EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Neural Acceleration Lab
Hwigeon Oh

Introduction

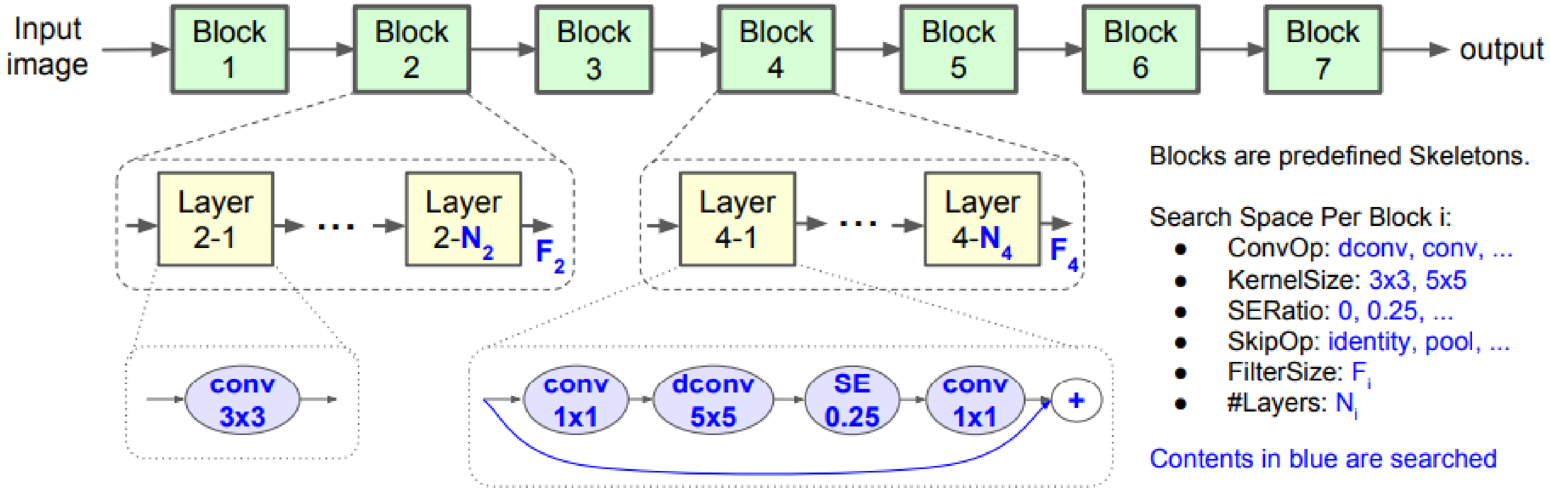
- Scaling up ConvNets is widely used to achieve better accuracy
- Ways to scale up: Depth (#layers) / Width (#channels) / Image resolution
- In previous work, it is common to scale only one of three
- This paper provides a principled method to scale up with ***compound scaling***

Problem Formulation

- A ConvNet Layer i can be defined as: $Y_i = \mathcal{F}_i(X_i)$
- A ConvNet $\mathcal{N}$ can be represented as: $\mathcal{N} = \mathcal{F}_k \odot \cdots \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1 \ldots k} \mathcal{F}_j(X_1)$
- By factorization with blocks: $\mathcal{N} = \bigodot_{i=1 \ldots s} {\mathcal{F}_i}^{L_i}(X_{<H_i, W_i, C_i>})$

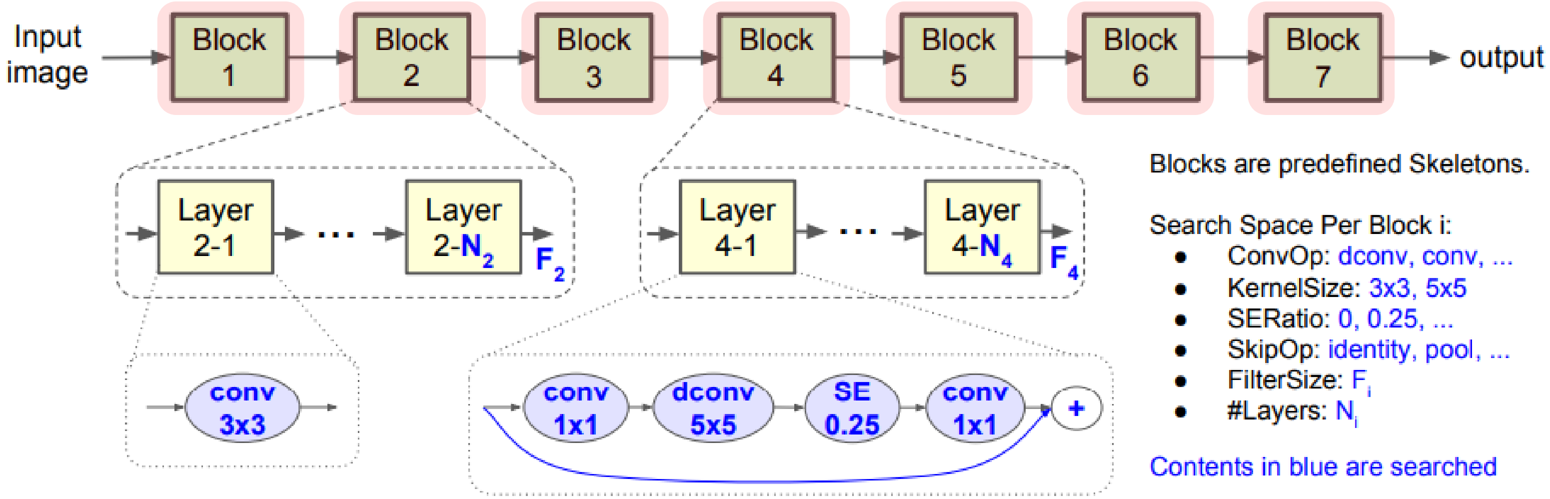
Problem Formulation

- A ConvNet Layer i can be defined as: $Y_i = \mathcal{F}_i(X_i)$
- A ConvNet $\mathcal{N}$ can be represented as: $\mathcal{N} = \mathcal{F}_k \odot \cdots \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1\ldots k} \mathcal{F}_j(X_1)$
- By factorization with blocks: $\mathcal{N} = \bigodot_{i=1\ldots s} \mathcal{F}_i^{L_i}(X_{\langle H_i, W_i, C_i \rangle})$



