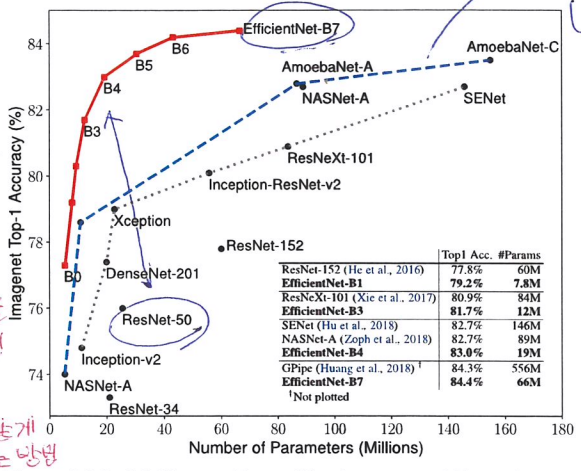
EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

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 84.4% top-1 / 97.1% top-5 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.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) [1]	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

[1] 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 84.4% 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.

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 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,

[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.

* FLOPS (Floating point Operations Per Second)
: 컴퓨터의 성능을 수치로 나타낼 때 주로 사용되는 단위. 초당 부동 소수점 연산이라는 뜻. Conventional scaling

* Partitioning (데이터 베이스 분할)
: 데이터 베이스 여러부분으로 분할.
중요한 튜닝기법. 데이터 너무 커져서 조회시간↑ 관리 용이성, 성능, 가용성 향상 이유.

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

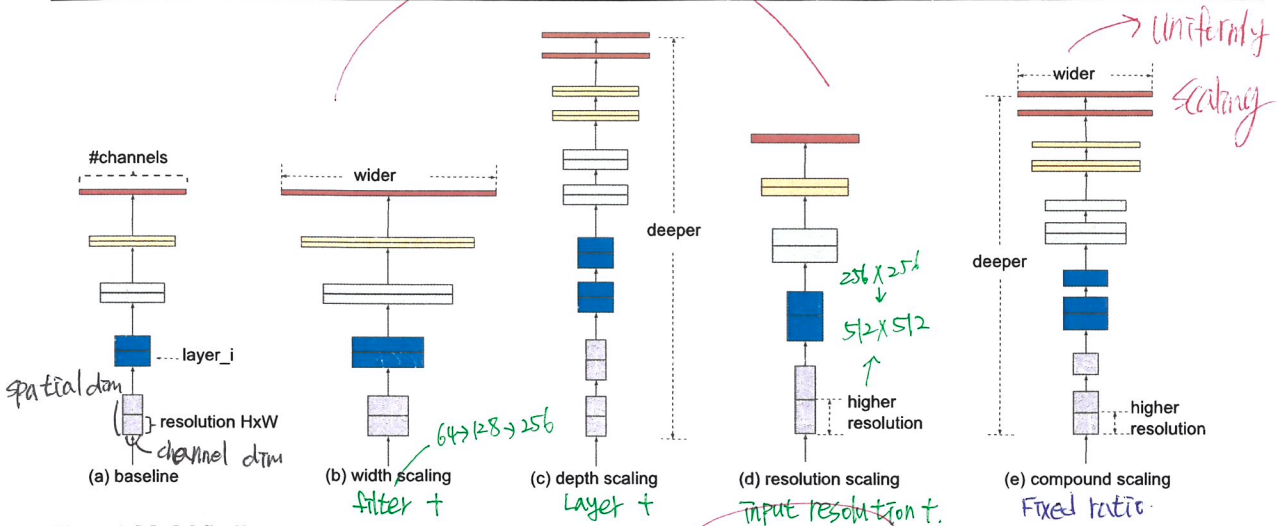


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

and resolution with a set of fixed scaling coefficients. For example, if we want to use 2^N times more computational resources, then we can simply increase the network depth by α^N, width by β^N, and image size by γ^N, where α, β, γ are constant coefficients determined by a small grid search on the original small model. Figure 2 illustrates the difference between our scaling method and conventional methods.

Intuitively, the compound scaling method makes sense because if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image. In fact, previous theoretical (Raghu et al., 2017; Lu et al., 2018) and empirical results (Zagoruyko & Komodakis, 2016) both show that there exists certain relationship between network width and depth, but to our best knowledge, we are the first to empirically quantify the relationship among all three dimensions of network width, depth, and resolution.

We demonstrate that our scaling method work well on existing MobileNets (Howard et al., 2017; Sandler et al., 2018) and ResNet (He et al., 2016). Notably, the effectiveness of model scaling heavily depends on the baseline network; to go even further, we use neural architecture search (Zoph & Le, 2017; Tan et al., 2019) to develop a new baseline network, and scale it up to obtain a family of models, called *EfficientNets*. Figure 1 summarizes the ImageNet performance, where our EfficientNets significantly outperform other ConvNets. In particular, our EfficientNet-B7 surpasses the best existing GPipe accuracy (Huang et al., 2018), but using 8.4x fewer parameters and running 6.1x faster on inference. Compared to the widely used ResNet-50 (He et al., 2016), our EfficientNet-B4 improves the top-1 accuracy from 76.3% to 83.0% (+6.7%) with similar FLOPS. Besides ImageNet, EfficientNets also transfer well and achieve state-of-the-art accuracy on 5 out of 8 widely used datasets, while reducing parameters by up to 21x than existing ConvNets.

2. Related Work

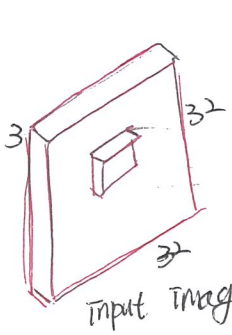
ConvNet Accuracy: Since AlexNet (Krizhevsky et al., 2012) won the 2012 ImageNet competition, ConvNets have become increasingly more accurate by going bigger: while the 2014 ImageNet winner GoogleNet (Szegedy et al., 2015) achieves 74.8% top-1 accuracy with about 6.8M parameters, the 2017 ImageNet winner SENet (Hu et al., 2018) achieves 82.7% top-1 accuracy with 145M parameters. Recently, GPipe (Huang et al., 2018) further pushes the state-of-the-art ImageNet top-1 validation accuracy to 84.3% using 557M parameters: it is so big that it can only be trained with a specialized pipeline parallelism library by partitioning the network and spreading each part to a different accelerator. While these models are mainly designed for ImageNet, recent studies have shown better ImageNet models also perform better across a variety of transfer learning datasets (Kornblith et al., 2019), and other computer vision tasks such as object detection (He et al., 2016; Tan et al., 2019). Although higher accuracy is critical for many applications, we have already hit the hardware memory limit, and thus further accuracy gain needs better efficiency.

