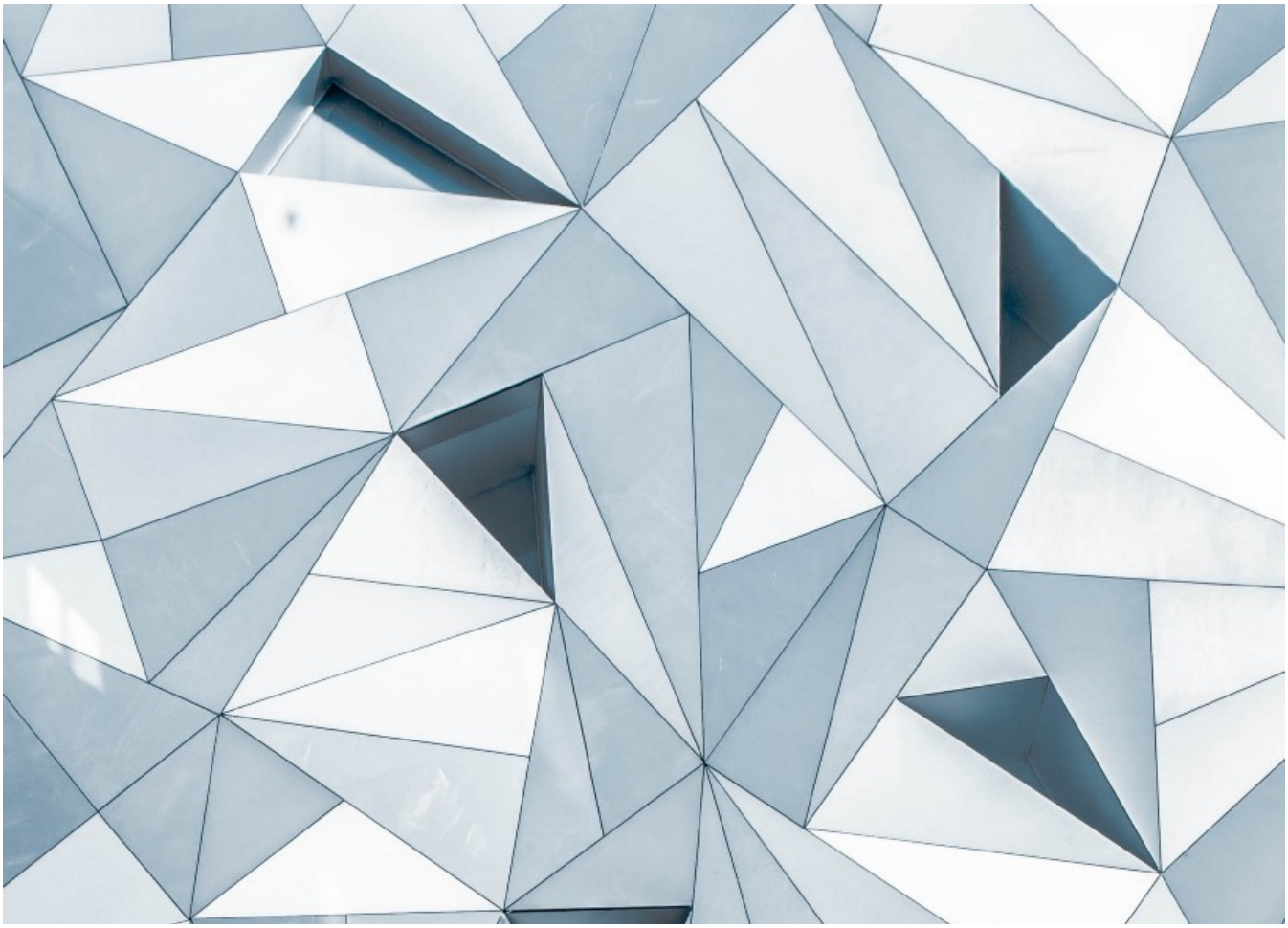
Complete Architectural Details of all EfficientNet Models

Let's dive deep into the architectural details of all the different EfficientNet Models and find out how they differ from each other.

