EfficientNet

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

A

OWOP season1 03
조유민

Paper

Mingxing Tan, Quoc V. Le (2020) EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan [1] Quoc V. Le [1]

Abstract

Convolutional Neural Networks (ConvNets) are commonly developed at a fixed resource budget, and then scaled up for better accuracy if more resources are available. In this paper, we systematically study model scaling and identify that carefully balancing network depth, width, and resolution can lead to better performance. Based on this observation, we propose a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective *compound coefficient*. We demonstrate the effectiveness of this method on scaling up MobileNets and ResNet.

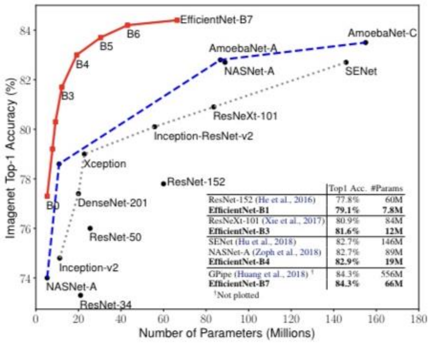
To go even further, we use neural architecture search to design a new baseline network and scale it up to obtain a family of models, called *EfficientNets*, which achieve much better accuracy and efficiency than previous ConvNets. In particular, our EfficientNet-B7 achieves state-of-the-art 84.3% top-1 accuracy on ImageNet, while being **8.4x smaller** and **6.1x faster** on inference than the best existing ConvNet. Our EfficientNets also transfer well and achieve state-of-the-art accuracy on CIFAR-100 (91.7%), Flowers (98.8%), and 3 other transfer learning datasets, with an order of magnitude fewer parameters. Source code is at https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet.

1. Introduction

Scaling up ConvNets is widely used to achieve better accuracy. For example, ResNet (He et al., 2016) can be scaled up from ResNet-18 to ResNet-200 by using more layers; Recently, GPipe (Huang et al., 2018) achieved 84.3% ImageNet top-1 accuracy by scaling up a baseline model four

[1]Google Research, Brain Team, Mountain View, CA. Correspondence to: Mingxing Tan <tanmingxing@google.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.



	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.

time larger. However, the process of scaling up ConvNets has never been well understood and there are currently many ways to do it. The most common way is to scale up ConvNets by their depth (He et al., 2016) or width (Zagoruyko & Komodakis, 2016). Another less common, but increasingly popular, method is to scale up models by image resolution (Huang et al., 2018). In previous work, it is common to scale only one of the three dimensions – depth, width, and image size. Though it is possible to scale two or three dimensions arbitrarily, arbitrary scaling requires tedious manual tuning and still often yields sub-optimal accuracy and efficiency.

In this paper, we want to study and rethink the process of scaling up ConvNets. In particular, we investigate the central question: is there a principled method to scale up ConvNets that can achieve better accuracy and efficiency? Our empirical study shows that it is critical to balance all dimensions of network width/depth/resolution, and surprisingly such balance can be achieved by simply scaling each of them with constant ratio. Based on this observation, we propose a simple yet effective *compound scaling method*. Unlike conventional practice that arbitrary scales these factors, our method uniformly scales network width, depth,

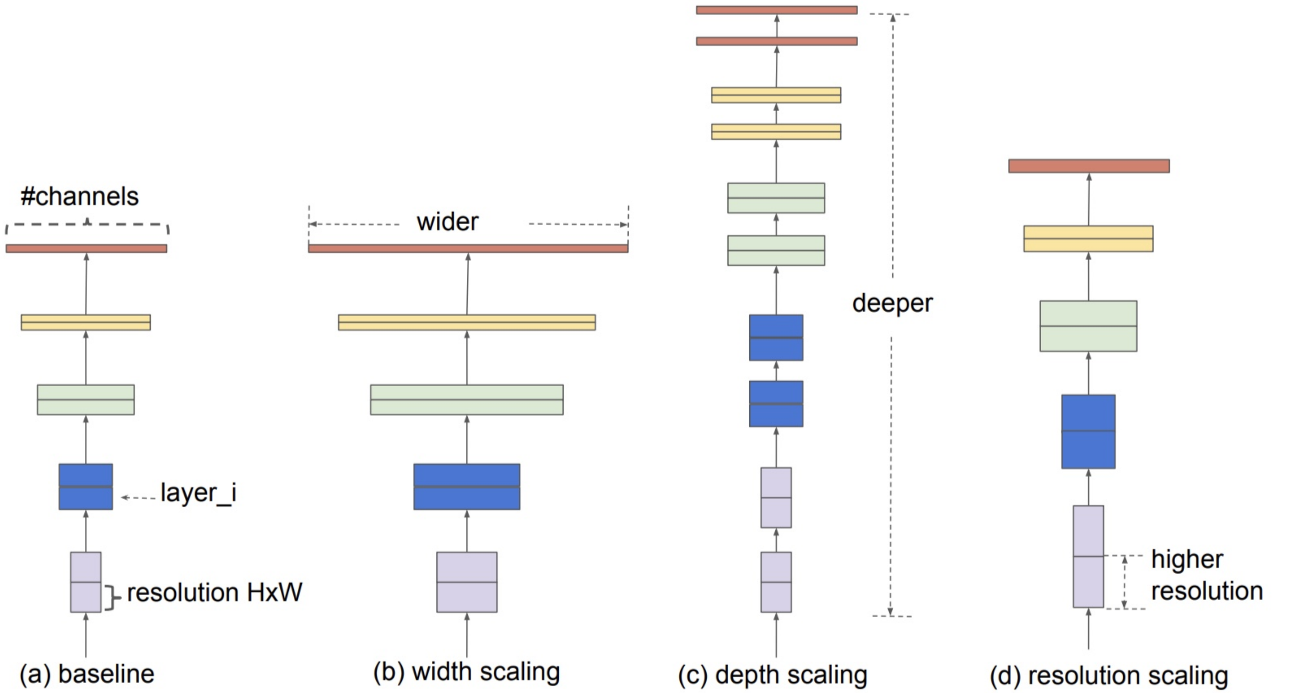
Accuracy vs Efficiency

Trade off between high accuracy and computational efficiency

-> Is there more principled method to obtain better **accuracy** and **efficiency**?