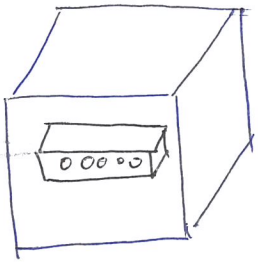
ConvNet Efficiency: Deep ConvNets are often over-parameterized. Model compression (Han et al., 2016; He et al., 2018; Yang et al., 2018) is a common way to reduce model size by trading accuracy for efficiency. As mobile phones become ubiquitous, it is also common to hand-craft efficient mobile-size ConvNets, such as SqueezeNets (Iandola et al., 2016; Gholami et al., 2018), MobileNets (Howard et al., 2017; Sandler et al., 2018), and ShuffleNets

* Model compression (모델 압축)
사전 학습된 모델에 SVD (특이값 분해) 적용.
사전 학습된 모델을 조금씩 잘라내며 특정값 이하로 내려가는 파라미터들 전부 0으로 바꿔 sparse matrix 생성하는 network pruning 방식 있음.

* receptive field (수용공간)

: 어떤 출력 레이어의 뉴런 하나에 영향을 미치는 입력 뉴런들의 공간의 크기 ⇒ 필터의 크기





입력이 32x32x3 인 경우 필터 5x5x3 일때
receptive field는 5x5x3. (=필터의 크기)

* fine grained

: Vision algorithm에서 coarse-to-fine-strategy 존재.
→ 간략화된 이미지 (lowered-resolution) 에서 세밀화된 이미지 (the finest resolution) 으로 변화시키며 KLT tracker 등 optimization 기반 알고리즘의 convergence basin 넓혀주는 역할.

- Coarse-grained programing : 큰 덩어리로 나누는 것
- fine-grained programing : 세밀 한 묘로 프로그래밍 나눠서 고려

* GPipe

: 파이프라인 병렬화 & 체크 포인팅 핵심.
tensorflow로 구현된 GPipe는 1개의 GPU로 학습불가능한 거대한 모델을 가능한 효율적으로 훈련 위한 모델 학습 방법
GPipe 도입하면 메모리 사용량이 큰 모델을 여러 GPU에서 병렬처리 해 GPU를 효율적으로 사용하면서 빠른 학습 가능.

① 파이프라인 병렬화.
: 여러 묶음으로 나눈 모델 각 GPU에 올리는 방식 (모델 병렬화 와 같음). But 모델 훈련이 여러 GPU에서 일정시간 동안 동시에 연산될 수 있으므로 미니 배치를 여러개로 쪼갠 마이크로 배치를 학습에 활용.
→ 중첩되어 처리되는 작업량 ↑↑. 같은 양의 데이터 더 짧은 시간 내 처리.

ex) 각 GPU가 100장 미니 배치 처리 10초. → 2nd GPU는 10초 뒤 작업 → 여러 GPU 최대 동시 동원
But, 10장 마이크로 배치 10개 훈련 → 2nd GPU는 1초 뒤 작업

마이크로 배치 크기 작아질수록 다음 GPU가 더 빨리 일 시작 ⇒ 유휴시간 줄어듦. But 마이크로 배치 크기 무한정 줄일수 X.

② 체크 포인팅.
: 은닉층 결과값 모두 버려 모델 차지하는 메모리 크기 ↓, 모델 나눈 묶음에 체크 포인트 적용하고 분할된 모델 서로 연결하는 은닉층만 메모리에 들림.
순전파 단계에서 분할된 모델 내부의 은닉층결과 값을 기억하지 않고 버림. 이 값은 역전파 단계에서 입력층 결과값과 계산 조건을 가지고 재계산 가능.
⇒ 모든 은닉층 결과값을 메모리에 올려둘 필요 없기 때문에 모델의 훈련에 사용하는 메모리 공간 줄일수 O.
역전파 과정에서 순전파 계산을 한번 더 진행하기에 모델 학습 속도 약 25% 느려지지만, 마이크로 배치 수에 반비례 하기 때문에 메모리 사용량 줄일수 있어 속도 희생가치 有

*Overhead
- 어떤 처리를 하기 위해 들어가는 간접적인 처리 시간, 메모리 ex) A 처리 실행 10초
안전성고려, 부가적인 B 처리 추가해 15초 걸렸으면 overhead는 5초.

*expressive power
- 컴퓨터 과학에서 언어의 표현력은 해당 언어로 표현되고 전달될 수 있는 아이디어의 폭. 표현력이 높을수록 표현하는데 사용할 수 있는 아이디어의 다양성과 양이 많아짐.

Neural architecture 검색

(Zhang et al., 2018; Ma et al., 2018). Recently, neural architecture search becomes increasingly popular in designing efficient mobile-size ConvNets (Tan et al., 2019; Cai et al., 2019), and achieves even better efficiency than hand-crafted mobile ConvNets by extensively tuning the network width, depth, convolution kernel types and sizes. However, it is unclear how to apply these techniques for larger models that have much larger design space and much more expensive tuning cost. In this paper, we aim to study model efficiency for super large ConvNets that surpass state-of-the-art accuracy. To achieve this goal, we resort to model scaling.

Model Scaling: There are many ways to scale a ConvNet for different resource constraints: ResNet (He et al., 2016) can be scaled down (e.g., ResNet-18) or up (e.g., ResNet-200) by adjusting network depth (#layers), while WideResNet (Zagoruyko & Komodakis, 2016) and MobileNets (Howard et al., 2017) can be scaled by network width (#channels). It is also well-recognized that bigger input image size will help accuracy with the overhead of more FLOPS. Although prior studies (Raghu et al., 2017; Lin & Jegelka, 2018; Sharir & Shashua, 2018; Lu et al., 2018) have shown that network depth and width are both important for ConvNets' expressive power, it still remains an open question of how to effectively scale a ConvNet to achieve better efficiency and accuracy. Our work systematically and empirically studies ConvNet scaling for all three dimensions of network width, depth, and resolutions.

3. Compound Model Scaling

In this section, we will formulate the scaling problem, study different approaches, and propose our new scaling method.

3.1. Problem Formulation

A ConvNet Layer i can be defined as a function: $Y_i = \mathcal{F}_i(X_i)$, where $\mathcal{F}_i$ is the operator, Y_i is output tensor, X_i is input tensor, with tensor shape $\langle H_i, W_i, C_i \rangle$[1], where H_i and W_i are spatial dimension and C_i is the channel dimension. A ConvNet $\mathcal{N}$ can be represented by a list of composed layers: $\mathcal{N} = \mathcal{F}_k \odot ... \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1...k} \mathcal{F}_j(X_1)$. In practice, ConvNet layers are often partitioned into multiple stages and all layers in each stage share the same architecture: for example, ResNet (He et al., 2016) has five stages, and all layers in each stage has the same convolutional type except the first layer performs down-sampling. Therefore, we can define a ConvNet as:

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

where $\mathcal{F}_i^{L_i}$ denotes layer $\mathcal{F}_i$ is repeated L_i times in stage i, $\langle H_i, W_i, C_i \rangle$ denotes the shape of input tensor X of layer i. Figure 2(a) illustrate a representative ConvNet, where the spatial dimension is gradually shrunk but the channel dimension is expanded over layers, for example, from initial input shape $\langle 224, 224, 3 \rangle$ to final output shape $\langle 7, 7, 512 \rangle$.

