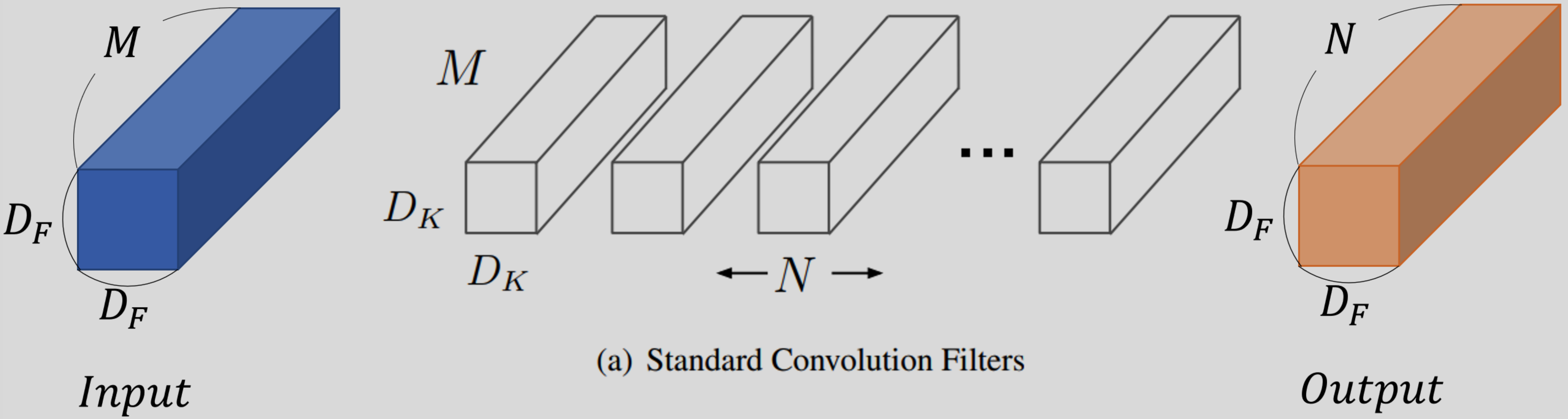
Computer Vision

Lecture 11: Efficient architectures

MobileNet

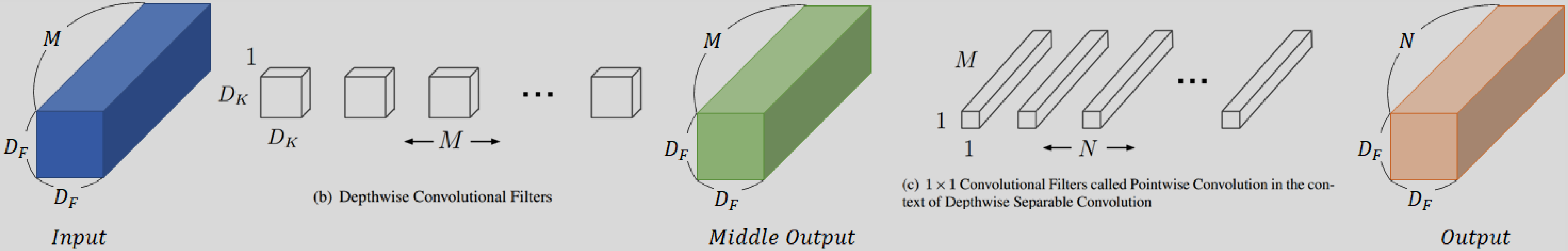


(a) Standard Convolution Filters

$$G_{convolution} = D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, ArXiv'17.

MobileNet



(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

$$G_{depthwise} = D_K \cdot D_K \cdot M \cdot D_F \cdot D_F$$

$$G_{pointwise} = M \cdot N \cdot D_F \cdot D_F$$

$$G_{dsc} = D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

$$Ratio_G = \frac{G_{dsc}}{G_{convolution}} = \frac{1}{N} + \frac{1}{D_K^2}$$

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, ArXiv'17.

MobileNet

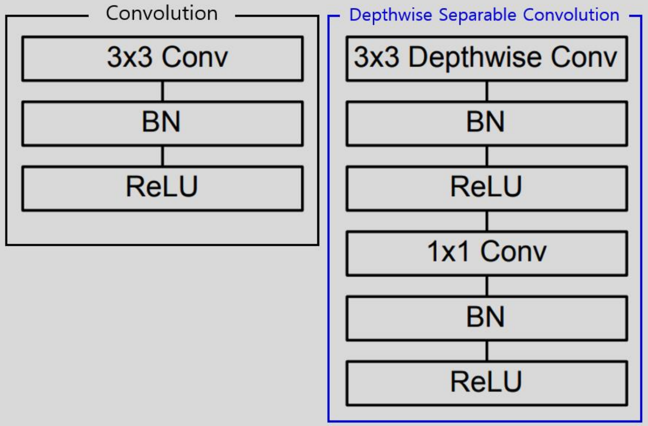


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, ArXiv’17.

MobileNet

Table 2. Resource Per Layer Type

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, ArXiv'17.

MobileNet

Width Multiplier : α

$$G_{dsc\cdot\alpha} = D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F$$

Resolution Multiplier : ρ

$$G_{dsc\cdot\rho} = D_K \cdot D_K \cdot M \cdot \rho D_F \cdot \rho D_F + M \cdot N \cdot \rho D_F \cdot \rho D_F$$

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, ArXiv'17.

MobileNet

Table 6. MobileNet Width Multiplier

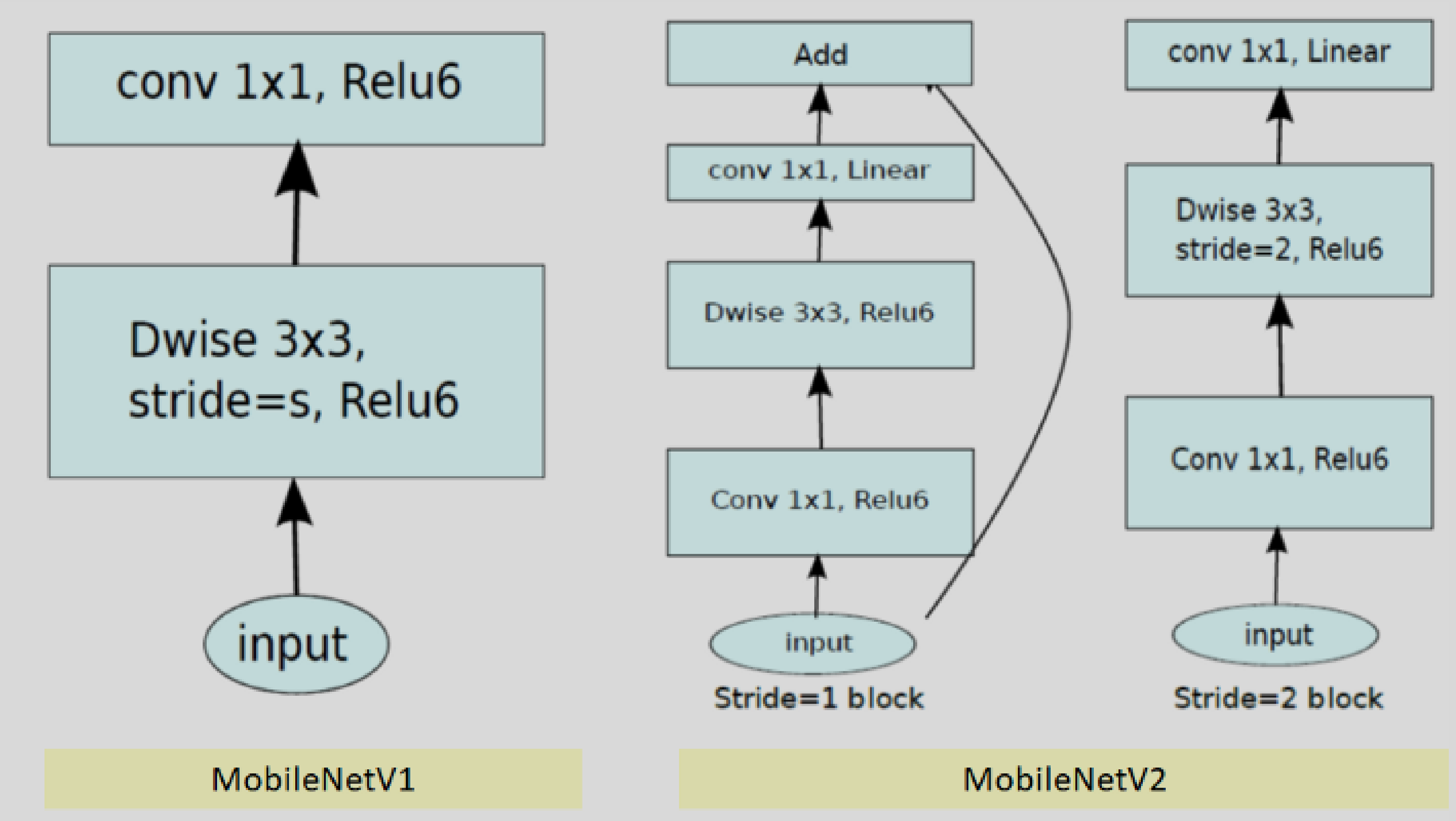
Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, ArXiv'17.