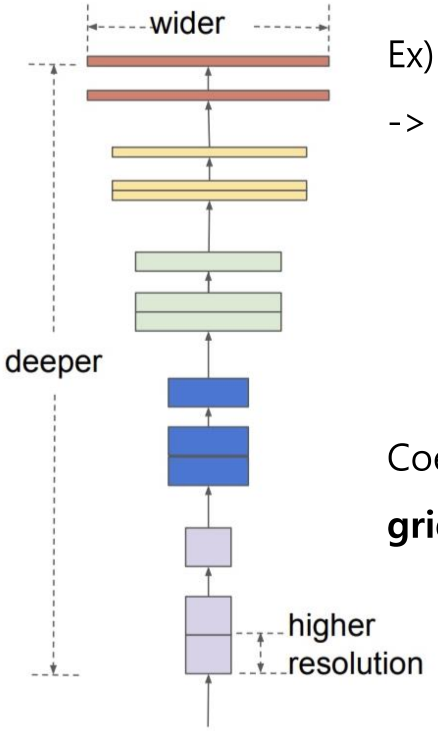
Model Scaling

Scaling up a baseline CNN can achieve high performance



Introduction

Compound scaling method



(e) compound scaling

Ex) 2^n times more computational resources

-> increase the network

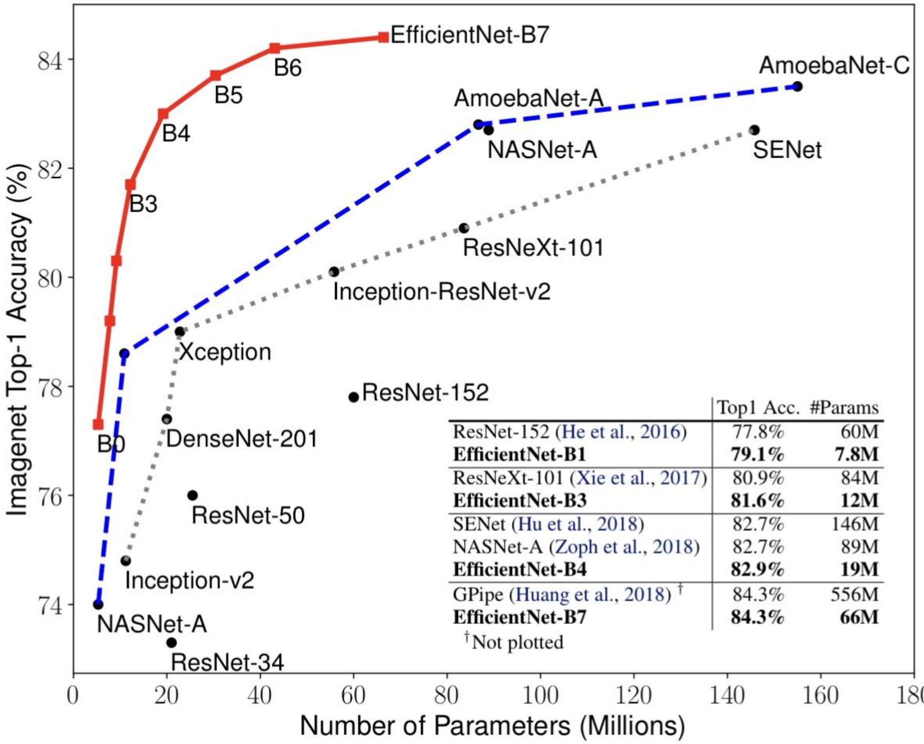
depth by α^n

width by β^n

image size by γ^n

Coefficients are determined by a small **grid search**

Uniformly(Not arbitrary) scales network width, depth, and resolution with a set of fixed **scaling coefficients**.



	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.

Related Work

ConvNet Accuracy

- AlexNet
- GoogleNet
- SENet
- GPipe

Already hit the hardware memory limit…

It needs better efficiency!!!

VS

ConvNet Efficiency

- SqueezeNets
- MobileNets
- ShuffleNets

Unclear how to apply these techniques for larger models!!!

It needs much more expensive tuning cost…

How to effectively scale a ConvNet?

Compound Model Scaling

Problem

Large design space to explore different L, H, W, C

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

$$\mathcal{N} = \mathcal{F}_k \odot \ldots \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1\ldots k} \mathcal{F}_j(X_1)$$

Solution

Reduce the design space by scaling layers uniformly with constant ratio

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

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

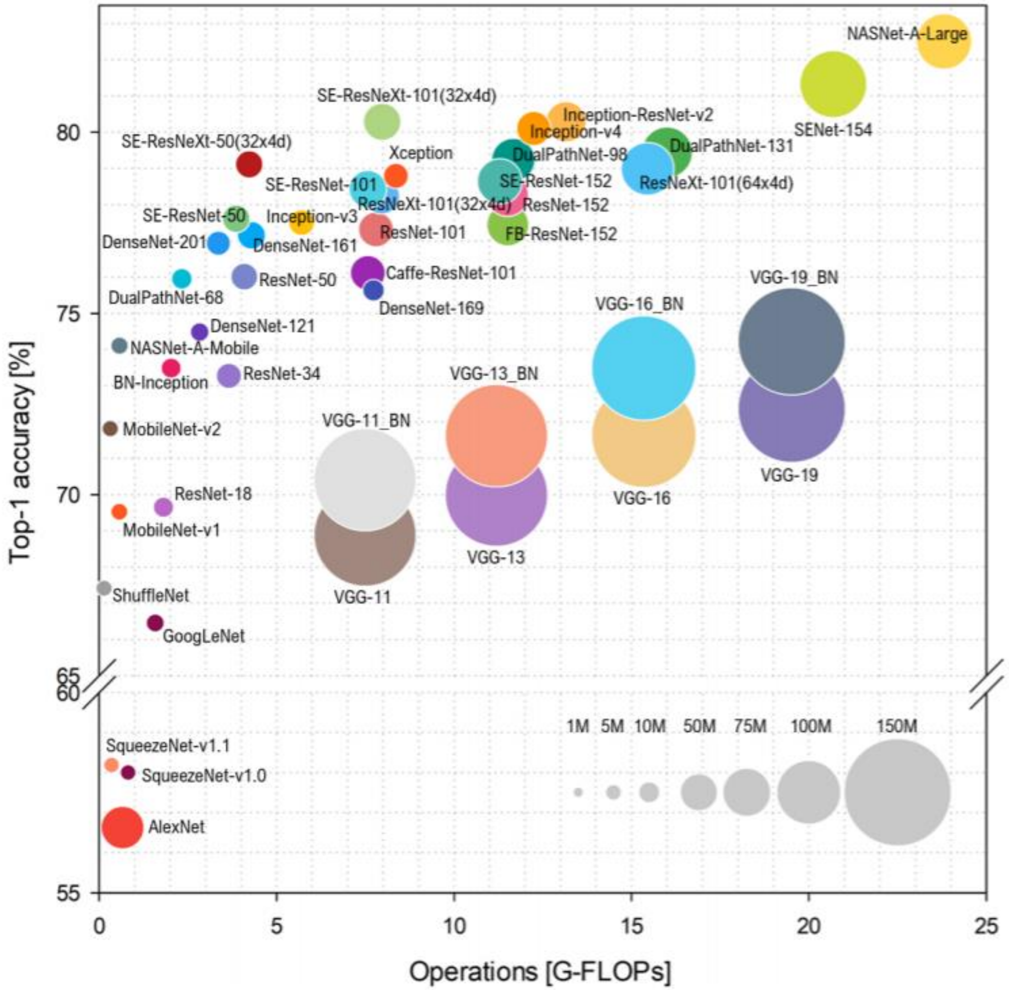
How to find the optimal d, w, r?