Unlike regular ConvNet designs that mostly focus on finding the best layer architecture $\mathcal{F}_i$, model scaling tries to expand the network length (L_i), width (C_i), and/or resolution (H_i, W_i) without changing $\mathcal{F}_i$ predefined in the baseline network. By fixing $\mathcal{F}_i$, model scaling simplifies the design problem for new resource constraints, but it still remains a large design space to explore different L_i, C_i, H_i, W_i for each layer. In order to further reduce the design space, we restrict that all layers must be scaled uniformly with constant ratio. Our target is to maximize the model accuracy for any given resource constraints, which can be formulated as an optimization problem:

$$\max_{d,w,r} \quad Accuracy(\mathcal{N}(d,w,r))$$
$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1...s} \hat{\mathcal{F}}_i^{d \cdot \hat{L}_i}(X_{\langle r \cdot \hat{H}_i, r \cdot \hat{W}_i, w \cdot \hat{C}_i \rangle})$$
$$\text{Memory}(\mathcal{N}) \le \text{target\_memory}$$
$$\text{FLOPS}(\mathcal{N}) \le \text{target\_flops} \quad (2)$$

where w, d, r are coefficients for scaling network width, depth, and resolution; $\hat{\mathcal{F}}_i, \hat{L}_i, \hat{H}_i, \hat{W}_i, \hat{C}_i$ are predefined parameters in baseline network (see Table 1 as an example).

3.2. Scaling Dimensions

The main difficulty of problem 2 is that the optimal d, w, r depend on each other and the values change under different resource constraints. Due to this difficulty, conventional methods mostly scale ConvNets in one of these dimensions:

Depth (d): Scaling network depth is the most common way used by many ConvNets (He et al., 2016; Huang et al., 2017; Szegedy et al., 2015; 2016). The intuition is that deeper ConvNet can capture richer and more complex features, and generalize well on new tasks. However, deeper networks are also more difficult to train due to the vanishing gradient problem (Zagoruyko & Komodakis, 2016). Although several techniques, such as skip connections (He et al., 2016) and batch normalization (Ioffe & Szegedy, 2015), alleviate the training problem, the accuracy gain of very deep network diminishes: for example, ResNet-1000 has similar accuracy as ResNet-101 even though it has much more layers. Figure 3 (middle) shows our empirical study on scaling a baseline model with different depth coefficient d, further suggesting the diminishing accuracy return for very deep ConvNets.

Width (w): Scaling network width is commonly used for small size models (Howard et al., 2017; Sandler et al., 2018;

[1] For the sake of simplicity, we omit batch dimension.

*high level feature
: 고차원의 특징

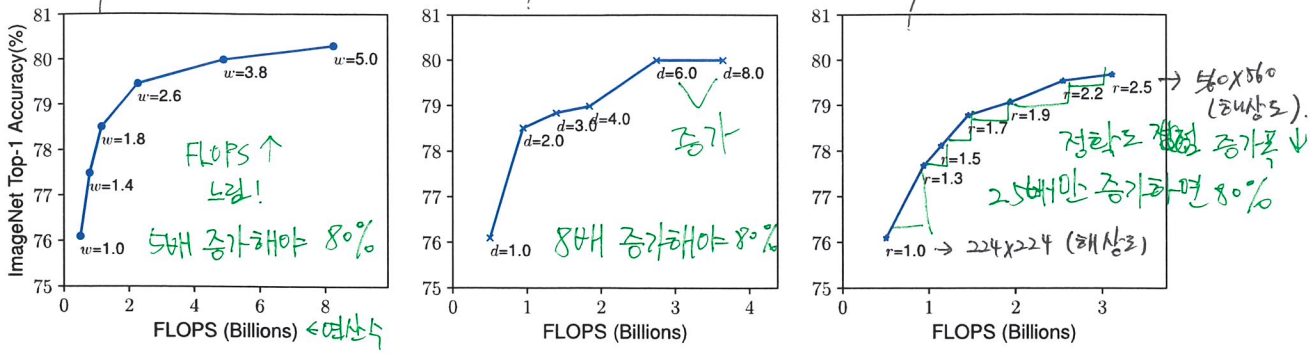


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single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

Width
(폭)
세밀한 feature 잘 포착
훈련 쉬움
(단점)
너무 넓고 얕은 Network
높은 수준의 피처 잡기 어려

Tan et al., 2019)[2]. As discussed in (Zagoruyko & Komodakis, 2016), wider networks tend to be able to capture more fine-grained features and are easier to train. However, extremely wide but shallow networks tend to have difficulties in capturing higher level features. Our empirical results in Figure 3 (left) show that the accuracy quickly saturates when networks become much wider with larger w. 가독한
→ 폭 넓어지면 정확도 ↑

Resolution
높은 img 해상도 ↑
fine grained pattern 잘 잡아냄
기존 ConvNet 224x224 해상도 사용
현재 299x299 해상도 331x331 사용
GPipe 480x480 해상도 사용 정확도 ↑
→ 600x600 같은 더 높은 해상도는 Object detection ConvNet 사용

Resolution (r): With higher resolution input images, ConvNets can potentially capture more fine-grained patterns. Starting from 224x224 in early ConvNets, modern ConvNets tend to use 299x299 (Szegedy et al., 2016) or 331x331 (Zoph et al., 2018) for better accuracy. Recently, GPipe (Huang et al., 2018) achieves state-of-the-art ImageNet accuracy with 480x480 resolution. Higher resolutions, such as 600x600, are also widely used in object detection ConvNets (He et al., 2017; Lin et al., 2017). Figure 3 (right) shows the results of scaling network resolutions, where indeed higher resolutions improve accuracy, but the accuracy gain diminishes for very high resolutions ($r = 1.0$ denotes resolution 224x224 and $r = 2.5$ denotes resolution 560x560).
→ 해상도 높아지면 정확도 높아지지만, 매우 큰 해상도에서는 정확도 감소.

