EfficientNet

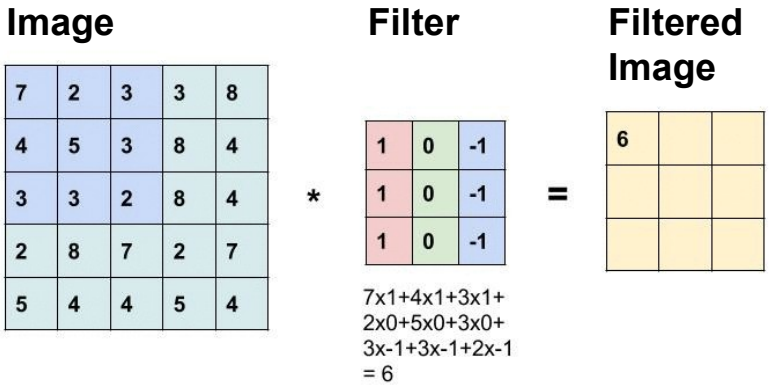
Rethinking model scaling for convolutional neural networks

https://arxiv.org/pdf/1905.11946.pdf

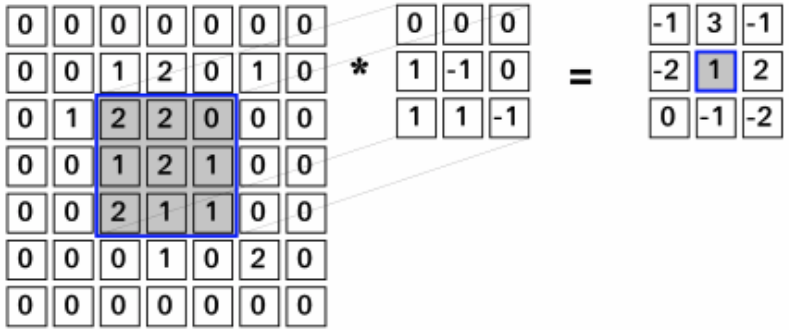
Outline

1. Review of CNNs
2. Model Scaling
3. Building EfficientNet
4. Results

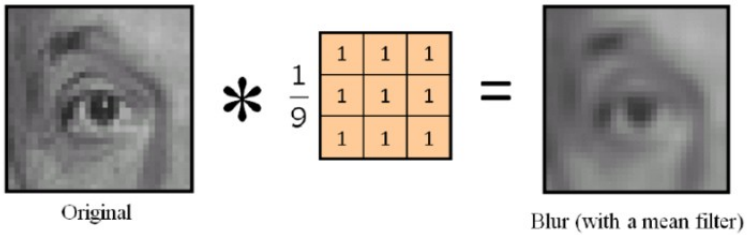
Sliding windows and filters



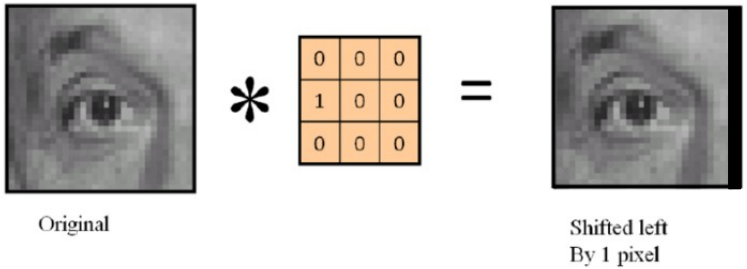
- The operation illustrated here is a convolution using a sliding window
- A scalar product is computed at each position
- Filters can accomplish sharpening, blurring, translation, etc.



