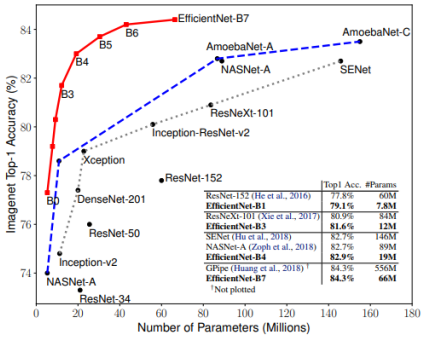
EfficientNet 논문

Introduction

- ResNet 발표 이후
  - 2015년 ILSVRC에서 ResNet이 사람보다 더 높은 성능을 가진다고 발표됨 (top-5 error: 3.57%)
  - 이후 Better Accuracy와 Better Efficiency로 연구 파트가 나뉘게 됨
  - Better Accuracy 모델들과 EfficientNet을 비교 -> Better Accuracy & Better Efficiency



	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

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,

2015년 이미지넷 대회에서 ResNet이 사람보다 높은 분류 성능을 지닌다고 발표가 되었다.
이후 ResNet을 참고해서 Accuracy를 더욱 높이는 연구와 적은 연산량으로 우수한 성능을 보이려는 연구가 활발히 이루어졌다.
아래 그림은 Accuracy를 높이려고 연구한 모델(파란 색 점)들과 이 논문에서 제안하는 EfficientNet을 비교한 것이다. 여기서 EfficientNet은 parameter 수도 적으면서 더 높은 성능을 보인다. 즉 Better Accuracy와 Better Efficiency를 모두 만족시켰다는 의미가 된다.
이것이 가능했던 이유는 ConvNet의 Scaling에 대해 연구했기 때문이다.
연구진은 width, depth, resolution을 모두 balance있게 조정했을 때 성능 향상을 보였다고 말한다.
따라서 이 논문은 제목 그대로 scaling 기법에 대해 다시 생각해보는 논문이다.

Introduction

- Model Scaling
  - 모델의 사이즈를 높이거나 줄이는 방법
  - Depth(레이어 수), Width(필터 수), Resolution(입력 해상도)를 조절함
  - 이 논문은 Depth, Width, Resolution을 균형있게 키우는 compound scaling 기법이 사용됨

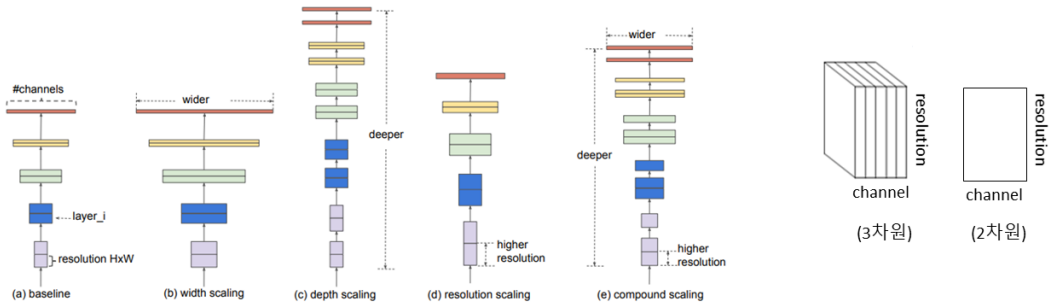


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

모델 스케일링은 모델의 사이즈를 높이거나 줄이는 방법을 의미한다. 이 때 Depth, Width, Resolution을 늘려서 모델을 키우거나 줄여서 모델을 줄이는 것이다. 그림은 3차원 Feature Map을 2차원으로 만들었을 때, 각각 width, depth, resolution에 대하여 scaling한 상태를 보여준다.
보통은 이렇게 1 dimension으로 scaling을 하지만 이 논문에서는 width, depth, resolution을 모두 높이려고 시도했다. 여기서 3 dimension을 균형있게 높이는 방법이 compound scaling이다.

