MicroNet: Improving Image Recognition with Extremely Low FLOPs

Yunsheng Li[1], Yinpeng Chen[2], Xiyang Dai[2], Dongdong Chen[2], Mengchen Liu[2],
Lu Yuan[2], Zicheng Liu[2], Lei Zhang[2], Nuno Vasconcelos[1]
[1] University of California San Diego [2] Microsoft
{yul554,nvasconcelos}@ucsd.edu,
{yiche,xidai,dochen,mengcliu,luyuan,zliu,leizhang}@microsoft.com

Abstract

This paper aims at addressing the problem of substantial performance degradation at extremely low computational cost (e.g. 5M FLOPs on ImageNet classification). We found that two factors, sparse connectivity and dynamic activation function, are effective to improve the accuracy. The former avoids the significant reduction of network width, while the latter mitigates the detriment of reduction in network depth. Technically, we propose micro-factorized convolution, which factorizes a convolution matrix into low rank matrices, to integrate sparse connectivity into convolution. We also present a new dynamic activation function, named Dynamic Shift Max, to improve the non-linearity via maxing out multiple dynamic fusions between an input feature map and its circular channel shift. Building upon these two new operators, we arrive at a family of networks, named MicroNet, that achieves significant performance gains over the state of the art in the low FLOP regime. For instance, under the constraint of 12M FLOPs, MicroNet achieves 59.4% top-1 accuracy on ImageNet classification, outperforming MobileNetV3 by 9.6%. Source code is at https://github.com/liyunsheng13/micronet.

1. Introduction

Recent progress in efficient CNN architectures [16, 13, 31, 12, 47, 28, 34] successfully decreases the computational cost of ImageNet classification from 3.8G FLOPs (ResNet-50 [11]) by two orders of magnitude to about 40M FLOPs (e.g. MobileNet, ShuffleNet), with a reasonable performance drop. However, they suffer from a significant performance degradation when reducing computational cost further. For example, the top-1 accuracy of MobileNetV3 degrades substantially from 65.4% to 58.0% and 49.8% when the computational cost drops from 44M to 21M and 12M MAdds, respectively. In this paper, *we aim at improving accuracy at the extremely low FLOP regime from 21M to 4M MAdds*, which marks the computational cost decrease of another order of magnitude (from 40M).

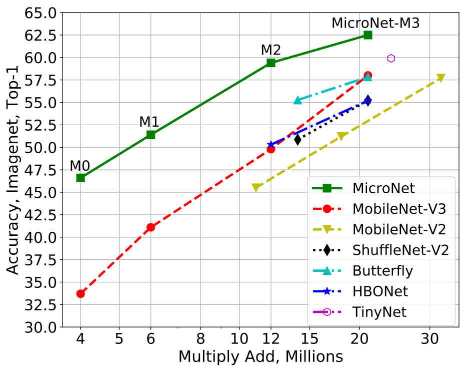

Figure 1. **Computational Cost (MAdds) vs. ImageNet Accuracy.** MicroNet significantly outperforms the state-of-the-art efficient networks at very low FLOPs (from 4M to 21M MAdds).

The problem of dealing with extremely low computational cost (4M–21M FLOPs) is very challenging, considering that 2.7M MAdds are consumed by a thin stem layer that contains a single 3×3 convolution with 3 input channels and 8 output channels over a 112×112 grid (stride=2). The remaining resources are too limited to design the convolution layers and 1,000 class classifier required for effective classification. As shown in Figure 1, a common strategy to reduce the width or depth of existing efficient CNNs (e.g. MobileNet [13, 31, 12] and ShuffleNet [47, 28]) results in a severe performance degradation. Note that we focus on new operator design while fixing the input resolution to 224×224 even for the budget of 4M FLOPs.

In this paper, we handle the extremely low FLOPs from two perspectives: *node connectivity* and *non-linearity*, which are related to the network width and depth. First, we show that lowering node connectivity to enlarge network width provides a good trade-off for a given computational budget. Second, we rely on improved layer nonlinearities to compensate for reduced network depth, which determines the non-linearity of the network. These two factors motivate the design of more efficient convolution and activation functions.

Regarding convolutions, we propose a *Micro-Factorized convolution (MF-Conv)* to factorize a pointwise convolution into two group convolution layers, where the group number G adapts to the number of channels C as:

$$G = \sqrt{C/R},$$

where R is the channel reduction ratio in between. As analyzed in Section 3.1, this equation achieves a good trade-off between the number of channels and node connectivity for a given computational cost. Mathematically, the pointwise convolution matrix is approximated by a block matrix ($G \times G$ blocks), whose blocks have rank-1. This guarantees minimal path redundancy (with only one path between any input-output pair) and maximum input coverage (per output channel), enabling more channels implementable by the network for a given computational budget.

With regards to non-linearities, we propose a new activation function, named *Dynamic Shift-Max (DY-Shift-Max)*, which non-linearly fuses channels with dynamic coefficients. In particular, the new activation forces the network to learn to fuse different circular channel shifts of the input feature maps, using coefficients that adapt to the input, and to select the best among these fusions. This is shown to enhance the representation power of the group factorization with little computational cost.

Based upon the two new operators (MF-Conv and DY-Shift-Max), we obtain a family of models, called *MicroNets*. Figure 1 summarizes the ImageNet performance, where MicroNets outperform the state-of-the-art by a large margin. In particular, our MicroNet models of 12M and 21M FLOPs outperform MobileNetV3 by 9.6% and 4.5% in terms of top-1 accuracy, respectively. For the extremely challenging regime of 6M FLOPs, MicroNet achieves 51.4% top-1 accuracy, outperforming by 1.6% over MobileNetV3, which is twice as complex (12M FLOPs).

Even though MicroNet is manually designed for theoretical FLOPs, it outperforms MobileNetV3 (which is searched over inference latency) with fast inference on edge devices. Furthermore, our MicroNet surpasses MobileNetV3 on object detection and keypoint detection, but uses substantially less computational cost.

