution `MobileViTv2` is a hybrid architecture that combines convolution and separable self-attention to learn visual representations. Unlike many ViT-based models (e.g., DeiT), `MobileViTv2` does not require adjustment to patch embeddings or positional biases for different input resolutions and is simple to finetune. We finetune `MobileViTv2` models at higher resolution (i.e., 384×384) for 10 epochs with a fixed learning rate of 10^{-3} using SGD.

Comparison with existing methods Table 2 and Fig. 2 compares `MobileViTv2`'s performance with recent methods[1]. We make following observations:

- When MHA in MobileViTv1 is replaced with separable self-attention, the resultant model, `MobileViTv2`, is faster and better (Fig. 2); validating the effectiveness of the proposed separable self-attention method for mobile ViTs.
- Compared to transformer-based (including hybrid) models, `MobileViTv2` models are fast on mobile devices. For example, `MobileViTv2` is about $8\times$ faster on a mobile device and delivers 2.5% better performance on the ImageNet-1k dataset than MobileFormer [35], even though MobileFormer is FLOP efficient (R9 vs. R10). However, on GPU, both MobileFormer and `MobileViTv2` run at a similar speed. The discrepancy in FLOPs and speed of MobileFormer across devices is primarily because of its architectural design. MobileFormer has conditional operations between mobile and former blocks. Such conditional operations, especially on resource-constrained devices, have a low degree of parallelism and create memory bottlenecks, resulting in a high latency network. Ma et al. [37] also makes a similar observation for CNN-based architectures.
- `MobileViTv2` bridges the latency gap between CNN- and ViT-based models on mobile devices while maintaining performance with similar or fewer parameters. For example, on a mobile device, ConvNexT [34] (CNN-based model) is $2\times$ and $3.6\times$ faster than `MobileViTv2` (hybrid model) and DeiT (transformer-based model) for similar performance respectively (see R11, R13, and R14). The low latency of fully CNN-based models on mobile devices can be attributed to several device-level optimizations that have been done for CNN-based models over the past few years (e.g., dedicated hardware implementations for convolutions and folding batch normalization with

[1]For additional results including ablations, see Appendix B, Appendix C, and Appendix E.

Table 3: **Semantic segmentation results on the ADE20k and the PASCAL VOC 2012 datasets.** Here, throughput, network parameters, and FLOPs are measured on the ADE20k dataset for an input with a spatial resolution of 512×512. mIoU (mean intersection over union) score is calculated for a single scale only. Throughput is calculated using a batch size of 32 images on a single NVIDIA V100 GPU with 32 GB memory and is an average of over 50 iterations (excluding 10 iterations for warmup). We do not report latency on a mobile device as some of the operations (e.g., pyramid pooling in PSPNet) are not optimally implemented for mobile devices. The baseline results are from the MMSegmentation library [45]. Rows are grouped by network parameters.

Seg.	ImageNet-1k	Image	Throughput ↑	# Params ↓	FLOPs ↓	mIoU ↑	
Model	Backbone	Size	(images/sec)	(in millions)	(in billions)	ADE20k [42]	PASCAL VOC [43]
PSPNet [40]	MobileViTv2-0.5 (Ours)	512^2	**439**	**3.6 M**	**15.4 G**	**31.8**	74.6
	MobileNetv2 [24]	512^2	276	13.7 M	53.1 G	29.7	–
PSPNet [40]	MobileViTv2-1.75 (Ours)	512^2	114	**22.5 M**	**95.9 G**	39.8	**80.2**
	ResNet-50 [44]	512^2	**119**	49.1 M	179.1 G	**41.1**	76.8
DeepLabv3 [41]	MobileViTv2-0.75 (Ours)	512^2	241	**9.6 M**	**40.0 G**	**34.7**	75.1 (α=0.5)
	MobileNetv2 [24]	512^2	**246**	18.7 M	75.4 G	34.1	–
DeepLabv3 [41]	MobileViTv2-2.0 (Ours)	512^2	90	**34.0 M**	**147.0 G**	40.9	**80.3** (α=1.5)
	ResNet-50 [44]	512^2	**103**	68.2 M	270.3 G	**42.4**	79.1

convolutions). ViT-based models still lack such optimizations and therefore, the resultant inference graphs are sub-optimal. Though MobileViTv2 bridges the latency gap between CNNs and ViTs, we believe the latency of ViT-based models will improve in the future with similar optimizations.