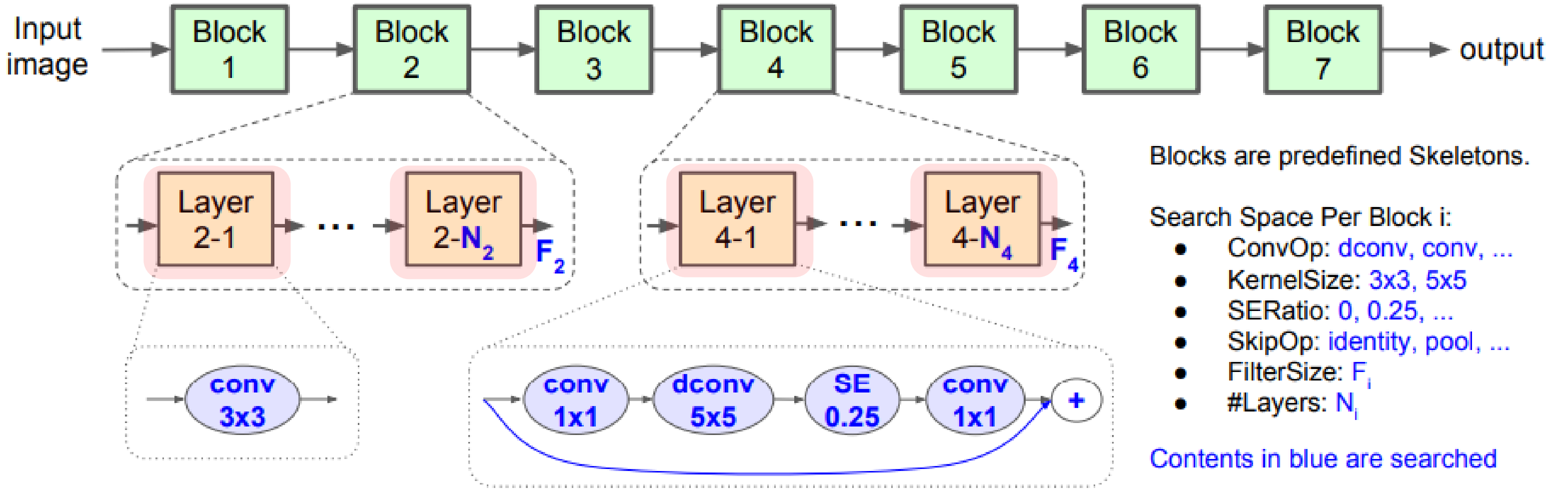
Problem Formulation

- A ConvNet Layer i can be defined as: $Y_i = \mathcal{F}_i(X_i)$
- A ConvNet $\mathcal{N}$ can be represented as: $\mathcal{N} = \mathcal{F}_k \odot \cdots \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1\ldots k} \mathcal{F}_j(X_1)$
- By factorization with blocks: $\mathcal{N} = \bigodot_{i=1\ldots s} \mathcal{F}_i^{L_i}(X_{\langle H_i, W_i, C_i \rangle})$



Problem Formulation

- A ConvNet Layer i can be defined as: $Y_i = \mathcal{F}_i(X_i)$
- A ConvNet $\mathcal{N}$ can be represented as: $\mathcal{N} = \mathcal{F}_k \odot \cdots \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1\ldots k} \mathcal{F}_j(X_1)$
- By factorization with blocks: $\mathcal{N} = \bigodot_{i=1\ldots s} \mathcal{F}_i^{L_i}(X_{<H_i,W_i,C_i>})$



Blocks are predefined Skeletons.

Search Space Per Block i:
- ConvOp: dconv, conv, ...
- KernelSize: 3x3, 5x5
- SERatio: 0, 0.25, ...
- SkipOp: identity, pool, ...
- FilterSize: F_i
- #Layers: N_i

Contents in blue are searched

Problem Formulation

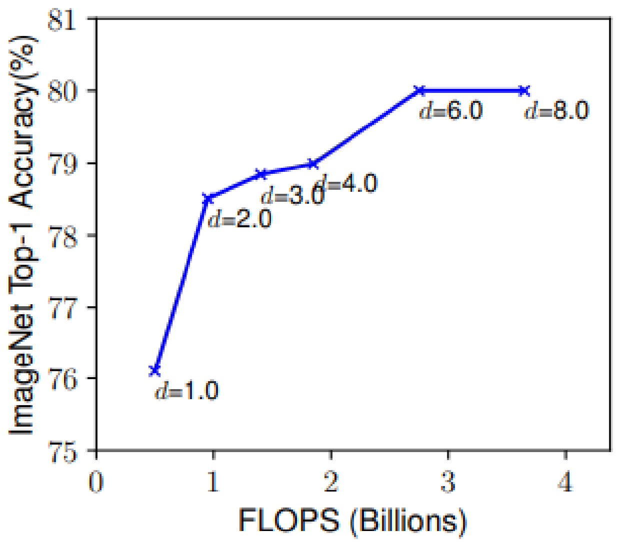
- **Model scaling tries to expand** the length(L_i), width(C_i), resolution(H_i, W_i)
- **Without changing layer architecture**($\mathcal{F}_i$)
- To reduce the design space, all layers are **scaled uniformly** with constant ratio