Related Work

- ConvNet Accuracy
  - AlexNet 이후 ImageNet 대회에서 더 정확도가 높은 모델들이 여럿 발표됨
  - 2018년 Gpipe 학습 기법이 발표됨 (state-of-the-art, top-1 error: 84.3%, 557M parameters)
  - 하드웨어 메모리 제한을 고려해서 효율적인 모델 개발도 필요함

- ConvNet Efficiency
  - 깊은 ConvNets은 종종 over-parameterized됨
  - 모델을 압축하는 여러 기법(Pruning, Deepth-wise seperable convolution 등)이 제안됨
  - 큰 모델에 위와 같은 기법을 어떻게 적용할지는 아직 unclear함

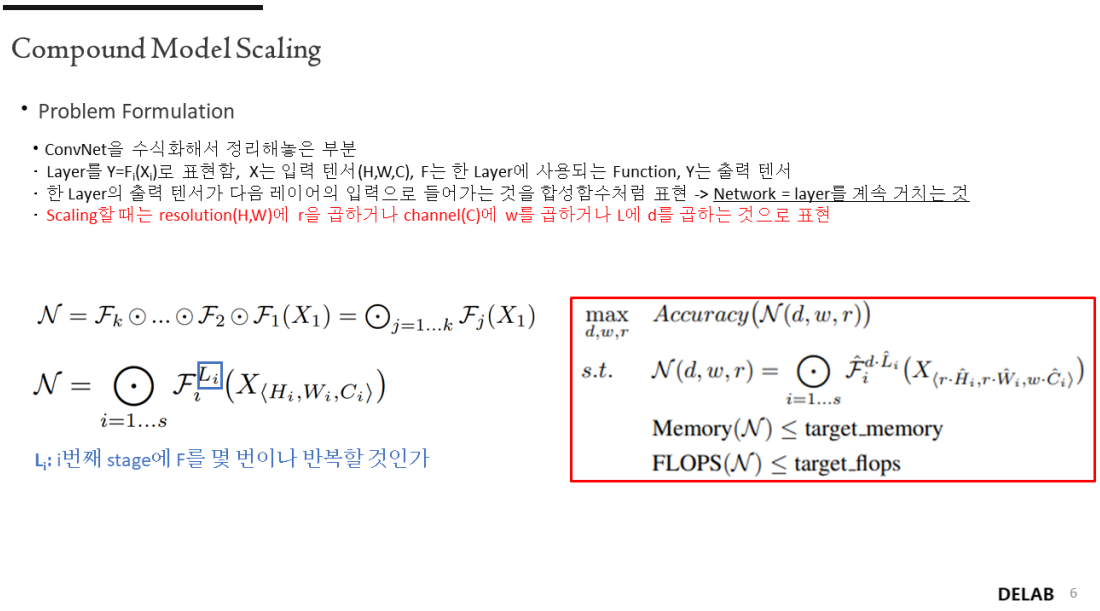
- Model Scaling
  - 기존에도 Depth, Width, Resolution를 키우는 방법을 취함
  - 효과적으로 Scaling하는 방법은 여전히 question으로 남아 있었음

첫 번째 장에서도 말했듯 연관 연구로는 Accuracy를 높이려는 연구들과 Efficiency를 높이려는 연구들이 있었다. 특히 Accuracy에서 2018년 GPipe라는 학습 기법이 발표되었다. 이 기법은 Google Brain에서 발표한 학습 기법으로, 메모리를 많이 차지하는 큰 모델을 효율적으로 학습시키는 데

유용하다. 이렇게 효율적인 학습을 하는 것도 중요하지만 논문에서는 하드웨어 메모리 제한을 고려해서 효율적인 모델 개발도 필요하다고 한다.

Efficiency 연구를 보면 over-parametered된 모델을 대비하려는 모델 압축 기법을 소개한다. 예를 들어 pruning이나 depth-wise convolution 등이 제안되었고 mobileNet, MNas-Net 등이 생겼다. 그러나 이런 기법들은 여전히 큰 네트워크에 적용하는 방법에 대해 unclear하다고 한다.