Sliding windows and filters example



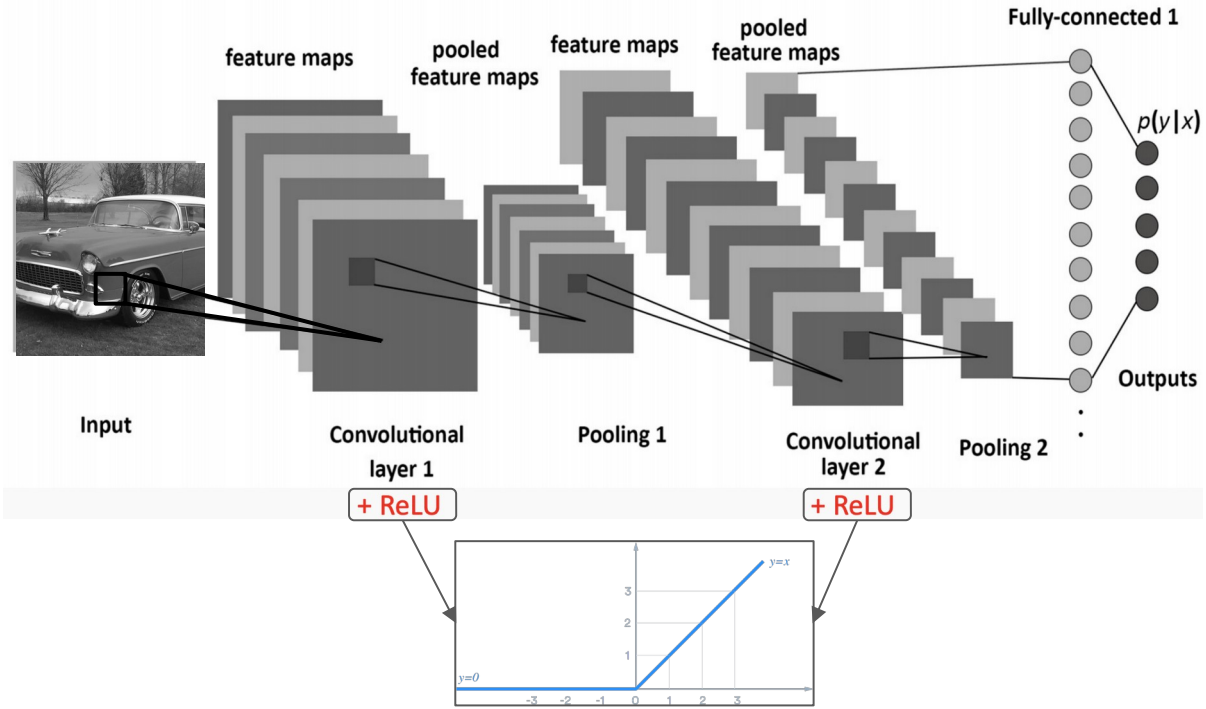
- Mean filter with a uniform weight of 1/9
- Output is a blurred image



- This is an example of a left shift by a single pixel
- A value of 0 in the image matrix is a black pixel

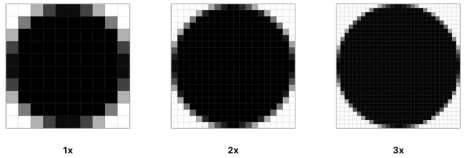
Image Classification Example

Apply filters → Activation function → Pooling → Repeat as needed

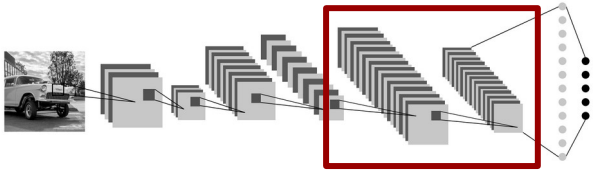


Overview of scaling

Resolution scaling



Depth scaling



Width scaling



Convolutional layer

Challenges of model scaling

- Neural networks are often scaled to achieve better accuracy, but there is no systematic method of scaling these networks.
- Arbitrary scaling methods tend to result in very large, slow models!