MobileNet V2



MobileNetV2: Inverted Residuals and Linear Bottlenecks, ArXiv'18.

MobileNet V2

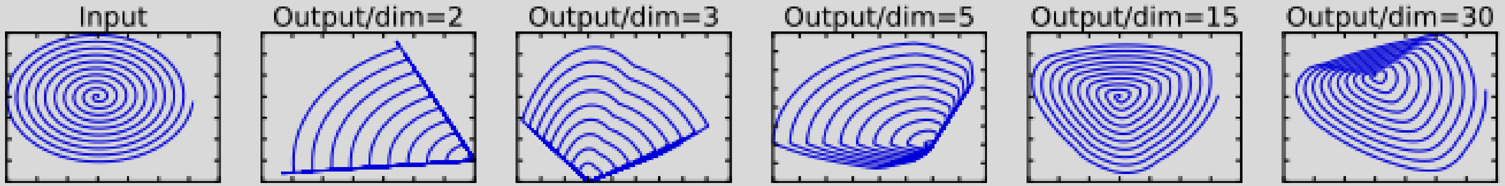


Figure 1: Examples of ReLU transformations of low-dimensional manifolds embedded in higher-dimensional spaces. In these examples the initial spiral is embedded into an n-dimensional space using random matrix T followed by ReLU, and then projected back to the 2D space using T^{-1}. In examples above $n = 2, 3$ result in information loss where certain points of the manifold collapse into each other, while for $n = 15$ to 30 the transformation is highly non-convex.

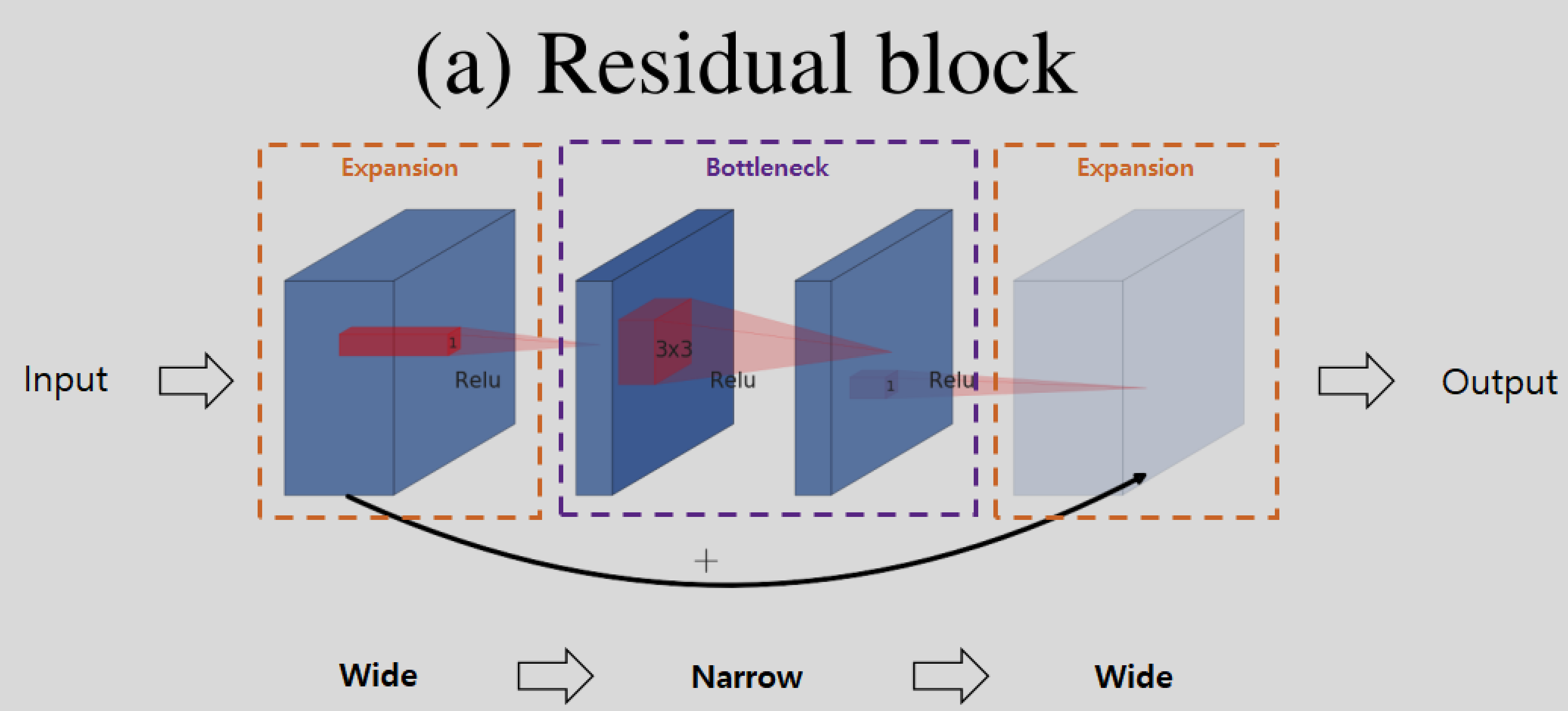
MobileNetV2: Inverted Residuals and Linear Bottlenecks, ArXiv’18.

MobileNet V2

- ReLU can reduce the information contained in the responses.
- MobileNetV2 involves the linear bottleneck layer that performs the linear transformation.

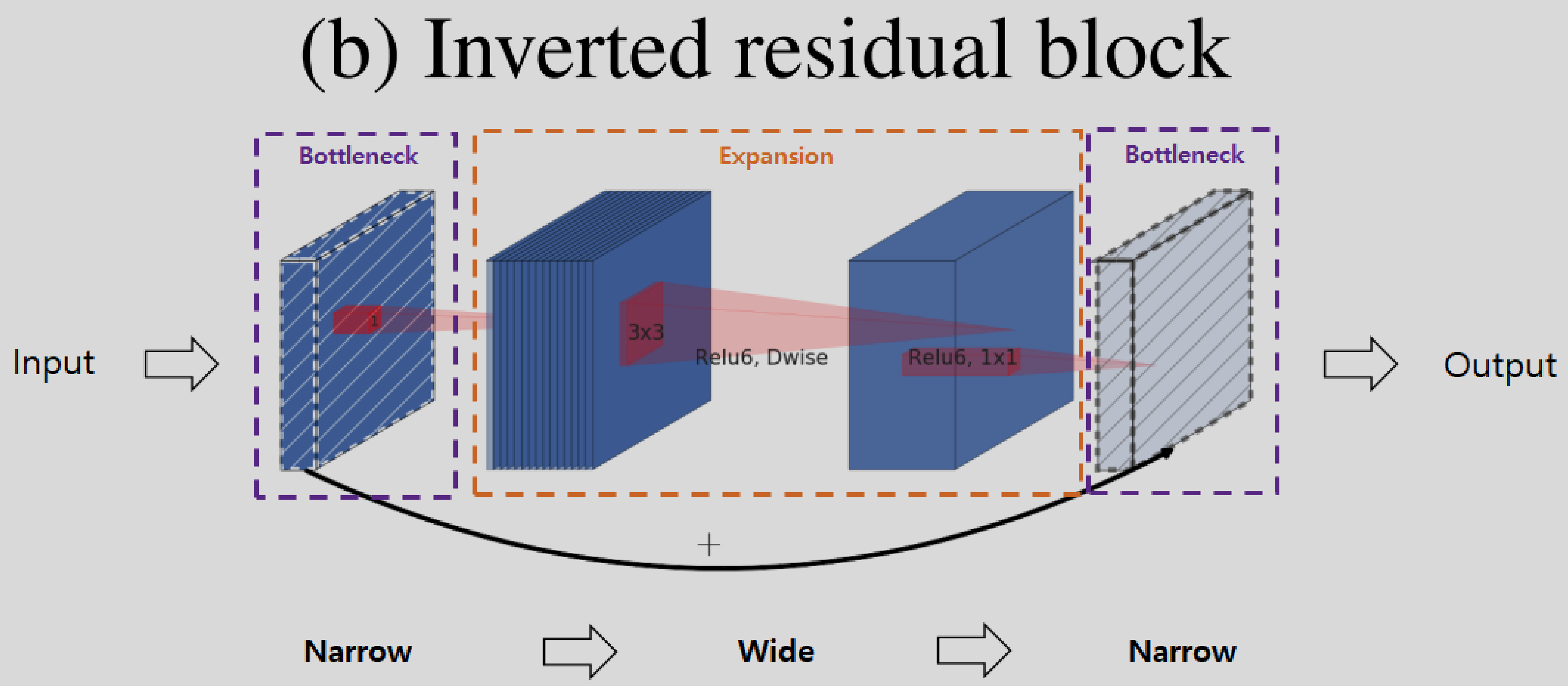
MobileNetV2: Inverted Residuals and Linear Bottlenecks, ArXiv'18.