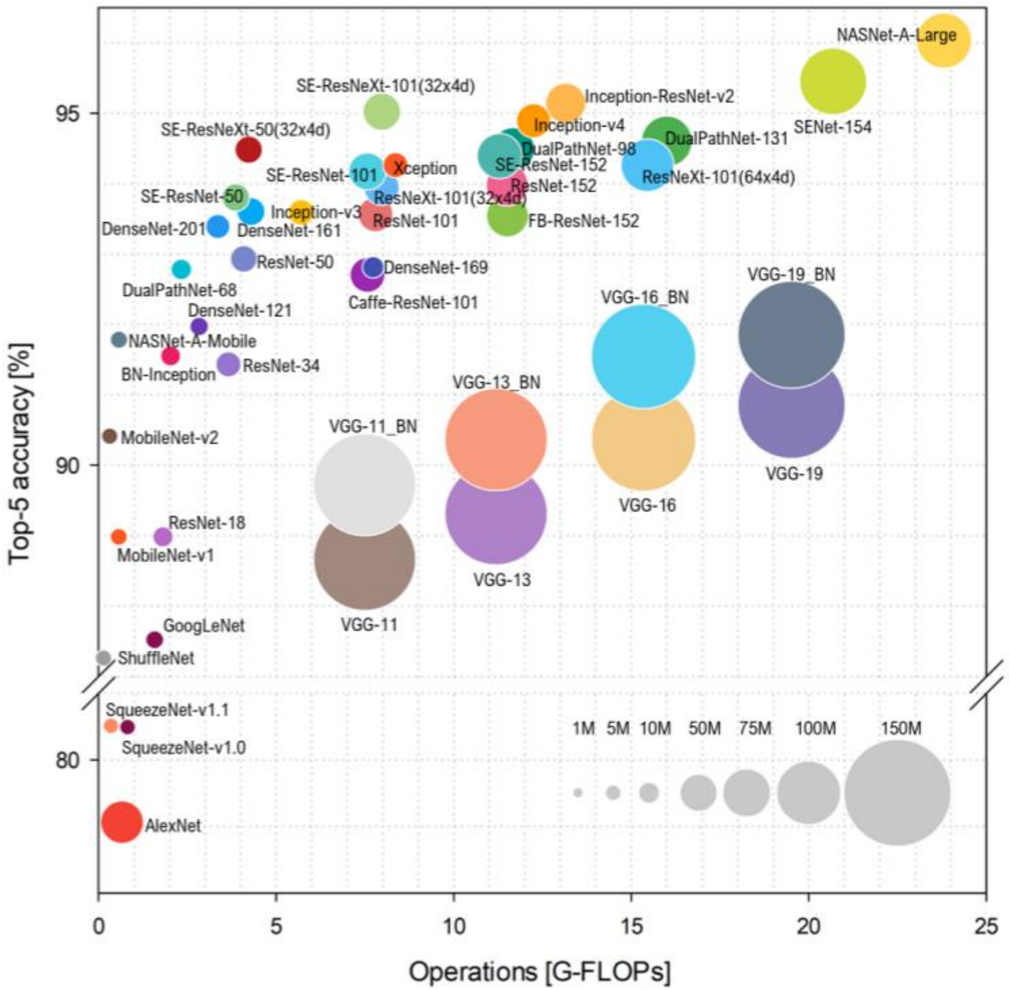
FLOPs

FLOPs(Floating-point operations per seconds)

1,000,000,000 FLOPs = 1 G-FLOPs(Giga FLOPs)

1,000 G-FLOPs = 1 T-FLOPs(Tera FLOPs)





Compound Model Scaling

Depth(d)

Diminishing accuracy for very deep ConvNet

Width(w)

Difficulties in capturing higher level features

Resolution(r)

Accuracy gain diminishes for very high resolutions

∴ Scaling up any dimension of network d, w, r improves accuracy, but the accuracy gain **diminishes for bigger models**

