

* Abstract.

기존의 network는 고정된 크기의 네트워크로 개발된다.

네트워크의 크기가 조정이 필요하다면 여러 실험을 거쳐 구조를 바꾼다.

논문에서는 NAS (Network Architecture Search)를 참고하여,

feature map의 크기, 깊이, 넓이를 조정한다

* Introduction

Network의 크기를 바꾸는 것은 보통 성능을 높이기 위해 사용된다.

하지만 그 방식이 정형화된(기입) 않고 연구가 필요하다.

평균적인 방식은 depth와 width를 바꾸는 것이다.

⇒ 수동 조성이 필요하며, 오직의 결과를 찾기 힘들다

본논문은 같은 정량도를 산출해내는 depth, width, resolution

공식이 있을 뿐 아니라 가능하다

→ Compound scaling method.

→ 일정한 scaling 계수를 설정해놓고 scaling을 조정한다

⇒ 별래 작은 model에 대해 2^N 만큼의 계산량 증가를 허용한다면,

depth, width, resolution 을 각각 $\alpha^N, \beta^N, \gamma^N$ 만큼의 scaling을 진행한다.

⇒ α, β, γ 는 고정값

사실 당연히 input의 차이면 reception field를 카우고, 세밀한

부분은 보자마자 depth와 channel 수를 가리는 것은 당연하다.

PerNet, MobileNet mm 같은 곳에서도 시로운 baseline을 위해

NAS를 쓰고, 적용한다

* Related Work

⇒ 어떤 ImageNet 모델

⇒ ImageNet에서 좋은 성능을 가진다면 다른 Task에서 좋은 성능을 가진다

- ConvNet efficiency

모델 경량화가 유익하고, 이미 NAS가 적어도 성능은 낮아지지 않음

매우 큰 모델보다 훨씬 빠름이다.

따라서 큰 모델이 대신 일반적인 efficient 방식を使う

- Model scaling

depth, width, resolution은 network의 power의 중요한 것은 사실이다

아직 open question이다.

* Compound Model Scaling

Scaling 공식화.

- Problem formulation

ConvNet layer는 합집합 형식으로 표현할 수 있다. F_i

$$Y_i = F_i(X_i), \quad X_i = (H_i, W_i, C_i)$$

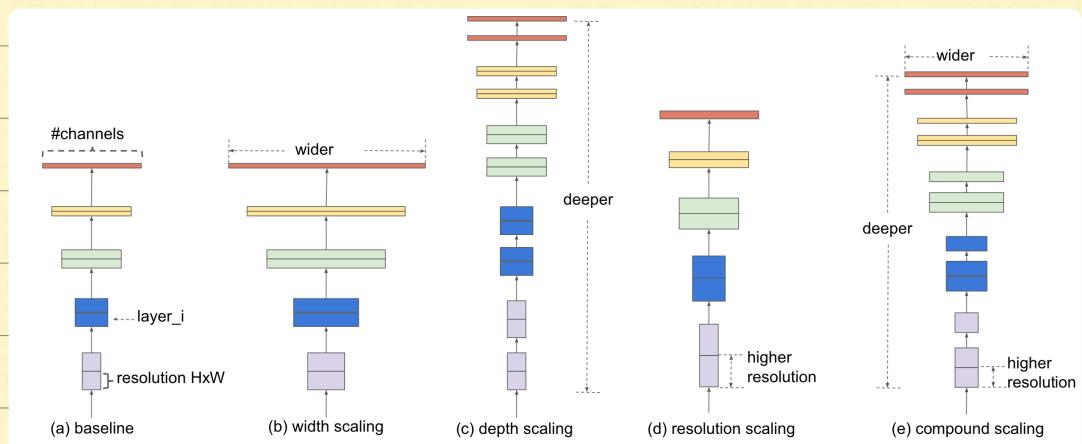
$$\therefore \text{model } N = F_k \circ F_{k-1} \circ \cdots \circ F_1(X_1)$$

$$= \bigcup_{j=1 \sim k} F_j(X_j)$$

$\Rightarrow F$ 는 다 같은 구조의 블록이 같은 개수 (Repetitive 구조)

$$\therefore N = \bigcup_{i=1 \dots s} F_i^{L_i}(X_{(H_i, W_i, C_i)})$$

$\Rightarrow L_i$ 는 stage), F_i 는 몇번 블록인지를



모델이 구조를 찾는 다른 논문과 다르게, L_i, C_i, H_i, W_i 를 조절

F_i 를 고정함으로써, 자원에 제약이 대비 모델 디자인 문제를 간단히 했고.

L, C, H, W 을 조절함으로써 모델을 조절하자,

모델 공간을 더 넓이기 위해 상수배로 λ scale을 조절하자.

간접된 자원 액세스 청탁도를 최대화하기 위한 공식화



$$\max_{d, w, r} \text{Accuracy}(N(d, w, r))$$

$$\text{subject to } N(d, w, r) = \bigodot_{i=1 \dots s} \hat{F}_i^{d \cdot \hat{L}_i} (X_{\langle r \cdot \hat{A}_i \cdot r \cdot \hat{w}_i \cdot n \cdot \hat{c}_i \rangle})$$

$\text{Memory}(N) \leq \text{target memory}$

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

- Scaling dimension

optimal d, w, r 은 목표의 영역도 뿐만 아니라 계산 효율성이 커야 한다.

