Friendly Introduction to EfficientNet

Rethinking Model Scaling for Convolutional Neural Networks

Ashik S 20 / 3 / 2021

Dusting the Jargon

- Baseline model : The first model you will create, which will be simple and easy to set up. You generally create a baseline and then try to make more complex solutions in order to get a better result. (Everyone can create this).

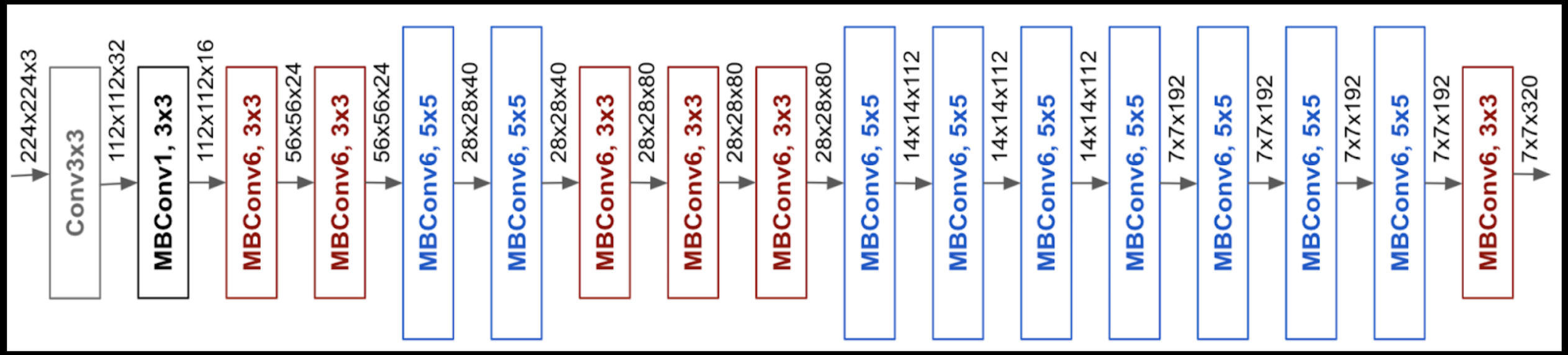
- Convolutional Neural Networks (CNN / ConvNets) : A deep learning algorithm which takes an Image as input, learn patterns in it and able to differentiate from one another. Stacked with n number of layers which help

- ImageNet : A benchmark dataset where people test their models (e.g. EfficientNet) on this to evaluate how good their model is. It is of 150 GB and has 1000 classes.

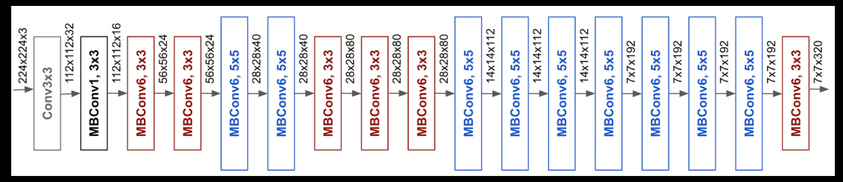
- Scaling up a model : Improve the model.

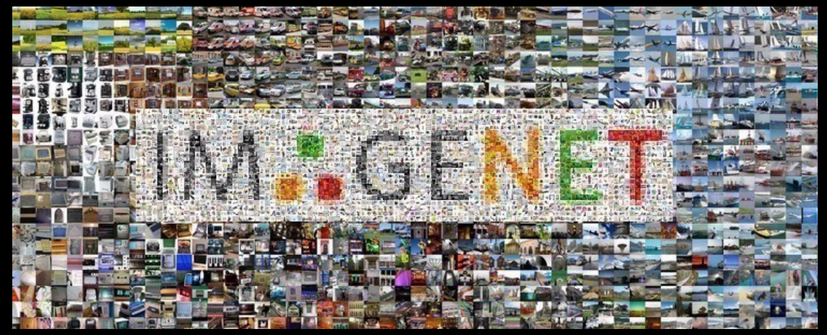
- Transfer Learning : Transfer Learning is leveraging a working model's existing architecture and learned patterns for our own problem.