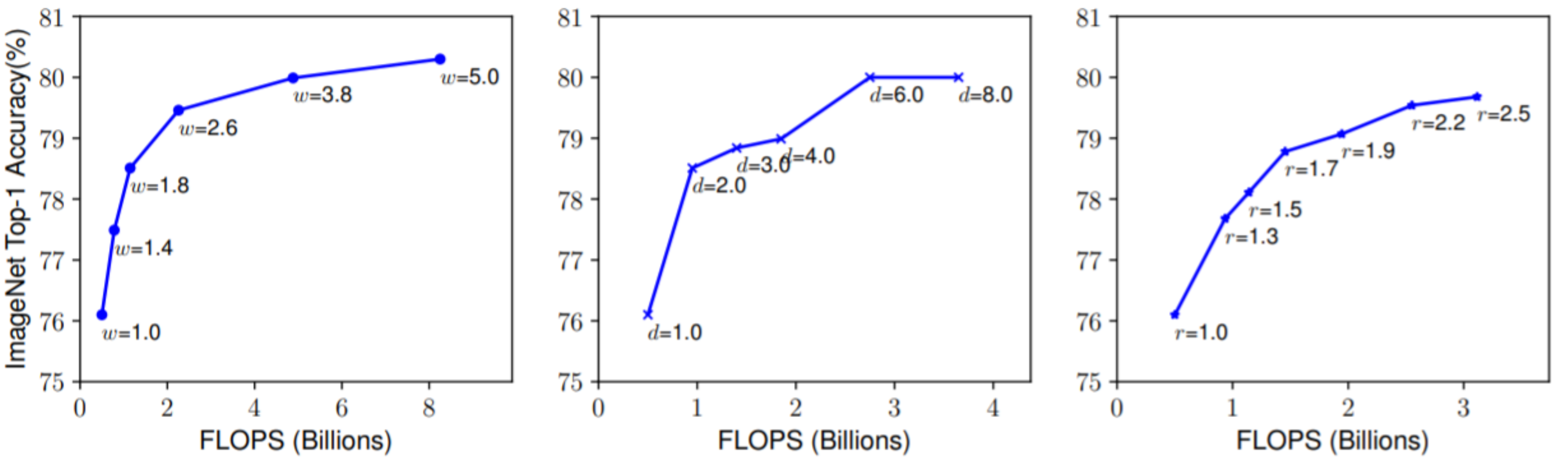


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single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

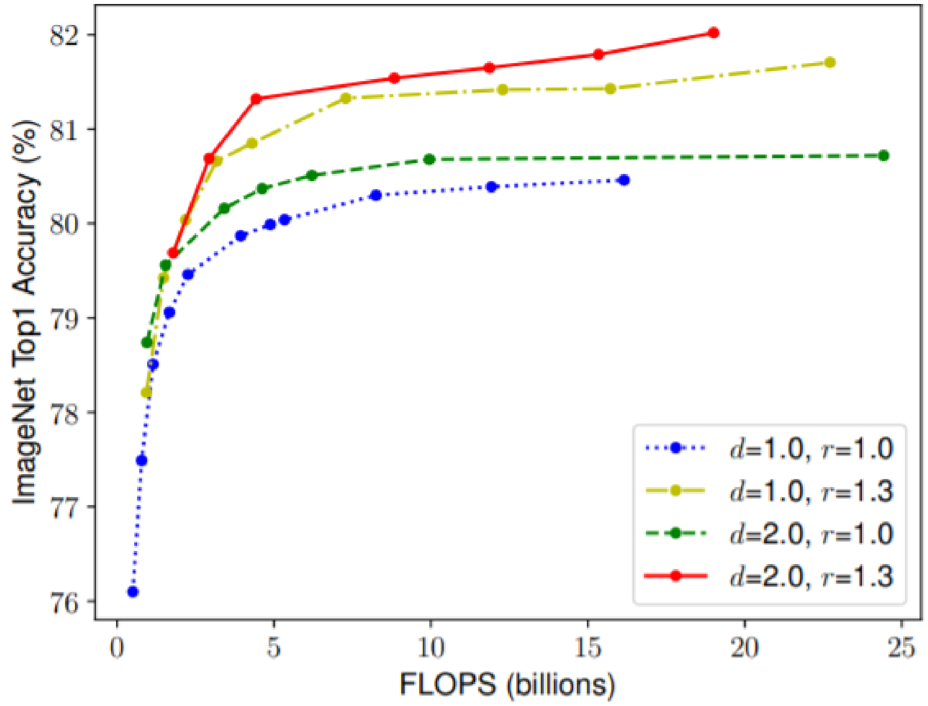
Compound Model Scaling

Intuition

Different scaling dimensions are **not independent**

-> It needs to coordinate and balance different scaling dimensions rather than conventional single-dimension scaling.

Compare w under different network d and r



∴ It is critical to **balance all dimensions of network** width, depth, and resolution during ConvNet scaling.

Model Architecture

Compound scaling method

Φ = how many more resources are available for model scaling

α, β, γ = how to assign these extra resources to network

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

Optimization

ACC, **FLOPS** = accuracy and FLOPS of model m

T = target FLOPS

w = hyperparameter to control the trade-off

$$ACC(m) \times [FLOPS(m)/T]^{w}$$

※ Unlike MNasNet, **Latency** is not included in the optimization goal

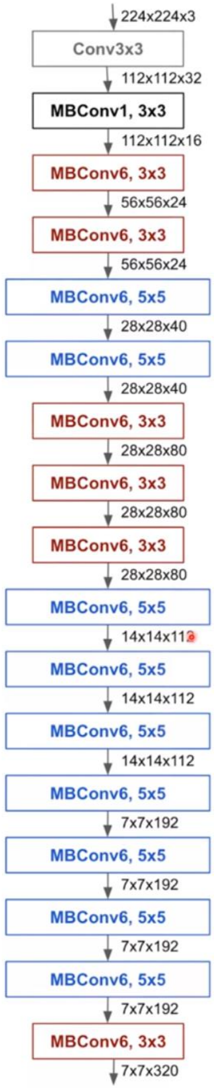
Baseline

Perform NAS using AutoML MNAS framework

It uses mobile inverted bottleneck convolution (**MBConv**), similar to **MobileNetV2 and MNasNet**, but slightly larger due to an **increased FLOP** budget

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



NasNet

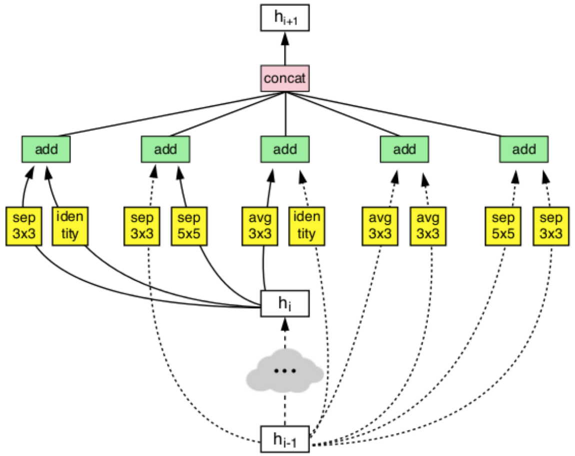
NasNet

Maintain existing **NAS(Neural Architecture Search)** methods, but finding a **new search space**