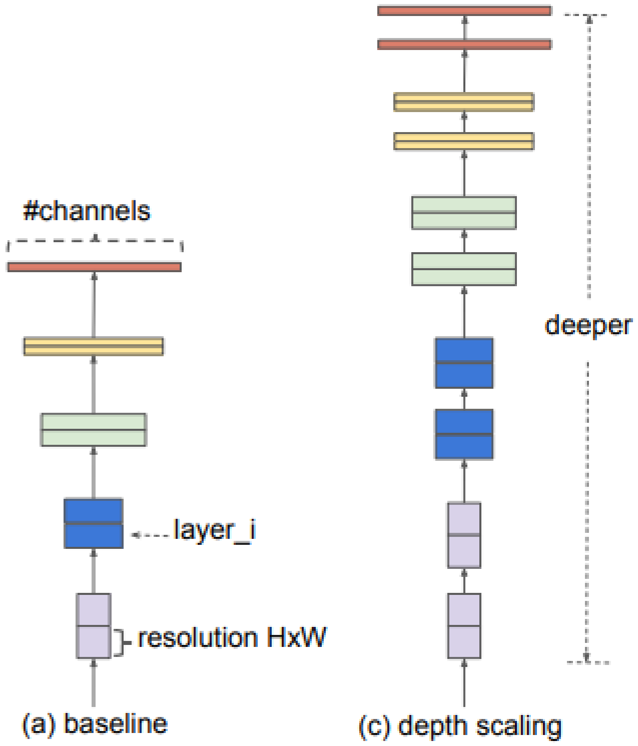
$$
\begin{aligned}
\max_{d,w,r} \quad & Accuracy\big(\mathcal{N}(d,w,r)\big) \\
s.t. \quad & \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\big(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\big) \\
& \text{Memory}(\mathcal{N}) \leq \text{target\_memory}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

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

Scaling Dimensions: depth

- Deeper network can capture **richer** and **more complex** features
- Ex. ResNet-50 < ResNet-152

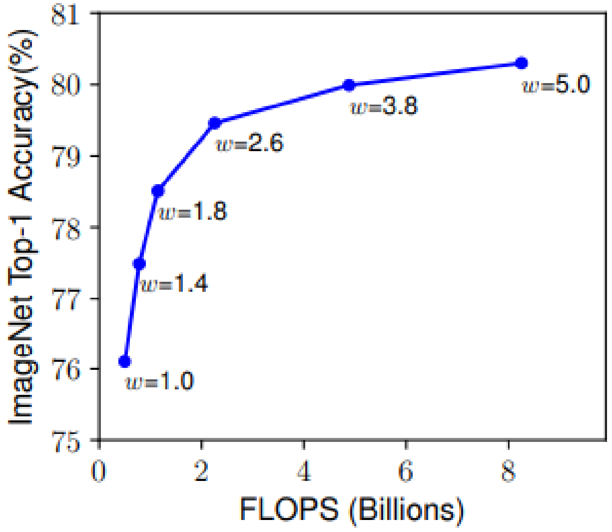


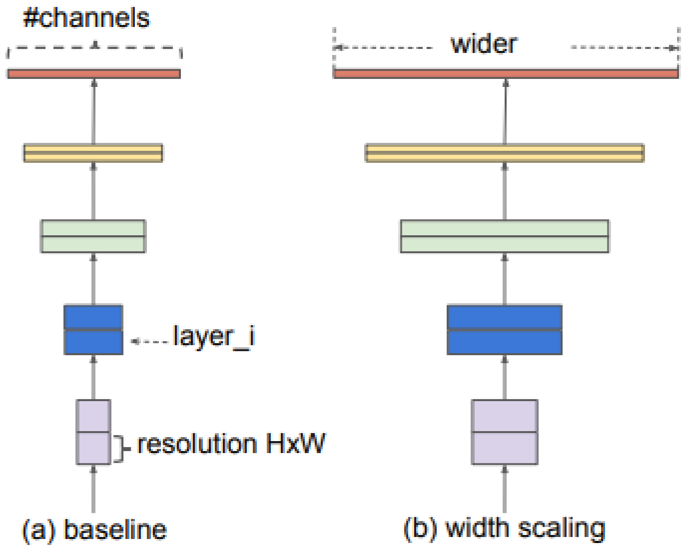


$$
\begin{aligned}
\max_{d,w,r} \quad & Accuracy\big(\mathcal{N}(d,w,r)\big) \\
s.t. \quad & \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\big(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\big) \\
& \text{Memory}(\mathcal{N}) \leq \text{target\_memory}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

Scaling Dimensions: width

- Wider network can capture **more fine-grained** features
- Ex. wide residual network





$$\max_{d,w,r} \quad Accuracy\big(\mathcal{N}(d,w,r)\big)$$

$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\big(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\big)$$

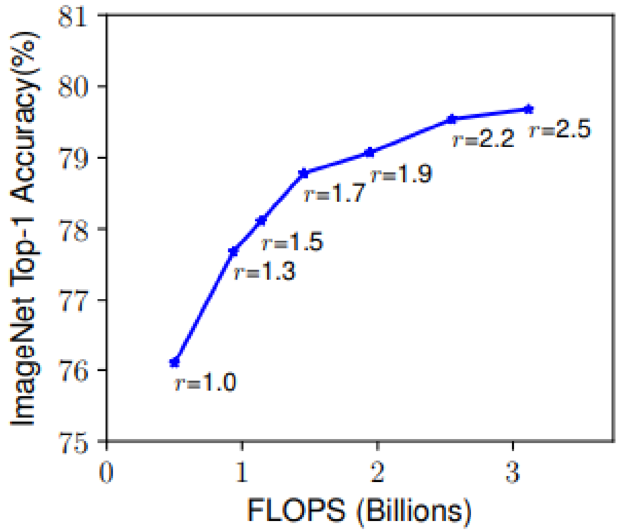
$$\text{Memory}(\mathcal{N}) \leq \text{target\_memory}$$