2. Related Work

Efficient CNNs: MobileNets [13, 31, 12] decompose $k \times k$ convolution into a depthwise and a pointwise convolution. ShuffleNets [47, 28] further simplify pointwise convolution by group convolution and channel shuffle. [35] uses MixConv to mix up multiple kernel sizes in a convolution. [38] uses butterfly transform to approximate pointwise convolution. EfficientNet [34, 36] proposes a compound scaling method to scale depth/width/resolution uniformly. AdderNet [2] trades massive multiplications for cheaper additions. GhostNet [9] generates more feature maps from cheap linear transformations. Sandglass [48] alleviates information loss by flipping the structure of inverted residual block. [45, 1] train one network to support multiple sub-networks.
Dynamic Neural Networks: Dynamic networks improve the representation capability by adapting architectures or parameters to the input. [22, 26, 39, 41] perform dynamic routing within a super-network. [39] and [41] use reinforcement learning to learn a controller for skipping part of an existing model. MSDNet [15] allows early-exit for easy samples based on the prediction confidence. [46] searches for the optimal MSDNet. [21] learns dynamic routing across scales for semantic segmentation. [44] adapts image resolution to achieve efficient inference. Another line of work keeps the architectures fixed, but adapts parameters. HyperNet [8] uses another network to generate parameters for the main network. SENet [14] adapt weights over channels based on squeezing global context. SKNet [20] adapts attention over kernels with different sizes. Dynamic convolution [43, 4] aggregates multiple convolution kernels based on their attention. Dynamic ReLU [5] adapts slopes and intercepts of two linear functions in ReLU [29, 17]. [27] uses grouped fully connected layer to generate convolutional weights directly. [3] presents spatial-aware dynamic convolution. [32] proposes dynamic group convolution. [37] applies dynamic convolution on instance segmentation.

3. Micro-Factorized Convolution

The goal of Micro-Factorized convolution is to optimize the trade-off between the number of channels and node connectivity. Here, the *connectivity* E of a layer is defined as the number of paths per output node, where a path connects an input node and an output node.

3.1. Micro-Factorized Pointwise Convolution

We propose the use of group-adaptive convolution to factorize a pointwise convolution. For conciseness, we assume the convolution kernel $\boldsymbol{W}$ has the same number of input and output channels ($C_{in} = C_{out} = C$) and ignore bias terms. The kernel matrix $\boldsymbol{W}$ is factorized into two group-adaptive convolutions, where the number of groups G depends on the number of channels C, according to

$$\boldsymbol{W} = \boldsymbol{P}\boldsymbol{\Phi}\boldsymbol{Q}^T, \tag{1}$$

where $\boldsymbol{W}$ is a $C \times C$ matrix, $\boldsymbol{Q}$ is a $C \times \frac{C}{R}$ matrix that compresses the number of channels by a factor of R, and $\boldsymbol{P}$ is a $C \times \frac{C}{R}$ matrix that expands the number of channels back to C. $\boldsymbol{P}$ and $\boldsymbol{Q}$ are diagonal block matrices with G blocks, each implementing the convolution of a group of channels. $\boldsymbol{\Phi}$ is a $\frac{C}{R} \times \frac{C}{R}$ permutation matrix, shuffling channels similarly to [47]. The computational complexity of the factorized layer is $\mathcal{O} = \frac{2C^2}{RG}$. Figure 2-Left shows an example of the factorization, for $C = 18$, $R = 2$ and $G = 3$.

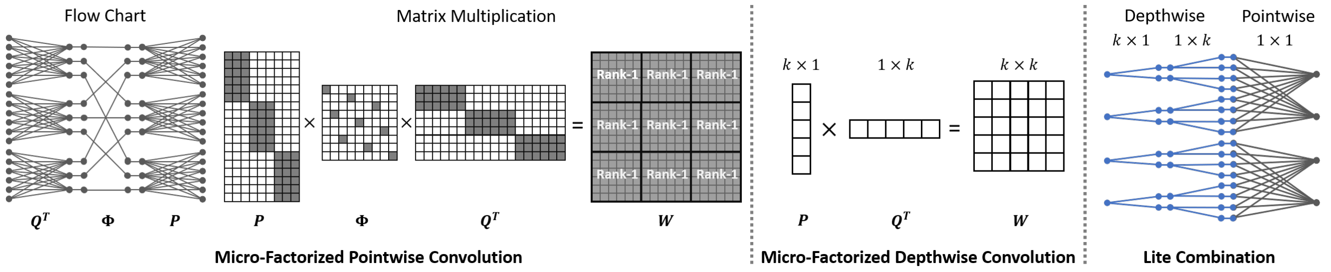

Figure 2. **Micro-Factorized pointwise and depthwise convolutions**. **Left:** factorizing a pointwise convolution into two group-adaptive convolutions, where the group number $G = \sqrt{C/R} = \sqrt{18/2} = 3$. The resulting matrix $\boldsymbol{W}$ can be divided into $G \times G$ blocks, of which each block has rank 1. **Middle:** factorizing a $k \times k$ depthwise convolution into a $k \times 1$ and a $1 \times k$ depthwise convolutions. **Right:** lite combination of Micro-Factorized pointwise and depthwise convolutions.