EfficientNet Architecture

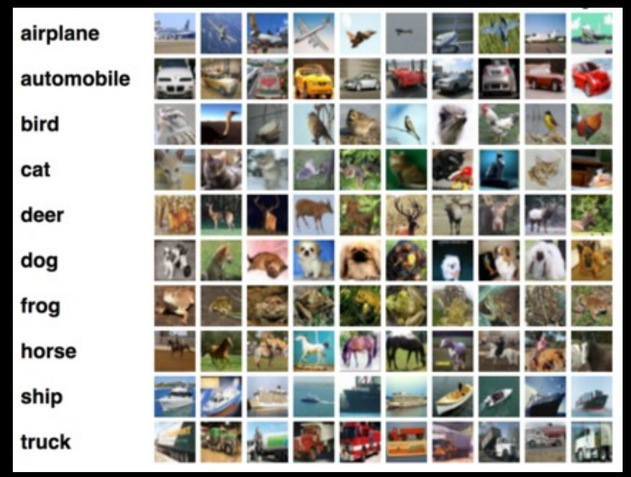


Benchmarks





(Imagenet) ⟶ 84 % and top 1%

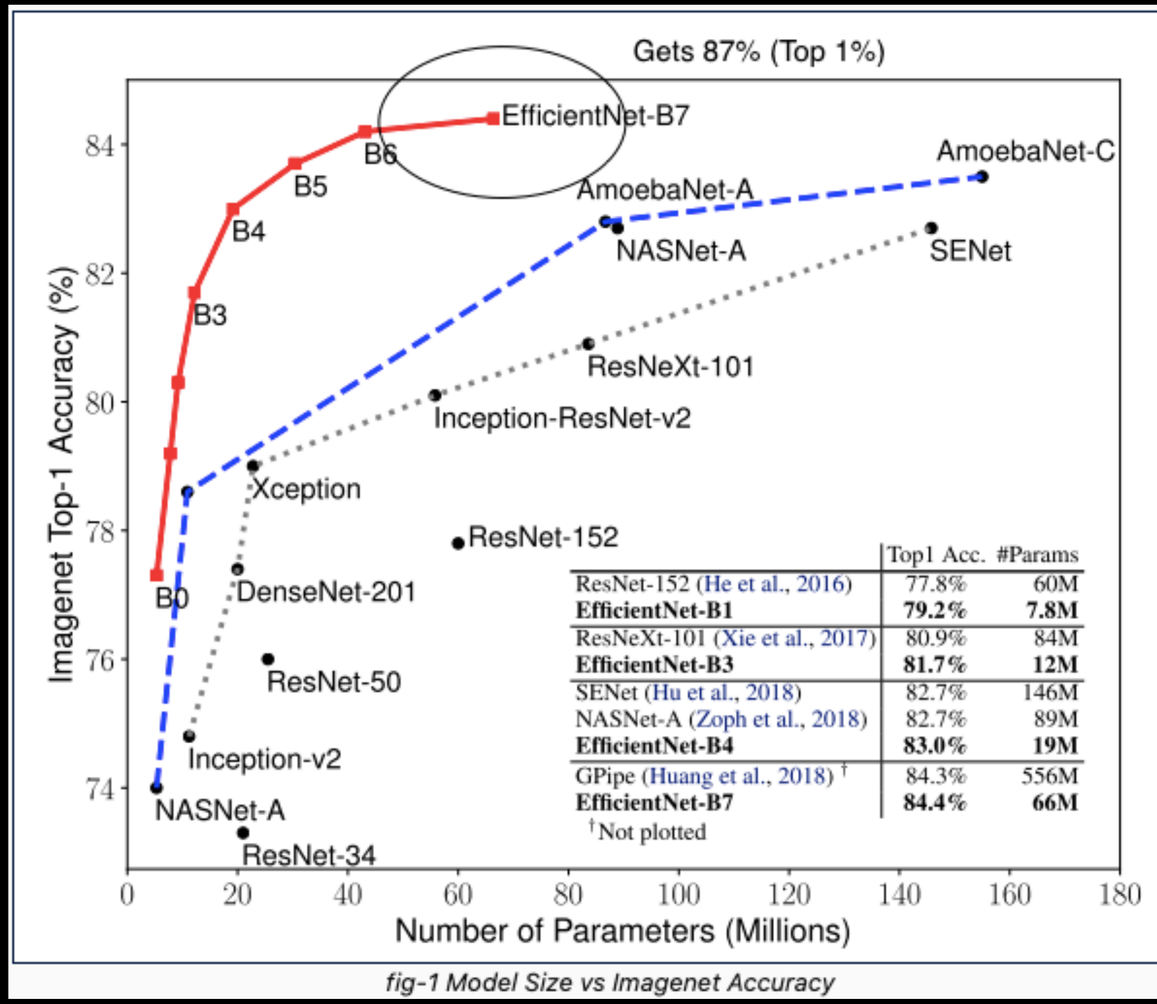


(Cifar - 10) ⟶ 96 % and top 1%

EfficientNet winning over!! 🏅

- Achieve's 87% (top-1) and 97.1% (top 5) accuracy on ImageNet and outperforms almost all ConvNets (CNN arch).
- While being 8.4x smaller and 6.1x times faster than all the best existing ConvNet so far. And work's great on transfer learning too (worked great on other data too).
- The great thing about EfficientNets is that not only do they have better accuracies compared to other models, yet they are also lightweight and thus, faster to run.