The above analyses lead us to the first observation:

Observation 1 – Scaling up any dimension of network width, depth, or resolution improves accuracy, but the accuracy gain diminishes for bigger models.
→ width, depth, resolution 크게 하면 정확도 상승
but 더 큰 모델에서는 정확도 ↓

3.3. Compound Scaling

We empirically observe that different scaling dimensions are not independent. Intuitively, for higher resolution images, we should increase network depth, such that the larger receptive fields can help capture similar features that include more pixels in bigger images. Correspondingly, we should also increase network width when resolution is higher, in

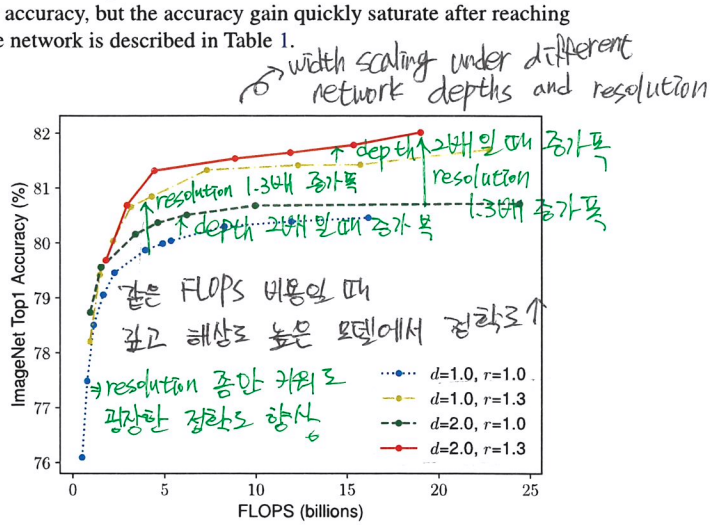


Figure 4. **Scaling Network Width for Different Baseline Networks.** Each dot in a line denotes a model with different width coefficient (w). All baseline networks are from Table 1. The first baseline network (d=1.0, r=1.0) has 18 convolutional layers with resolution 224x224, while the last baseline (d=2.0, r=1.3) has 36 layers with resolution 299x299.

order to capture more fine-grained patterns with more pixels in high resolution images. These intuitions suggest that we need to coordinate and balance different scaling dimensions rather than conventional single-dimension scaling. → 3 dim 조화 중요

To validate our intuitions, we compare width scaling under different network depths and resolutions, as shown in Figure 4. If we only scale network width (w) without changing depth (d=1.0) and resolution (r=1.0), the accuracy saturates 가독한 quickly. With deeper (d=2.0) and higher resolution (r=2.0), width scaling achieves much better accuracy under the same FLOPS cost. These results lead us to the second observation:
스케일링 → 더 깊고 해상도 큰 모델이 정확도도 높다

Observation 2 – In order to pursue better accuracy and efficiency, it is critical to balance all dimensions of network width, depth, and resolution during ConvNet scaling.
→ 정확도, 효율성 더 낮게 하려면 3 dim 조화 중요.

[2] In some literature, scaling number of channels is called "depth multiplier", which means the same as our width coefficient w.

3개 dim 조정 → 중요!
해상도 ↑
깊이 ↑

해상도 ↑ → 깊이 ↑
큰 이미지에서 더 많은 픽셀 포함하는 더 큰 수용영역(필터)가 유용한 특징 포착 가능
해상도 ↑ → 폭 ↑
더 많은 픽셀에서 세밀한 패턴 발견 위해

In fact, a few prior work (Zoph et al., 2018; Real et al., 2019) have already tried to arbitrarily balance network width and depth, but they all require tedious manual tuning.

In this paper, we propose a new **compound scaling method**, which use a compound coefficient ϕ to uniformly scales network width, depth, and resolution in a principled way:

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

where α, β, γ are constants that can be determined by a small grid search. Intuitively, ϕ is a user-specified coefficient that controls how many more resources are available for model scaling, while α, β, γ specify how to assign these extra resources to network width, depth, and resolution respectively. Notably, the FLOPS of a regular convolution op is proportional to d, w^2, r^2, i.e., doubling network depth will double FLOPS, but doubling network width or resolution will increase FLOPS by four times. Since convolution ops usually dominate the computation cost in ConvNets, scaling a ConvNet with equation 3 will approximately increase total FLOPS by $(\alpha \cdot \beta^2 \cdot \gamma^2)^\phi$. In this paper, we constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ such that for any new ϕ, the total FLOPS will approximately[3] increase by 2^ϕ.

4. EfficientNet Architecture

Since model scaling does not change layer operators $\hat{\mathcal{F}}_i$ in baseline network, having a good baseline network is also critical. We will evaluate our scaling method using existing ConvNets, but in order to better demonstrate the effectiveness of our scaling method, we have also developed a new mobile-size baseline, called EfficientNet.

Inspired by (Tan et al., 2019), we develop our baseline network by leveraging a multi-objective neural architecture search that optimizes both accuracy and FLOPS. Specifically, we use the same search space as (Tan et al., 2019), and use $ACC(m) \times [FLOPS(m)/T]^w$ as the optimization goal, where $ACC(m)$ and $FLOPS(m)$ denote the accuracy and FLOPS of model m, T is the target FLOPS and w=-0.07 is a hyperparameter for controlling the trade-off between accuracy and FLOPS. Unlike (Tan et al., 2019; Cai et al., 2019), here we optimize FLOPS rather than latency since we are not targeting any specific hardware device. Our search produces an efficient network, which we name EfficientNet-B0. Since we use the same search space as (Tan et al., 2019), the architecture is similar to MnasNet, except our EfficientNet-B0 is slightly bigger due to the larger FLOPS target (our FLOPS target is 400M). Table 1 shows the architecture of EfficientNet-B0. Its main building block is mobile inverted bottleneck MBConv (Sandler et al., 2018; Tan et al., 2019), to which we also add squeeze-and-excitation optimization (Hu et al., 2018).

[3] FLOPS may differ from theoretical value due to rounding.

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

Starting from the baseline EfficientNet-B0, we apply our compound scaling method to scale it up with two steps:

- STEP 1: we first fix $\phi = 1$, assuming twice more resources available, and do a small grid search of α, β, γ based on Equation 2 and 3. In particular, we find the best values for EfficientNet-B0 are $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$, under constraint of $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$.
- STEP 2: we then fix α, β, γ as constants and scale up baseline network with different ϕ using Equation 3, to obtain EfficientNet-B1 to B7 (Details in Table 2).

Notably, 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 on larger models. Our method solves this issue by only doing search once on the small baseline network (step 1), and then use the same scaling coefficients for all other models (step 2).

5. Experiments

In this section, we will first evaluate our scaling method on existing ConvNets and the new proposed EfficientNets.

5.1. Scaling Up MobileNets and ResNets

As a proof of concept, we first apply our scaling method to the widely-used MobileNets (Howard et al., 2017; Sandler et al., 2018) and ResNet (He et al., 2016). Table 3 shows the ImageNet results of scaling them in different ways. Compared to other single-dimension scaling methods, our compound scaling method improves the accuracy on all these models, suggesting the effectiveness of our proposed scaling method for general existing ConvNets.

★ Mobile Inverted Bottle Neck.

⇒ MobileNet V2. //

(DSC)

이전의 MobileNet : 일반적 conv 무겁 → depthwise seperable conv 사용

Mobile Net V2 기법

① depthwise seperable conv. ← Mobile Net 참고하세요!