Scaling Type	Example	Results (ImageNet)
Network Depth (More Layers)	ResNet-50 to ResNet-152 (He et al., 2016)	ResNet-50: 76% top1 ResNet-152: 77.8% top1
Network Width (More filters per layer)	Wide Residual Networks (Zagoruyko & Komodakis, 2016)	78.1% top1
Input Resolution	[331x331] NASNet (Zoph et al., 2018) [480x480] GPipe (Huang et al., 2018)	NASNet: 82.7% top1 GPipe: 84.3% top1

Define a model

- The goal of a model is to minimize error

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

Maximize accuracy of the model by varying d,w,r

$$\mathcal{N} = \bigodot_{i=1\ldots s} \mathcal{F}_i^{L_i}\left(X_{\langle H_i, W_i, C_i\rangle}\right)$$

The model is defined by stages (*i*) composed of layers (*F*) each repeated L_i times per stage.

$\mathcal{F}_i^{L_i}$ denotes layer F_i is repeated L_i times in stage i

Given:
d = depth coef
w = width coef
r = resolution coef

Input:
H_i = Layer Height
W_i = Layer Width
C_i = Layer Channels

EfficientNet model

- EfficientNet tries to maximize the model accuracy for any given resource constraints (FLOPS, Memory)

$$\max_{d,w,r} \; Accuracy(\mathcal{N}(d,w,r)) \quad \text{s.t.} \begin{cases} \text{Memory}(\mathcal{N}) \leq \text{target_memory} \\ \text{FLOPS}(\mathcal{N}) \leq \text{target_flops} \end{cases}$$

Maximize accuracy of the model by varying d,w,r **given memory/flop constraints**

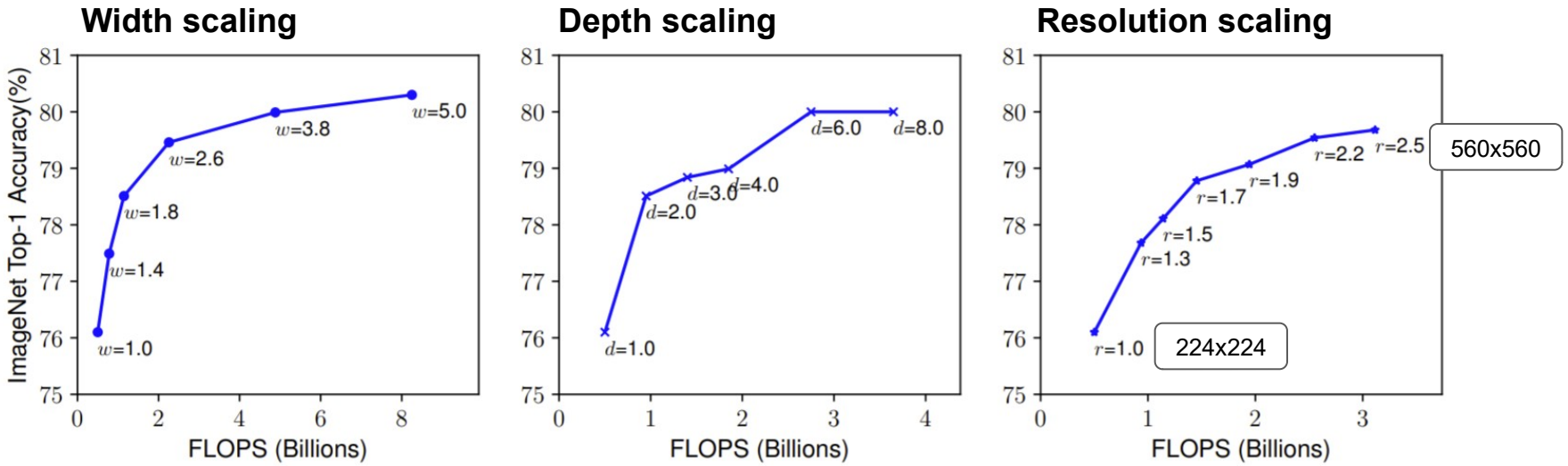
$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\left(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\right)$$