- For every model they should able to provide the same result even when transferring them to another dataset (transfer learning), where EfficientNet even performs well on other datasets.
- Less parameters but great result. Act as a good baseline.

Performance on ImageNet



	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) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

† Not plotted

fig-1 Model Size vs Imagenet Accuracy

But Why EfficientNet 🤔

Drawback of Traditional Method

Traditional Method

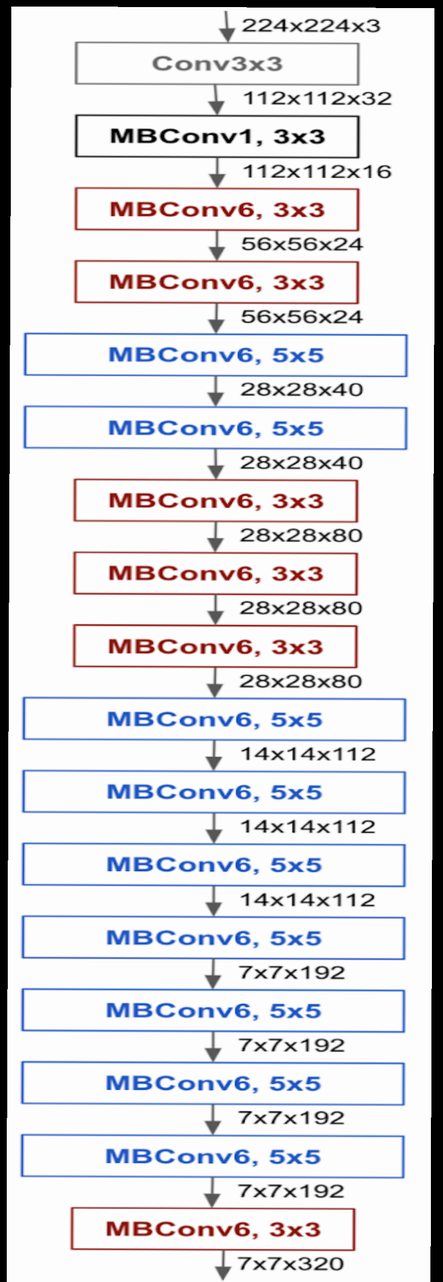
Bridging the Gap 🛠️

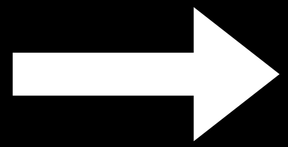
Compound Scaling

After a certain period, scaling doesn't improve the performance and it starts to adversely affects the performance.