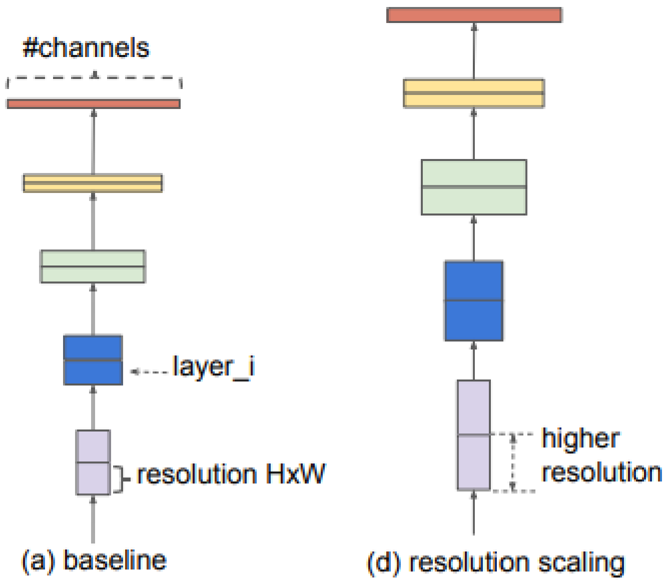
$$\text{FLOPS}(\mathcal{N}) \leq \text{target\_flops}$$

Scaling Dimensions: resolution

- With higher resolution input, can capture **more fine-grained** patterns

- Ex. GPipe



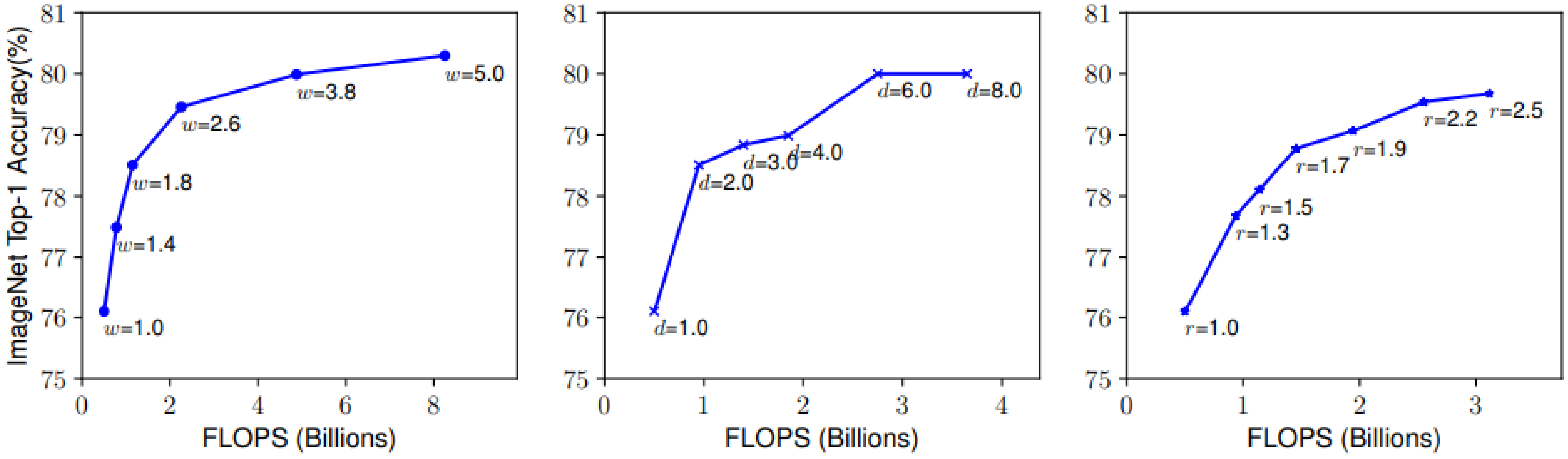


$$
\begin{aligned}
\max_{d,w,r} \quad & Accuracy\big(\mathcal{N}(d,w,r)\big) \\
s.t. \quad & \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\big(X_{\langle r\cdot\hat{H}_i,\, r\cdot\hat{W}_i,\, w\cdot\hat{C}_i\rangle}\big) \\
& \text{Memory}(\mathcal{N}) \leq \text{target\_memory}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

Scaling Dimensions

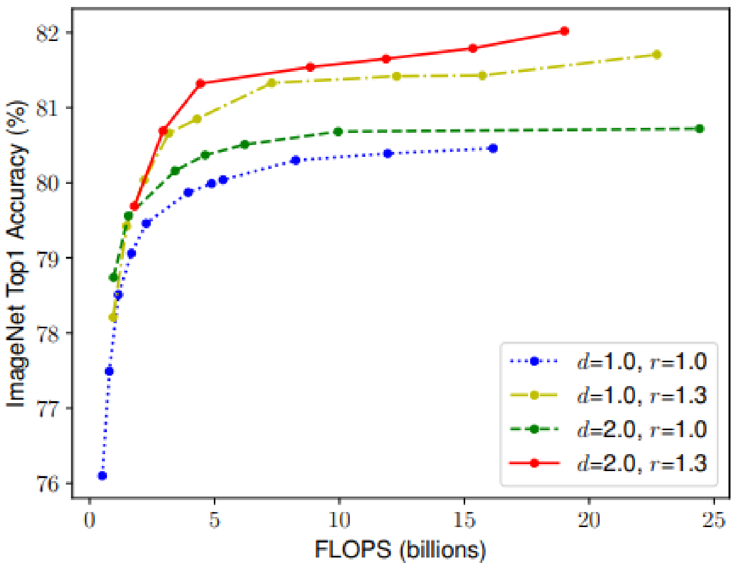
- Scaling up any dimension improves accuracy