- The delta in speed (on GPU) between ConvNext and MobileViTv2 (R15-R18) at higher model complexities reduces from $1.6\times$ to $1.3\times$ when input resolution is increased from 224×224 (or 256×256) to 384×384, suggesting ViT-based (including hybrid) models exhibit better scaling properties as compared to CNNs. This is because of a higher degree of parallelism that ViT-based models offer at a large scale [1, 38]. Our results on down-stream tasks in Section 4.2 and previous work on scaling ViTs [1, 39] further supports this observation.

4.2 Evaluation on down-stream tasks

Semantic segmentation We integrate MobileViTv2 with two standard segmentation architectures, PSPNet [40] and DeepLabv3 [41], and study it on two standard semantic segmentation datasets, ADE20k [42] and PASCAL VOC 2012 [43]. For training details including hyper-parameters, see supplementary material.

Table 3 and Fig. 2c compares the segmentation performance in terms of validation mean intersection over union (mIOU) of MobileViTv2 with different segmentation methods. MobileViTv2 delivers competitive performance at different complexities while having significantly fewer parameters and FLOPs. Interestingly, the inference speed of MobileViTv2 models is comparable to CNN-based models, including light-weight MobileNetv2 and heavy-weight ResNet-50 [44] model. This is consistent with our observation in Section 4.1 (R17 vs. R18; Table 2) where we also observe that ViT-based models scale better than CNN's at higher input resolutions and model complexities.

Object detection We integrate MobileViTv2 with SSDLite [24] (SSD head [46] with separable convolutions) for mobile object detection, and study its performance on MS-COCO dataset [47]. We follow [4] for training detection models. Table 4 and Fig. 2b compares SSDLite's detection performance in terms of validation mean average precision (mAP) using different ImageNet-1k backbones. MobileViTv2 delivers competitive performance to models with different capacities, further validating the effectiveness of the proposed self-separable attention method.

5 Visualizations of self-separable attention scores

Fig. 5 visualizes what the context scores learn at different output strides[2] of MobileViTv2 network. We found that separable self-attention layers pay attention to low-, mid-, and high-level features, and allow MobileViTv2 to learn representations from semantically relevant image regions.

[2]Output stride is the ratio of the spatial dimension of the input to the feature map.

Table 4: **Object detection using SSDLite on the MS-COCO dataset.** Here, throughput is measured with a batch of 128 images on the NVIDIA V100 GPU, and is an average over 50 iterations (excluding 10 iterations for warmup). Latency on a mobile device is not reported as some operations (e.g., hard swish) are not optimally implemented for such devices. Rows are grouped by network parameters.

ImageNet-1k backbone	Image Size	Throughput ↑ (images/sec)	# Params ↓ (in millions)	FLOPs ↓ (in billions)	mAP ↑
MobileViTv1-XXS	320^2	2246	**1.7 M**	0.9 G	19.9
`MobileViTv2-0.5` (Ours)	320^2	2782	2.0 M	0.9 G	21.2
`MobileViTv2-0.75` (Ours)	320^2	1876	3.6 M	1.8 G	**24.6**
Mobilenetv2	320^2	3052	4.3 M	0.8 G	22.1
MobileNetv3	320^2	3884	5.0 M	**0.6 G**	22.0
MobileNetv1	320^2	**4330**	5.1 M	1.3 G	22.2
`MobileViTv2-1.75`	320^2	780	**14.9 M**	**9.0 G**	**29.5**
ResNet-50	300^2	744	22.9 M	20.2 G	25.2

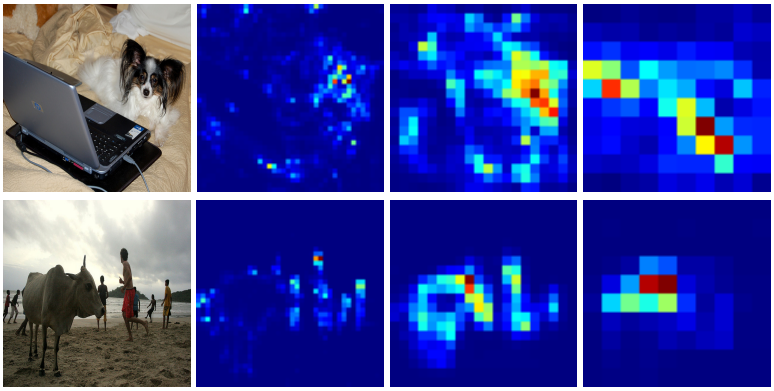


Figure 5: **Context score maps at different output strides (OS) of `MobileViTv2` model.** Observe how context scores pay attention to semantically relevant image regions. (**Left to right:** input image, context scores at OS=8, context scores at OS=16, and context scores at OS=32). For more examples and details about context score map generation, see Appendix D.