EfficientNet introduces Compound Scaling to fix this drawback!

Architecture to Code





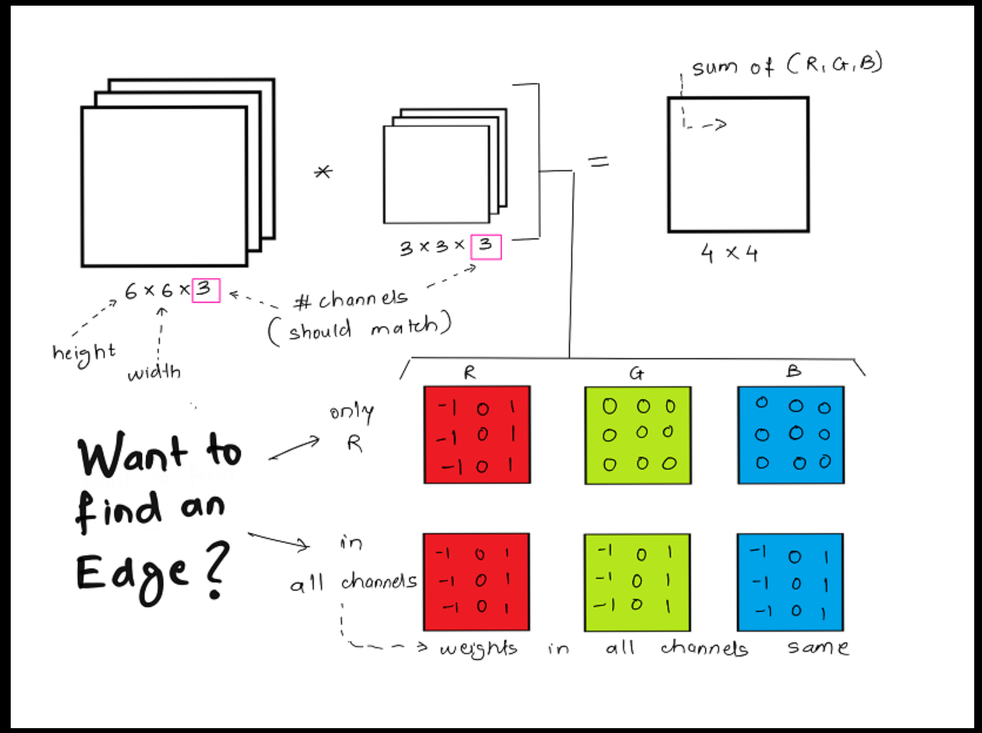
```python
# Pytorch Implementation of Efficient Net

from efficientnet_pytorch import EfficientNet
model = EfficientNet.from_pretrained('efficientnet-b0')
```

Compound Scaling (way to improve model)

- Before the EfficientNet came long the common to scale up a model is by either by one of three dimensions:
  - Width (number of channels)
  - Depth (number of layers)
  - Image Resolution (image size)

Even though it's possible to scale up two or three things, but it requires tedious manual tuning and we never know whether they can perform well