- **The accuracy gain diminishes for bigger models**



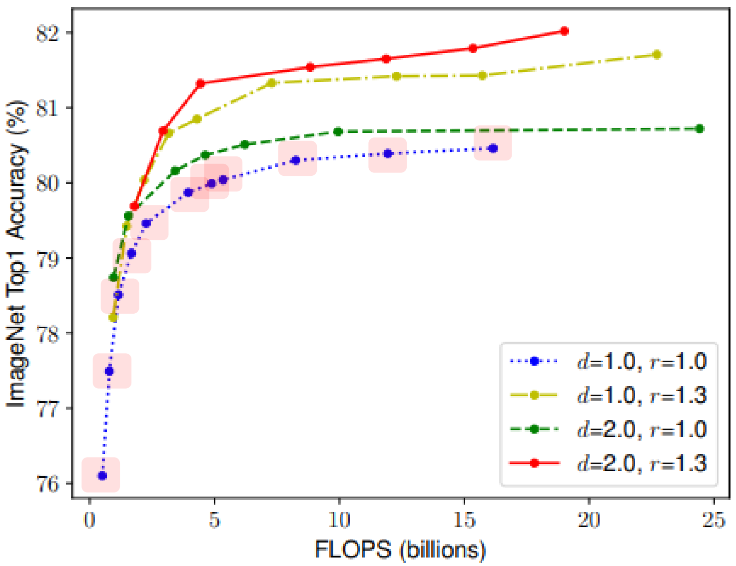
Compound Scaling

- Scaling dimensions are not independent (empirical observation)
- Increase depth to make larger receptive field for higher resolution images
- Increase width to capture more fine-grained patterns with more pixels



Compound Scaling

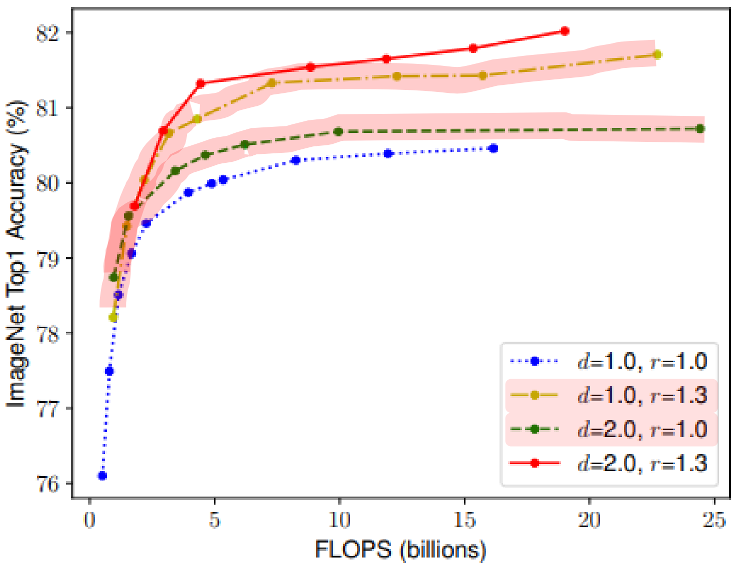
- Scaling dimensions are not independent (empirical observation)
- Increase depth to make larger receptive field for higher resolution images
- Increase width to capture more fine-grained patterns with more pixels



- Each dot in a line denotes a model with different width
- Accuracy saturates quickly if only scale width

Compound Scaling

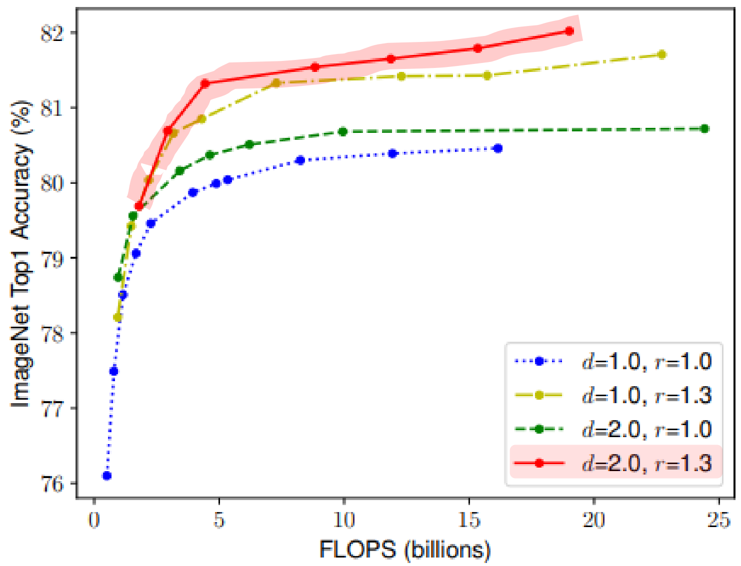
- Scaling dimensions are not independent (empirical observation)
- Increase depth to make larger receptive field for higher resolution images
- Increase width to capture more fine-grained patterns with more pixels



- Each dot in a line denotes a model with different width
- Accuracy saturates quickly if only scale width
- Scaling width with depth or resolution makes better

Compound Scaling

- Scaling dimensions are not independent (empirical observation)
- Increase depth to make larger receptive field for higher resolution images
- Increase width to capture more fine-grained patterns with more pixels



- Each dot in a line denotes a model with different width
- Accuracy saturates quickly if only scale width
- Scaling width with depth or resolution makes better
- **It is critical to balance all three dimensions**