6 Conclusions

Transformer-based vision models are slow on mobile devices as compared to CNN-based models because multi-headed self-attention is expensive on resource-constrained devices. In this paper, we introduce a *separable self-attention* method that has linear complexity and can be implemented using hardware-friendly element-wise operations. Experimental results on standard datasets and tasks demonstrate the effectiveness of the proposed method over multi-headed self-attention.

Acknowledgements

We are grateful to Ali Farhadi, Peter Zatloukal, Oncel Tuzel, Rick Chang, Fartash Faghri, Farzad Abdolhosseini, Lailin Chen, and Max Horton for their helpful comments. We are also thankful to Apple's infrastructure and open-source teams for their help with training infrastructure and open-source release of the code and pre-trained models.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, 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. URL `https://openreview.net/forum?id=YicbFdNTTy`.

[2] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[3] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[4] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=vh-0sUt8HlG`.

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

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

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[8] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[9] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[10] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[11] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[12] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*, 2019.

[13] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems*, 33:21665–21674, 2020.

[14] Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Yuwei Fang, Siqi Sun, Yu Cheng, and Jingjing Liu. Cluster-former: Clustering-based sparse transformer for question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3958–3968, 2021.

[15] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[16] Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Delight: Deep and light-weight transformer. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=ujmgfuxSLrO`.

[17] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31, 2021.

[18] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11936–11945, 2021.

[19] Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. *Advances in Neural Information Processing Systems*, 34, 2021.

[20] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021.

[21] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

[22] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=shpkpVXzo3h.

[23] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2021. URL https://arxiv.org/abs/2106.04560.

[24] 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.

[25] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[26] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[27] 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.

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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

[30] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

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

[32] Sachin Mehta, Farzad Abdolhosseini, and Mohammad Rastegari. Cvnets: High performance library for computer vision. *CoRR*, 2022.

[33] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

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

[35] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. *arXiv preprint arXiv:2108.05895*, 2021.

[36] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[37] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

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

[39] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *CoRR*, abs/2106.04560, 2021.

[40] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[41] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[42] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.

[43] Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111 (1):98–136, 2015.

[44] 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.

[45] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

[46] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[47] 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.

[48] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[49] Sachin Mehta, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9190–9200, 2019.

[50] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

[51] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. CrossVit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[52] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021.

[53] 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.

[54] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[55] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[56] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

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

[58] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, 2020.

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

A Detailed architecture of MobileViTv2

MobileViTv2's architecture follows MobileViTv1 [4] and is given in Table 5. MobileViTv2 block, shown in Fig. 6, makes two changes to the MobileViTv1 block: (1) it replaces the multi-headed self-attention with the proposed separable self-attention to learn global representations and (2) it does not use fusion block and skip-connection (see Fig. 1b in [4]) as they improve the performance marginally (see Fig. 12 in [4]). The expansion factor in MobileNetv2 [24] blocks and feed-forward layers is two. Similar to [4], we use Swish [48] as a non-linear activation function. Unlike MobileViTv1 that creates three specific architectures (XXS, XS, and S) for mobile devices, we uniformly scale the width of MobileViTv2 network using a width multiplier $\alpha \in 0.5, 2.0$ to create models at different complexities.

B MobileViTv2's classification performance

ImageNet-1k Table 6 shows the results of MobileViTv2 on the ImageNet-1k dataset. Finetuning MobileViTv2 models at higher resolution (384×384) shows improvement across the board. For example, the performance of MobileViTv2-0.50 with 1.4 million parameters improves by about 2% when finetuned at higher resolution (R1 vs. R2). Similarly, pre-training on the ImageNet-21k-P dataset helps improve the performance of MobileViTv2 models. For example, ImageNet-21k-P pretraining improves the performance of MobileViTv2-2.0 improves by 1.2% (R17 vs. R18). Notably, MobileViTv2 models pretrained on the ImageNet-21k-P are able to achieve the similar performance with fewer FLOPs to models finetuned on ImageNet-1k with a higher resolution (e.g., R10 vs. R11; R14 vs. R15; R18 vs. R19 in Table 6).