The $\frac{C}{R}$ channels of matrix $\boldsymbol{\Phi}$ are denoted *hidden channels*. The grouping structure limits the number of these channels that are affected by (affect) each input (output) of the layer. Specifically, each hidden channel connects to $\frac{C}{G}$ input channels and each output channel connects to $\frac{C}{RG}$ hidden channels. The number $E = \frac{C^2}{RG^2}$ of input-output connections per output channel denotes the *connectivity* E of the layer. When the computational budget $\mathcal{O} = \frac{2C^2}{RG}$ and the compression factor R are fixed, the number of channels C and connectivity E change with G in opposite directions,

$$C = \sqrt{\frac{\mathcal{O}RG}{2}}, \quad E = \frac{\mathcal{O}}{2G}. \tag{2}$$

This is illustrated in Figure 3. As the number of groups G increases, C increases but E decreases. The two curves intersect ($C = E$) when

$$G = \sqrt{C/R}, \tag{3}$$

in which case each output channel connects to all input channels exactly once ($E = C$). This guarantees that no redundant paths exist between any input-output pair (minimum path redundancy) while guaranteeing the existence of a path between each pair (maximum input coverage). Eq. 3 is a defining property of micro-factorized pointwise convolution. It implies that the number of groups G is *not* fixed, but defined by the number of channels C and the compression factor R, according to a square root law that optimally balances the number of channels C and input/output connectivity. Mathematically, the resulting convolution matrix $\boldsymbol{W}$ is divided into $G \times G$ rank-1 blocks, as shown in Figure 2-Left.

3.2. Micro-Factorized Depthwise Convolution

Figure 2-Middle shows how micro-factorization can be applied to a $k \times k$ depthwise convolution. The convolution kernel is factorized into a $k \times 1$ and a $1 \times k$ kernel. This follows Eq. 1, with per channel $k \times k$ kernel matrix $\boldsymbol{W}$,

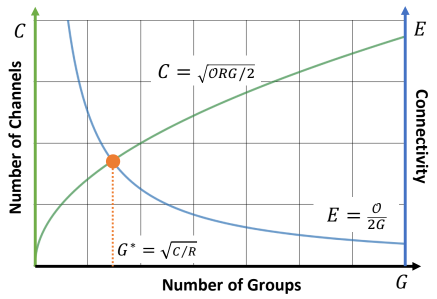

Figure 3. **Number of Channels** C **vs. Connectivity** E over number of groups G. We assume that the computational cost $\mathcal{O}$ and the reduction ratio R are fixed. Best viewed in color.

$k \times 1$ vector $\boldsymbol{P}$, $1 \times k$ vector $\boldsymbol{Q^T}$ and $\boldsymbol{\Phi}$ a scalar of value 1. This low rank approximation reduces the computational complexity from $\mathcal{O}(k^2C)$ to $\mathcal{O}(kC)$.

Combining Micro-Factorized Pointwise and Depthwise Convolutions: Micro-Factorized pointwise and depthwise convolutions can be combined in two different ways: (a) regular combination, and (b) lite combination. The former simply concatenates the two convolutions. The lite combination, shown in Figure 2-Right, uses Micro-Factorized depthwise convolutions to expand the number of channels, by applying multiple spatial filters per channel. It then applies one group-adaptive convolution to fuse and squeeze the number of channels. Compared to its regular counterpart, it spends more resources on learning spatial filters (depthwise) by saving channel fusion (pointwise) computations, which is empirically validated to be more effective for implementation of lower network layers.

4. Dynamic Shift-Max

So far, we have discussed the design of efficient static networks, which do not change their weights according to the input. We now introduce dynamic Shift-Max (DY-Shift-Max), a new dynamic non-linearity that strengthens connections between the groups created by micro-factorization.

Output	M0				M1				M2				M3			
	Block	k	C	$\frac{C}{R}$	Block	k	C	$\frac{C}{R}$	Block	k	C	$\frac{C}{R}$	Block	k	C	$\frac{C}{R}$
112×112	stem	3	4	2	stem	3	6	3	stem	3	8	4	stem	3	12	4
56×56	Micro-A	3	16	8	Micro-A	3	24	8	Micro-A	3	32	12	Micro-A	3	48	16
28×28	Micro-A	3	32	12	Micro-A	3	32	16	Micro-A	3	48	16	Micro-A	3	64	24
									Micro-B	3	144	24	Micro-B	3	144	24
14×14	Micro-B	5	64	16	Micro-B	5	96	16	Micro-C	5	192	32	Micro-C	3	192	32
	Micro-C	5	128	32	Micro-C	5	192	32	Micro-C	5	192	32	Micro-C	5	192	32
									Micro-C	5	384	64	Micro-C	5	384	64
													Micro-C	5	480	80
													Micro-C	5	480	80
7×7	Micro-C	5	256	64	Micro-C	5	384	64	Micro-C	5	576	96	Micro-C	5	720	120
	Micro-C	3	384	96	Micro-C	3	576	96	Micro-C	3	768	128	Micro-C	3	720	120
													Micro-C	3	864	144
1×1	avg pool → 2fc → softmax															
	4M MAdds, 1.0M Param				6M MAdds, 1.8M Param				12M MAdds, 2.4M Param				21M MAdds, 2.6M Param			

Table 1. **MicroNet Architectures**. "stem" refers to the stem layer. "Micro-A", "Micro-B", and "Micro-C" refers to three Micro-Blocks (see section 5.1 and Figure 4 for more details). k is the kernel size, C is the number of output channels, R is the channel reduction ratio in Micro-Factorized pointwise convolution. Note that for "Micro-A" (see Figure 4a), C is the number of output channels in Micro-Factorized depthwise convolution, $\frac{C}{R}$ is the number of output channels for the block.

This is complementary to Micro-Factorized pointwise convolution, which focuses on connections within a group.