MobileNet V2



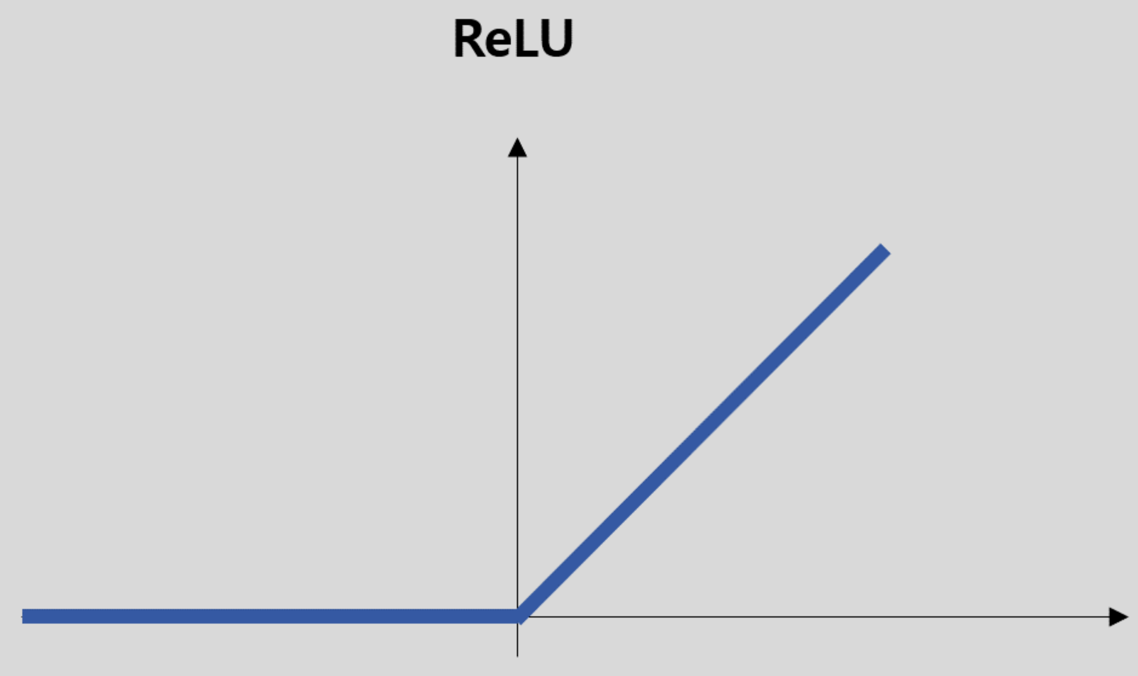
MobileNetV2: Inverted Residuals and Linear Bottlenecks, ArXiv'18.

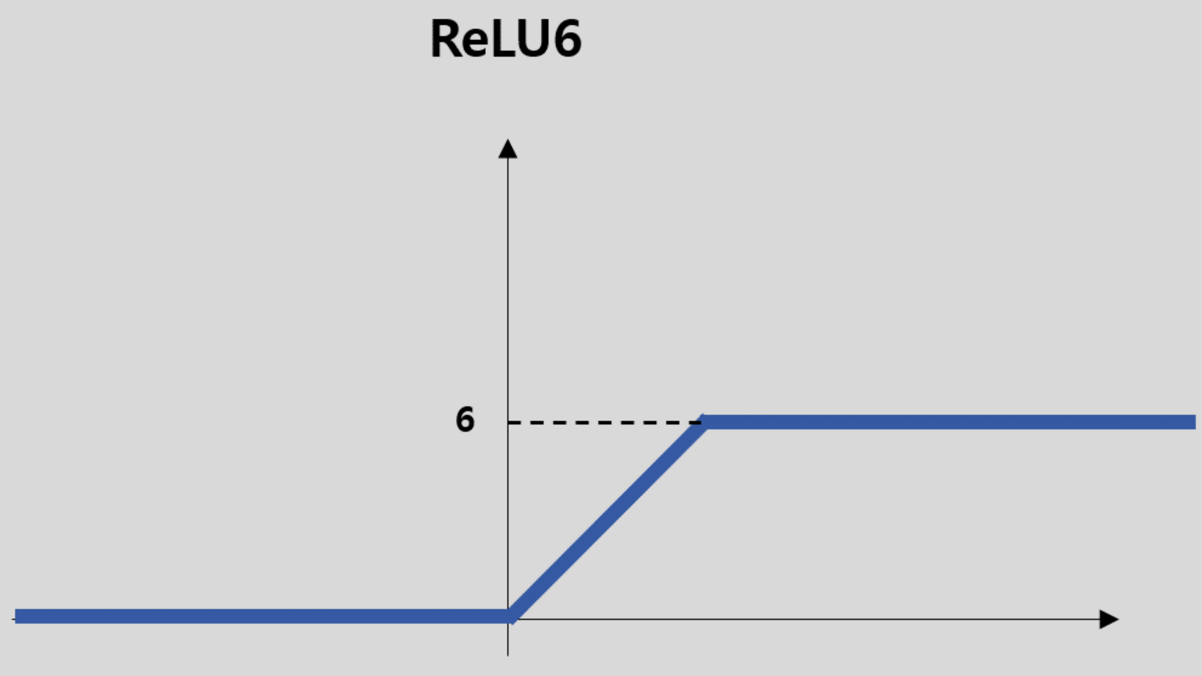
MobileNet V2



MobileNetV2: Inverted Residuals and Linear Bottlenecks, ArXiv'18.

MobileNet V2





MobileNetV2: Inverted Residuals and Linear Bottlenecks, ArXiv’18.

MobileNet V2

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

MobileNetV2: Inverted Residuals and Linear Bottlenecks, ArXiv'18.

MobileNet V2

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

MobileNetV2: Inverted Residuals and Linear Bottlenecks, ArXiv'18.

MobileNet V2

Size	MobileNetV1	MobileNetV2	ShuffleNet (2x,g=3)
112x112	64/1600	16/400	32/800
56x56	128/800	32/200	48/300
28x28	256/400	64/100	400/600K
14x14	512/200	160/62	800/310
7x7	1024/199	320/32	1600/156
1x1	1024/2	1280/2	1600/3
max	1600K	**400K**	600K

Table 3: The max number of channels/memory (in Kb) that needs to be materialized at each spatial resolution for different architectures. We assume 16-bit floats for activations. For ShuffleNet, we use $2x, g = 3$ that matches the performance of MobileNetV1 and MobileNetV2. For the first layer of MobileNetV2 and ShuffleNet we can employ the trick described in Section 5 to reduce memory requirement. Even though ShuffleNet employs bottlenecks elsewhere, the non-bottleneck tensors still need to be materialized due to the presence of shortcuts between the non-bottleneck tensors.

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Table 4: Performance on ImageNet, comparison for different networks. As is common practice for ops, we count the total number of Multiply-Adds. In the last column we report running time in milliseconds (ms) for a single large core of the Google Pixel 1 phone (using TF-Lite). We do not report ShuffleNet numbers as efficient group convolutions and shuffling are not yet supported.

MobileNetV2: Inverted Residuals and Linear Bottlenecks, ArXiv'18.