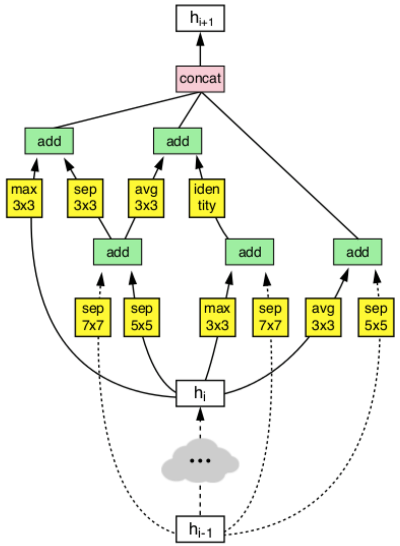
Find the best layers that work well on CIFAR-10(Blocks or cells are searched RL and RNN)

and then **transfer** the best learned architecture to ImageNet image classification and COCO objec

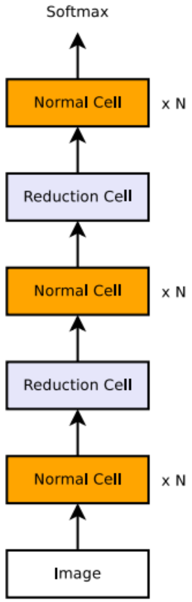
detection.