Compound Scaling Method

- Depth: $d = \alpha^{\phi}$, Width: $w = \beta^{\phi}$, Resolution: $r = \gamma^{\phi}$
- Use a compound coefficient ϕ to uniformly scales
- The FLOPS of a convolution op is proportional to d, w^2, r^2
- With constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$, the total FLOPS will increase by 2^{ϕ}
- ϕ controls how many more resources are available
- α, β, γ specify how to assign extra resources

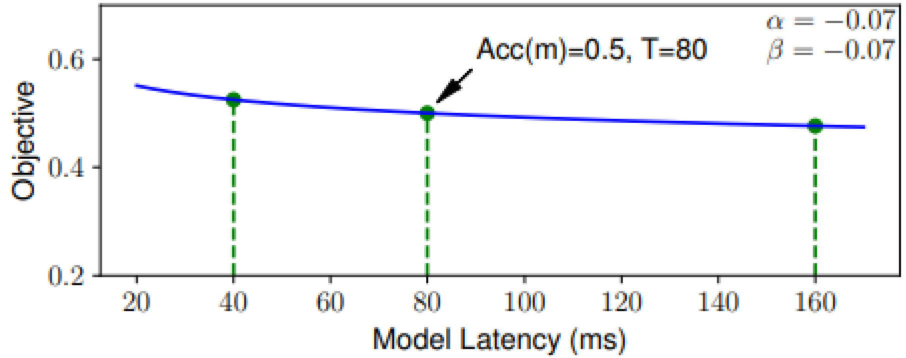
MnasNet (Tan et al., 2019)

- Multi-objective neural architecture search that optimized both accuracy and latency
- Use weighted product method to approximate Pareto optimal solutions
- Use empirical observation: x2 the latency brings about 5% relative accuracy gain
  - With this condition, β in the below equation is ≈ -0.07

$$\begin{aligned} \underset{m}{\text{maximize}} \quad & ACC(m) \\ \text{subject to} \quad & LAT(m) \le T \end{aligned}$$

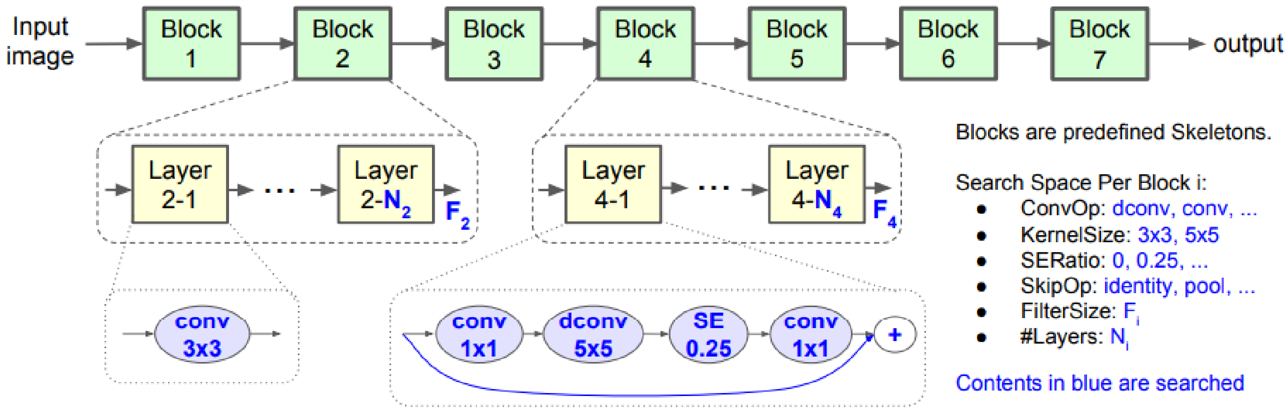
$$\underset{m}{\text{maximize}} \quad ACC(m) \times \left[\frac{LAT(m)}{T}\right]^{w}$$

$$w = \begin{cases} \alpha, & \text{if } LAT(m) \le T \\ \beta, & \text{otherwise} \end{cases}$$



MnasNet (Tan et al., 2019)

- Use Factorized Hierarchical Search Space
- Network layers are grouped into a number of blocks
- Each block contains a variable number of repeated identical layers
- For each block, search for the operations for a single layer and the number of layers



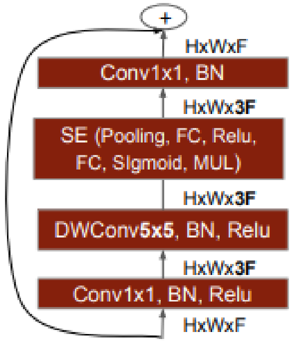
- Convolutional ops $ConvOp$: regular conv (conv), depthwise conv (dconv), and mobile inverted bottleneck conv [29].
- Convolutional kernel size $KernelSize$: 3x3, 5x5.
- Squeeze-and-excitation [13] ratio $SERatio$: 0, 0.25.
- Skip ops $SkipOp$: pooling, identity residual, or no skip.
- Output filter size F_i.
- Number of layers per block N_i.

EfficientNet Architecture

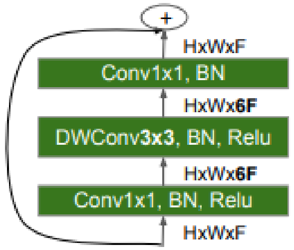
- Since model scaling does not change layer, having a good baseline is critical
- Use $ACC(m) \times [\frac{FLOPS(m)}{T}]^w$ as the optimization goal where $w = -0.07$

Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Notations are adopted from equation 2.