MobileNet V3

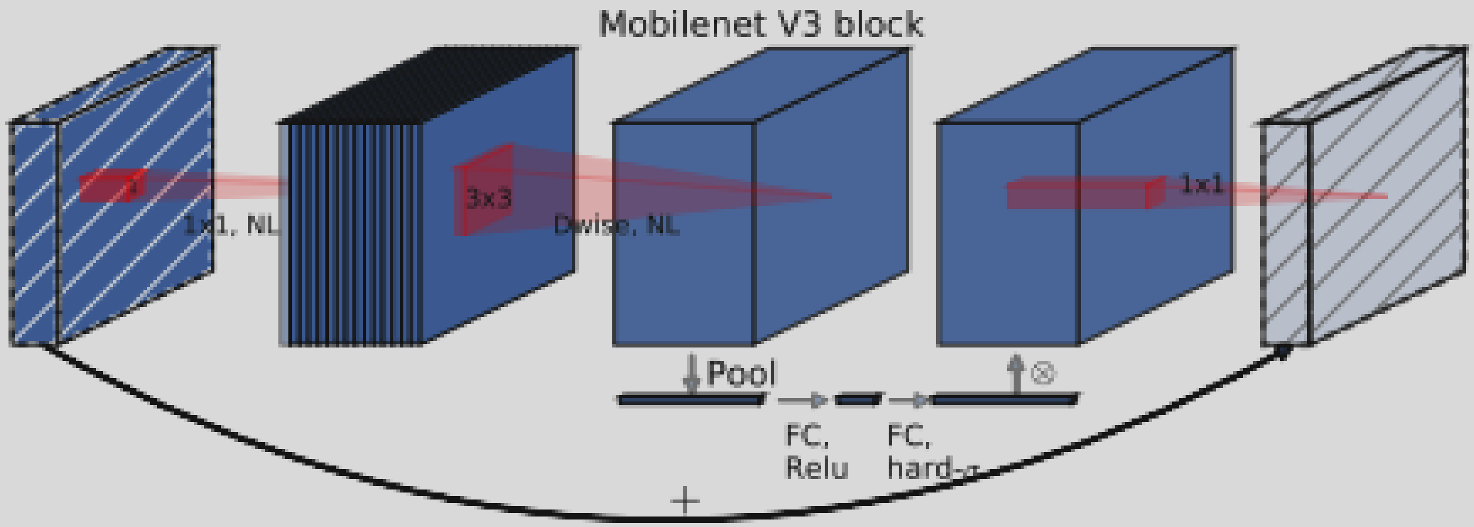
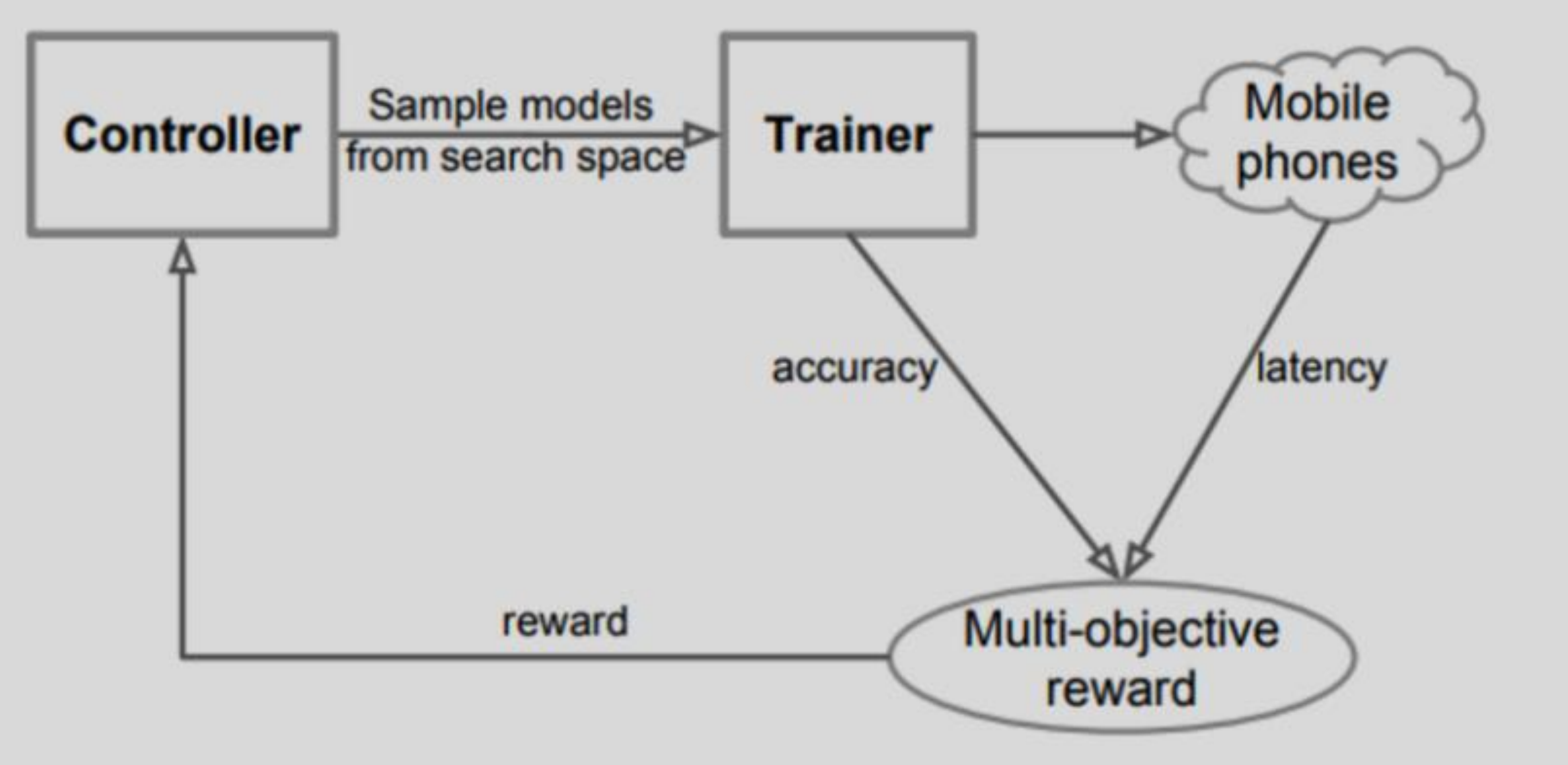


Figure 4. MobileNetV2 + Squeeze-and-Excite [20]. In contrast with [20] we apply the squeeze and excite in the residual layer. We use different nonlinearity depending on the layer, see section 5.2 for details.

Searching for MobileNetV3, ICCV'19.

Platform-aware NAS for entire architecture search



MnasNet: Platform-Aware Neural Architecture Search for Mobile, CVPR'19.

NetAdapt for each layer

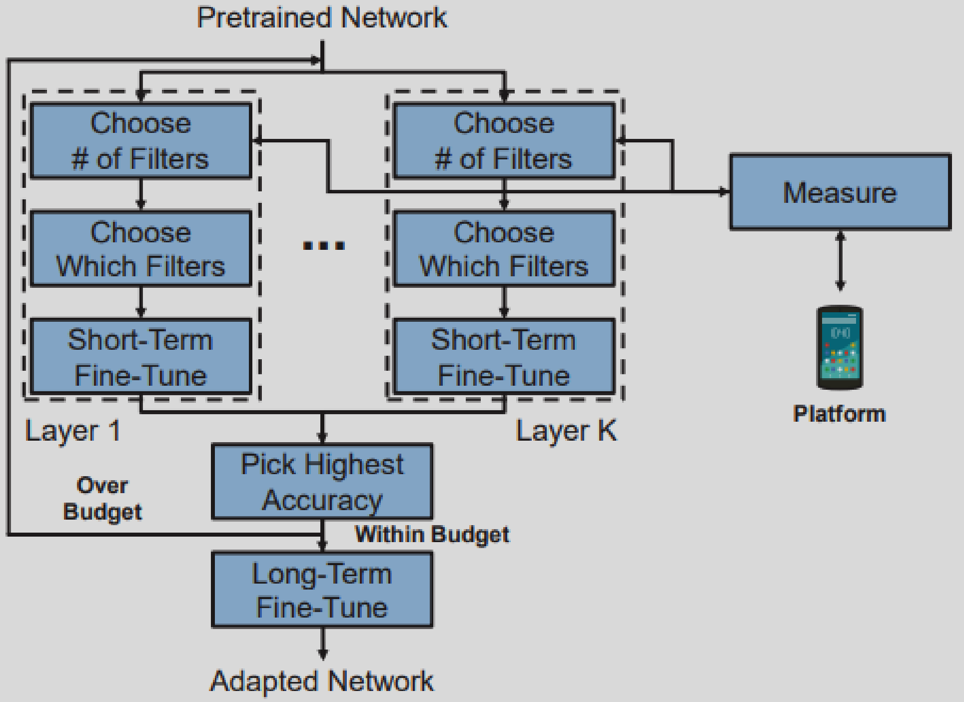


Fig. 2. This figure visualizes the algorithm flow of NetAdapt. At each iteration, NetAdapt decreases the resource consumption by simplifying (i.e., removing filters from) one layer. In order to maximize accuracy, it tries to simplify each layer individually and picks the simplified network that has the highest accuracy. Once the target budget is met, the chosen network is then fine-tuned again until convergence.

NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications, ECCV'18.