ImageNet-21k-P Table 7 shows the results on the ImageNet-21k-P validation dataset. The performance of MobileViTv2 improves with increase in model size.

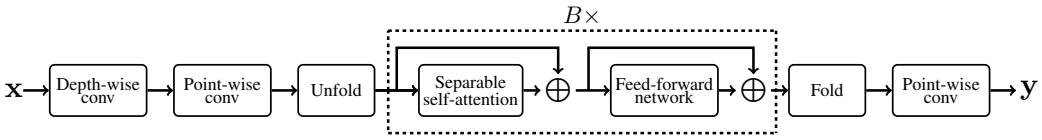


Figure 6: MobileViTv2 **block**. Here, depth-wise convolution uses a kernel size of 3×3 to encode local representations. Similar to [4], unfolding and folding operations uses a patch height and width of two respectively. The separable self-attention and feed-forward layers are repeated $B\times$ before applying the folding operation.

Table 5: MobileViTv2 **architecture.** Here, d represents dimensionality of the input to the separable self-attention layer, B denotes the repetition of transformer block with separable self-attention inside the MobileViTv2 block (Fig. 6), and MV2 indicates MobileNetv2 block. Similar to MobileViTv1 block, we set kernel size as three and spatial dimensions of patch (height h and width w) as two in the MobileViTv2 block.

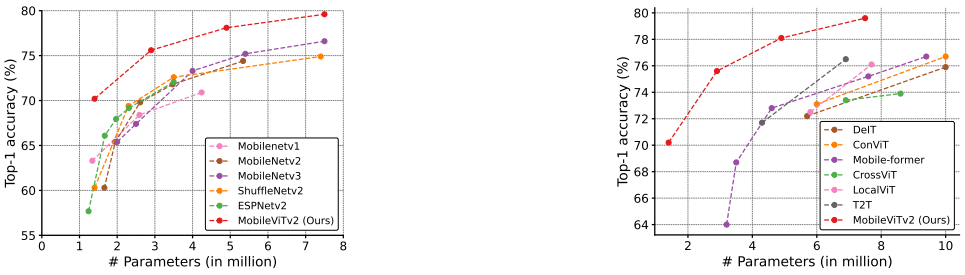
Layer	Output size	Output stride	Repeat	Output channels
Image	256×256	1		
Conv-3×3, $\downarrow 2$	128×128	2	1	32α
MV2			1	64α
MV2, $\downarrow 2$	64×64	4	1	128α
MV2			2	128α
MV2, $\downarrow 2$	32×32	8	1	256α
MobileViTv2 block (Fig. 6; $B = 2$)			1	$256 * \alpha$ ($d = 128\alpha$)
MV2, $\downarrow 2$	16×16	16	1	384α
MobileViTv2 block (Fig. 6; $B = 4$)			1	384α ($d = 192\alpha$)
MV2, $\downarrow 2$	8×8	32	1	512α
MobileViTv2 block (Fig. 6; $B = 3$)			1	512α ($d = 256\alpha$)
Global pool	1×1	256	1	512α
Linear				1000

Table 6: **Classification performance of `MobileViTv2` on the ImageNet-1k dataset.** Here, † indicates finetuning at higher resolution.