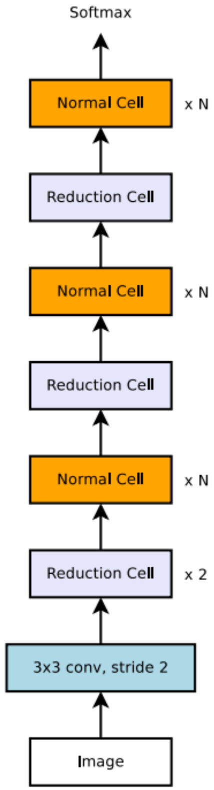
Normal Cell



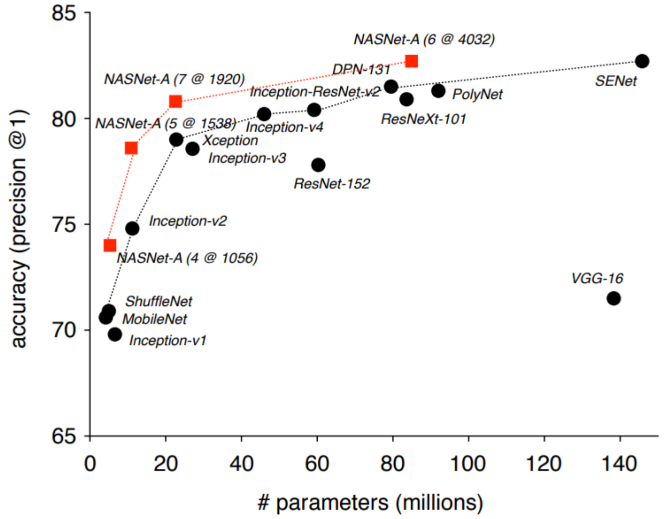
Reduction Cell

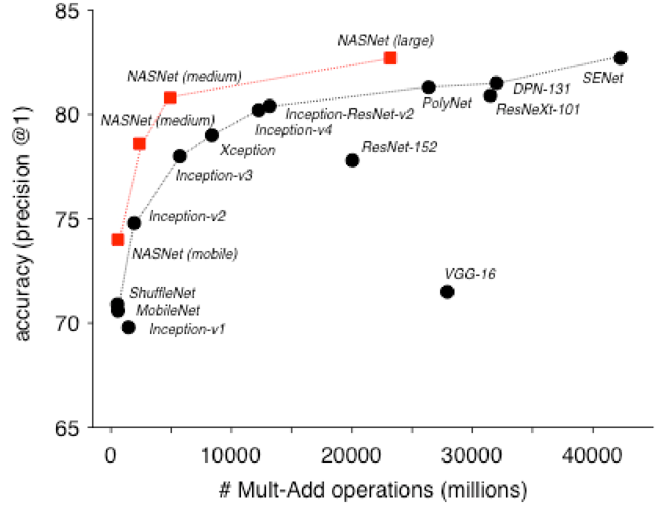


CIFAR10 Architecture



ImageNet Architecture



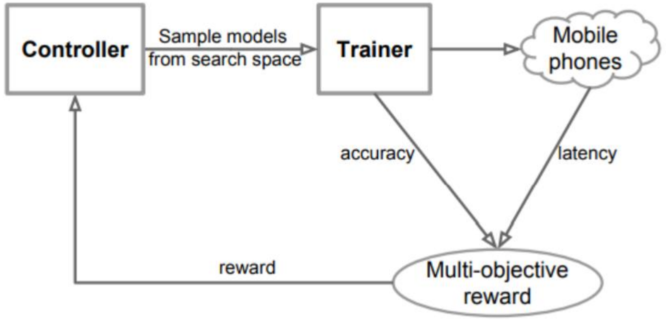


MNasNet

MNasNet

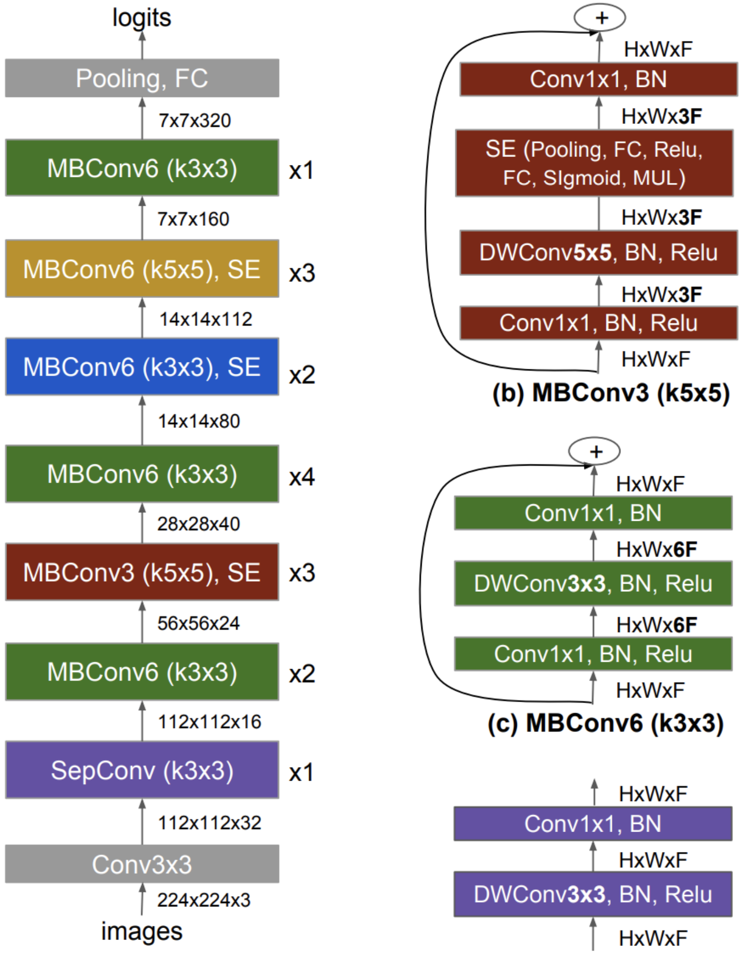
Mobile platform aware neural architecture search algorithm

Incorporates model **latency** explicitly into the main objective



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

Model	Type	#Params	#Mult-Adds	Top-1 Acc. (%)	Top-5 Acc. (%)	Inference Latency
MobileNetV1 [11]	manual	4.2M	575M	70.6	89.5	113ms
SqueezeNext [5]	manual	3.2M	708M	67.5	88.2	-
ShuffleNet (1.5x) [33]	manual	3.4M	292M	71.5	-	-
ShuffleNet (2x)	manual	5.4M	524M	73.7	-	-
ShuffleNetV2 (1.5x) [24]	manual	-	299M	72.6	-	-
ShuffleNetV2 (2x)	manual	-	597M	75.4	-	-
CondenseNet (G=C=4) [14]	manual	2.9M	274M	71.0	90.0	-
CondenseNet (G=C=8)	manual	4.8M	529M	73.8	91.7	-
MobileNetV2 [29]	manual	3.4M	300M	72.0	91.0	75ms
MobileNetV2 (1.4x)	manual	6.9M	585M	74.7	92.5	143ms
NASNet-A [36]	auto	5.3M	564M	74.0	91.3	183ms
AmoebaNet-A [26]	auto	5.1M	555M	74.5	92.0	190ms
PNASNet [19]	auto	5.1M	588M	74.2	91.9	-
DARTS [21]	auto	4.9M	595M	73.1	91	-
MnasNet-A1	**auto**	**3.9M**	**312M**	**75.2**	**92.5**	**78ms**
MnasNet-A2	**auto**	**4.8M**	**340M**	**75.6**	**92.7**	**84ms**
MnasNet-A3	**auto**	**5.2M**	**403M**	**76.7**	**93.3**	**103ms**



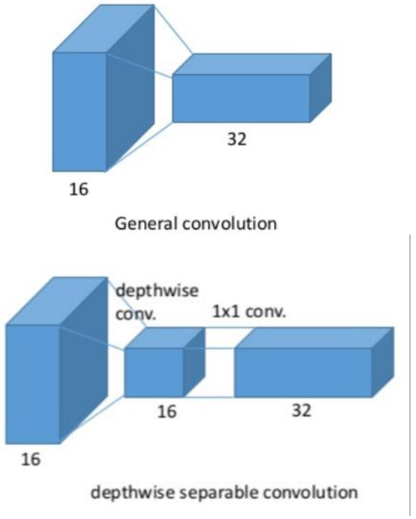
(a) MnasNet-A1

(b) MBConv3 (k5x5)

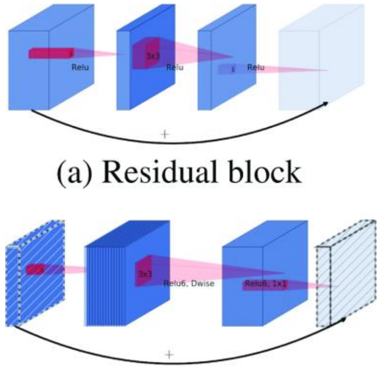
(c) MBConv6 (k3x3)

(d) SepConv (k3x3)

MobileNet V1



MobileNet V2



(a) Residual block

(b) Inverted residual block

Model Architecture & Results

Scale up

Step1.

Grid search once on the small baseline network(**fixing Φ=1**)

Best values for EfficientNet-B0

α = 1.2, β =1.1, γ = 1.1

Step2.

Use the same scaling coefficients for all other models

Fix α, β, γ and scale up with **different Φ**

∴ Obtain EfficientNet-B1 to B7

Table 2. **EfficientNet Performance Results on ImageNet** (Russakovsky et al., 2015). All EfficientNet models are scaled from our baseline EfficientNet-B0 using different compound coefficient ϕ in Equation 3. ConvNets with similar top-1/top-5 accuracy are grouped together for efficiency comparison. Our scaled EfficientNet models consistently reduce parameters and FLOPS by an order of magnitude (up to 8.4x parameter reduction and up to 16x FLOPS reduction) than existing ConvNets.

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-B3	**81.6%**	**95.7%**	**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	**82.9%**	**96.4%**	**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	**83.6%**	**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	**84.0%**	**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).