Let $\boldsymbol{x} = \{x_i\}$ $(i = 1, \dots, C)$ denote an input vector (or tensor) with C channels that are divided into G groups of $\frac{C}{G}$ channels each. The j-group circular shift (shifting $j\frac{C}{G}$ channels) of $\boldsymbol{x}$ is the vector $\hat{\boldsymbol{x}}^j$ such that $\hat{x}_i^j = x_{(i+j\frac{C}{G}) \bmod C}$. Dynamic Shift-Max outputs the maximum of K fusions, each of which combines multiple (J) group shifts as:

$$y_i = \max_{1 \leq k \leq K} \{\sum_{j=0}^{J-1} a_{i,j}^k(\boldsymbol{x}) x_{(i+j\frac{C}{G}) \bmod C}\}, \tag{4}$$

where $a_{i,j}^k(\boldsymbol{x})$ is a dynamic weight, i.e. a weight that depends on the input $\boldsymbol{x}$. It is implemented as a hyper-function (with CJK output dimension) that consists of a sequence of average pooling, two fully connected layers, and a sigmoid layer, as in Squeeze-and-Excitation [16].

In this way, DY-Shift-Max implements two forms of non-linearity: it (a) outputs the maximum of K fusions of J groups, and (b) weighs each fusion by a dynamic parameter $a_{i,j}^k(\boldsymbol{x})$. The first non-linearity is complementary to Micro-Factorized pointwise convolution, which focuses on connectivity within each group, strengthening the connections between groups. The second enables the network to tailor this strengthening to the input $\boldsymbol{x}$. The two operations increase the representation power of the network, compensating for the loss inherent to the reduced number of layers.

DY-Shift-Max synthesizes CJK weights $a_{i,j}^k(\boldsymbol{x})$ from input $\boldsymbol{x}$. Its computational complexity is a sum of (a) average pooling $\mathcal{O}(HWC)$, (b) generation of the $a_{i,j}^k(\boldsymbol{x})$ weights $\mathcal{O}(C^2JK)$, and (c) application of dynamic Shift-Max per channel and spatial location $\mathcal{O}(HWCJK)$. This leads to a light-weight model when J and K are small. Empirically, a good trade-off between classification performance and complexity is achieved when $J = 2$ and $K = 2$.

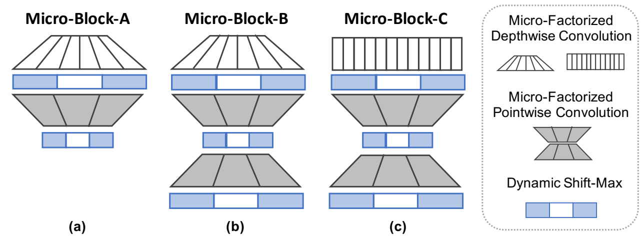
Micro-Block-A
Micro-Block-B
Micro-Block-C
Micro-Factorized Depthwise Convolution
Micro-Factorized Pointwise Convolution
Dynamic Shift-Max
(a)
(b)
(c)

Figure 4. **Diagram of three Micro-Blocks. (a) Micro-Block-A** that uses the lite combination of Micro-Factorized pointwise and depthwise convolutions (see Figure 2-Right). **(b) Micro-Block-B** that connects Micro-Block-A and Micro-Block-C. **(c) Micro-Block-C** that uses the regular combination of Micro-Factorized pointwise and depthwise convolutions. See Table 1 for their usage.

5. MicroNet

Below we describe in detail the design of MicroNet, using Micro-Factorized convolution and dynamic Shift-Max.

5.1. Micro-Blocks

MicroNet models consist of three Micro-Blocks of Figure 4, which combine Micro-Factorized pointwise and depthwise convolutions in different ways. All of the Micro-Blocks use the dynamic Shift-Max activation function.

Micro-Block-A: The Micro-Block-A of Figure 4a, uses the lite combination of Micro-Factorized pointwise and depthwise convolutions of Figure 2-Right. It expands the number of channels with Micro-Factorized depthwise convolution, and compresses them with a group-adaptive convolution. It is best suited to implement lower network layers of higher resolution (e.g. 112×112 or 56×56).

Micro-Block-B: The Micro-Block-B of Figure 4b is used to connect Micro-Block-A and Micro-Block-C. Different from Micro-Block-A, it uses a full Micro-Factorized pointwise convolution, which includes two group-adaptive convolutions. Hence, it both compresses and expands the number of channels. All MicroNet models have a single Micro-Block-B (see Table 1).

Micro-Block-C: The Micro-Block-C of Figure 4c implements the regular combination of Micro-Factorized depthwise and pointwise convolutions. It is best suited for the higher network layers (see Table 1) since it assigns more computation to channel fusion (pointwise) than the lite combination. The skip connection is used when the input and output have the same dimension.

Each micro-block has three hyper-parameters: kernel size k, number of output channels C, compression factor R of the bottleneck of Micro-Factorized pointwise convolution. Note that the number of groups in the two group-adaptive convolutions is determined by Eq. 3.

5.2. Architectures

All models are manually designed to optimize for FLOPs, which is a theoretical and device independent metric. We hope this can be leveraged by new hardware design and optimization for edge devices. We aware that FLOPs is *not* equivalent to inference latency at existing hardware and will show in experiment that MicroNet also improves accuracy and latency. We propose four models (M0, M1, M2, M3) of different computational cost (4M, 6M, 12M, 21M MAdds) based on the Micro-Blocks above. Table 1 presents their full specification. These networks follow the same pattern from low to high layers: stem layer $\rightarrow$ Micro-Block-A $\rightarrow$ Micro-Block-B $\rightarrow$ Micro-Block-C. All models are handcrafted, without network architecture search (NAS). The network hyper-parameters are selected based on simple rules: R is fixed (4 for M0, 6 for MicroNet-M1,M2,M3), C increases from low to high levels, depth increases from M0 to M3. For the deepest model (M3), we only use one dynamic Shift-Max layer per block after the depthwise convolution. The stem layer includes a 3×1 convolution and a 1×3 group convolution, and is followed by a ReLU. The second convolution expands the number of channels.