MobileNet V3

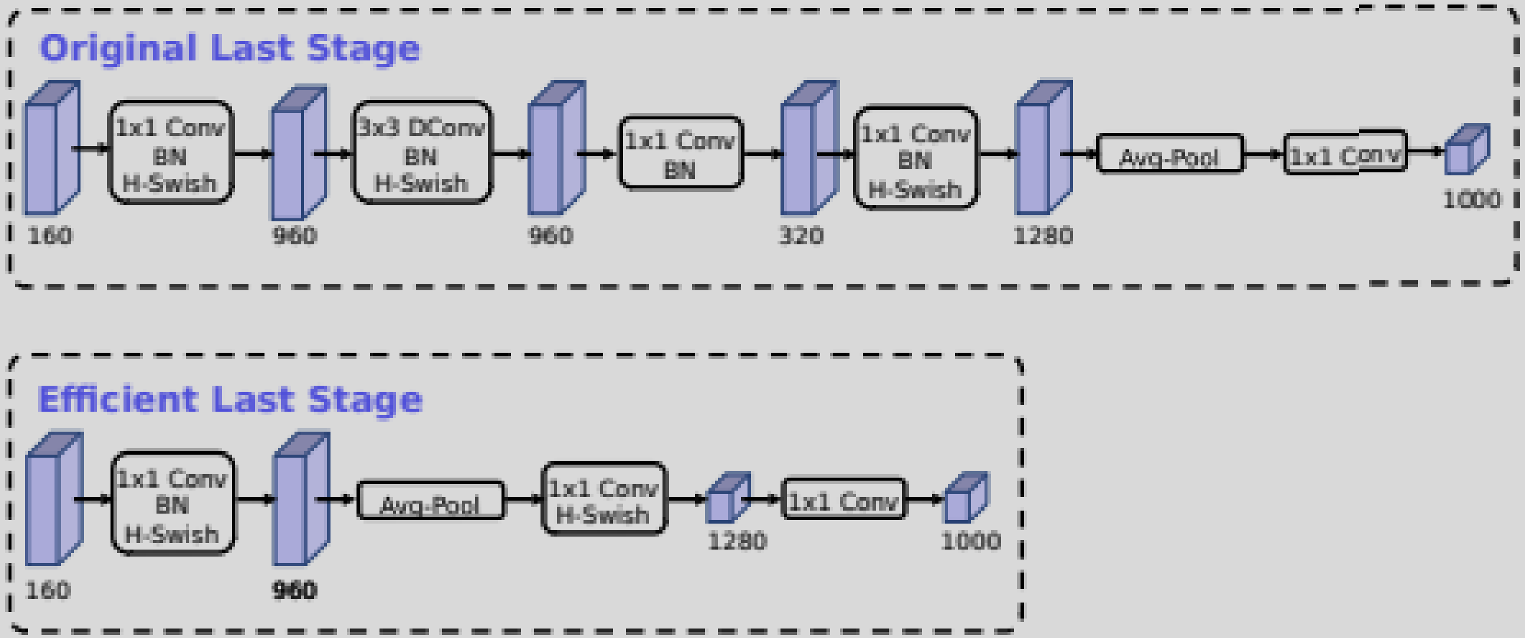


Figure 5. Comparison of original last stage and efficient last stage. This more efficient last stage is able to drop three expensive layers at the end of the network at no loss of accuracy.

Searching for MobileNetV3, ICCV'19.

MobileNet V3

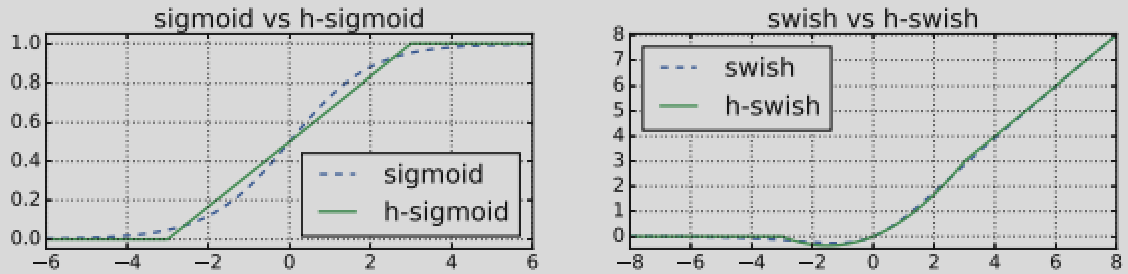


Figure 6. Sigmoid and swish nonlinearities and ther "hard" counterparts.

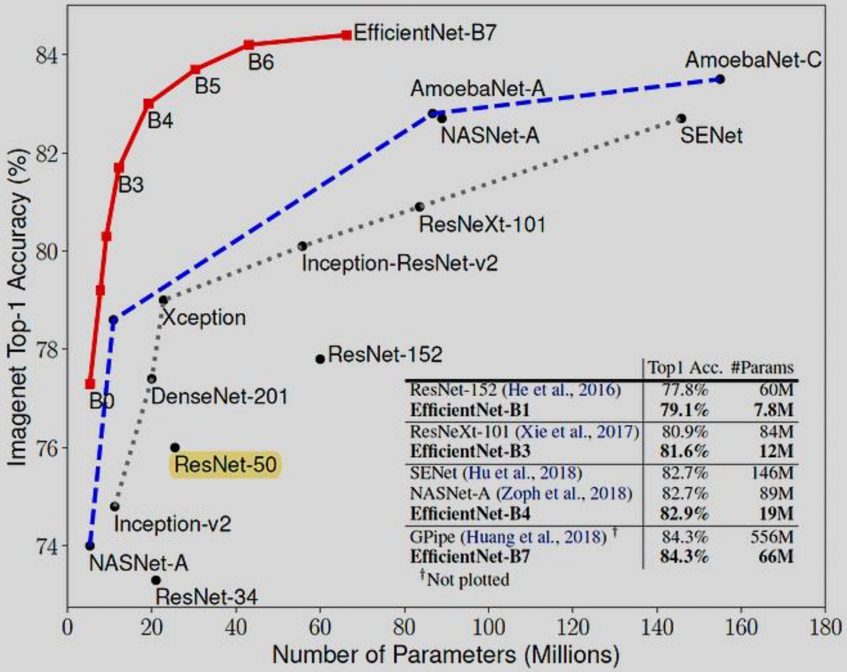
$$Swish(x) = x * sigmoid(x)$$

$$H - ReLU6(x) = \frac{ReLU6(x+3)}{6}$$

$$H - Swish = x * H - ReLU(6) = x * \frac{ReLU6(x+3)}{6}$$

Searching for MobileNetV3, ICCV'19.

EfficientNet



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML'19.

EfficientNet

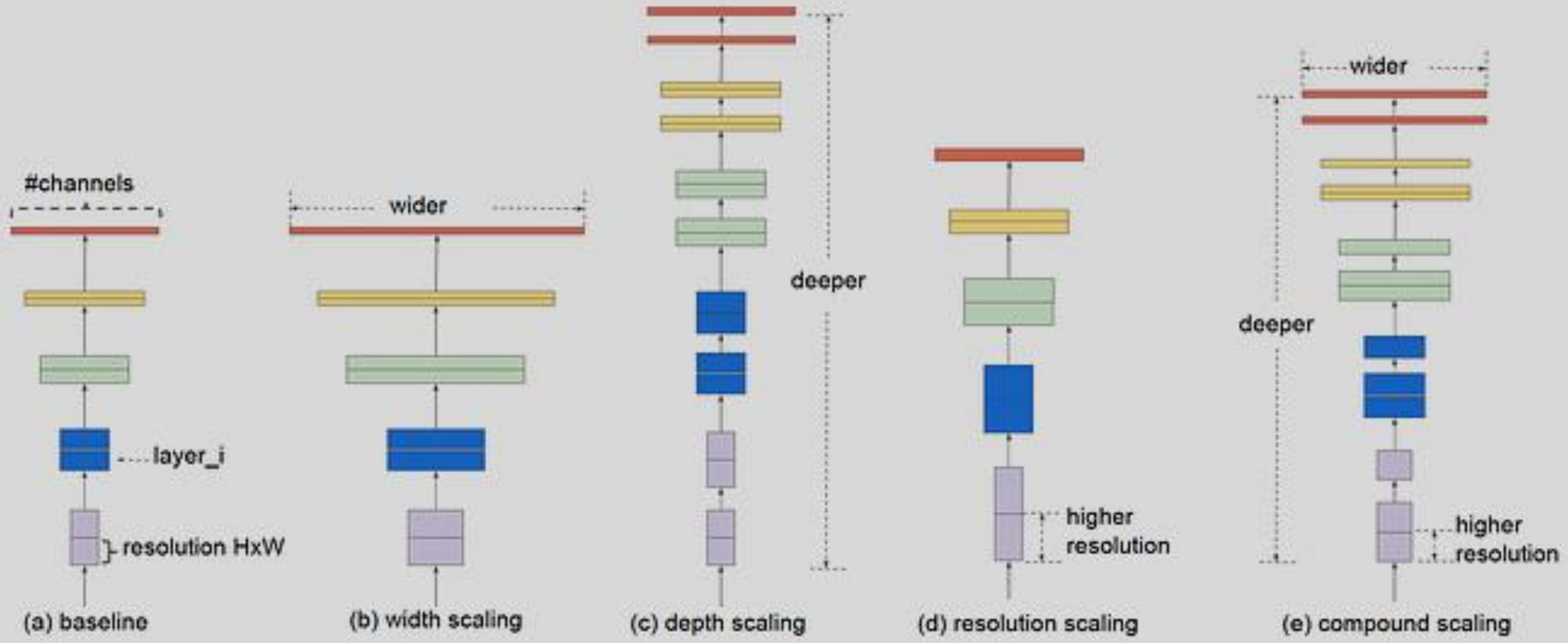


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML'19.

EfficientNet

$$
\begin{aligned}
\max_{d,w,r} \quad & Accuracy\big(\mathcal{N}(d,w,r)\big) \\
s.t. \quad & \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot \hat{L}_i}\big(X_{\langle r\cdot \hat{H}_i, r\cdot \hat{W}_i, w\cdot \hat{C}_i\rangle}\big) \\
& \text{Memory}(\mathcal{N}) \leq \text{target\_memory} \\
& \text{FLOPS}(\mathcal{N}) \leq \text{target\_flops}
\end{aligned}
$$

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML’19.