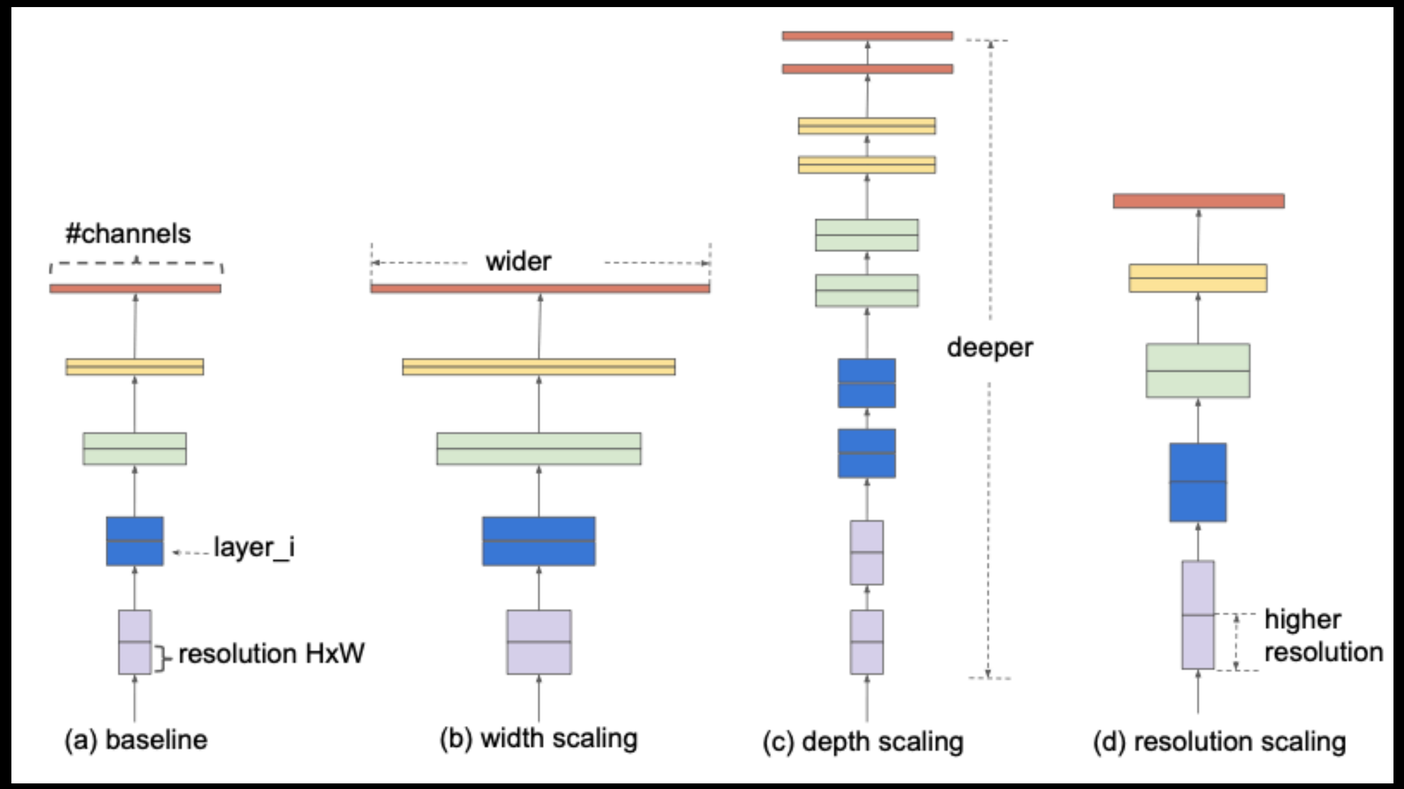


Working of a Convolutional Neural Network

(Traditional Methods)