5.3. Relation to Prior Work

MicroNet has various connections to the recent deep learning literature. It is related to the popular MobileNet [13, 31, 12] and ShuffleNet [47, 28] models. It shares the inverted bottleneck structure with MobileNet and the use of group convolution with ShuffleNet. In contrast, MicroNet differs from these models in both its convolutions and activation functions. First, it factorizes pointwise convolutions into group-adaptive convolutions, with the number of groups $G = \sqrt{C/R}$ that is channel adaptive and guarantees minimum path redundancy. Second, it factorizes depthwise convolution. Third, it relies on a novel activation function, dynamic Shift-Max, to strengthen group connectivity in a non-linear and input dependent manner. Dynamic Shift-Max itself generalizes the recently proposed dynamic ReLU [5] (i.e. dynamic ReLU is a special case where $J = 1$ and each channel is activated alone).

	Micro-Fac Conv			Shift-Max				
	DW	PW	Lite	static	dynamic	Param	MAdds	Top-1
Mobile						1.3M	10.6M	44.9
Micro	✓					1.7M	10.6M	46.4
	✓	✓				1.7M	10.6M	50.0
	✓	✓	✓			1.8M	10.5M	51.7
	✓	✓	✓	✓		1.9M	11.8M	54.4
	✓	✓	✓		✓	2.4M	12.4M	**58.5**

Table 2. **The path from MobileNet to MicroNet** evaluated on ImageNet classification. Here, we modify MobileNet-V2 such that it has similar FLOPs (about 10.6M) to three Micro-Factorized convolution options: depthwise (DW), pointwise (PW), and lite combination at low levels (Lite). We also compare dynamic Shift-Max with its static counterpart (static $a_{i,j}^k$ in Eq. 4).

6. Experiments

We evaluate MicroNet on three tasks: (a) image classification, (b) object detection, and (c) keypoint detection. In this section, the baseline MobileNetV3-Small in [12] is denoted as MobileNetV3, for conciseness.

6.1. ImageNet Classification

We start by evaluating the four MicroNet models (M0–M3) on the task of ImageNet [6] classification. ImageNet has 1000 classes, including 1,281,167 images for training and 50,000 images for validation.

All models are trained using an SGD optimizer with 0.9 momentum. The image resolution is 224×224. Data augmentation of standard random cropping and flipping is used. We use a mini-batch size of 512, and a learning rate of 0.02. Each model is trained for 600 epochs with cosine learning rate decay. The weight decay is 3e-5 and dropout rate is 0.05 for smaller MicroNets (M0, M1, M2). For the largest model M3, the weight decay is 4e-5 and dropout rate is 0.1.

6.1.1 Ablation Studies

Several ablations were performed using MicroNet-M2. All models are trained for 300 epochs. The default hyper parameters of DY-Shift-Max were set as J=2, K=2.

From MobileNet to MicroNet: Table 2 shows the path from MobileNet to MicroNet. Both share the inverted bottleneck structure. Here, we modify MobileNetV2 (without SE [14]) such that it has complexity (10.6M MAdds) similar to the static Micro-Factorized convolution variants of row 2–4. The introduction of Micro-Factorized depthwise convolutions improves performance by 1.5%. Micro-Factorized pointwise convolutions adds another 3.6% and the lite combination at lower layers adds a final gain of

G	Param	MAdds	Top-1
1	1.3M	10.6M	48.8
2	1.5M	10.5M	50.2
4	1.7M	10.6M	50.7
8	1.7M	10.6M	50.8
$G = \sqrt{C/R}$	1.8M	10.5M	**51.7**

(a) **Fixed group number** G.

$\lambda = \frac{G}{\sqrt{C/R}}$	Param	MAdds	Top-1
0.25	1.5M	10.5M	50.2
0.5	1.7M	10.6M	51.6
✶ 1.0	1.8M	10.5M	**51.7**
2.0	2.1M	10.5M	50.6
4.0	2.2M	10.7M	47.6

(b) **Adaptive group number** G.

Levels				
low	high	Param	MAdds	Top-1
		1.7M	10.6M	50.0
✶ ✓		1.8M	10.5M	**51.7**
✓	✓	2.0M	10.6M	51.2

(c) **Lite combination** at different levels

Table 3. **Ablations of Micro-Factorized convolution** on ImageNet classification. ✶ indicates the default choice for the rest of the paper.

1.7%. Altogether the three factorizations boost the top-1 accuracy of the static network from 44.9% to 51.7%. The addition of static and dynamic Shift-Max further increases this gain by 2.7% and 6.8% respectively, for a small increase in computation. This demonstrates that both *Micro-Factorized Convolutions* and *Dynamic Shift-Max* are effective and complementary mechanisms for the implementation of networks with extremely low computational cost.

Number of Groups G: Micro-Factorized pointwise convolution includes two group-adaptive convolutions, with a number of groups equal to the integer closest to $G = \sqrt{C/R}$. Table 3a compares this to networks of similar structure and FLOPs (about 10.5M MAdds), but using a fixed group cardinality. Group-adaptive convolution achieves higher accuracy, demonstrating the importance of its optimal trade-off between input/output connectivity and the number of channels.

This is further confirmed by Table 3b, which compares different options for the adaptive number of groups. This is controlled by a multiplier λ such that $G = \lambda\sqrt{C/R}$. Larger λ corresponds to more channels but less input/output connectivity (see Figure 3). The optimal balance is achieved when λ is between 0.5 and 1. Top-1 accuracy drops when λ either increases (more channels but less connectivity) or decreases (fewer channels but more connectivity) from this optimal point. The value $\lambda = 1$ is used in the remainder of the paper. Note that all models in Table 3b have similar computational cost (about 10.5M MAdds).

Lite combination: Table 3c compares using the lite combination of Micro-Factorized pointwise and depthwise convolutions (Figure 2-Right) at different layers. The lite combination is more effective for lower layers. Compared to the regular combination, it saves computations from channel fusion (pointwise) to allow more spatial filters (depthwise).