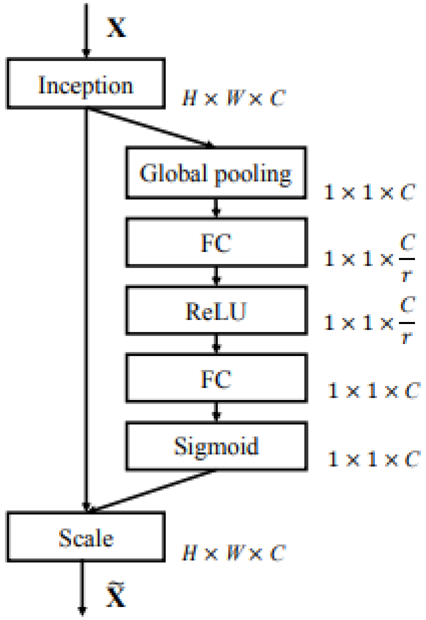
Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1



(b) MBConv3 (k5x5)



(c) MBConv6 (k3x3)



SE-Inception Module

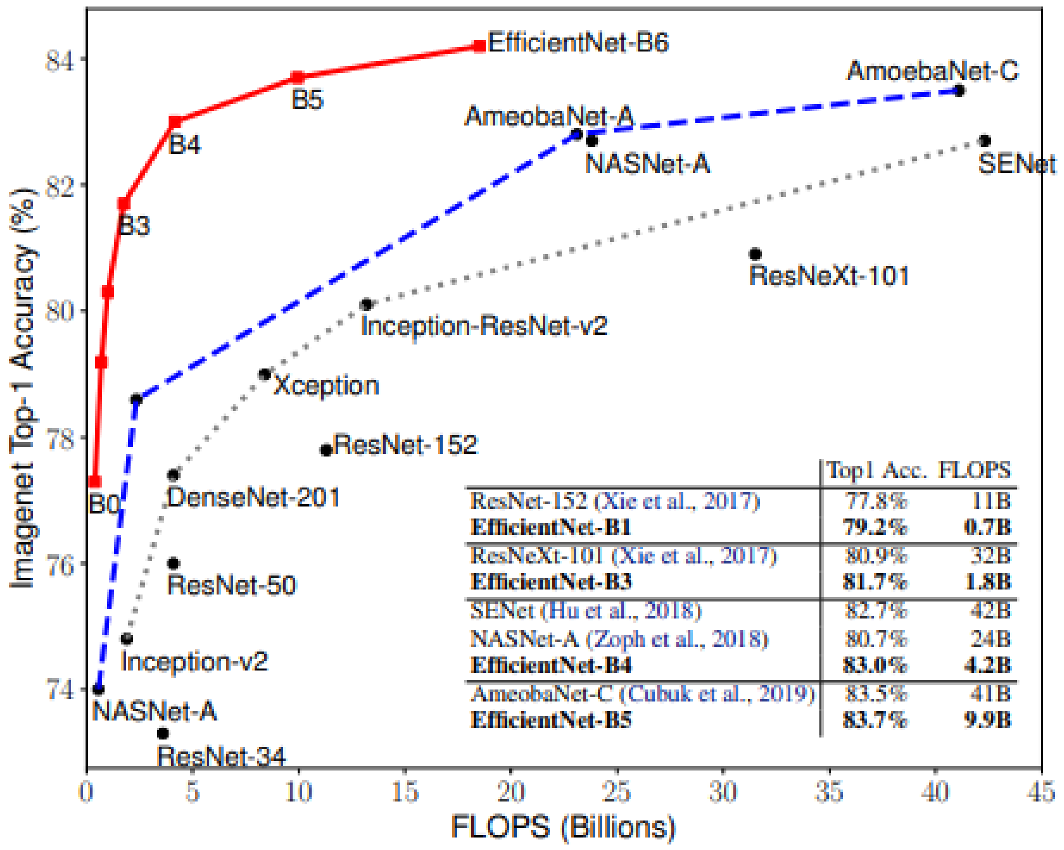
EfficientNet Architecture

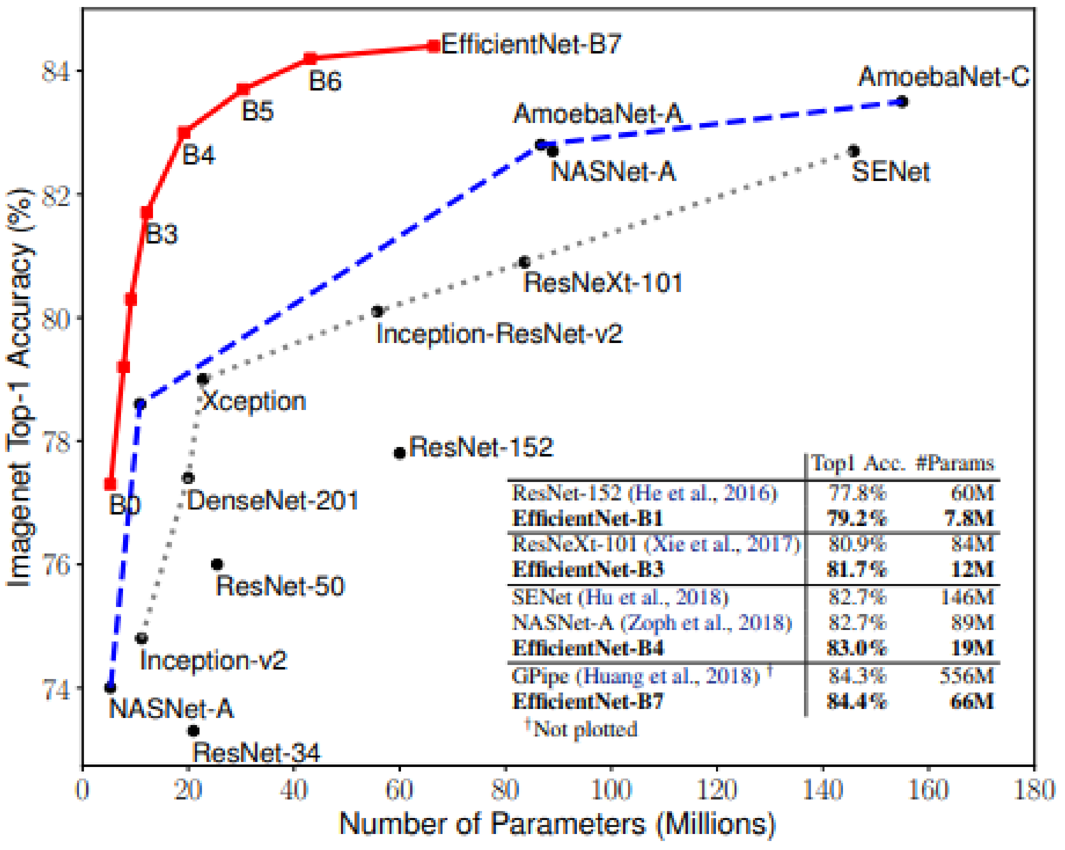
- Starting from the baseline EfficientNet-B0, apply compound scaling method:
  - STEP1: Fix $\phi = 1$, assuming x2 resources. Do a grid search to find α, β, γ
  - STEP2: Fix α, β, γ as constants and scale up baseline network with different ϕ
- Resultant $\alpha = 1.2$, $\beta = 1.1$, $\gamma = 1.15$