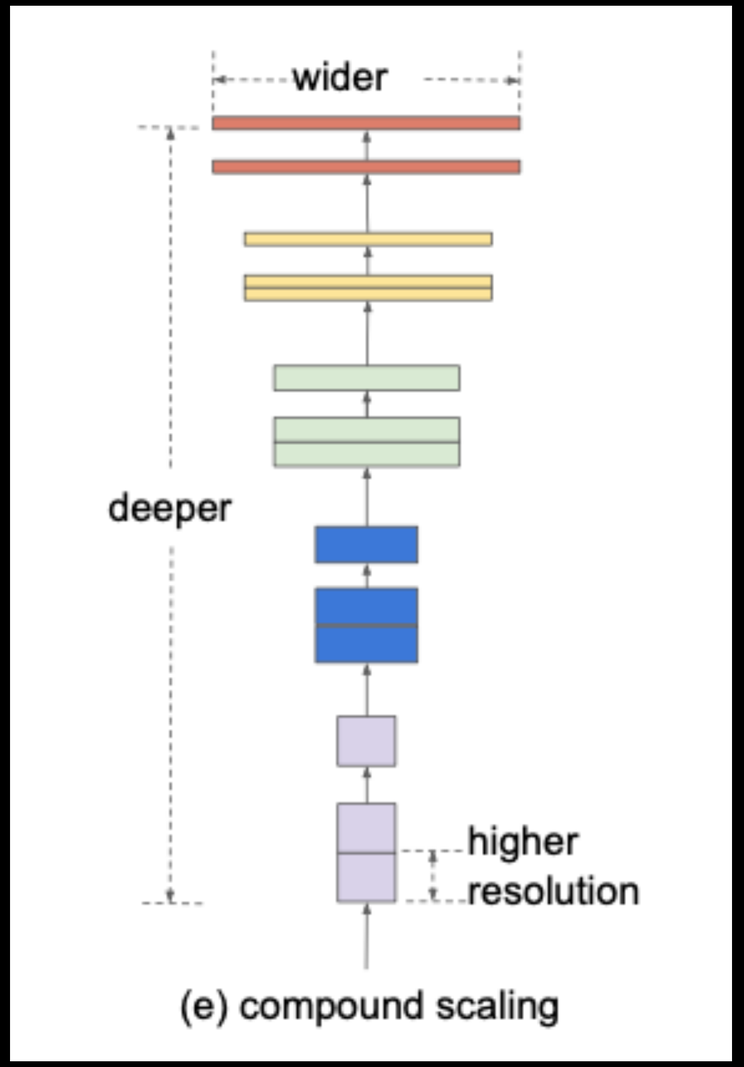
Whereas, **EfficientNets** on the other hand perform Compound Scaling, which means scale all three above dimensions at once and maintain a balance between other dimensions of the network and gives great results.

Compound Scaling (way to improve model)



Conventional Scaling of the dimensions

(tedious and one at a time)



Compound Scaling

Breaking down the dimensions

(width , depth , image size)

Depth

- Most common way of scaling up an ConvNets.
- More the layers —> the network studies more rich features and complex ones too.
- Hard to train a deep layered model, due to vanishing gradient problem
- When that happens it results in, a model with 1000 layers has same accuracy to a model having 100 layers.

Width

- This means increasing the number of channels in the layers itself.
- Most commonly used for smaller sized models.
- As they are wider and wider, they tend to capture more fine-grained features and easier to train.
- When that happens it results in, a model with 1000 layers has same accuracy to a model having 100 layers.

Image Size (resolution)

- Increasing the image size let our ConvNets to improve it's accuracy.
- Most commonly used for smaller sized models.
- Default image size we would use is (224 , 224) which comprises of width and height of a image respectively.
- Using image with bigger size might want us to use reliable GPU to process the computations of our image.

EfficientNet Vs Other Models

- EfficientNet (B0……B7), higher the number more complex the model is (larger).
- EfficientNet B0 : The first baseline model obtained.
- EfficientNet B1 - B7 : Obtained by scaling up the baseline model.
- EfficientNet B7 achieves new state-of-the-art 84% top 1 / 97.5% top-5 accuracy.
- FLOPS —> Floating point operations per second, it's a unit of speed. Which measures the computing power of a hardware.

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	-	-

But wait…

Something Groundbreaking happened!

Sadly while preparing for this seminar, a new improved model challenged EfficientNet on 15 March.