딥러닝이 때문에 어렵다

딥러닝 전통적인 scaling은 이중 단계의 계산에서만 진행된다.

- Depth(d)

depth 조절은 가장 일반적인 방법이다.

직관적으로 depth가 커지면, network는 더 풍부하고 복잡한 feature를

잡아 대답이 새로운 테스트에서 대답을 잘 동작한다.

하지만 동시에 훈련하기 힘들어지고, gradient vanishing 문제가 나타난다.

BN, Residual 등의 방식으로 불구하고 Deep NN의 성능 평가는 줄어든다.

Fig 3의 축간은 diminish accuracy의 예시다 (d의 따른)

- Width(w)

width 조절은 작은 크기의 모듈이 주로 쓰인다.

wide network는 복잡한 feature를 보고, 훈련이 용이하다

하지만 극단적으로 넓지만 얕은 네트워크는 얼굴 인식, 객체 탐지 등

고난관의 Task에서는 성능이 떨어지는 경향이 있다.

- resolution (r)

resolution이 즐을수록 더 세밀하게 학습되는 힘을 갖다.

更深도가 즐아지는 경향이 있다.

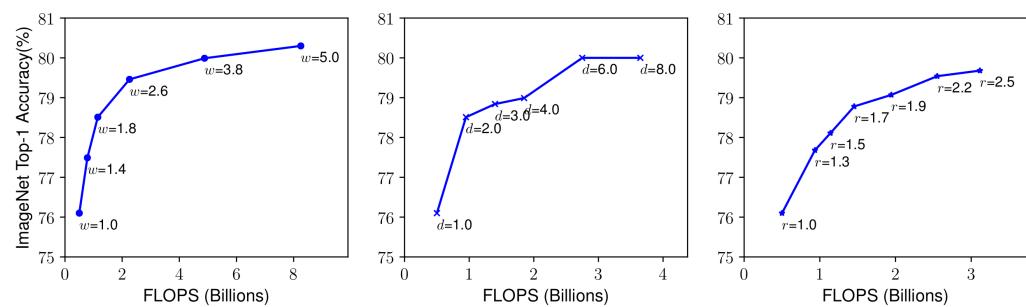


Figure 3. Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients. Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single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

\Rightarrow 2d dimension when scaling up은 있는데 한 성능상한도가 있다.

큰 모델이라는 표현이다.

- Compound scaling

d, w, r 은 독립적이지 않다.

직관적으로 높이가 커지면, receptive field를 넓혀, 큰 이미지를 처리하는 능력을 확장한다.

즉 d 와 w, r 은 상호작용한다.

마찬가지로, 큰 r 은 높은 feature를 처리하는데 있어 w 도 고려해야 한다.

\Rightarrow compound scaling을 생각하는 이유

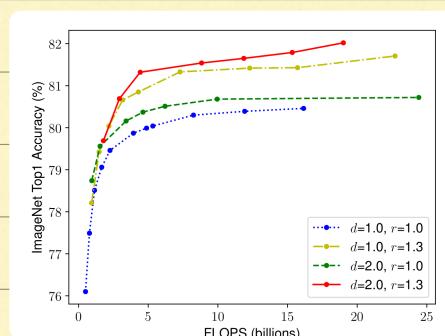
이를 증명하기 위해 다른 r, d, w 들을

w 를 scaling

\Rightarrow

$\Rightarrow d, r$ 은 수용 saturation

되는 듯하다.



\Rightarrow certain compound scaling을 찾는다

NAS 는 이전 연구들로 domain划分 balancing을 찾지만
모두는 tuning이 너무 많다

certain rule로 compound scaling 방식을 제시한다.

compound coefficient ϕ 사용

$$\left\{ \begin{array}{l} \text{depth: } d = \alpha^\phi \\ \text{width: } w = \beta^\phi \\ \text{resolution: } r = \gamma^\phi \\ \text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \end{array} \right.$$

$d \geq 1, \beta \geq 1, \gamma \geq 1$

α, β, γ 는 small grid search에 의해 찾은 constant로 간주된다.

ϕ 는 model scaling의 성능에 따라 사용되는 값이다.

α, β, γ 는 같은 차원의 d, m, r 을 어떻게 디자인할지에 따라 결정된다.

w, r 은 거듭제곱 FLOPs는 계산으로 각각의 디자인

$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ 의 식으로 constraint를 걸었고,

0(m) FLOPs는 2^ϕ 만큼 증가하게 된다

\Rightarrow 그 말고 다른 속성을 땐?

* EfficientNet Architecture.

Model scaling의 모델의 기본 구조를 바꾸진 않고 때문에

base line network의 구조는 유지된다.