EfficientNet

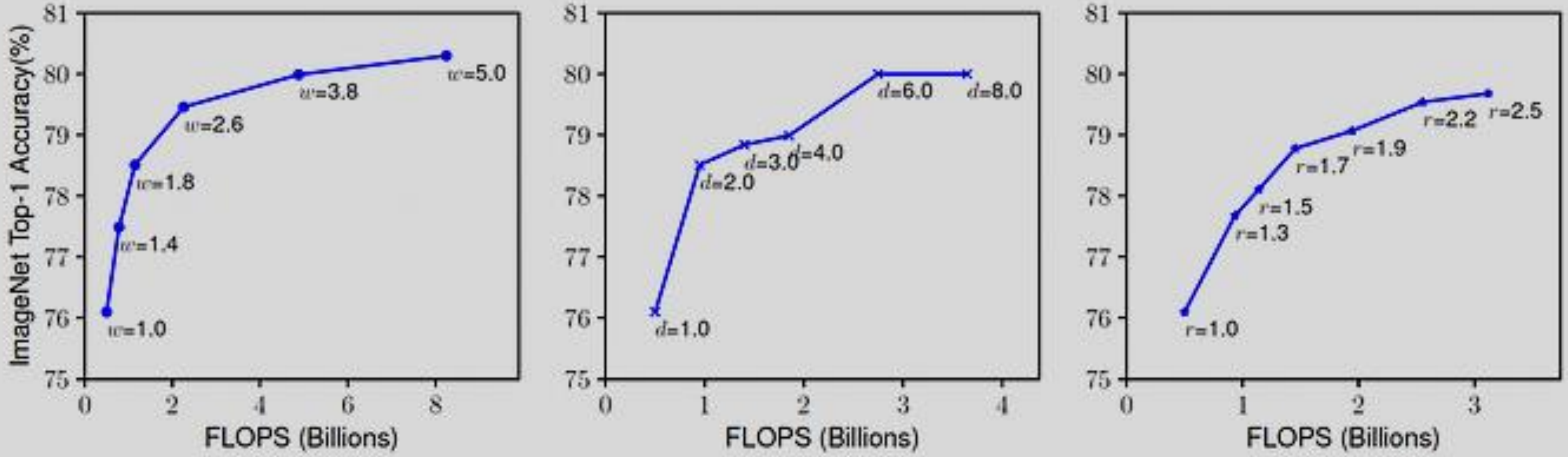


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single dimension scaling. Baseline network is described in Table 1.

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML’19.

EfficientNet

$$
\begin{aligned}
\text{depth: } & d = \alpha^{\phi} \\
\text{width: } & w = \beta^{\phi} \\
\text{resolution: } & r = \gamma^{\phi} \\
\text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\
& \alpha \geq 1, \beta \geq 1, \gamma \geq 1
\end{aligned} \tag{3}
$$

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML'19.

EfficientNet

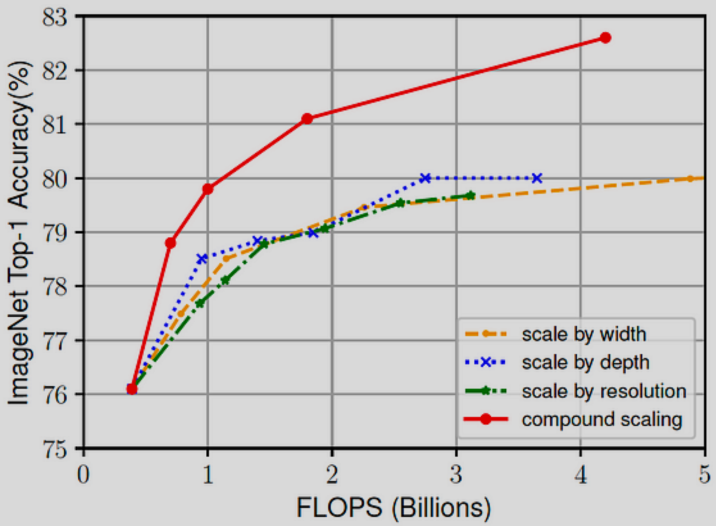


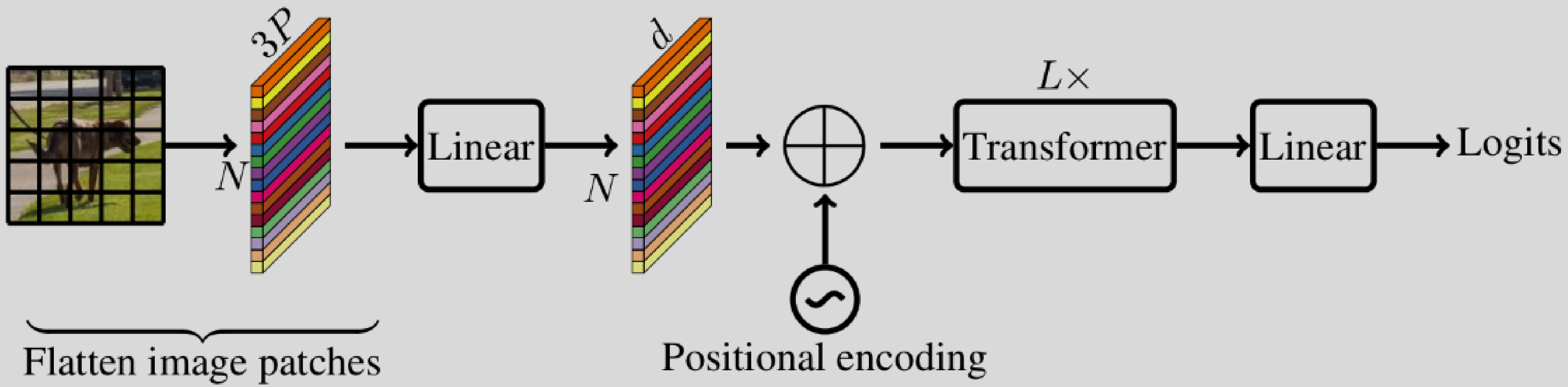
Figure 8. **Scaling Up EfficientNet-B0 with Different Methods.**

Table 7. **Scaled Models Used in Figure 7.**

Model	FLOPS	Top-1 Acc.
Baseline model (EfficientNet-B0)	0.4B	77.3%
Scale model by depth (d=4)	1.8B	79.0%
Scale model by width (w=2)	1.8B	78.9%
Scale model by resolution (r=2)	1.9B	79.1%
Compound Scale (d=1.4, w=1.2, r=1.3)	**1.8B**	**81.1%**

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML'19.

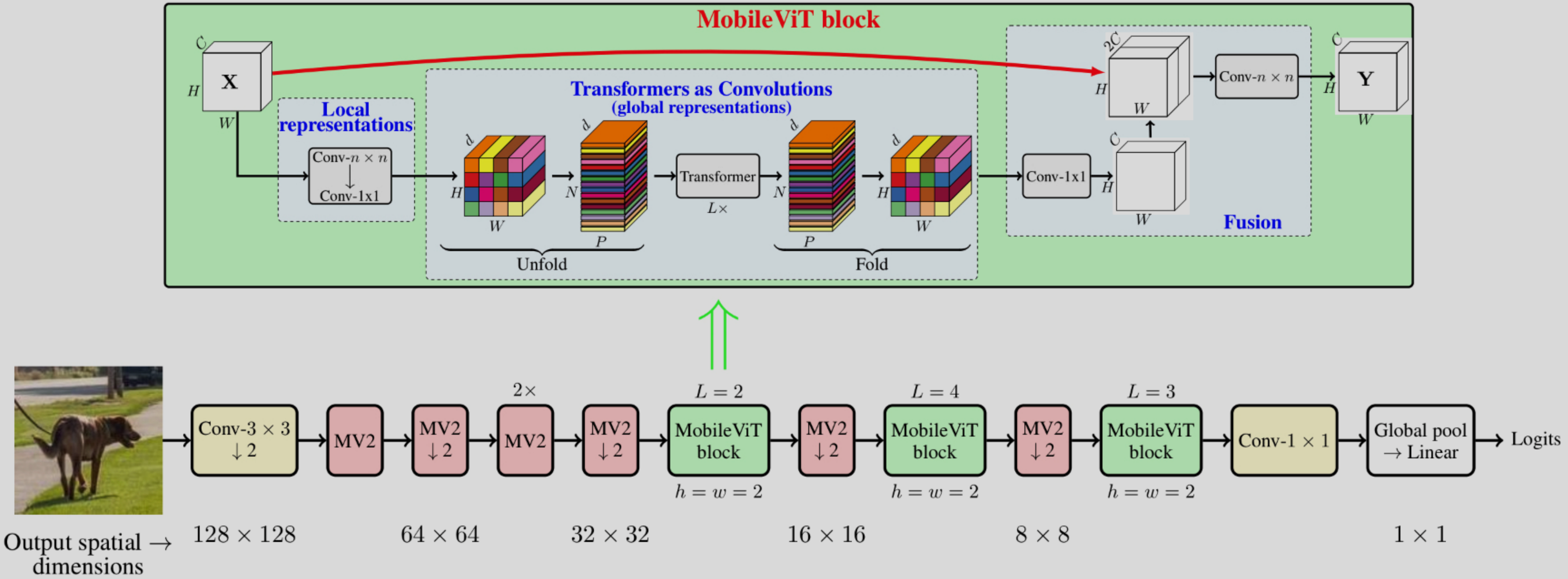
MobileViT



(a) **Standard visual transformer (ViT)**

MOBILEVIT: LIGHT-WEIGHT, GENERAL-PURPOSE, AND MOBILE-FRIENDLY VISION TRANSFORMER, ICLR'22.