Row #	Model	Image size	Extra data	# Params ↓	FLOPs ↓	Top-1 ↑
R1	MobileViTv2-0.50	256^2	None	1.4 M	0.5 G	70.2
R2	MobileViTv2-0.50†	384^2	None	1.4 M	1.0 G	72.1
R3	MobileViTv2-0.75	256^2	None	2.9 M	1.0 G	75.6
R4	MobileViTv2-0.75†	384^2	None	2.9 M	2.3 G	77.0
R5	MobileViTv2-1.00	256^2	None	4.9 M	1.8 G	78.1
R6	MobileViTv2-1.00†	384^2	None	4.9 M	4.1 G	79.7
R7	MobileViTv2-1.25	256^2	None	7.5 M	2.8 G	79.6
R8	MobileViTv2-1.25†	384^2	None	7.5 M	6.3 G	80.9
R9	MobileViTv2-1.50	256^2	None	10.6 M	4.0 G	80.4
R10	MobileViTv2-1.50	256^2	ImageNet-21k-P	10.6 M	4.0 G	81.5
R11	MobileViTv2-1.50†	384^2	None	10.6 M	9.1 G	81.5
R12	MobileViTv2-1.50†	384^2	ImageNet-21k-P	10.6 M	9.1 G	82.6
R13	MobileViTv2-1.75	256^2	None	14.3 M	5.5 G	80.8
R14	MobileViTv2-1.75	256^2	ImageNet-21k-P	14.3 M	5.5 G	81.9
R15	MobileViTv2-1.75†	384^2	None	14.3 M	12.3 G	82.0
R16	MobileViTv2-1.75†	384^2	ImageNet-21k-P	14.3 M	12.3 G	82.9
R17	MobileViTv2-2.00	256^2	None	18.5 M	7.2 G	81.2
R18	MobileViTv2-1.75	256^2	ImageNet-21k-P	18.5 M	7.2 G	82.4
R19	MobileViTv2-2.00†	384^2	None	18.5 M	16.1 G	82.2
R20	MobileViTv2-1.50†	384^2	ImageNet-21k-P	18.5 M	16.1 G	83.4

Table 7: Performance of `MobileViTv2` on the ImageNet-21k-P validation set.

Width factor α	# Params ↓	FLOPs ↓	Top-1 ↑	Top-5 ↑
1.50	17.9 M	4.1 G	44.5	74.5
1.75	22.7 M	5.5 G	45.8	75.8
2.00	28.1 M	7.2 G	46.4	76.6



(a) Comparison with light-weight CNNs

(b) Comparison with light-weight ViTs

Figure 7: **Comparison with light-weight CNN- and ViT-based models.** `MobileViTv2` is smaller and better, which is desirable for mobile devices.

C Comparisons with light-weight networks on the ImageNet-1k dataset

Comparison with light-weight CNNs. Fig. 7a shows that `MobileViTv2` outperforms light-weight CNNs across different network sizes (MobileNetv1 [27], MobileNetv2 [24], ShuffleNetv2 [37], ESPNetv2 [49], and MobileNetv3 [25]).

Comparison with light-weight ViTs. Fig. 7b shows that `MobileViTv2` achieves better performance than previous light-weight ViT-based models acorss different network sizes (DeIT [2], T2T [50], CrossViT [51], LocalViT [52], ConViT [53], and Mobile-former [35]).

D Visualizations of separable self-attention scores

The `MobileViTv2` block, Fig. 6, unfolds the input $\mathbf{x} \in \mathbb{R}^{d \times H \times W}$ to obtain $\mathbf{x_u} \in \mathbb{R}^{d, \times M \times N}$, where $N = \frac{HW}{hw}$ are the number of patches, each patch with width w and height h ($M = hw$ pixels per patch). This unfolded feature map is fed to separable self-attention module to learn non-local representations. To better understand how separable self-attention processes $\mathbf{x_u}$, we visualize context scores $\mathbf{c_s}$.