The model **parameterized by d, w, r**

Given:
d = depth coef
w = width coef
r = resolution coef

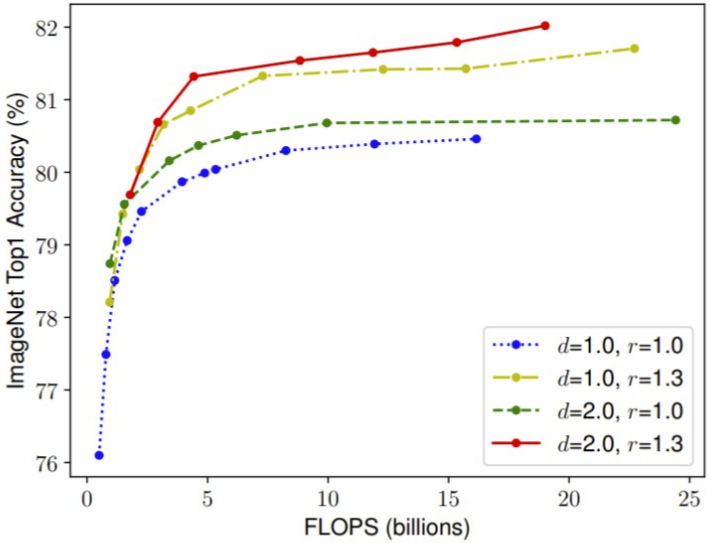
Input:
H_i = Layer Height
W_i = Layer Width
C_i = Layer Channels

Single dimension scaling



Compound scaling experiment

Width scaling



- Each model has a fixed depth and resolution coefficient
- Each dot represents a model with a different width coefficient

- The blue line is a network where (*d=1.0, r=1.0)* and has 18 convolutional layers with resolution 224x224
- The red line is a network where (*d=2.0, r=1.3)* and has 36 convolutional layers with resolution 299x299

Model scaling - variable definitions

Parameters

$$\begin{cases} \text{depth: } d = \alpha^{\phi} \\ \text{width: } w = \beta^{\phi} \\ \text{resolution: } r = \gamma^{\phi} \end{cases}$$

Constraints

$$\begin{cases} \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{cases}$$

- Φ defines ratio of resources available for model scaling relative to a baseline
  - For any Φ, total flops available will increase by 2^{Φ}
    - $\Phi = 0$ is the baseline model
    - $\Phi = 1$ requires 2x resources
- $(\alpha^{\Phi}, \beta^{\Phi}, \gamma^{\Phi})$ determine how the network width, depth, and resolution is scaled

FLOPS scaling of a convolution operation

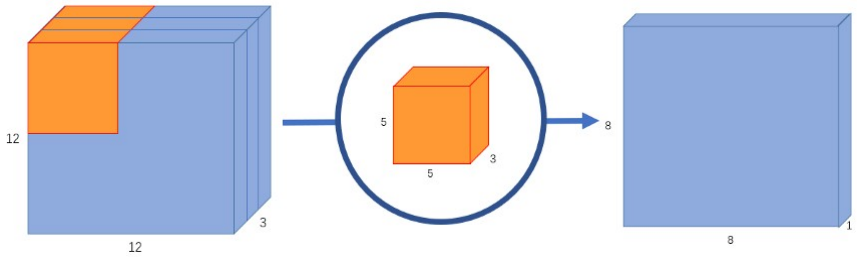
- FLOPS of a regular convolution operation is proportional to d, r^2, w^2
  - Doubling d would double the number of convolution layers
  - Doubling r would double the width and height dimensions, requiring r^2 more flops
  - Doubling w would double the number of filters, requiring w^2 more flops (*)
    - Example on next slide

$$\mathbf{y}^{(m)} = f\left(\sum_n \mathbf{W}^{(n,m)}\,\mathbf{x}^{(n)} + b^{(m)}\right)$$

$$\begin{bmatrix}\mathbf{y}^{(m)}\end{bmatrix} = \begin{bmatrix}\mathbf{W}^{(n,m)}\end{bmatrix}\begin{bmatrix}\mathbf{x}^{(n)}\end{bmatrix} + \begin{bmatrix}b^{(m)}\end{bmatrix}$$