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

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

Results on ImageNet

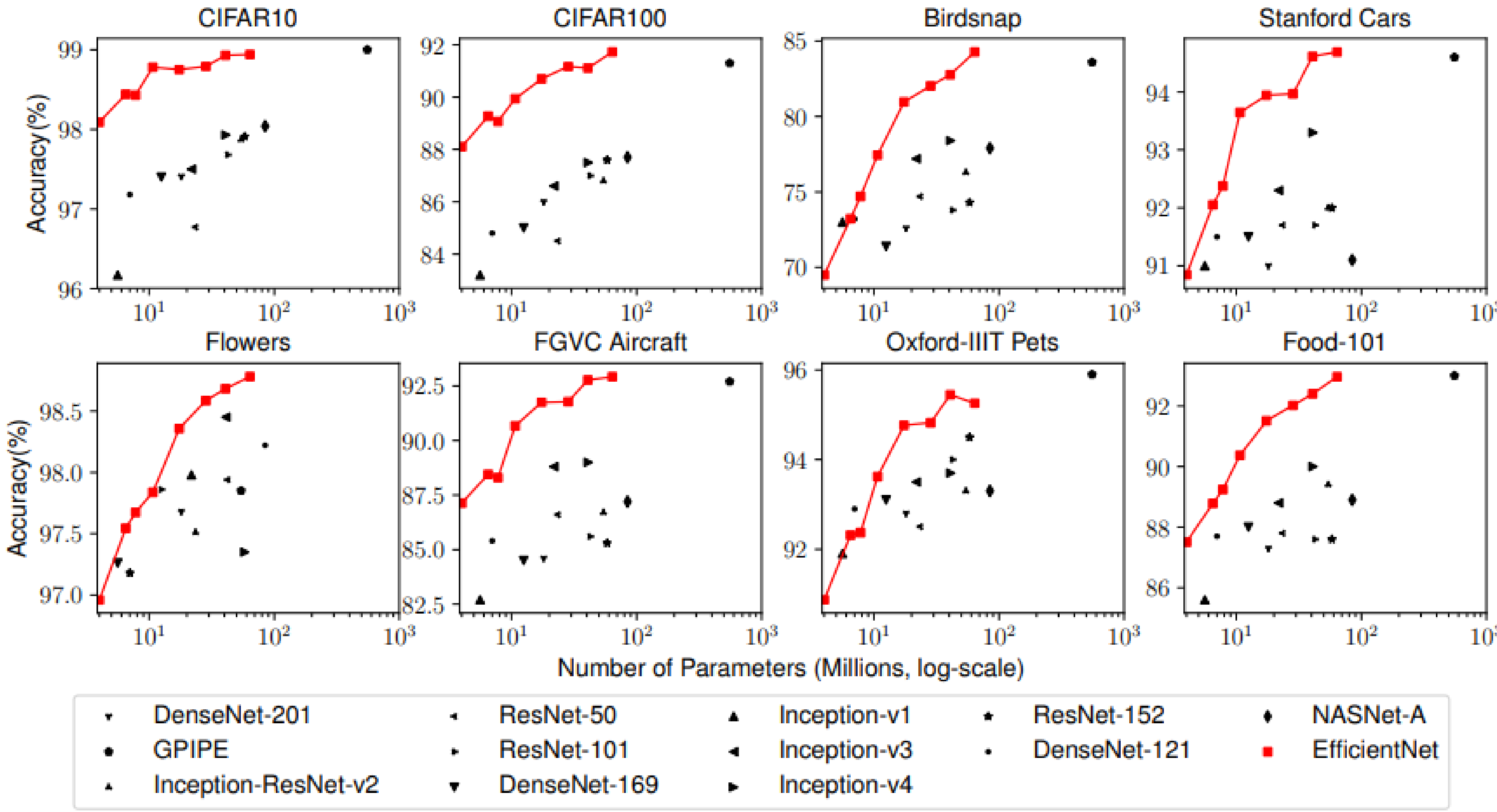




Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**77.3%**	**93.5%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.2%**	**94.5%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.3%**	**95.0%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.7%**	**95.6%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**83.0%**	**96.3%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.7%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.2%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.4%**	**97.1%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

Results on Transfer Learning



Comparisons with different scaling methods

- The model with compound scaling tends to focus more relevant regions