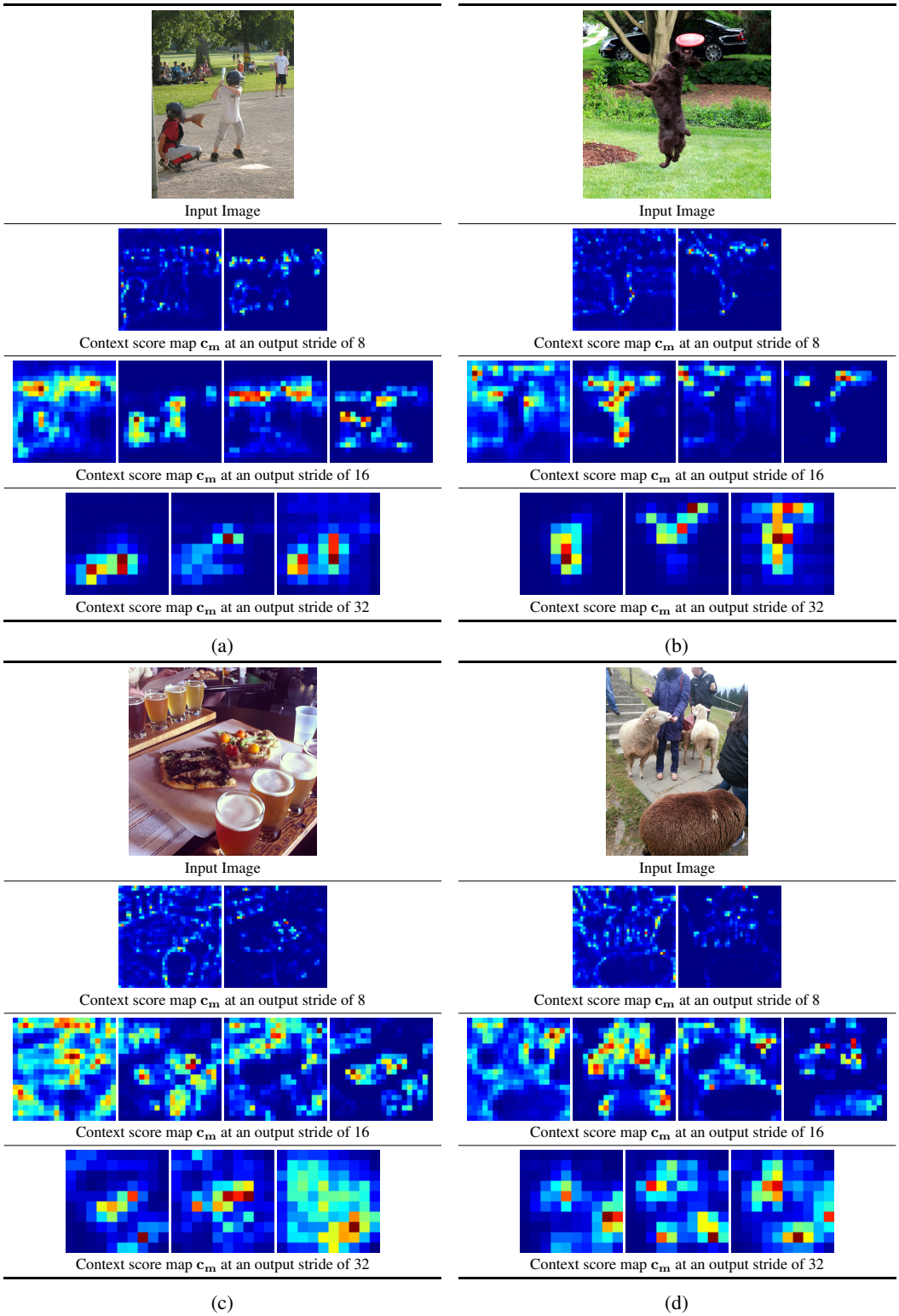


Figure 8: Layer-wise visualization of context score maps $\mathbf{c_m}$ at different output strides. Recall that `MobileViTv2` (Fig. 6 and Table 5) applies $B = 2$, $B = 4$, and $B = 3$ separable self-attention layers at an output strides of 8, 16, and 32 respectively. Therefore, we have 2, 4, and 8 context score maps at an output stride of 8, 16, and 32 respectively

The separable self-attention in `MobileViTv2` block computes context scores $\mathbf{c_s}$ for M pixels simultaneously across N patches. Therefore, $\mathbf{c_s}$ has a dimensions of $M \times N$. To visualize context scores, we fold $\mathbf{c_s} \in \mathbb{R}^{M \times N}$ to the same spatial dimensions as the input and obtain context score map $\mathbf{c_m} \in \mathbb{R}^{H \times W}$. For ease of visualization, we scale $\mathbf{c_m}$ using min-max normalization.

The context score maps for different input images at different output strides of `MobileViTv2` model are shown in Fig. 8. These visualizations show that the proposed separable self-attention method is able to (1) aggregate information from entire image under different settings, including complex backgrounds, illumination & view-point changes, and different objects, and (2) learn high-, mid-, and low-level representations.

E `MobileViTv2`'s ablation studies on the ImageNet-1k dataset

In this section, we study the effect on different methods on the performance of `MobileViTv2` models, including augmentation methods.

Standard vs. advanced augmentation We study two different augmentation methods: (1) standard augmentation that uses Inception-style augmentation [54], i.e., random resized cropping and horizontal flipping and (2) advanced augmentation that uses RandAugment [55], CutMix [56], MixUp [57], and RandomErase [58] along with standard augmentation methods. The effect of these augmentations on the performance of `MobileViTv2` is shown in Figure 9. Smaller models (< 4.5 million parameters) benefit from standard augmentation while larger models (≥ 4.5 million parameters) benefit from advanced augmentation. For simplicity, we use advanced augmentation for all variants of `MobileViTv2` in this paper.