(*)
https://www.math.purdue.edu/~nwinovic/deep_learning.html
https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

FLOPS scaling example



$$\mathbf{y}^{(m)} = f\left(\sum_n \mathbf{W}^{(n,m)}\,\mathbf{x}^{(n)} + b^{(m)}\right)$$

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

- Requires 5x5x3=75 multiplications per output filter every time the filter slides (8x8 times)
  - 4800 total operations per filter
- Doubling the number of input/output filters (w=2) requires 5x5x6=150 multiplications for 2 filters (8x8 times)
  - 19200 total operations = 4800*2*2=4800*w^2

Model Scaling

Model Scaling - constraint explanation

- Taking desired scaling 2^{Φ} and above relationship into account:

$$2^{\phi} \approx (d * w^2 * r^2)$$
$$2^{\phi} \approx ((\alpha^{\phi}) * (\beta^{\phi})^2 * (\gamma^{\phi})^2)$$
$$2^{\phi} \approx (\alpha * \beta^2 * \gamma^2)^{\phi}$$

  - Therefore $\alpha * \beta^2 * \gamma^2 \sim 2$

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

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

Baseline EfficientNet architecture

- Determine initial architecture through Neural Architecture Search, with $ACC(m) \times [FLOPS(m)/T]^w$ as the optimization goal.
  - Given model m
  - Target flops T
  - Constant $w=-0.07$ is a hyperparameter for controlling the trade-off between accuracy and flops

EfficientNet-B0 Architecture

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

Scaling up EfficientNet

- $\alpha=1.2$, $\beta=1.1$, $\gamma=1.15$ were found through grid search with $\Phi = 1$ (2x more resources available than for EfficientNet-B0)
  - Maximizing Accuracy while remaining within target memory/flop constraints
- Network Coefficients are therefore
  - $d = 1.2^{\Phi}$
  - $w = 1.1^{\Phi}$
  - $r = 1.15^{\Phi}$
- Scaling up Φ from 1-7 and plugging into the equation from before provides us with the EfficientNet architectures!

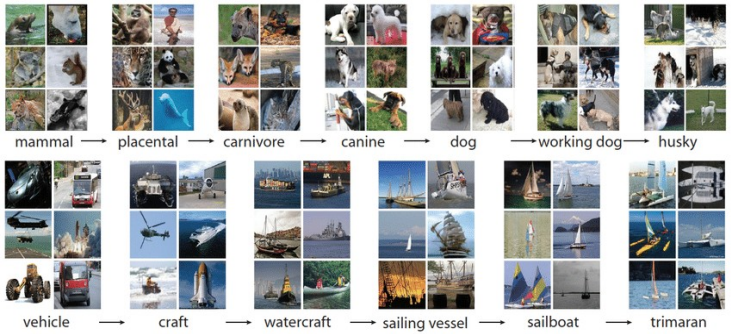
$$\mathcal{N}(d, w, r) = \bigodot_{i=1 \ldots s} \hat{\mathcal{F}}_i^{d \cdot \hat{L}_i} \left(X_{\langle r \cdot \hat{H}_i, r \cdot \hat{W}_i, w \cdot \hat{C}_i \rangle} \right)$$

- **Note:** It is possible to achieve even better performance by searching for α, β, γ directly around a large model, but the search cost becomes prohibitively more expensive

Results

ImageNet Benchmark

- ImageNet is a database of images depicting hundreds of different nouns
- Each noun is associated with thousands of images
- ImageNet is a benchmark for evaluating CNN performance across industry and academia alike