Activation functions: Dynamic Shift-Max is compared to three previous activation functions: ReLU [29], SE+ReLU [14], and dynamic ReLU [5]. Table 4 shows that dynamic Shift-Max outperforms all three by a clear margin (at least 2.5%). Note that dynamic ReLU is the special case of dynamic Shift-Max with $J = 1$ (see Eq. 4).

Location of DY-Shift-Max: Table 5 shows the top-1 accuracy when dynamic Shift-Max is implemented in different combinations of the three layers of the micro-blocks of Figure 4. When used in a single layer, dynamic Shift-Max should be placed after the depthwise convolution. This improves the top-1 accuracy over a network with ReLU activations by 4.2%. Adding a Dynamic Shift-Max activation at the Micro-Block output further improves performance by 2%. Finally, using three layers of Dynamic Shift-Max further increases the gain over the ReLU network to 6.8%.

Activation	Param	MAdds	Top-1	Top-5
ReLU[29]	1.8M	10.5M	51.7	74.3
SE[14]+ReLU	2.1M	10.9M	54.4	76.8
Dynamic ReLU [5]	2.4M	11.8M	56.0	78.0
Dynamic Shift-Max	2.4M	12.4M	**58.5**	**80.1**

Table 4. **Dynamic Shift-Max vs. other activation functions** on ImageNet classification. MicroNet-M2 is used.

	A_1	A_2	A_3	Param	MAdds	Top-1	Top-5
ReLU	–	–	–	1.8M	10.5M	51.7	74.3
Dynamic Shift-Max	✓	–	–	2.1M	11.3M	55.9	77.9
	–	✓	–	2.0M	10.6M	53.3	76.0
	–	–	✓	2.1M	11.2M	54.8	77.2
	✓	✓	–	2.2M	11.5M	56.6	78.3
	✓	–	✓	2.3M	12.2M	57.9	79.6
	–	✓	✓	2.2M	11.4M	55.5	77.8
	✓	✓	✓	2.4M	12.4M	**58.5**	**80.1**

Table 5. **Dynamic Shift-Max at different layers** evaluated on ImageNet. MicroNet-M2 is used. A_1, A_2, A_3 indicate three activation layers sequentially in Micro-Block-B and Micro-Block-C (see Figure 4). Micro-Block-A only includes A_1 and A_2.

J	K	Param	MAdds	Top-1	Top-5
1	1	2.1M	10.9M	54.4	76.8
2	1	2.2M	11.8M	55.9	78.2
✶ 2	2	2.4M	12.4M	58.5	80.1
2	3	2.6M	13.8M	58.1	79.7
1	2	2.2M	11.2M	55.5	77.6
✶ 2	2	2.4M	12.4M	58.5	80.1
3	2	2.6M	14.2M	59.0	**80.3**
3	3	2.8M	15.3M	**59.1**	**80.3**

Table 6. **Ablations of two hyper parameters in dynamic Shift-Max** (J, K in Eq. 4) on ImageNet classification. ✶ indicates the default choice for the rest of the paper.

Hyper-parameters in DY-Shift-Max: Table 6 shows the results of using different combinations of K and J in Eq. 4. We add a ReLU when $K = 1$ as only one element is left in the max operator. The baseline of the first row ($J = 1$, $K =$

1) is equivalent to SE+ReLU [14]. For fixed $J = 2$ (fusion of two groups), the best of two fusions ($K = 2$) is better than a single fusion ($K = 1$), but adding a third fusion does not help, since it only adds path redundancy. When K is fixed at $K = 2$ (best of two fusions), fusing more groups J is consistently better but requires more FLOPs. A good tradeoff is achieved with $J = 2$ and $K = 2$, enabling a gain of 4.1% over the baseline, for an additional 1.5M MAdds.

6.1.2 Comparison to Prior Networks

Table 7 compares MicroNet to the state-of-the-art models, which have complexity less than 24M FLOPs. As the prior works lack of reported results within 10M FLOPs budget, we extend the popular MobileNetV3 to 6M and 4M FLOPs as baseline, by using width multiplier 0.2 and 0.15 respectively. They share the same training setup with MicroNet.

To make comparison fair, two variations of M1–M3 (e.g. M3$^{\#}$ and M3) are used. The former (M3$^{\#}$) requires similar model size to but fewer FLOPs than the baseline (MobileNetV3 0.5×). The latter (M3) requires similar FLOPs but allows more parameters (up to 1M), best serving scenarios that FLOPs is more critical than memory. This is due to the difficulty to match both model size and FLOPs, except for the smallest model (M0). Note that M3$^{\#}$ has similar structure to M3, only shrinking the model size by reducing network width and parameters in dynamic Shift-Max.

In all cases, MicroNet outperforms all prior networks by a clear margin. For instance, MicroNet-M1$^{\#}$, M2$^{\#}$, M3$^{\#}$ outperform their MobileNetV3 counterpart by 8.3%, 8.4%, and 3.3%, respectively. Given another 1M budget on model size, MicroNet-M1, M2, M3 increase these gains by 2.0%, 1.2% and 1.2%, respectively. MicroNet-M0 outperforms MobileNetV3 0.15× by 12.9% (46.6% vs. 33.7%), demonstrating its better handle of cutting computational cost from 6M to 4M MAdds. In particular, the top-1 accuracy drops by 4.8% from MicroNet-M1 to M0, while the accuracy degrades by 7.4% from MobileNetV3 ×0.2 to ×0.15. When compared to recent MobileNet and ShuffleNet improvements, such as ButterflyTransforms [38] and TinyNet [10], MicroNet models have gains of more than 2.6% top-1 accuracy but use less FLOPs. This demonstrates the effectiveness of MicroNet at extremely low FLOPs.