마지막으로는 Model Scaling에 대한 연구인데, 기존에 depth나 width, resolution을 키우려는 시도는 있었지만 효과적인 scaling방법에 대해서는 잘 모른 체로 진행을 하였다고 한다.



위 식은 convNet을 scaling할 때 직관적으로 보이려고 수식 정리를 한 것이다.

이 때 Layer는 Y=F(x)로 표현하고 X는 입력 텐서, F는 한 layer에 사용되는 Function, Y는 출력 텐서를 의미한다. 식을 살펴 보면 한 Layer의 출력 텐서(Y)가 다음 레이어(F(X))의 입력으로 들어가는 것을 합성함수처럼 표현하였는데 이것은 결국 Network가 Layer를 계속 거치는 것이라는 의미이다.

수식을 정리해서도 나타낼 수 있는데 여기서 L_i는 i번째 stage에 F를 몇 번이나 반복할 것인지를 나타낸다. 이 수식을 이용해서 scaling을 표현할 때는 H,W에 r을 곱하거나 C에 w를 곱하거나 L에 d를 곱하는 것으로 표현한다.

Compound Model Scaling

- Scaling Dimensions
  - Width: 각 레이어의 width를 키우면 정확도가 높아지지만 계산량이 제곱에 비례하여 증가함
  - Depth: 네트워크의 깊이가 증가할수록 더 복잡한 feature를 잡아냄, vanishing gradien는 Batch Norm과 Residual Connection으로 해결
  - Resolution: 해상도를 키울수록 더 세부적인 feature를 잡아내 정확도가 높아지지만 계산량이 제곱에 비례하여 증가함
  - accuracy gain(정확도 증가량)은 셋 다 점점 줄어드는 형태

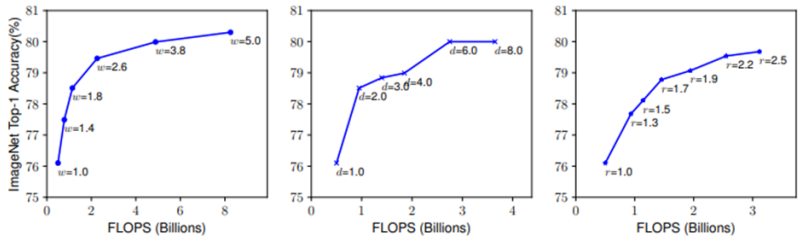


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single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

다음은 scaling dimensions에 대한 설명이다. Width는 채널 수(필터 수)로 width를 키우면 더 많은 feature가 저장돼서 정확도가 높아진다. 하지만 width를 증가하면 계산량은 제곱에 비례하여 증가한다. 예를 들어 width가 2배 증가하면 계산량은 2^2배 증가한다.

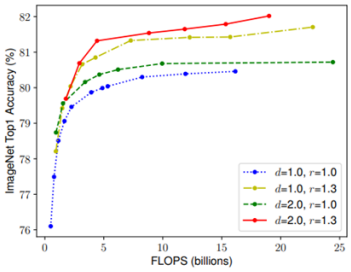
Depth는 레이어 수로 네트워크의 깊이가 증가할수록 더 복잡한 feature이자 더 abstract한 feature를 잡아낸다. 이 때 발생하는 vanishing gradient는 batch normalization이나 residual block으로 해결한다. Depth는 2배 증가하면 계산량도 2배 증가하게 된다.

Resolution은 해상도로 해상도를 키울수록 더 세부적인 feature를 잡아내 정확도가 높아진다. 하지만 resolution도 계산량이 제곱에 비례하다는 특징이 있다.

이 세 dimension모두 증가할수록 정확도가 증가하게 되고 정확도 증가량은 점점 줄어든다고 한다.