Loss functions CutMix and Mixup augmentations mixes the samples in a batch. As a result, each sample has multiple labels. Therefore, in presence of these augmentations, ImageNet classification can be thought as a multi-label classification task. Similar to [59], we trained `MobileViTv2` by minimizing binary cross-entropy loss. Unlike [59], we did not observe any improvements in the performance when cross-entropy loss with label smoothing is replaced with binary cross-entropy loss. Therefore, we use cross-entropy with label smoothing for training `MobileViTv2` models.

Effect of multiple latent tokens Similar to multi-head attention in transformers, the proposed separable self-attention can have multiple latent tokens. When we changed the number of latent tokens from 1 to 8, the performance improvements on the ImageNet-1k dataset were negligible (within ± 0.1 top-1 accuracy). Therefore, we use only one latent token in our experiments.

We note that changing the number of heads from 4 to 1 in multi-headed self-attention in the transformer block of the MobileViTv1-S architecture dropped the top-1 accuracy by 0.7%. This observation is similar to Vaswani et al. [5], who also found that multiple heads in multi-headed self-attention improve transformers performance on the task of neural machine translation.

Improving FLOP-efficiency via pixel- and patch-sampling The MobileViTv1 model [4] unfolds an input feature map into N patches, each patch with $M = hw$ pixels and applies a transformer block for each pixel in a patch independently, where h and w are patch's height and width respectively. Because pixels in a patch are spatially correlated, one can sub-sample m pixels from M pixels and learn non-local representations by applying self-attention layers on m pixels only. Such sub-sampling methods should help in reducing model FLOPs.

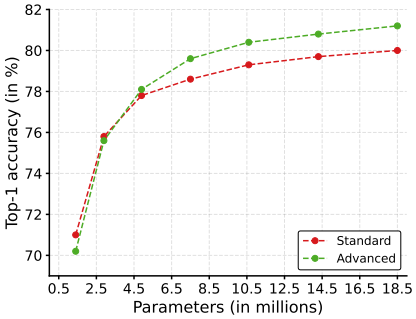


Figure 9: **Impact of data augmentation on the performance of `MobileViTv2` models on the ImageNet-1k dataset.** For smaller models (< 4.5 million parameters), standard augmentation works best while larger models (≥ 4.5 million parameters) benefit from advanced augmentation.

We tried following sampling methods at pixel- as well as patch-level:

- **Random sampling**, wherein m pixels (or n patches) from M pixels (or N patches) are randomly selected during training and uniformly during validation.
- **Top-m (or top-n) sampling**, wherein top-m pixels (or top-n patches) are selected based on their magnitude computed using L2 norm.
- **Uniform sampling**, wherein m pixels (or n patches) are sampled uniformly from M pixels (or N patches).

We found that these methods can reduce the FLOPs by $1.2\times$ to $1.6\times$ with little or no drop in top-1 accuracy on the ImageNet-1k dataset for both MobileViTv1 (with multi-headed self-attention) and `MobileViTv2` (with the proposed separable self-attention) models. However, these improvements in FLOPs did not translate to latency improvements on a mobile device. In fact, models with these sampling methods were significantly slower than the models without these methods. The high-latency of models with these sampling methods on mobile devices can be attributed to their high memory access cost, as these methods change the memory order of tensor. Because of their high-latency on mobile devices, we did not use these methods in the `MobileViTv2` model.

F `MobileViTv2` training configurations

Configurations for training and finetuning `MobileViTv2`-2.0 on the ImageNet-1k and ImageNet-21k-P datasets are given in Table 8 and Table 9 respectively while configurations for finetuning `MobileViTv2` on downstream tasks are given in Table 10.