MobileViT



(b) **MobileViT**. Here, Conv-$n \times n$ in the MobileViT block represents a standard $n \times n$ convolution and MV2 refers to MobileNetv2 block. Blocks that perform down-sampling are marked with $\downarrow 2$.

MOBILEVIT: LIGHT-WEIGHT, GENERAL-PURPOSE, AND MOBILE-FRIENDLY VISION TRANSFORMER, ICLR'22.

MobileViT

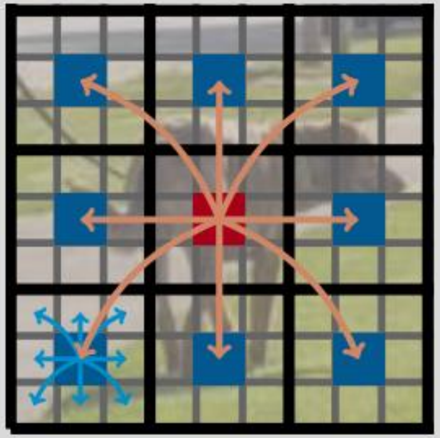
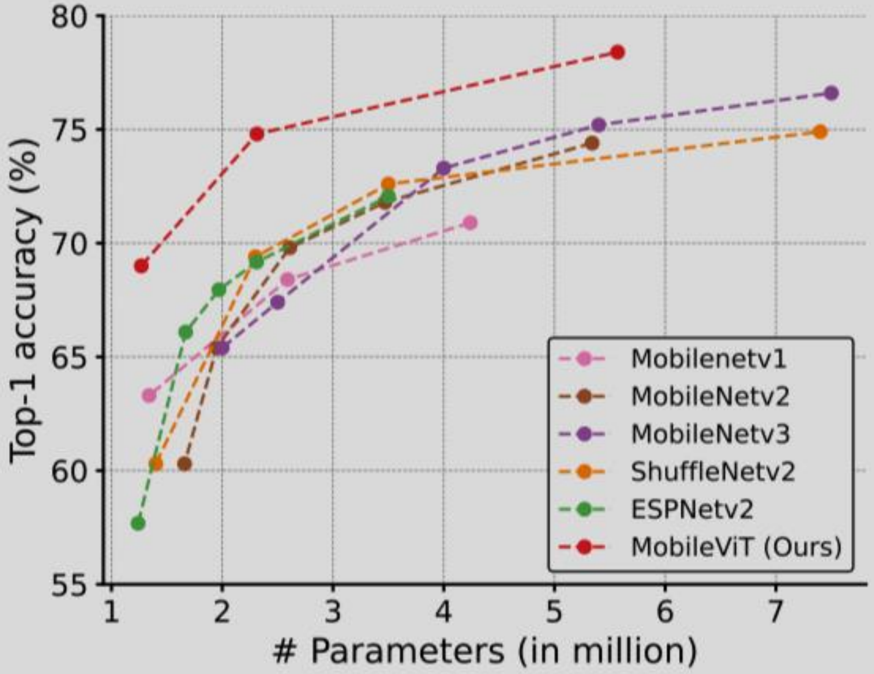


Figure 4: **Every pixel sees every other pixel in the MobileViT block.** In this example, the **red** pixel attends to **blue** pixels (pixels at the corresponding location in other patches) using transformers. Because **blue** pixels have already encoded information about the neighboring pixels using convolutions, this allows the **red** pixel to encode information from all pixels in an image. Here, each cell in **black** and **gray** grids represents a patch and a pixel, respectively.

MOBILEVIT: LIGHT-WEIGHT, GENERAL-PURPOSE, AND MOBILE-FRIENDLY VISION TRANSFORMER, ICLR'22.

MobileViT



(a) Comparison with light-weight CNNs

Model	# Params. ⇓	Top-1 ⇑
MobileNetv1	2.6 M	68.4
MobileNetv2	2.6 M	69.8
MobileNetv3	2.5 M	67.4
ShuffleNetv2	2.3 M	69.4
ESPNetv2	2.3 M	69.2
MobileViT-XS (Ours)	2.3 M	**74.8**

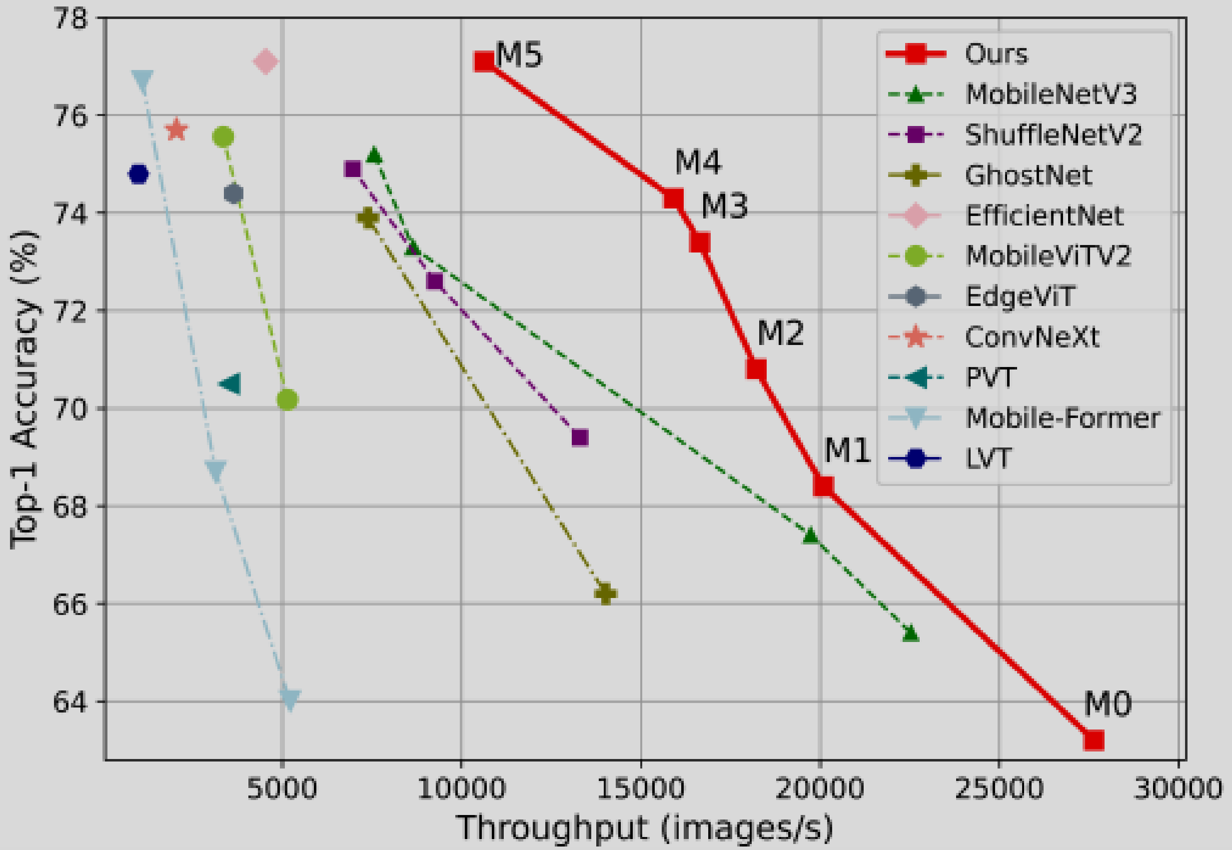
(b) Comparison with light-weight CNNs (similar parameters)

Model	# Params. ⇓	Top-1 ⇑
DenseNet-169	14 M	76.2
EfficientNet-B0	5.3 M	76.3
ResNet-101	44.5 M	77.4
ResNet-101-SE	49.3 M	77.6
MobileViT-S (Ours)	5.6 M	**78.4**