Compound Model Scaling

- compound Scaling
  - Width, Depth, Resolution을 모두 증가시킬 때 성능이 가장 좋음 (width만 변경하고 실험한 그림 참고)
  - Grid Search와 Hyper Parameter-Φ(phi)를 이용해서 scaling을 하는 개념 ($d=a^{phi}$, $w=b^{phi}$, $r=r^{phi}$)
  - Width와 Resolution은 w,r의 제곱에 비례하여 연산량 증가, depth는 d배만큼 연산량 증가 -> total FLOPS 계산할 때 고려함



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

Grid Search -> a=1.2, B=1.1, r=1.15

total FLOPS by $(\alpha \cdot \beta^2 \cdot \gamma^2)^\phi$.

-> phi로 모델 사이즈를 조절하고
2^{phi}로 total FLOPs를 계산할 수 있음

왼쪽 그림은 width, depth, resolution을 모두 증가시켰을 때 성능이 가장 좋았음을 나타낸다. width만 변경하고실험한 그림을 참고하면 된다. (파란색: width만 변경, 초록색: width, depth만 변경, 노란색:width,resolution만 변경, 빨간색: 전부 변경)

Compound scailing을 할 때는 phi와 a,b,r이라고 하는 파라미터를 이용한다. a,b,r은 Grid Search를 이용해서 찾고 phi는 사용자가 지정하는 값이다.
여기서 Grid Serarch란 모델에 넣을 수 있는 값들을 순차적으로 입력한 뒤에 가장 높은 성능을 보이는 하이퍼 파라미터들을 찾는 탐색 기법이다.
참고로 efficientnet에 적합한 a,b,r은 1.2, 1.1, 1.15였다고 한다.
하이퍼 파라미터들은 FLOPS 증가량과 관계가 있는데, $d=a^{phi}$, $w=b^{phi}$, $r=r^{phi}$로 나타냈을 때 FLOPS 식은 $(a*b^2*r^2)^{phi}$ 로 나타낼 수 있다. 왜냐하면 width와 resolution은 제곱에 비례하여 증가하고 depth는 d배만큼 증가하기 때문이다.
a,b,r이 1.2, 1.1, 1.15일 때 $(a*b^2*r^2)$는 2와 비슷한 값이 나오므로 phi가 1이면 연산량은 2배가 증가하고 phi가 n이면 연산량은 2^n배가 증가한다고 생각하면 된다.
결국 Compound Scaling은 Grid Search로 모델에 적합한 하이퍼 파라미터(a,b,r)을 찾은 뒤 이용자가 phi를 조절함으로써 모델을 키우거나 줄이든 기법을 의미한다.

Architecture

- EfficientNet Architecture
  - baseline network는 Mnas-Net과 유사하게 AutoML을 사용해서 제작
  - EfficientNet-B0는 grid search로 최적의 parameter(a,b,r)를 찾고 phi=1일 때의 네트워크
  - EfficientNet-B1~B7은 a,b,r은 고정시키고 phi만 변화시킨 네트워크

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

- 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).

efficientnet 구조는 MNas-Net과 유사하다. MNas-Net은 AutoML(머신러닝 모델 제작을 위한 머신러닝)을 사용해서 제작된 네트워크로 모바일에서 실행 가능하는 모델을 목적으로 제작되었다.
여기서 EfficientNet-B0는 MNas-Net과 유사한 구조에 phi=1일 때의 네트워크이다. 즉 depth를 1.2배, width를 1.1배, resolution을 1.15배한 Mnas-Net 유사 모델이다.
이후 EfficientNet-B1부터 B7까지는 a,b,r을 고정시키고 phi만 변화시킨 모델들이다.
사이즈가 커졌기에 a,b,r도 다시 grid search하는 방법도 있지만 grid search 알고리즘이 자원을 심하게 차지하기 때문에 굳이 하지 않았다고 한다.

Experiments

- Result
  - Accuracy가 비슷한 모델끼리 성능을 비교 -> EfficientNet이 Parameter, FLOPs가 더 작고 Accuaracy도 더 높음

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**

Parameter, FlOPs만큼 8배 가량 좋지는 않지만
5배 이상 속도가 빨라짐

EfficientNetB0~B7 모델들과 정확성이 비슷한 모델들 각각을 비교한 사진이다.