ResNet - RS

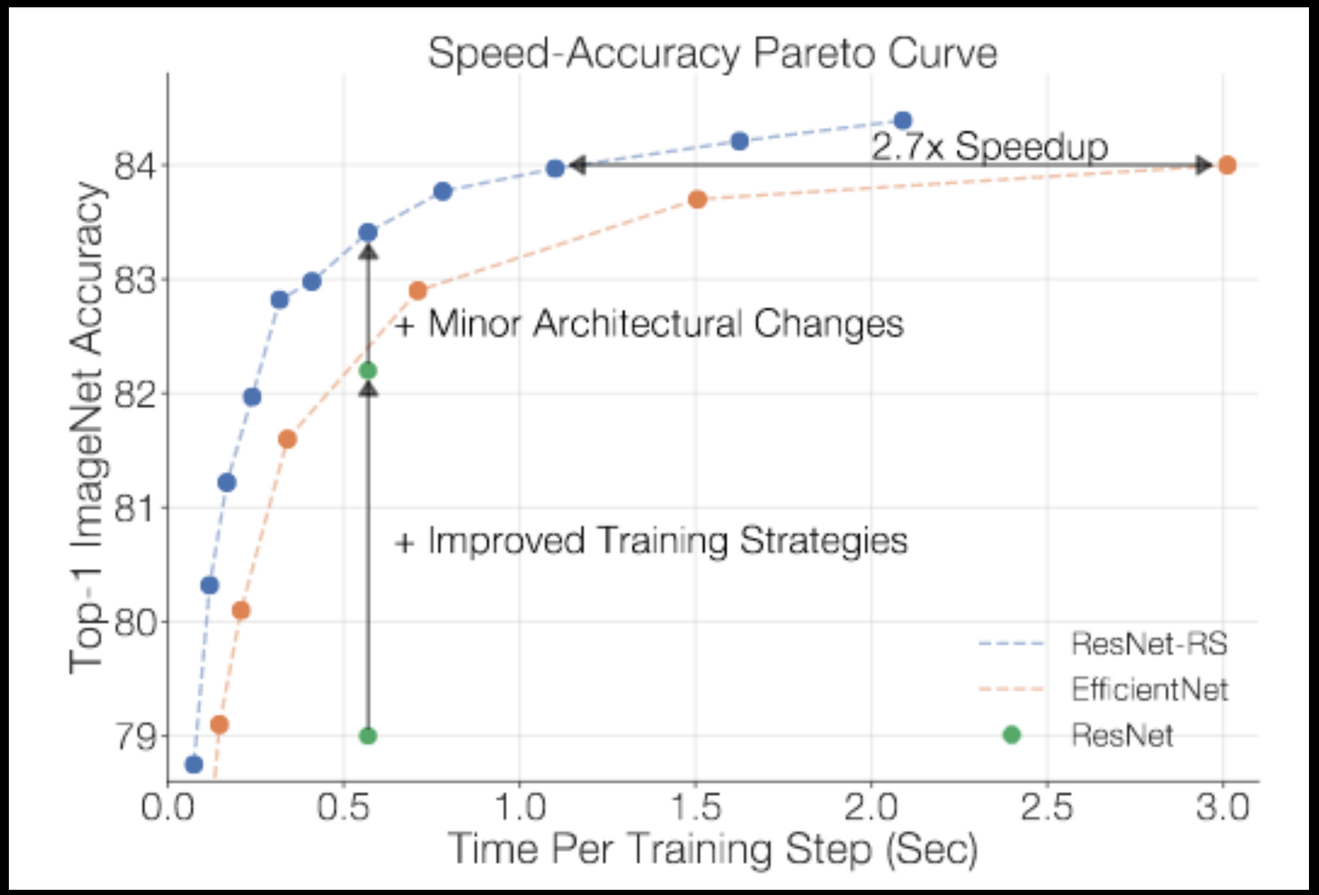
A Glimpse of Resnet - RS

- Resnet - RS which are 1.7X - 2.7X faster than EfficientNet's on TPU.
- Both offers more or less similar accuracy but ResNet wins in the efficiency and speed clearly.
- In large-scale, or difficult tasks Resnet - RS achieves 86.2% on top-1 ImageNet, while being 4.7X faster than EfficientNet
- New State-of-the-art model for Image Problems.

ResNet - RS

EfficientNet

ResNet



Reference

- Revisiting ResNets: Improved Training and Scaling Strategies
- EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks Blog - Aman Arora
- EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks:
    - MingXing Tan
    - Quoc V. Le

Thank You