② Inverted Residuals

↘

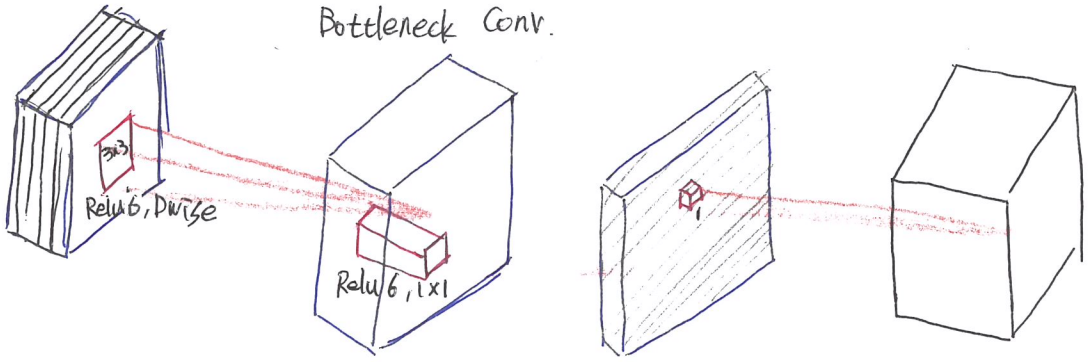
Linear Bottlenecks

- 방식 : Residual Learning 방식을 통해 네트워크를 깊게 구성할 경우, 연산량 ↑
  1 by 1 conv 활용해 채널 수 감소. cf. Residual Learning ?: 학습을 위한 weight layer 거치고 초기 input 값을 더해주는 형태
- 일반적 : 3x3 conv 실시
- Bottle Neck 방식 : 1x1 conv로 채널수 ↓ → 3x3 conv 실시 → 다시 1x1 conv로 채널수 복구
- 왜 Inverted Residuals 인가?

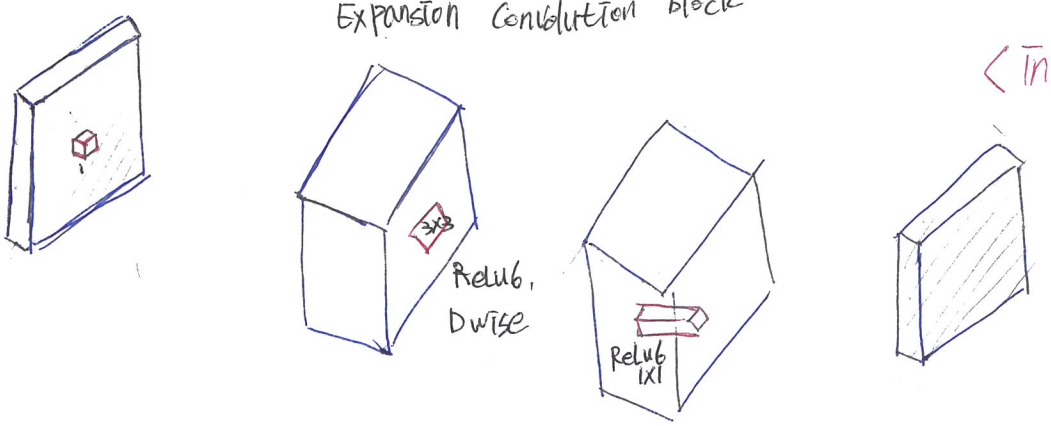
  : 기존의 bottleneck은 채널감소 → 학습(DSC) → 채널복구.

  Inverted Residual은 채널 증가 → 학습(DSC) → 채널감소

  <Inverted 됨> ⇒ input data와 output data의 크기 작아 메모리 효율적.



<기존의 bottle neck>



<Inverted Residual>

★ MnasNet

: Neural Architecture Search for Mobile.

Nas → Neural Architecture search 구글에서 딥러닝 모델 아키텍처 타는 또다른 딥러닝

MnasNet

: 모바일에 최적화된 네트워크 찾아주는 모델

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	**77.3%**	**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).

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%**

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**

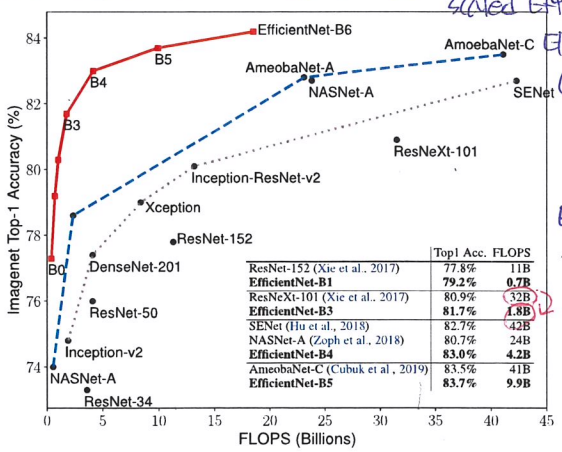


Figure 5. **FLOPS vs. ImageNet Accuracy** – Similar to Figure 1 except it compares FLOPS rather than model size.

5.2. ImageNet Results for EfficientNet

We train our EfficientNet models on ImageNet using similar settings as (Tan et al., 2019): RMSProp optimizer with decay 0.9 and momentum 0.9; batch norm momentum 0.99;

*RMSProp

: AdaGrad는 최소값에 도달하기 전에 학습률을 0에 수렴하게 만들 수도 있다.
AdaGrad가 간단한 convex function에서는 잘 동작 but 복잡한 다차원 곡면 함수를 잘 탐색하도록 설계되지 X. 기울기의 단순한 누적 방법으로 충분X.
RMSProp은 기울기 단순 누적 X. 지수 가중 이동 평균 (Exponentially weighted moving average) 사용해 최신 기울기들이 더 크게 반영되도록 함.

$$h_i \leftarrow \rho h_{i-1} + (1-\rho) \frac{\partial L_i}{\partial W} \odot \frac{\partial L_i}{\partial W}$$

이전 AdaGrad의 h에 새로운 hyper parameter ρ 추가
⇒ h가 무한히 커지지 않으면서 ρ가 작을 수록 가장 최신의 기울기를 더 크게 반영.

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).