공통적인 특징은 EfficientNet이 Parameter와 FLOPs가 더 작고(8배 가량 더 좋음) Accuracy도 더 좋거나 같다는 점이다. Latency(시간)같은 경우 8배까지는 아니지만 5배 정도 더 좋다는 점을 확인할 수 있다.

Experiments

- Result (Scaling Up)
  - 다른 모델에도 compound scaling을 적용하고 비교 -> FLOPS는 비슷하지만 성능이 좀 더 향상됨

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%**

다른 모델에 compound scaling을 적용한 결과이다. 한 dimension만 변경한 것과 비교했을 때 FLOPS는 비슷하지만 성능이 좀 더 향상됐다고 한다.

Experiments

- Result (Transfer Learning)
  - ImageNet Data로 Pre-training -> 8 datasets에 대해 Fine Tuning -> 8개 중 5개의 dataset에 대해 SOTA가 됨

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).

ImageNet Data로 사전학습한 모델을 cifar-10이나 food-101 등 8개의 dataset에 fine-tuning한 결

과이다. 무려 8개 중 5개의 dataset에 대해 SOTA 성능이 나왔다고 한다.

Experiments

- without Compound Scaling
  - width, depth, resolution 중 하나로 scaling을 하면 FLOPS는 더 낮지만 Accuracy가 떨어짐

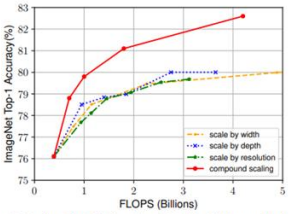
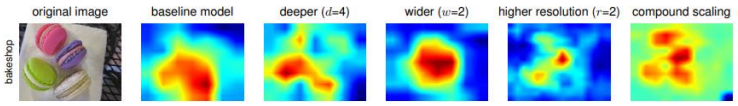


Figure 8. **Scaling Up EfficientNet-B0 with Different Methods.**



Class Activation Map(CAM)으로 결과에 가장 영향을 많이 준 영역을 찾아낸 결과 -> Compound Scaling이 가장 우수

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%**

마지막으로 1-dimension scaling과 compound-scaling그래프를 비교하고 CAM을 찍었다.
다른 모델에 적용한 것과 마찬가지로 FLOPS는 compound-scaling이 비슷하거나 좀 더 높고(1-dimension이 더 낮고) Accuracy는 더 좋았다.

CAM은 결과에 가장 영향을 많이 준 영상 영역을 찾는 기법인데, 이 역시도 compound scaling이 마카롱 세 영역을 모두 찾아내면서 가장 좋은 결과를 보였다.

END

- 논문 정리
  - Accuracy를 높이기 위해서는 width, depth, resolution을 모두 높여야 한다. 그러나 무작위로 셋 다 높이면 자원 낭비가 심해 효율적이지 않을 뿐더러 효과적이지도 않다. (Accuracy gain이 점점 작아지기 때문)
    논문에서 제안하는 compound scaling 기법을 이용하면 Efficient하게 Accuracy를 높일 수 있다.
    특히 MNas-Net처럼 이미 효율적인 모델에 적용하면 Accuracy를 위해 연구된 모델 이상의 성능을 보인다.
    따라서 적은 연산량으로 높은 성능을 도출할 수가 있다.

- 참고자료
  - EfficientNet 논문
    - https://arxiv.org/pdf/1905.11946.pdf
  - PR-169: EfficientNet
    - https://www.youtube.com/watch?v=Vhz0quyvR7I&t=594s
  - EfficientNet 논문 설명
    - https://greeksharifa.github.io/computer%20vision/2022/03/01/EfficientNet/

논문 정리 내용이다. (사진에 적은 글 참고) 이 논문은 PVANet처럼 기존에 소개된 모델(ex)

GoogleNet)에서 연산량을 줄이고 성능은 비슷하게 하자는 취지는 아니다.
모델 성능이 더 잘 나오도록 할 때 최대한 연산량을 줄이면서 잘 나오게 하자는 취지인 것 같다.