IMAGENET

http://www.image-net.org/

ImageNet results

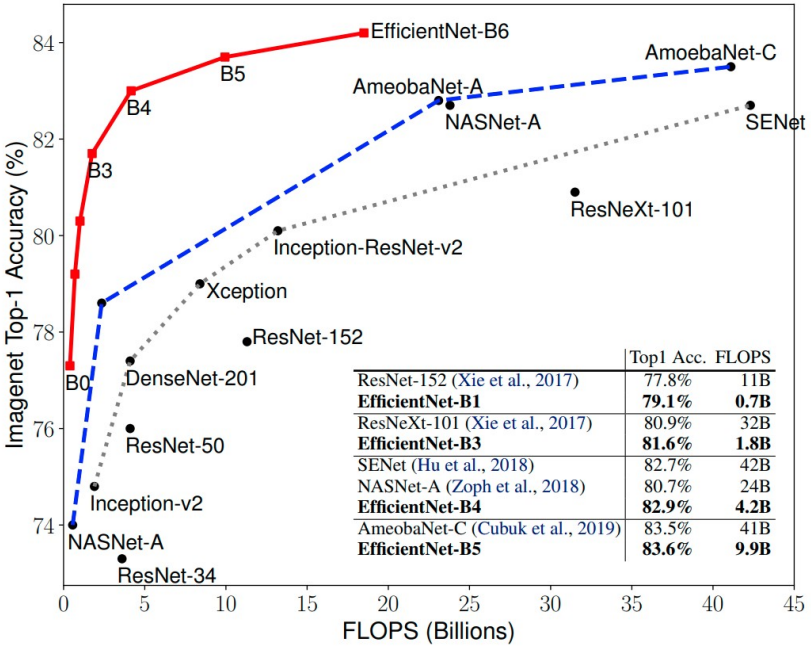
Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**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	**80.1%**	**94.9%**	**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-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**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

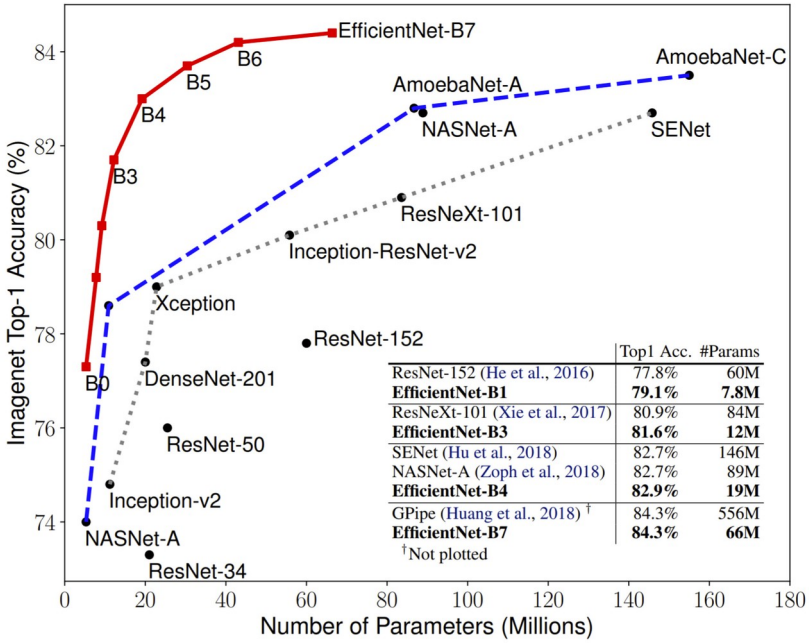
ImageNet top-1 accuracy vs flops/#params

FLOPS Required (lower is better)



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.1%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.6%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.9%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.6%**	**9.9B**

Model parameters required (lower is better)



	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

Results

Compound scaling results and latency comparison

Model	FLOPS	Top-1 Acc.	
Baseline MobileNetV1 (Howard et al., 2017)	0.6B	70.6%	Baseline
Scale MobileNetV1 by width (w=2)	2.2B	74.2%	
Scale MobileNetV1 by resolution (r=2)	2.2B	72.7%	
compound scale (d=1.4, w=1.2, r=1.3)	**2.3B**	**75.6%**	C.S.
Baseline MobileNetV2 (Sandler et al., 2018)	0.3B	72.0%	Baseline
Scale MobileNetV2 by depth (d=4)	1.2B	76.8%	
Scale MobileNetV2 by width (w=2)	1.1B	76.4%	
Scale MobileNetV2 by resolution (r=2)	1.2B	74.8%	
MobileNetV2 compound scale	**1.3B**	**77.4%**	C.S.
Baseline ResNet-50 (He et al., 2016)	4.1B	76.0%	Baseline
Scale ResNet-50 by depth (d=4)	16.2B	78.1%	
Scale ResNet-50 by width (w=2)	14.7B	77.7%	
Scale ResNet-50 by resolution (r=2)	16.4B	77.5%	
ResNet-50 compound scale	**16.7B**	**78.8%**	C.S.