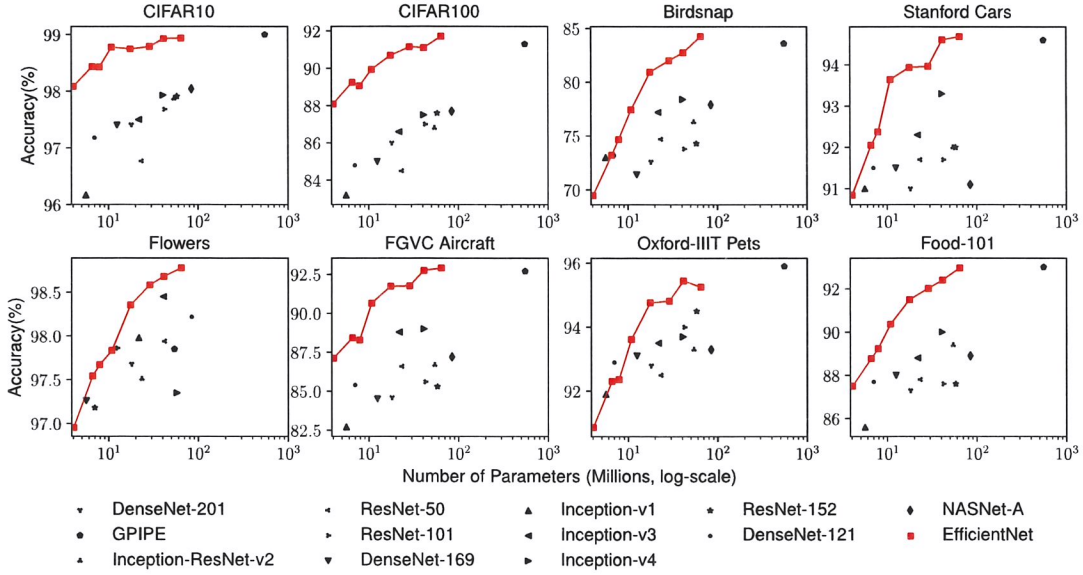


Figure 6. **Model Parameters vs. Transfer Learning Accuracy –** All models are pretrained on ImageNet and finetuned on new datasets.

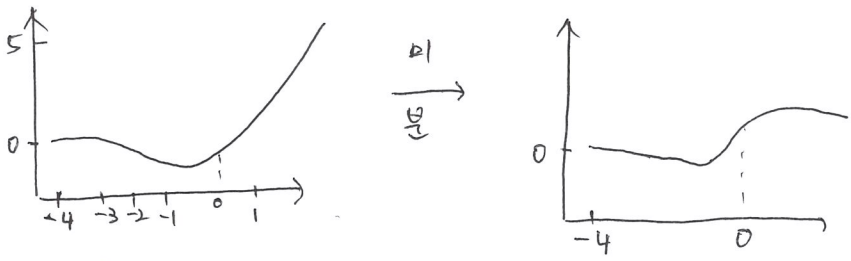
weight decay 1e-5; initial learning rate 0.256 that decays by 0.97 every 2.4 epochs. We also use swish activation (Ramachandran et al., 2018; Elfwing et al., 2018), fixed AutoAugment policy (Cubuk et al., 2019), and stochastic depth (Huang et al., 2016) with survival probability 0.8. As commonly known that bigger models need more regularization, we linearly increase dropout (Srivastava et al., 2014) ratio from 0.2 for EfficientNet-B0 to 0.5 for EfficientNet-B7.

Table 2 shows the performance of all EfficientNet models that are scaled from the same baseline EfficientNet-B0. Our EfficientNet models generally use an order of magnitude fewer parameters and FLOPS than other ConvNets with similar accuracy. In particular, our EfficientNet-B7 achieves 84.4% top1 / 97.1% top-5 accuracy with 66M parameters and 37B FLOPS, being more accurate but **8.4x smaller** than the previous best GPipe (Huang et al., 2018).

Figure 1 and Figure 5 illustrates the parameters-accuracy and FLOPS-accuracy curve for representative ConvNets, where our scaled EfficientNet models achieve better accuracy with much fewer parameters and FLOPS than other ConvNets. Notably, our EfficientNet models are not only small, but also computational cheaper. For example, our EfficientNet-B3 achieves higher accuracy than ResNeXt-101 (Xie et al., 2017) using **18x fewer FLOPS**.

To validate the computational cost, we have also measured the inference latency for a few representative CovNets on a real CPU as shown in Table 4, where we report average latency over 20 runs. Our EfficientNet-B1 runs **5.7x faster** than the widely used ResNet-152 (He et al., 2016), while EfficientNet-B7 runs about **6.1x faster** than GPipe (Huang et al., 2018), suggesting our EfficientNets are indeed fast on real hardware.

* Swish (Sigmoid Linear Unint, SiLU)



$f(x) = \frac{x}{1+e^{-x}} = x \cdot \sigma(x)$

$f'(x) = f(x) + \sigma(x)(1-f(x))$

- 2차원에서 학인 했을 때, linear 또는 ReLU 보다 훨씬 부드러운 형태.
- ReLU 및 다른 활성화함수를 대체하기 위해 만든 함수.
  논문에서 CIFAR 에서에서 실험 결과, ReLU 및 다른 활성화 함수보다 좋은 성능

* Auto Augment

-Augmentation

: 딥러닝 모델을 충분히 훈련하는데 필요한 데이터를 확보하는 기법 중 하나.
적은 양의 훈련 데이터에 인위적 변화를 가해 새로운 훈련 데이터를 대량 확보하는 방법론.

ex) 이미지 상하좌우로 뒤집기 (flipping), 자르기 (cropping) → 새로운 데이터 확보

: 현실 세계 에서도 실제로 존재할 법한 데이터 생성하여 좀더 일반화된 모델 얻는것 목표

(ex) 글자이미지를 좌우/상하좌우 뒤집기 : 글자의미 태곡
심장 촬영 영상 뒤집기 기법 부적절

문제점

: 데이터에 최적화된 Augmentation 방법론 탐색 어렵. 현실과 동떨어지거나 기존 특징 태곡할수도 있는 기법은 오히려 학습 난이도와 성능에 안 좋은 영향.
데이터를 취급 하는 방식이 다른 점도 Augmentation 기법에 영향끼침.

(ex) 이미지 촬영 때 대상물체 크기 달라짐 : 이미지 여러크기로 자동 조정하는 기법이 도움됨
의료영상 - 장기의 크기는 거의 바뀌지 않아도 데이터 획득 방식이나 환자의 상태에 따라 장기에 변이有 경우.
: 노이즈 적절하게 생성하는 기법 필요.

-Auto Augment

: ML로 데이터 수 늘리는 자동화 방법 찾는 연구. 강화학습을 통해 이미지 데이터셋에 가장 적합한 Augmentation 정책 (Policy)를 자동으로 찾아주는 알고리즘. 탐색공간 (search space)는 데이터에 적합한 Augmentation 기법을 다양하게 찾으려는 목적에서 하나의 정책을 5개의 하위정책으로 구성.
하위 정책은 그개의 이미지 오퍼레이션 (operation)으로 구성. 각 오퍼레이션은 augmentation 기법 ($T=\{$회전, 대비, 밝기, ⋯, 첨예도$\}$, $n(T)=16$)을 어떤 확률 ($P=\{0, 0.1, \cdots 1\}$, $n(P)=11$), 어느 수준의 강도 ($M=\{0, 1, \cdots, 9\}$, $n(M)=10$)로 실행할 지 결정. 한 하위 정책이 생성하는 augmentation 기법은 총 $(11\times11\times10)^2$개
결과적으로 Auto Augmentation에서 정의된 이미지 Augmentation에는 총 2.9×10^{32}개 후보군 有
일일이 학인 불가능, 탐색 깊이, 너비 줄이는 알고리즘 필요! Auto Augmentation은 임의로 선택한 Augmentation 기법의 훈련 데이터를 증강, 네트워크 훈련 초기는 성능 Good, bad 모두 골고루 탐색 이후 모델 성능 높이는 방향으로 보상 획득해가며 Auto Augmentation은 점점 더 높은 성능 을 내는 기법 찾아감 ⇒ Auto Augmentation은 동일한 모델구조 최적화 기술 활용할 때 학습에 적합한 데이터 증강 기법 적용하는 것만으로도 성능 개선 의의.
Auto Augmentation은 데이터셋 규모를 축소했을 때 더 큰 성능 개선 효과. 레이블링 데이터를 사용한 기존의 준 감독 학습 (semi-supervised learning)과 달리 또려 데이터 규모 축소 상태에서 Auto Augment가 찾아낸 방법론은 성능 훨씬 도드라짐. But, 계산 복잡도 ↑, 일반적인 연구환경에 적용 어려움.