6.1.3 Inference Latency

We also measure the inference latency of MicroNet on an Intel(R) Xeon(R) CPU E5-2620 v4 (2.10GHz). Following the common settings in [31, 12], we test under single-threaded mode with batch size 1. The average inference latency of 5,000 images (with resolution 224×224) is reported. Figure 5-Right shows the comparison between MicroNet and MobileNetV3-Small. To achieve similar performance, MicroNet clearly consumes less runtime than MobileNetV3. For example, MicroNet with 55% accuracy has a latency less than 7ms, while MobileNetV3 requires about 9.5ms. The accuracy-latency curve is slightly degraded when using MicroNet with fewer parameters (M1$^{\#}$, M2$^{\#}$, M3$^{\#}$), but it still outperforms MicroNetV3. Although the largest MicroNet model (M3) only slightly outperforms MobileNetV3 for the same latency, MicroNet gains significantly more improvement over MobileNetV3 when the latency decreases. In particular, at a latency of 4ms, MicroNet improves over MobileNetV3 by 10%, demonstrating its strength at low computational cost.

Model	#Param	MAdds	Top-1	Top-5
MobileNetV3 0.15×†	1.0M	4M	33.7	57.2
MicroNet-M0	1.0M	4M	**46.6**	**70.6**
MobileNetV3 0.2×†	1.2M	6M	41.1	65.2
MicroNet-M1$^{\#}$	1.2M	5M	49.4	72.9
MicroNet-M1	1.8M	6M	**51.4**	**74.5**
ShuffleNetV1 0.25× [47]	–	13M	47.3	–
MobileNetV3 0.35× [12]	1.4M	12M	49.8	–
HBONet (96×96) [19]	–	12M	50.3	73.8
MobileNetV3+BFT 0.5× [38]	–	15M	55.2	–
MicroNet-M2$^{\#}$	1.4M	11M	58.2	80.1
MicroNet-M2	2.4M	12M	**59.4**	**80.9**
HBONet (128×128) [19]	–	21M	55.2	78.0
ShuffleNetV2+BFT [38]	–	21M	57.8	–
MobileNetV3 0.5× [12]	1.6M	21M	58.0	–
TinyNet-E (106×106) [10]	2.0M	24M	59.9	81.8
MicroNet-M3$^{\#}$	1.6M	20M	61.3	82.9
MicroNet-M3	2.6M	21M	**62.5**	**83.1**

Table 7. **ImageNet [6] classification results**. $^{\#}$ stands for the MicroNet variation that has similar model size to but fewer MAdds than the corresponding MobileNetV3-Small baseline. † indicates our implementation under the same training setup with MicroNet. "–": not available in the original paper. Note that input resolution 224×224 is used for MicroNet and related works other than HBONet/TinyNet, whose input resolution is shown in the bracket.

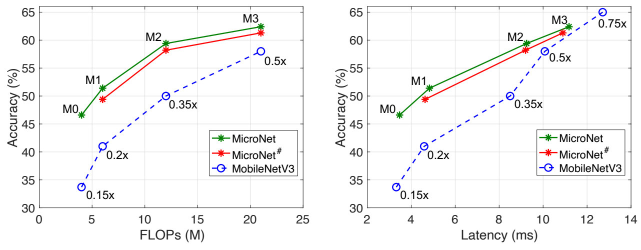

Figure 5. Evaluation on ImageNet classification. **Left**: *top-1 accuracy vs. FLOPs*. **Right**: *top-1 accuracy vs. latency*. Note that MobileNetV3 ×0.75 is added to facilitate the comparison. MicroNet outperforms MobileNetV3, especially at extremely low computational cost (more than 5% gain on top-1 accuracy when FLOPs is less than 15M or latency is less than 9ms).

6.1.4 Discussion

As shown in Figure 5, MicroNet clearly outperforms MobileNetV3 under the same FLOPs, but the gap shrinks under the same latency. This is due to two reasons. First,

Backbone	DET Framework	MAdds	mAP
MobileNetV3 $\times 1.0$	R-CNN	56M	25.9
MicroNet-M3		21M	**26.2**
MicroNet-M2		12M	22.7
MobileNetV3 $\times 1.0$	RetinaNet	56M	24.0
MicroNet-M3		21M	**25.4**
MicroNet-M2		12M	22.6

Table 8. **COCO object detection results**. All models are trained on `train2017` for 36 epochs (3×) and tested on `val2017`. MAdds is computed on image size 224×224.

Backbone	Head	Param	MAdds	AP	$AP^{0.5}$	$AP^{0.75}$	AP^{M}	AP^{L}
MobileNetV3 $\times 1.0$	Mobile-Blocks	2.1M	726.9M	57.1	83.8	63.7	55.0	62.2
MicroNet-M3	Micro-Blocks	2.2M	163.2M	**58.7**	**84.0**	**65.5**	**56.0**	**64.2**
MicroNet-M2	Micro-Blocks	1.8M	116.8M	54.9	82.0	60.3	53.2	59.6

Table 9. **COCO keypoint detection results**. All models are trained on `train2017` and tested on `val2017`. Input resolution 256×192 is used. The baseline applies MobileNetV3-Small ×1.0 as backbone and the head structure in [4] (which includes bilinear upsampling and inverted residual bottleneck blocks). Compared to the baseline, MicroNet-M3 has similar model size, consumes significantly less MAdds, but achieves higher accuracy.

different from MobileNetV3 that is optimized for latency by search, MicroNet is manually designed based on theoretical FLOPs. Second, the implementation of group convolution and dynamic Shift-Max are not optimized (we use PyTorch for implementation). We observe that the latency of group convolution is not proportionally reduced as the number of groups increases, and dynamic Shift-Max is significantly slower than convolution with the same FLOPs.

We believe that the runtime performance of MicroNet can be further improved by using hardware-aware architecture search to find latency friendly combination of Micro-Factorized convolution and dynamic Shift-Max. MicroNet can also leverage the improvement of optimization in group convolution [7] and dynamic Shift-Max to speed up. We will investigate these in the future work.