(c) Comparison with heavy-weight CNNs

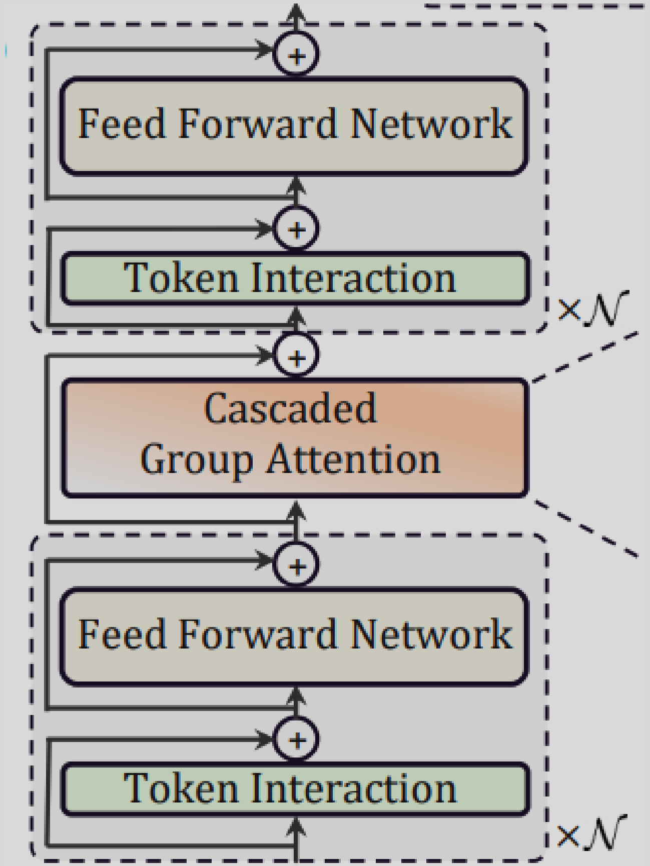
MOBILEVIT: LIGHT-WEIGHT, GENERAL-PURPOSE, AND MOBILE-FRIENDLY VISION TRANSFORMER, ICLR'22.

EfficientViT



EfficientViT: Memory Efficient Vision Transformer with Cascaded Group Attention, CVPR’23.

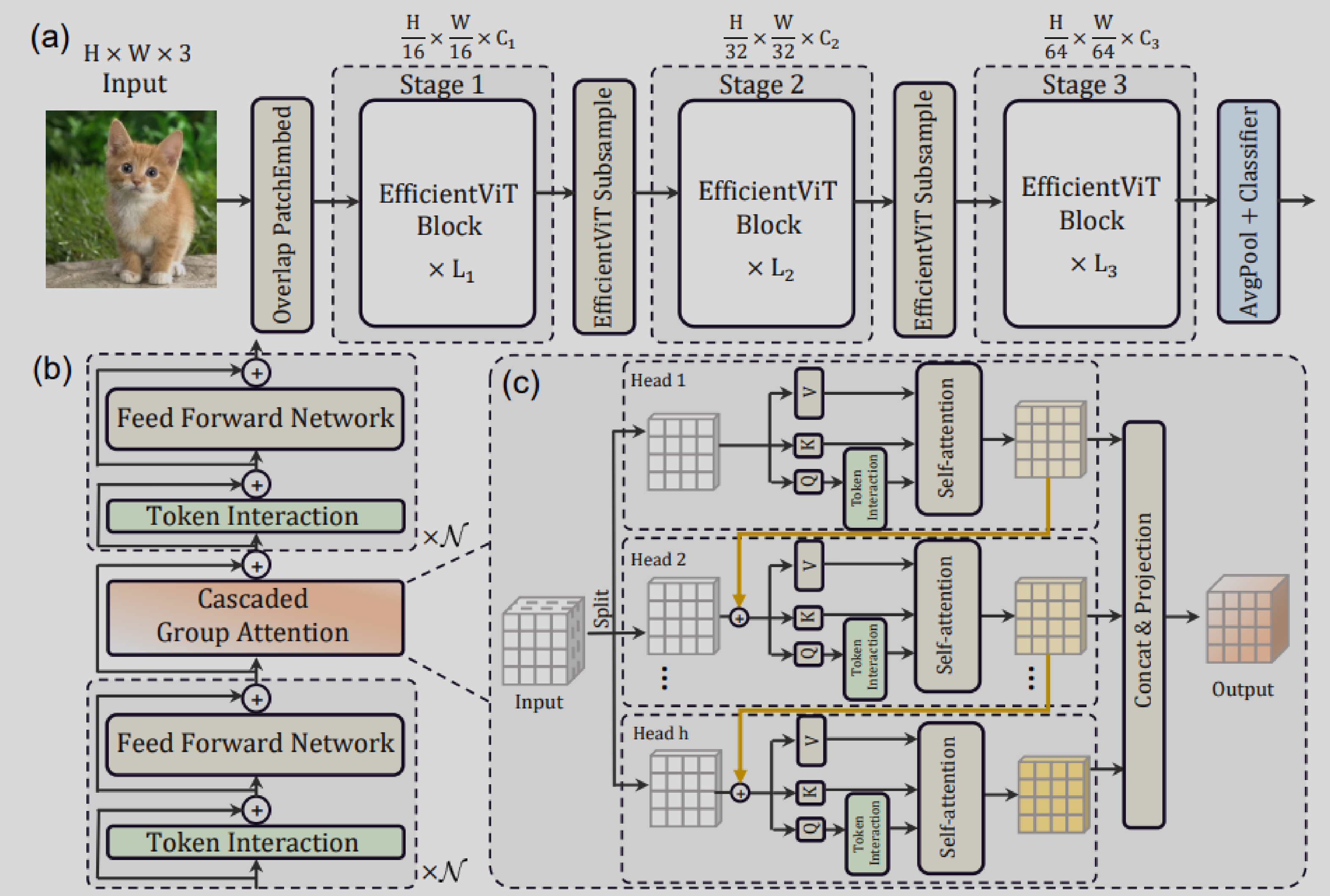
EfficientViT



$$X_{i+1} = \prod^{N} \Phi_i^F(\Phi_i^A(\prod^{N} \Phi_i^F(X_i)))$$

EfficientViT: Memory Efficient Vision Transformer with Cascaded Group Attention, CVPR'23.

EfficientViT



EfficientViT: Memory Efficient Vision Transformer with Cascaded Group Attention, CVPR'23.

EfficientViT

Model	Top-1 (%)	Top-5 (%)	Throughput (images/s) GPU	CPU	ONNX	Flops (M)	Params (M)	Input	Epochs
EfficientViT-M0	**63.2**	85.4	27644	228.4	340.1	79	2.3	224	300
MobileNetV3-Small [26]	67.4	-	19738	156.5	231.7	57	2.5	224	600
EfficientViT-M1	**68.4**	88.7	20093	126.9	215.9	167	3.0	224	300
Mobile-Former-52M [9]	68.7	-	3141	32.8	21.5	52	3.5	224	450
MobileViT-XXS [50]	69.0	-	4456	29.4	41.7	410	1.3	256	300
ShuffleNetV2 1.0× [48]	69.4	88.9	13301	106.7	177.0	146	2.3	224	300
MobileViTV2-0.5 [51]	70.2	-	5142	34.4	44.9	466	1.4	256	300
EfficientViT-M2	**70.8**	90.2	18218	121.2	158.7	201	4.2	224	300
MobileOne-S0 [70]	71.4	-	11320	67.4	128.6	274	2.1	224	300
MobileNetV2 1.0× [63]	72.0	91.0	6534	32.5	80.4	300	3.4	224	300
EfficientViT-M3	**73.4**	91.4	16644	96.4	120.8	263	6.9	224	300
GhostNet 1.0× [23]	73.9	91.4	7382	57.3	77.0	141	5.2	224	300
NASNet-A-Mobile [89]	74.1	-	2623	19.8	25.5	564	5.3	224	300
EfficientViT-M4	**74.3**	91.8	15914	88.5	108.6	299	8.8	224	300
EdgeViT-XXS [56]	74.4	-	3638	28.2	29.6	556	4.1	224	300
MobileViT-XS [50]	74.7	-	3344	11.1	20.5	986	2.3	256	300
ShuffleNetV2 2.0× [48]	74.9	92.4	6962	37.9	52.3	591	7.4	224	300
MobileNetV3-Large [26]	75.2	-	7560	39.1	70.5	217	5.4	224	600
MobileViTV2-0.75 [51]	75.6	-	3350	16.0	22.7	1030	2.9	256	300
MobileOne-S1 [70]	75.9	-	6663	30.7	51.1	825	4.8	224	300
GLiT-Tiny [5]	76.4	-	3516	17.5	15.7	1333	7.3	224	300
EfficientNet-B0 [67]	77.1	93.3	4532	30.2	29.5	390	5.3	224	350
EfficientViT-M5	**77.1**	93.4	10621	56.8	62.5	522	12.4	224	300
EfficientViT-M4↑384	**79.8**	95.0	3986	15.8	22.6	1486	12.4	384	330
EfficientViT-M5↑512	**80.8**	95.5	2313	8.3	10.5	2670	12.4	512	360

EfficientViT: Memory Efficient Vision Transformer with Cascaded Group Attention, CVPR'23.

Thank you!

UNIST

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

2007