Training config	`MobileViTv2`-2.0	
Dataset	ImageNet-1k	ImageNet-21k-P
# Training samples	1.28 M	11 M
# Validation samples	50 k	523 k†
Train resolution	256×256	256×256
Val resolution	256×256	256×256
RandAug	✓	✓
CutMix	✓	✓
MixUp	✓	✓
Random resized crop	✓	✓
Random horizontal flip	✓	✓
Random erase	✓	✓
Stochastic depth	✗	✗
Label smoothing	✓	✓
Loss	CE	CE
Optimizer	AdamW	AdamW
Weight decay	0.05	0.05
Scheduler	Cosine	Cosine
Warm-up iterations	20 k	None
Warm-up init LR	$1e^{-6}$	None
Warm-up scheduler	Linear	None
Base LR	0.002	0.0003
Epochs	300	80
Batch size	1024	4096
Layer-wise LR decay	✗	✗
Grad. clip	10	10
Exp. moving average	✓	✓
Weight init	Random	ImageNet-1k

Table 8: Configuration for training `MobileViTv2`-2.0 on the ImageNet-1k/22k-P datasets. † The validation set in ImageNet-21k-P does not overlap with ImageNet-1k validation set, and is created following Ridnik et al. [33].

Training config	`MobileViTv2-2.0`		
Dataset	ImageNet-1k	ImageNet-1k	ImageNet-1k
# Training samples	1.28 M	1.28 M	1.28 M
# Validation samples	50 k	50 k	50 k
Train resolution	384×384	256×256	384×384
Val resolution	384×384	256×256	384×384
Weight init	ImageNet-1k	ImageNet-21k-P	ImageNet-21k-P-1k†
RandAug	✗	✓	✓
CutMix	✗	✓	✓
MixUp	✗	✓	✓
Random resized crop	✓	✓	✓
Random horizontal flip	✓	✓	✓
Random erase	✗	✓	✓
Stochastic depth	✗	✗	✗
Label smoothing	✓	✓	✓
Loss	CE	CE	CE
Optimizer	SGD	SGD	SGD
Weight decay	$4e^{-5}$	$4e^{-5}$	$4e^{-5}$
Scheduler	Fixed	Cosine	Fixed
Warm-up iterations	None	None	None
Warm-up init LR	None	None	None
Warm-up scheduler	None	None	None
Base LR	0.001	0.01	0.001
Epochs	10	50	10
Batch size	128	256	128
Layer-wise LR decay	✗	✗	✗
Grad. clip	10	10	10
Exp. moving average	✓	✓	✓

Table 9: Configuration for finetuning `MobileViTv2-2.0` on the ImageNet-1k dataset. Here, † denotes that the ImageNet-21k-P model finetuned on the ImageNet-1k dataset at 256×256 image resolution is used for initializing the weights.

Training config	SSDLite-`MobileViTv2-1.75`	DeepLabv3-`MobileViTv2-1.75`	DeepLabv3-`MobileViTv2-1.75`
Dataset	MS-COCO	ADE20k	PASCAL VOC 2012
Extra Data	None	None	COCO
Task	Detection	Segmentation	Segmentation
# Training samples	117 k	20 k	128 k
# Validation samples	5 k	2 k	1.45 k
Train resolution	320×320	512×512	512×512
Val resolution	320×320	Shortest side 512	Shortest side 512
Weight init	ImageNet-1k	ImageNet-1k	ImageNet-1k
SSD Cropping	✓	✗	✗
Photometric distortion	✓	✓	✓
Random horizontal flip	✓	✓	✓
Resize	✓	✗	✗
Random short size resize	✗	✓	✓
Random Crop	✗	✓	✓
Random Gaussian blur	✗	✓	✓
Random rotation	✗	✓	✓
Loss	Smooth L1 + CE	CE	CE
Optimizer	AdamW	SGD	AdamW
Weight decay	0.05	$1e^{-4}$	0.05
Scheduler	Cosine	Cosine	Cosine
Warm-up iterations	500	None	500
Warm-up init LR	$9e^{-5}$	None	$5e^{-5}$
Warm-up scheduler	Linear	None	Linear
Base LR	0.0009	0.02	0.0005
Epochs	200	120	50
Batch size	128	16	128
Layer-wise LR decay	✗	✗	✗
Grad. clip	10	10	10
Exp. moving average	✓	✓	✓

Table 10: Configuration for finetuning `MobileViTv2` on downstream tasks. For Ade20k, we found that SGD was more stable as compared to AdamW across different `MobileViTv2` configurations, and therefore, we used SGD for finetuning on Ade20k dataset. The configurations for `MobileViTv2` with PSPNet are the same as Deeplabv3 on both PASCAL VOC and Ade20k datasets.