- Consistent improvement in resulting accuracy when compound scaling (C.S.) is applied across various networks
- Significant improvement in latency as well:

Table 4. **Inference Latency Comparison** – Latency is measured with batch size 1 on a single core of Intel Xeon CPU E5-2690.

	Acc. @ Latency		Acc. @ Latency
ResNet-152	77.8% @ 0.554s	GPipe	84.3% @ 19.0s
EfficientNet-B1	78.8% @ 0.098s	EfficientNet-B7	84.4% @ 3.1s
Speedup	**5.7x**	**Speedup**	**6.1x**

Transfer learning results

	Comparison to best reported results					
	Model	Acc.	#Param	Our Model	Acc.	#Param(ratio)
CIFAR-10	†Gpipe	**99.0%**	556M	EfficientNet-B7	98.9%	64M (8.7x)
CIFAR-100	Gpipe	91.3%	556M	EfficientNet-B7	**91.7%**	64M (8.7x)
Birdsnap	GPipe	83.6%	556M	EfficientNet-B7	**84.3%**	64M (8.7x)
Stanford Cars	‡DAT	**94.8%**	-	EfficientNet-B7	94.7%	-
Flowers	DAT	97.7%	-	EfficientNet-B7	**98.8%**	-
FGVC Aircraft	DAT	92.9%	-	EfficientNet-B7	**92.9%**	-
Oxford-IIIT Pets	GPipe	**95.9%**	556M	EfficientNet-B6	95.4%	41M (14x)
Food-101	GPipe	93.0%	556M	EfficientNet-B7	**93.0%**	64M (8.7x)
Geo-Mean						**(9.6x)**

Table 5. **EfficientNet Performance Results on Transfer Learning Datasets**. Our scaled EfficientNet models achieve new state-of-the-art accuracy for 5 out of 8 datasets, with 9.6x fewer parameters on average.

Results

Class activation map comparison

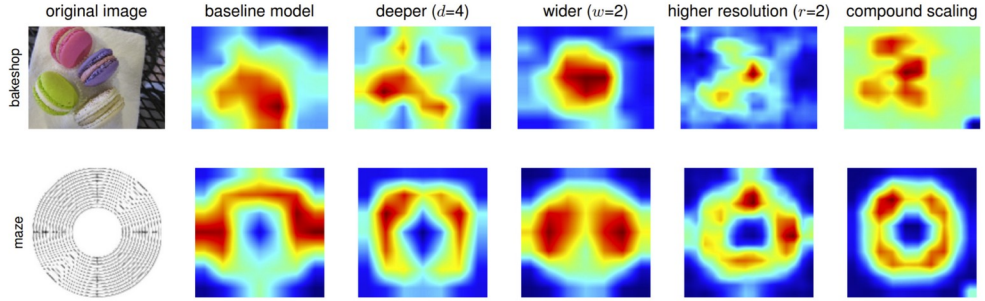


Figure 7. **Class Activation Map (CAM) (Zhou et al., 2016) for Models with different scaling methods-** Our compound scaling method allows the scaled model (last column) to focus on more relevant regions with more object details. Model details are in Table 7.

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%**

Questions

- Could a similar scaling technique be used on transformer-based networks?
- How does this technique relate to the Lottery Ticket Hypothesis (related to neural network pruning)?
- How are the paradigms of scaling vs pruning related to each other?
- My Keras Implementation: https://github.com/Sumobot1/efficientnet-keras