6.2. Object Detection

We evaluate the generalization ability of MicroNet on COCO object detection [25]. All models are trained on `train2017` and evaluated in mean Average Precision (mAP) on `val2017`. Following [9], MicroNet is used as a drop-in replacement for the backbone feature extractor in both the two-stage Faster R-CNN [30] with Feature Pyramid Networks (FPN) [23] and the one-stage RetinaNet [24]. All models are trained using SGD for 36 epochs (3×) from ImageNet pretrained weights with the hyper-parameters and data augmentation suggested in [40].

The detection results are shown in Table 8, where the backbone FLOPs are calculated using image size 224 × 224 as common practice. With significantly lower backbone FLOPs (21M *vs* 56M), MicroNet-M3 achieves higher mAP than MobileNetV3-Small ×1.0 both on Faster R-CNN and RetinaNet frameworks, demonstrating its capability to transfer to detection task.

6.3. Human Pose Estimation

We also evaluate MicroNet on COCO single person keypoint detection. All models are trained on `train2017` that includes $57K$ images and $150K$ person instances labeled with 17 keypoints, and evaluated on `val2017` that contains 5000 images, using the mean average precision (AP) over 10 object key point similarity (OKS) thresholds. Similar to object detection, two MicroNet models (M2, M3) are considered. The models are modified for the keypoint detection task, by increasing the resolution (×2) of a select set of blocks (all blocks with stride of 32). Each model contains a head with three micro-blocks (one of stride 8 and two of stride 4) and a pointwise convolution that generates heatmaps for 17 keypoints. Bilinear upsampling is used to increase the head resolution, and the spatial attention mechanism of [5] is used. Both models are trained from scratch for 250 epochs using Adam optimizer [18]. The human detection boxes are cropped and resized to 256×192. The training and testing follow the setup of [42, 33].

Table 9 compares MicroNet-M3 and M2 with a strong efficient baseline, which only requires 726.9M MAdds and 2.1M parameters. The baseline applies MobileNetV3-Small ×1.0 as backbone and mobile blocks (inverted residual bottleneck blocks) in the head (see [4] for details). Our MicroNet-M3 only consumes 22% (163.2M/726.9M) of the FLOPs used by the baseline but achieves higher performance, demonstrating its effectiveness for low-complexity keypoint detection. MicroNet-M2 provides a good handle for even lower complexity (116.8M FLOPs).

7. Conclusion

In this paper, we have presented MicroNet to handle extremely low computational cost. It builds on two proposed operators: Micro-Factorized convolution and Dynamic Shift-Max. The former balances between the number of channels and input/output connectivity via low rank approximations on both pointwise and depthwise convolutions. The latter fuses consecutive channel groups dynamically, enhancing both node connectivity and non-linearity to compensate for the depth reduction. A family of MicroNets achieve solid improvement for three tasks (image classification, object detection and human pose estimation) under extremely low FLOPs. We hope this work provides good baselines for efficient CNNs on multiple vision tasks.

Acknowledgement

This work was partially funded by NSF awards IIS-1924937, IIS-2041009.

References

[1] Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *ArXiv*, abs/1908.09791, 2019. 2

[2] Hanting 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. 2

[3] Jyun-Ruei Chen, Xijun Wang, Zichao Guo, X. Zhang, and J. Sun. Dynamic region-aware convolution. *ArXiv*, abs/2003.12243, 2020. 2

[4] 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. 2, 8

[5] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. In *ECCV*, 2020. 2, 5, 6, 8

[6] 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. 5, 7

[7] P. Gibson, J. Cano, J. Turner, E. J. Crowley, M. O'Boyle, and A. Storkey. Optimizing grouped convolutions on edge devices. In *2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 189–196, 2020. 8

[8] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. *ICLR*, 2017. 2

[9] 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. 2, 8

[10] Kai Han, Yunhe Wang, Qiulin Zhang, Wei Zhang, Chunjing XU, and Tong Zhang. Model rubika's cube: Twisting resolution, depth and width for tinynets. In *Advances in Neural Information Processing Systems*, volume 33, pages 19353–19364, 2020. 7

[11] 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. 1

[12] 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, 5, 7

[13] 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. 1, 2, 5

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 5, 6, 7

[15] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018. 2

[16] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$1mb model size. *CoRR*, abs/1602.07360, 2016. 1, 4

[17] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *The IEEE International Conference on Computer Vision (ICCV)*, 2009. 2

[18] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 8

[19] Duo Li, Aojun Zhou, and Anbang Yao. Hbonet: Harmonious bottleneck on two orthogonal dimensions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3316–3325, 2019. 7

[20] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[21] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[22] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems*, pages 2181–2191. 2017. 2

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 8

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 8

[25] 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. 8

[26] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 2

[27] Ningning Ma, X. Zhang, J. Huang, and J. Sun. Weightnet: Revisiting the design space of weight networks. volume abs/2007.11823, 2020. 2

[28] 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. 1, 2, 5

[29] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 2, 6

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 8

[31] 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, 5, 7

[32] Zhuo Su, Linpu Fang, Wen xiong Kang, D. Hu, M. Pietikäinen, and Li Liu. Dynamic group convolution for accelerating convolutional neural networks. In *ECCV*, August 2020. 2

[33] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 8

[34] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114, Long Beach, California, USA, 09–15 Jun 2019. 1, 2

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

[36] 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

[37] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, August 2020. 2

[38] 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, 7

[39] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[40] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 8

[41] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[42] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *European conference on computer vision*, 04 2018. 8

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

[44] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[45] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2019. 2

[46] Zhihang Yuan, Bingzhe Wu, Z. Liang, Shiwan Zhao, Weichen Bi, and Guangyu Sun. S2dnas: Transforming static cnn model for dynamic inference via neural architecture search. *ArXiv*, abs/1911.07033, 2019. 2

[47] 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. 1, 2, 5, 7

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