*transfer learning
: 기존의 만들어진 모델을 사용해 새로운 모델 만들 때 학습 빠르게 하며 예측을 더 높이는 방법.

→ transfer learning 왜 사용?
① 실질적으로 Conv. Net 처음부터 학습시킬 일 많지 X. 대부분의 문제는 이미 학습된 모델 사용해 문제 해결 가능.

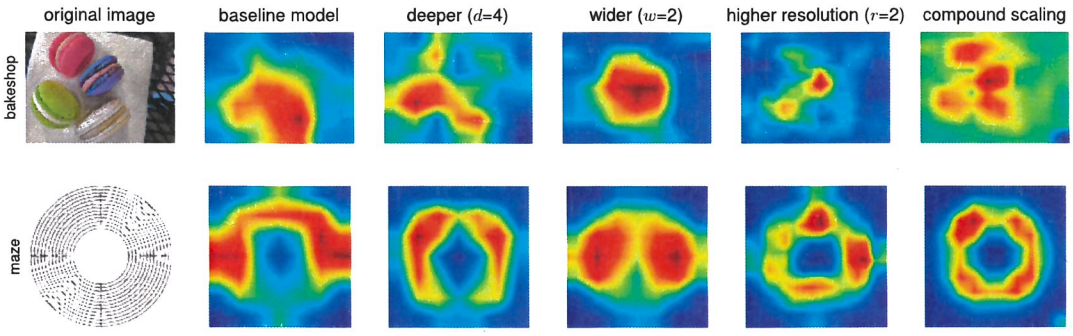


Figure 7. **Class Activation Map (CAM) (Zhou et al., 2016) for Models with different scaling methods-** Our compound scaling method allows the scaled model (last column) to focus on more relevant regions with more object details. Model details are in Table 7.

ImageNet 데이터 알고 다른 데이터

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

5.3. Transfer Learning Results for EfficientNet

다른 데이터에 대해 EfficientNet 사용 데이터

We have also evaluated our EfficientNet on a list of commonly used transfer learning datasets, as shown in Table 6. We borrow the same training settings from (Kornblith et al., 2019) and (Huang et al., 2018), which take ImageNet pretrained checkpoints and finetune on new datasets.

→ table 6 설명 : ImageNet 에서의 모델들과 비슷한 세팅에서 EfficientNet 평가.

공식적으로 이용가능 모델과 EfficientNet 비교 → NASNet-A, Inception-v4, EfficientNet 최고 정확도, 파라미터 많이감소

Table 5 shows the transfer learning performance: (1) Compared to public available models, such as NASNet-A (Zoph et al., 2018) and Inception-v4 (Szegedy et al., 2017), our EfficientNet models achieve better accuracy with 4.7x average (up to 21x) parameter reduction. (2) Compared to state-of-the-art models, including DAT (Ngiam et al., 2018) that dynamically synthesizes training data and GPipe (Huang et al., 2018) that is trained with specialized pipeline parallelism, our EfficientNet models still surpass their accuracy in 5 out of 8 datasets, but using 9.6x fewer parameters

→ table 5 설명 : 다른모델과 비교

Figure 6 compares the accuracy-parameters curve for a variety of models. In general, our EfficientNets consistently achieve better accuracy with an order of magnitude fewer parameters than existing models, including ResNet (He et al., 2016), DenseNet (Huang et al., 2017), Inception (Szegedy et al., 2017), and NASNet (Zoph et al., 2018).

→ 파라미터수 / 정확도 다른 모델과 비교

DAT train data 극적증가

GPipe 특화된 파이프라인 병렬화

3개 중 5개에서 리커가 정확도 더 적은 파라미터

6. Discussion

To disentangle the contribution of our proposed scaling method from the EfficientNet architecture, Figure 8 compares the ImageNet performance of different scaling methods for the same EfficientNet-B0 baseline network. In general, all scaling methods improve accuracy with the cost of more FLOPS, but our compound scaling method can further improve accuracy, by up to 2.5%, than other single-dimension scaling methods, suggesting the importance of our proposed compound scaling.

→ 3개 다 소개되길 바라는게 좋다!

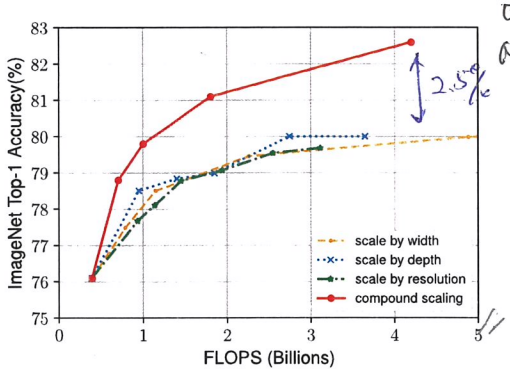


Figure 8. **Scaling Up EfficientNet-B0 with Different Methods.**

더 적은 FLOPS 사용 accuracy 증가.

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%**

In order to further understand why our compound scaling method is better than others, Figure 7 compares the class activation map (Zhou et al., 2016) for a few representative models with different scaling methods. All these models are scaled from the same baseline, and their statistics are shown in Table 7. Images are randomly picked from ImageNet validation set. As shown in the figure, the model with compound scaling tends to focus on more relevant regions with more object details, while other models are either lack of object details or unable to capture all objects in the images.

compound scaling은 더 관련 있는 지역에 집중! 더 디테일함

다른 모델들은 디테일 부족, 모든 물체 포착X

*pretrained
: 이미 학습이 끝난 모델 / 모델구성요소 (ex. 임베딩).
선행 학습된 임베딩을 신경망에 입력하는 경우 有
모델에서 선행학습된 임베딩에 의존하지 않고 임베딩 자체를 학습하는 경우 有

*checkpoint
: 특정시점에 모델 변수의 상태 포착한 데이터. 체크포인트를 통해 모델 가중치 내보내고 여러 세션 (session)을 넘나들며 학습 수행할 수 있음. 또한 체크포인트로부터 과거의 오류 (ex. 작업 선점)를 이어받아 학습할 수 있음. 단, 그래프 자체는 체크포인트에 포함X.

* transfer learning.