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**

-> Fast on real hardware

Results

ImageNet Results

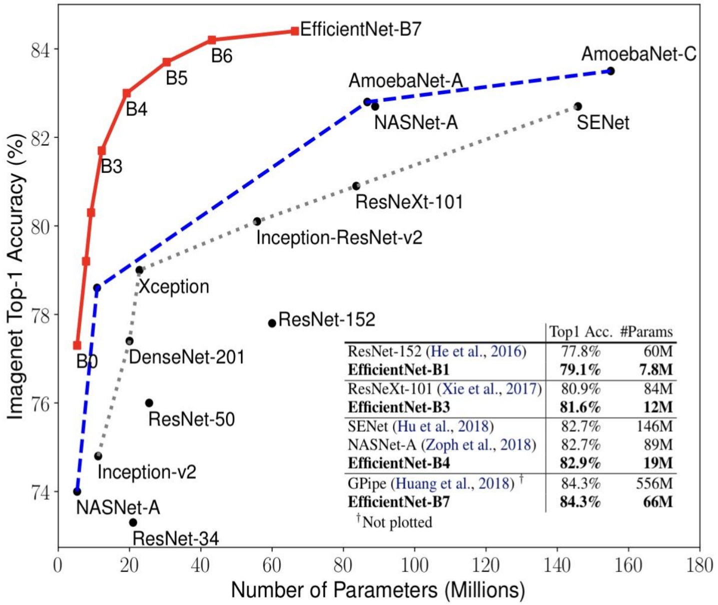


Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1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

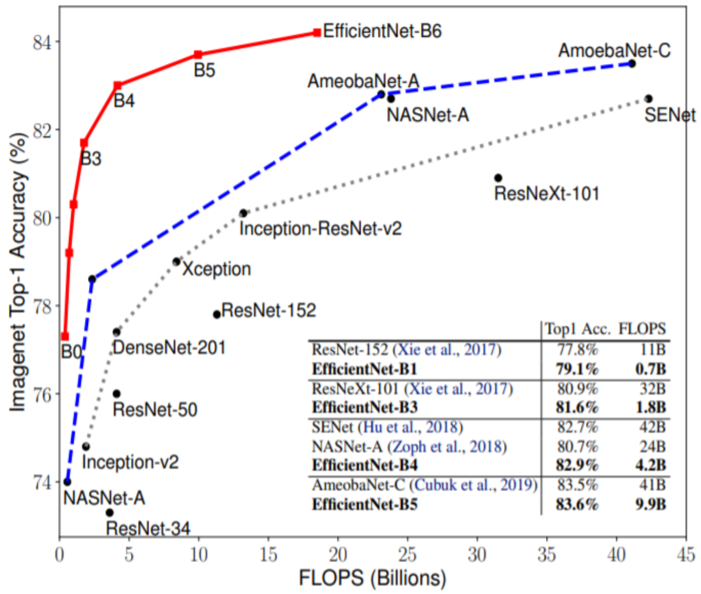


Figure 5. **FLOPS vs. ImageNet Accuracy –** Similar to Figure 1 except it compares FLOPS rather than model size.

-> Better accuracy with much fewer parameters and FLOPs

Apply on other models

Table 3. **Scaling Up MobileNets and ResNet.**

Model	FLOPS	Top-1 Acc.
Baseline MobileNetV1 (Howard et al., 2017)	0.6B	70.6%
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%**
Baseline MobileNetV2 (Sandler et al., 2018)	0.3B	72.0%
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%**
Baseline ResNet-50 (He et al., 2016)	4.1B	76.0%
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%**

Results

Transfer Learning Results

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.

	Comparison to best public-available results						Comparison to best reported results					
	Model	Acc.	#Param	Our Model	Acc.	#Param(ratio)	Model	Acc.	#Param	Our Model	Acc.	#Param(ratio)
CIFAR-10	NASNet-A	98.0%	85M	EfficientNet-B0	98.1%	4M (21x)	†Gpipe	**99.0%**	556M	EfficientNet-B7	98.9%	64M (8.7x)
CIFAR-100	NASNet-A	87.5%	85M	EfficientNet-B0	88.1%	4M (21x)	Gpipe	91.3%	556M	EfficientNet-B7	**91.7%**	64M (8.7x)
Birdsnap	Inception-v4	81.8%	41M	EfficientNet-B5	82.0%	28M (1.5x)	GPipe	83.6%	556M	EfficientNet-B7	**84.3%**	64M (8.7x)
Stanford Cars	Inception-v4	93.4%	41M	EfficientNet-B3	93.6%	10M (4.1x)	‡DAT	**94.8%**	-	EfficientNet-B7	94.7%	-
Flowers	Inception-v4	98.5%	41M	EfficientNet-B5	98.5%	28M (1.5x)	DAT	97.7%	-	EfficientNet-B7	**98.8%**	-
FGVC Aircraft	Inception-v4	90.9%	41M	EfficientNet-B3	90.7%	10M (4.1x)	DAT	92.9%	-	EfficientNet-B7	**92.9%**	-
Oxford-IIIT Pets	ResNet-152	94.5%	58M	EfficientNet-B4	94.8%	17M (5.6x)	GPipe	**95.9%**	556M	EfficientNet-B6	95.4%	41M (14x)
Food-101	Inception-v4	90.8%	41M	EfficientNet-B4	91.5%	17M (2.4x)	GPipe	93.0%	556M	EfficientNet-B7	**93.0%**	64M (8.7x)
Geo-Mean						**(4.7x)**						**(9.6x)**

†GPipe (Huang et al., 2018) trains giant models with specialized pipeline parallelism library.
‡DAT denotes domain adaptive transfer learning (Ngiam et al., 2018). Here we only compare ImageNet-based transfer learning results.
Transfer accuracy and #params for NASNet (Zoph et al., 2018), Inception-v4 (Szegedy et al., 2017), ResNet-152 (He et al., 2016) are from (Kornblith et al., 2019).