다만 기존의 쌍층은 convolution scaling이 short, 성능은 평균화되지만

scaling의 효율성이 증명을 위해 작은 사이즈의 baseline인 efficientNet을 개발한다

mNASNet은 기본으로 하였고, optimization goal은

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

T 는 target FLOPS, $w = -0.072$, accuracy와 FLOPS 사이의

trade-off를 고려하는 hyper param of CL.

mNASNet은 latency를 optim을 하지만 efficientnet에서는 특정 device를 두지 않기

대신 FLOPS를 optim 한다

efficientNet-B0은 baseline이고, target FLOPs는 400M이다.

Stage i	Operator \hat{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	MBCConv1, k3x3	112×112	16	1
3	MBCConv6, k3x3	112×112	24	2
4	MBCConv6, k5x5	56×56	40	2
5	MBCConv6, k3x3	28×28	80	3
6	MBCConv6, k5x5	14×14	112	3
7	MBCConv6, k5x5	14×14	192	4
8	MBCConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

\Rightarrow MB conv (mobileNetV1) or SENet을 쓰인다.

- EfficientNet-B0를 기준으로 compound scaling을 하는 단계.

STEP1. $\phi=1$ 은 고정하고, α, β, γ 는 small grid search를 한다

$$\text{그 결과 } \alpha=1.2, \beta=1.1, \gamma=1.15, \rightarrow \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

STEP2. α, β, γ 고정시하고 ϕ 를 뻬닝하거라

$$\beta_1 \sim \beta_7 \text{을 구한다.}$$

\Rightarrow 따라서 큰 모델의 성능은 모델을 구현하는

큰 모델은 NAS가 고려한 데 β_1 을 찾고 단계의 ϕ 를 늘려라

찾아 모델의 넓기 찾음

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).

* Experiments.

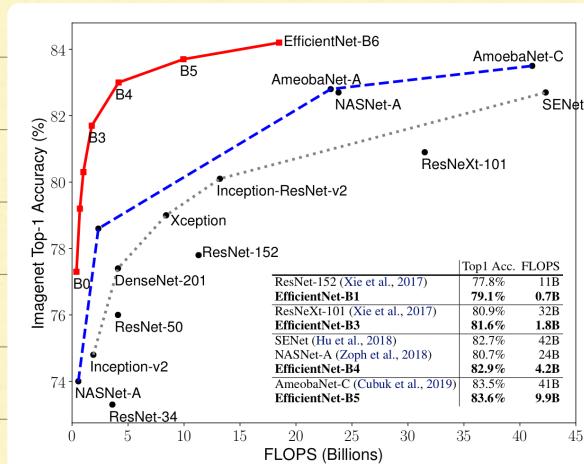
- ImageNet

in NASNet 2L 201. 실습 78

\Rightarrow RMS Prop(0.9, 0.9), BN, weight decay $1e-5$,

init LR = 0.256 decay 0.9n every 1.4 epoch, SLU

Auto arguments, stochastic depth 0.8 1/f, dropout



\Rightarrow 다른 dataset 2 FLOPs, accuracy 612

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

\Rightarrow Latency 2 %

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.

	Model	Comparison to best public-available results				Model	Acc.	#Param	Comparison to best reported results			
		Acc.	#Param	Our Model	Acc.				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).

\Rightarrow 다른 dataset on transfers 25%

* Discussion.

Compound scaling의 단점은 scaling rate는 각각 다른 흐름을 갖다

더 정확한 이해를 위해 class activation map을 I2로

↳ classification을 위한 GT와는

GAP (Global Average Pooling)을 사용함

Localization을 가능케

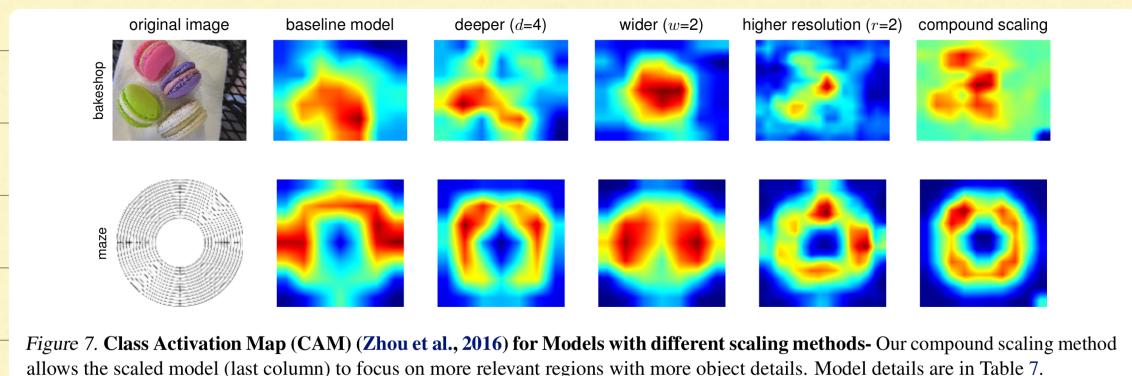


Figure 7. Class Activation Map (CAM) (Zhou et al., 2016) for Models with different scaling methods- Our compound scaling method allows the scaled model (last column) to focus on more relevant regions with more object details. Model details are in Table 7.

⇒ compound scaling은 object의 더 정밀한 위치를 찾는다.