: 기존의 만들어진 모델을 이용하여 새로운 모델 만들 시 빠르게 학습하며, 예측 더 높임.

왜 사용?

① 실질적으로 처음부터 CNN 학습하는 일 많지 X. 대부분의 문제는 이미 학습된 모델을 사용해 문제해결 가능.

② 복잡한 모델일수록 학습 시키기 어렵. 어떤 모델은 수주 이상, 비싼 GPU 여러대 사용.

③ layer의 개수, activation, hyperparameter 등 고려사항 多多.
실질적으로 처음부터 학습시키려면 많은 시도 필요.

⇒ 이미 잘 훈련된 모델이 있고, 해당 model과 유사한 문제 해결 시 transfer learning 사용

↓ 실질적 사용.

- 새로 훈련할 데이터 적지만, original 데이터와 유사한 경우.
: 데이터 양이 적어 fine-tune (전체 모델에 대해 backpropagation 진행) 은 over-fitting 우려이 있어 하지 X 새로 학습할 데이터는 original 데이터와 유사하기 때문에 이 경우 최종 linear classifier 레이어만 학습.

- 새로 훈련할 데이터 매우 많으며 original 데이터와 유사한 경우.
: 새로 학습할 데이터의 양이 많다는 것은 over-fitting의 위험이 낮다는 뜻이므로, 전체 레이어에 대해 fine-tune 함.

- 새로 훈련할 데이터가 적으며 original 데이터와 다른경우.
: 데이터의 양이 적기 때문에 최종 단계의 linear classifier 레이어 학습 좋을 것.
반면, 데이터가 서로 다르기 때문에 거의 마지막 부분 (the top of the network) 만 학습 좋지 X.
서로 상충되는 내용이지만, 이경우 네트워크 초기부분 어딘가 activation 이후에 특정레이어 학습시키는게 좋음.

- 새로 훈련할 데이터가 많지만 original 데이터와 다른 경우.
: 데이터 많기 때문에 아예 새로운 ConvNet 만들 수 있지만, 실질적으로 transfer learning이 더 효율 좋음. 전체 네트워크에 대해 fine-tune 해도 됨.

* Class Activation Map

: 기존 CNN의 모델은 classification 를 이미지 내에서 어떤 특징 (feature)을 보고 판단해는지 알수 X
하지만 출력의 결과를 시각화 하여 실제 모델이 이미지의 어떤 부분을 보고 분류하는지 알수 있게 하는 것이 Class Activation Map (CAM)

compound scaling 방법으로
모바일 크기의 EfficientNet 모델이 효과적으로 스케일 ↑ 가능.
→ 정확도 ↑, 파라미터수 ↓ : FLOPS ↓,
(ImageNet과 5개의 흔히 사용되는 transfer learning dataset에서

3개 dim 조라줄도 하나라도 늘릴시 더 좋은 정확도의 효율성 X
↓
Compound scaling : 모델 효율을 유지하면서 추가적으로 어떤 추가자원 강제할지에 대해 ConvNet 베이스라인을 쉽게 스케일 업 가능

7. Conclusion

In this paper, we systematically study ConvNet scaling and identify that carefully balancing network width, depth, and resolution is an important but missing piece, preventing us from better accuracy and efficiency. To address this issue, we propose a simple and highly effective compound scaling method, which enables us to easily scale up a baseline ConvNet to any target resource constraints in a more principled way, while maintaining model efficiency. Powered by this compound scaling method, we demonstrate that a mobile-size EfficientNet model can be scaled up very effectively, surpassing state-of-the-art accuracy with an order of magnitude fewer parameters and FLOPS, on both ImageNet and five commonly used transfer learning datasets.

Acknowledgements

We thank Ruoming Pang, Vijay Vasudevan, Alok Aggarwal, Barret Zoph, Hongkun Yu, Xiaodan Song, Samy Bengio, Jeff Dean, and Google Brain team for their help.

References

Berg, T., Liu, J., Woo Lee, S., Alexander, M. L., Jacobs, D. W., and Belhumeur, P. N. Birdsnap: Large-scale fine-grained visual categorization of birds. *CVPR*, pp. 2011–2018, 2014.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101–mining discriminative components with random forests. *ECCV*, pp. 446–461, 2014.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. *CVPR*, pp. 1610–02357, 2017.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *CVPR*, 2019.

Elfwing, S., Uchibe, E., and Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., Zhao, S., and Keutzer, K. Squeezenext: Hardware-aware neural network design. *ECV Workshop at CVPR'18*, 2018.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. *ICCV*, pp. 2980–2988, 2017.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. Amc: Automl for model compression and acceleration on mobile devices. *ECCV*, 2018.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. *CVPR*, 2018.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. *ECCV*, pp. 646–661, 2016.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. *CVPR*, 2017.

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V., and Chen, Z. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1808.07233*, 2018.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, pp. 448–456, 2015.

Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? *CVPR*, 2019.

Krause, J., Deng, J., Stark, M., and Fei-Fei, L. Collecting a large-scale dataset of fine-grained cars. *Second Workshop on Fine-Grained Visual Categorizatio*, 2013.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1097–1105, 2012.

Lin, H. and Jegelka, S. Resnet with one-neuron hidden layers is a universal approximator. *NeurIPS*, pp. 6172–6181, 2018.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. *CVPR*, 2017.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. *ECCV*, 2018.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. The expressive power of neural networks: A view from the width. *NeurIPS*, 2018.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *ECCV*, 2018.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. Exploring the limits of weakly supervised pretraining. *arXiv preprint arXiv:1805.00932*, 2018.

Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Ngiam, J., Peng, D., Vasudevan, V., Kornblith, S., Le, Q. V., and Pang, R. Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv:1811.07056*, 2018.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. *ICVGIP*, pp. 722–729, 2008.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. Cats and dogs. *CVPR*, pp. 3498–3505, 2012.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. On the expressive power of deep neural networks. *ICML*, 2017.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2018.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. *AAAI*, 2019.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. *CVPR*, 2018.

Sharir, O. and Shashua, A. On the expressive power of overlapping architectures of deep learning. *ICLR*, 2018.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. *CVPR*, pp. 1–9, 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. *CVPR*, pp. 2818–2826, 2016.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. *AAAI*, 4:12, 2017.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. MnasNet: Platform-aware neural architecture search for mobile. *CVPR*, 2019.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. *CVPR*, pp. 5987–5995, 2017.

Yang, T.-J., Howard, A., Chen, B., Zhang, X., Go, A., Sze, V., and Adam, H. Netadapt: Platform-aware neural network adaptation for mobile applications. *ECCV*, 2018.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *BMVC*, 2016.

Zhang, X., Li, Z., Loy, C. C., and Lin, D. Polynet: A pursuit of structural diversity in very deep networks. *CVPR*, pp. 3900–3908, 2017.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CVPR*, 2018.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. *CVPR*, pp. 2921–2929, 2016.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *ICLR*, 2017.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. *CVPR*, 2018.