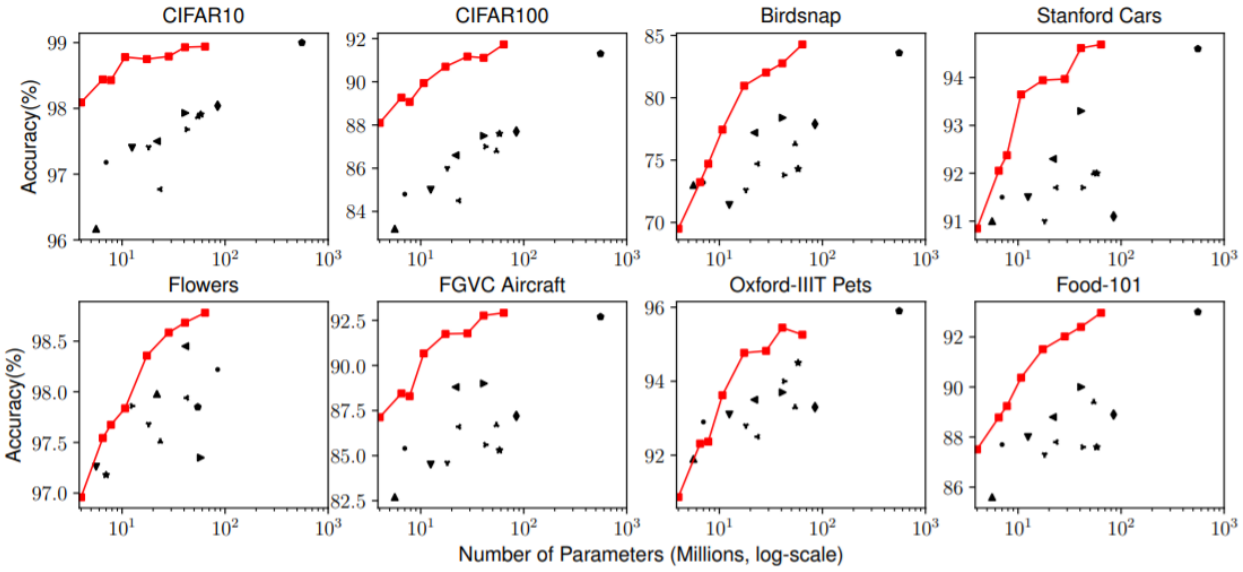
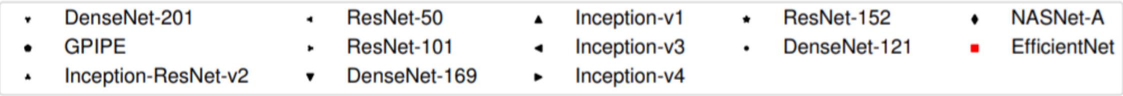


Table 6. **Transfer Learning Datasets.**

Dataset	Train Size	Test Size	#Classes
CIFAR-10 (Krizhevsky & Hinton, 2009)	50,000	10,000	10
CIFAR-100 (Krizhevsky & Hinton, 2009)	50,000	10,000	100
Birdsnap (Berg et al., 2014)	47,386	2,443	500
Stanford Cars (Krause et al., 2013)	8,144	8,041	196
Flowers (Nilsback & Zisserman, 2008)	2,040	6,149	102
FGVC Aircraft (Maji et al., 2013)	6,667	3,333	100
Oxford-IIIT Pets (Parkhi et al., 2012)	3,680	3,369	37
Food-101 (Bossard et al., 2014)	75,750	25,250	101



Discussion

Different scaling methods on Baseline

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%**

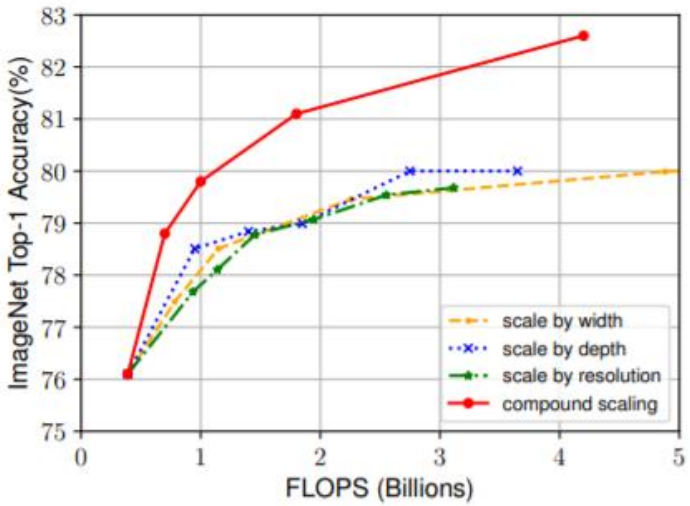


Figure 8. **Scaling Up EfficientNet-B0 with Different Methods.**

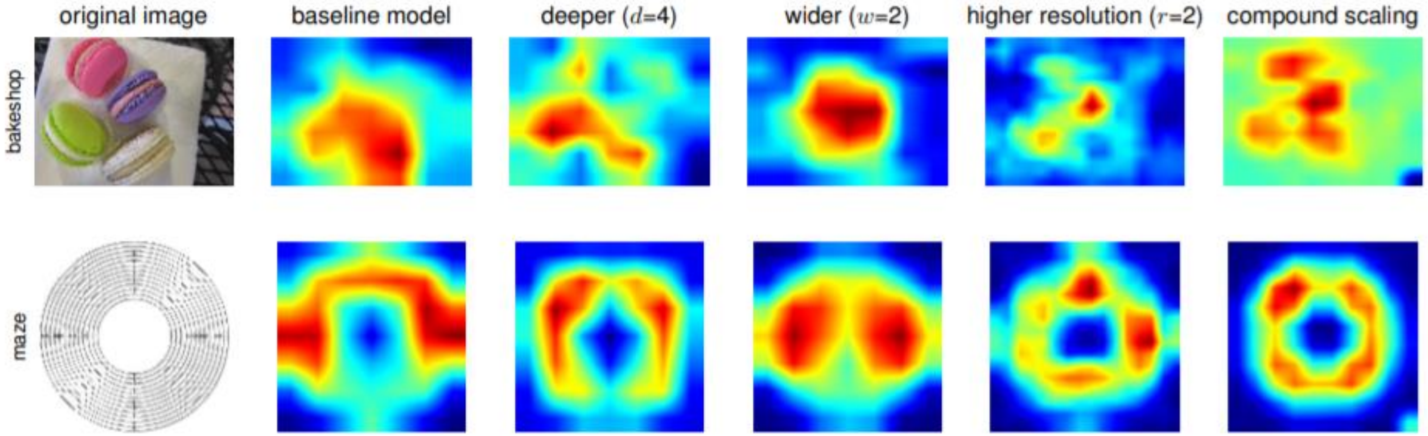


Figure 7. **Class Activation Map (CAM) (Zhou et al., 2016) for Models with different scaling methods-** Our compound scaling method allows the scaled model (last column) to focus on more relevant regions with more object details. Model details are in Table 7.

References

[Paper]
https://arxiv.org/abs/1905.11946

[Code]
https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet