EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

한현수

Abstract

이 논문에서는 Convolutional Neural Networks는 고정된 리소스로 개발한 후 더 많은 리소스를 사용할 수 있다면 accuracy를 위해 network의 depth, width, resolution을 scale up하는데, 효율적인 scale up 방법을 제안한다.

1. Introduction

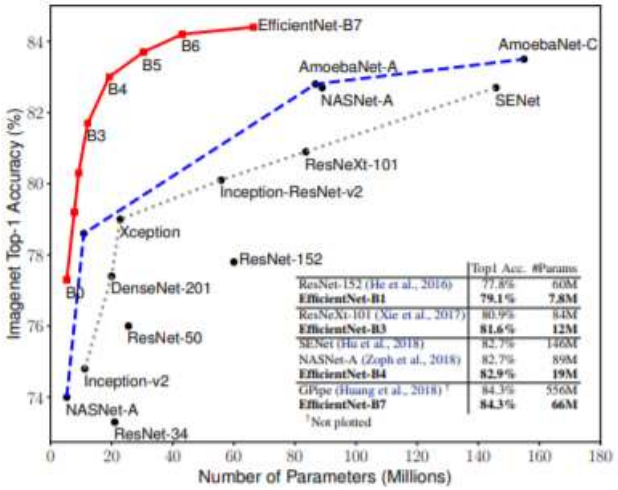

Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1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

이 논문에서는 CNN에서 Depth , Width , Resolution의 최적의 조합을 찾는다. 그 결과 위의 그림을 보면 다른 network들에 비해 적은 parameters로 높은 accuracy를 보임을 알 수 있다.

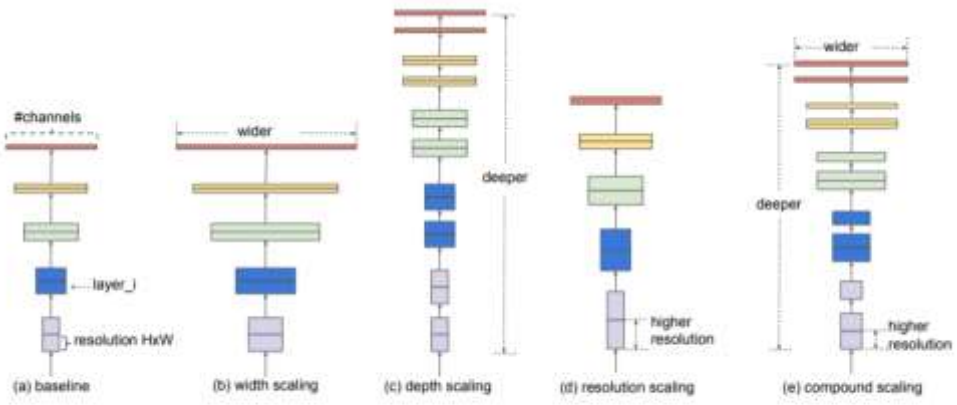

Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

위의 그림과 같이 CNN은 width(W), depth(D), resolution(R) 이 3가지 scaling 방법이 있다.

- Width : Convolution channel
- Depth : Network layer
- Resolution : Image Resolution(해상도)

이 논문에서는 제한된 computational resource에서의 W, D, R의 최적의 조합을 찾는 compound scaling을 진행한다. 즉 computational resource가 2^N 더 사용할 수 있다면, depth는 α^N만큼, width는 γ^N, resolution은 β^N 만큼 더 늘릴 수 있다.

3. Compound Model Scaling

3.1. Problem Formulation

ConvNet Layer i : $Y_i = F_i(X_i)$, $X_i = input$, $X_i\ shape = (H_i, W_i, C_i)$

Define ConvNet : $N = F_k \odot ... \odot F_2 \odot F_1(X_1) = \odot_{j=1...k} F_j(X_1)$

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

$F_i^{L^i}$는 ResNet은 layer를 몇 개의 stage로 나누는데, F_i가 stage i 가 L_i번 반복됨을 의미한다.

그래서 최종 목표를 식으로 정리하면,

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

로

$d \cdot \hat{L}_i$는 depth scaling, $(r \cdot \hat{H}_i\ , r \cdot \hat{W}_i)$는 resolution scaling, $w \cdot \hat{c}_i$는 width scaling을 뜻하며, memory 와 flops가 computational resource보다 작은 N 중에서 Accuracy가 가장 높은 N 를 grid search를 통해 찾는 것이다.

3.2. Scaling Dimension

Depth : network가 깊어질수록 성능은 좋아진다. 하지만 ResNet-1000과 ResNet-101의 성능이 비슷한 걸 보아 network를 깊게 쌓는 것은 한계가 있다.

Width : width를 넓게 할수록 fine-grained features를 잘 잡아낼 수 있다.

Resolution : 큰 해상도의 이미지를 사용할수록 fine-grained patterns를 잘 잡아 낼 수 있다.

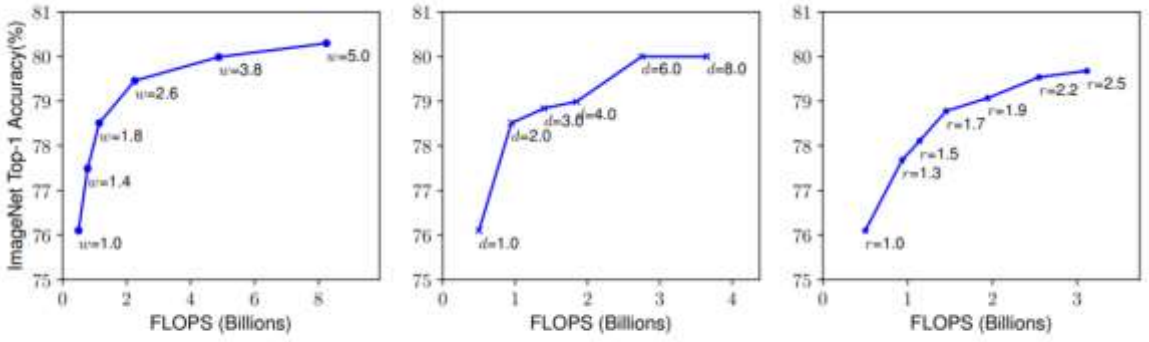

Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single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

위의 그래프들을 보면 d , w , r 이 높아질수록 Accuracy는 높아지지만, 점점 커질수록 그에 따른 accuracy증가폭이 적어지는 것을 확인할 수 있다.

3.3. Compound Scaling

CNN에서 이미지의 resolution이 높을수록 각 픽셀의 feature를 잘 학습하기 위해 depth를 키워야 한다. 그러므로 depth, width, resolution의 관계가 독립적이지 않음을 알 수 있다.

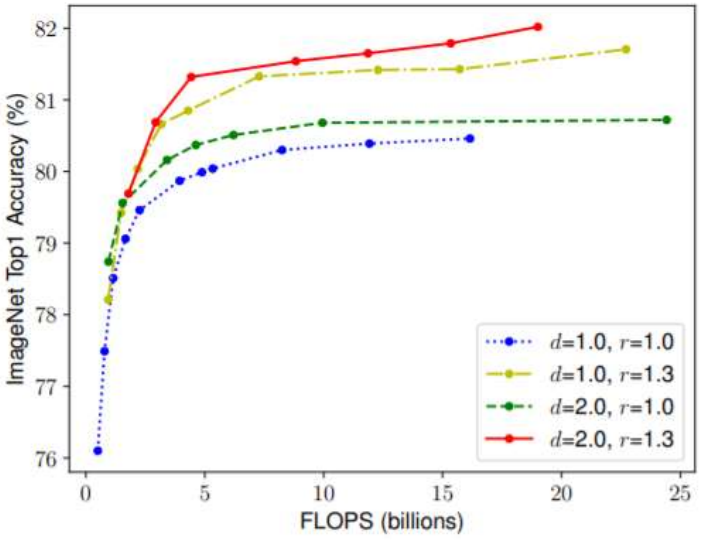

Figure 4. **Scaling Network Width for Different Baseline Networks.** Each dot in a line denotes a model with different width coefficient (w). All baseline networks are from Table 1. The first baseline network (d=1.0, r=1.0) has 18 convolutional layers with resolution 224x224, while the last baseline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

위의 그래프는 depth와 resolution을 고정한 채로 width를 변화시키면서 테스트한 결과이다. 이 때 다양한 크기의 depth와 resolution을 테스트하는데, 동일한 flops에서 width, depth, resolution에 따라 성능차이가 생김을 확인할 수 있다. 그래서 이 논문에서는 위의 조합을 찾기 위해 compound scaling method를 제시한다.

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

α, β, γ는 small grid search에 의해 결정된 상수이다. $\emptyset$는 임의로 설정 가능한 변수로 computational resource에 따라 적절한 값을 설정한다. 여기서 제곱이 들어가 있는 이유는 depth가 2배가 되면 FLOPS가 2배가 되지만, width, resolution은 2배가 되면 FLOPS가 4배가 되기 때문이다. 그래서 최종적인 FLOPS $\propto (\alpha, \beta^2, \gamma^2)^\emptyset$ 가 된다. 이 논문에서는 $\alpha, \beta^2, \gamma^2 \approx 2$라는 제약을 사용해 FLOPS는 약 $2^\emptyset$가 된다.

4. EfficientNet Architecture

Base model에 따라 기본 성능차이가 많이 발생하므로, 기존에 알려진 좋은 model을 base model로 설정한다. Depth, width, resolution을 찾기 위해 MNasNet과 동일한 search space를 사용하였으며, 최적화 식은 MNasNet은 $ACC(m) \times [LAT(m)/T]^w$를 사용하였지만 본 논문에서는 디바이스들을 대상으로 하지 않기 때문에 latency 대신 flops를 사용하여 $ACC(m) \times [FLOPS(m)/T]^w$을 최적화식으로 사용하였다.

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

이렇게 만들어진 기본 모델인 EfficientNet-B0의 구조이다. MNasNet과 유사하나, SE(Squeeze-and-Excitation)block이 추가되고, 더 큰 target FLOPS를 설정하였으므로 모델이 좀 더 크다. 이 모델을

시작으로 scale을 확장하는데 그 방법은 아래와 같다.

STEP 1

먼저, $\emptyset$ = 1로 고정한 뒤에 small grid search를 사용하여 위의 식 2, 3을 만족하는 α, β, γ 값을 찾는다. 이때 EfficientNet-B0 에서의 α, β, γ 값은 $\alpha = 1.2$, $\beta = 1.1$, $\gamma = 1.15$ 이다.

STEP 2

STEP 1에서 나온 α, β, γ 값을 고정한 후, $\emptyset$을 식 3에 만족하도록 바꿔 scale up한다. 그렇게 나온 결과가 EfficientNet-B1~B7이다.

이렇게 base line model을 step1, step2를 이용하여 scale up 할 수 있다.

5. Experiments

위의 scale up 방법이 맞는지 증명하기 위해 MobileNets과 ResNet에 적용하여 실험을 진행하였다.

5.1. Scaling Up MobileNets and ResNets

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).

이는 EfficientNet-B0 ~ B7까지로 비슷한 Accuracy를 가짐에도, parameters 수가 확연히 적음을 확인할 수 있었다.

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**

Table 3을 보면 scale up 을 1가지만 하는 것보다 compound scale 하는 것이 효과적임을 알 수 있다. Table 4는 latency를 측정한 것인데, 최적화 식으로 latency 대신 FLOPS를 사용하였음에도 latency가더 낮음을 알 수 있다.

5.2. ImageNet Results for EfficientNet

EffcientNet을 ImageNet으로 학습 한 결과이다. 이때 RMSProp optimizer(decay=0.9, momentum=0.9), batch norm momentum=0.99, weight decay=1e-5, initial learning rate=0.256(decay=0.97 every 2.4 epochs), SiLU Activation, Auto Augment, stochastic depth with survival probability=0.8, dropout rate는 EfficientNet-B0 = 0.2, EfficientNet-B7 = 0.5를 사용하여 실험을 진행하였으며 아래는 Transfer Learning 결과이다.

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)	[1]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)	[1]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)**

[1]GPipe (Huang et al., 2018) trains giant models with specialized pipeline parallelism library.
[1]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).

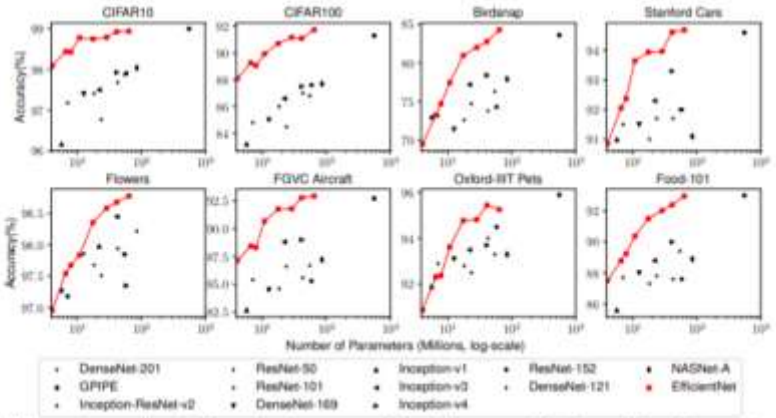

Figure 6. **Model Parameters vs. Transfer Learning Accuracy** – All models are pretrained on ImageNet and finetuned on new datasets.

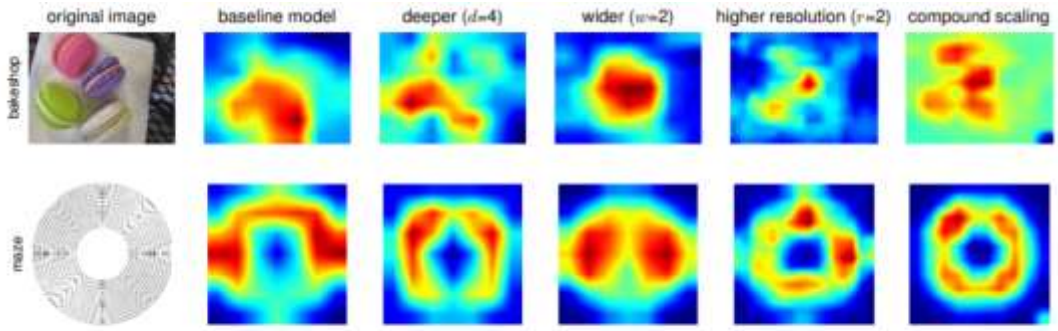

Figure 7. **Class Activation Map (CAM) (Zhou et al., 2016) for Models with different scaling methods-** Our compound scaling method allows the scaled model (last column) to focus on more relevant regions with more object details. Model details are in Table 7.

이는 Class Activation Map으로 compound scaling이 각 객체들을 잘 찾아냄을 볼 수 있다.

7. Conclusion

이 논문에서는 ConvNet에서의 depth, width, resolution를 무작정 scale up 하는 것이 아닌, balance를 맞춰 scale up 하는 방법을 제시하였다. 이에 대한 이점으로 한정된 computational resources에 맞는 model을 설정할 수 있다.