Vardan Agarwal May 24, 2020 · 6 min read ★



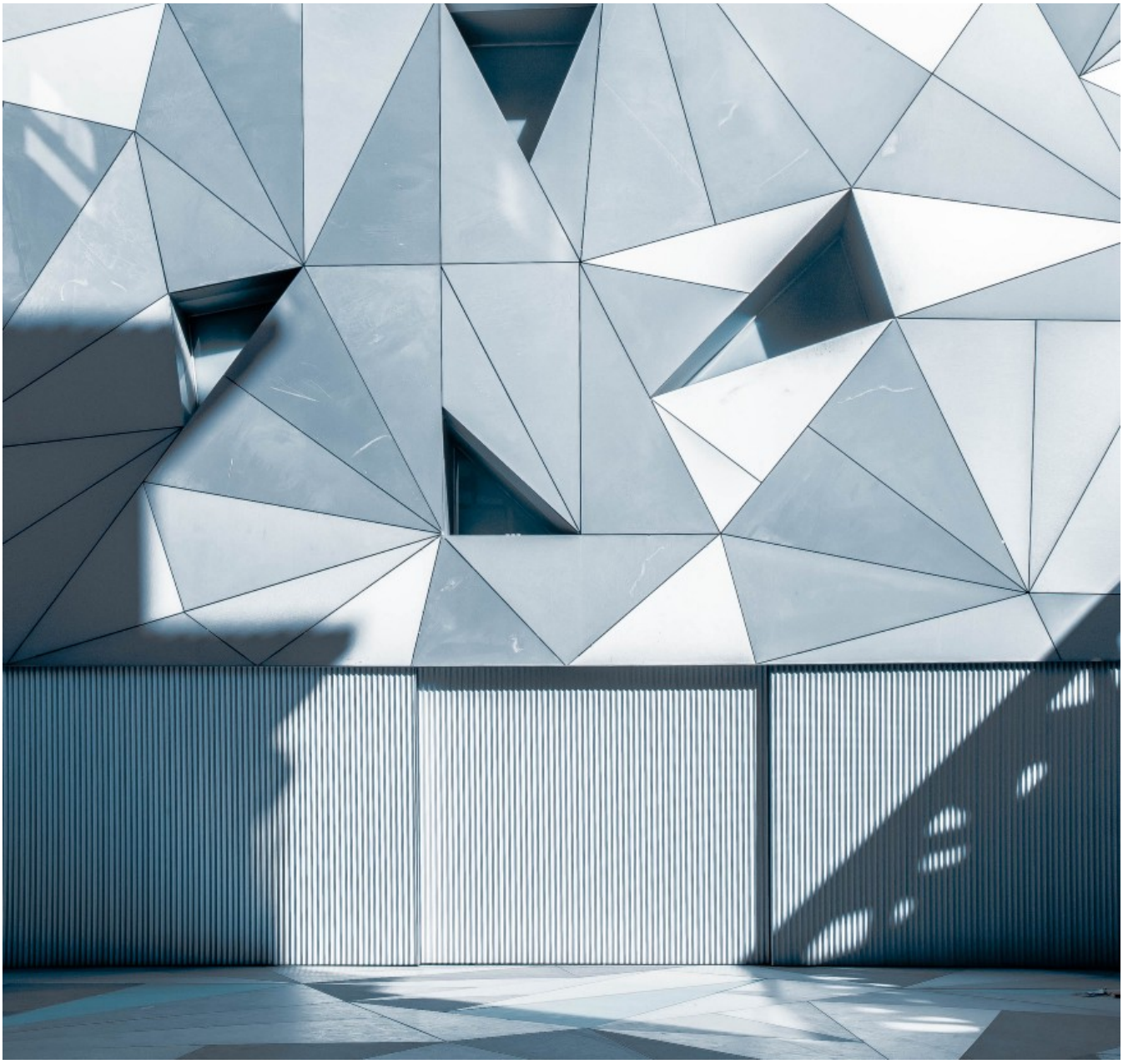
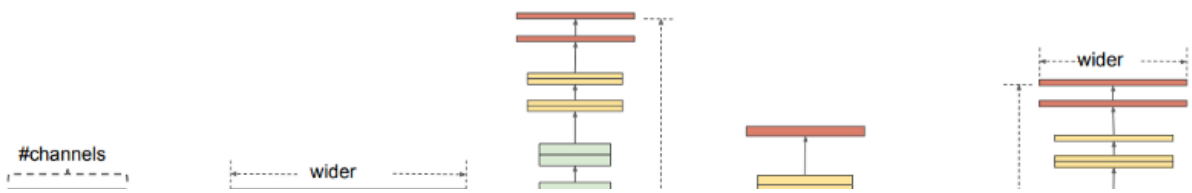
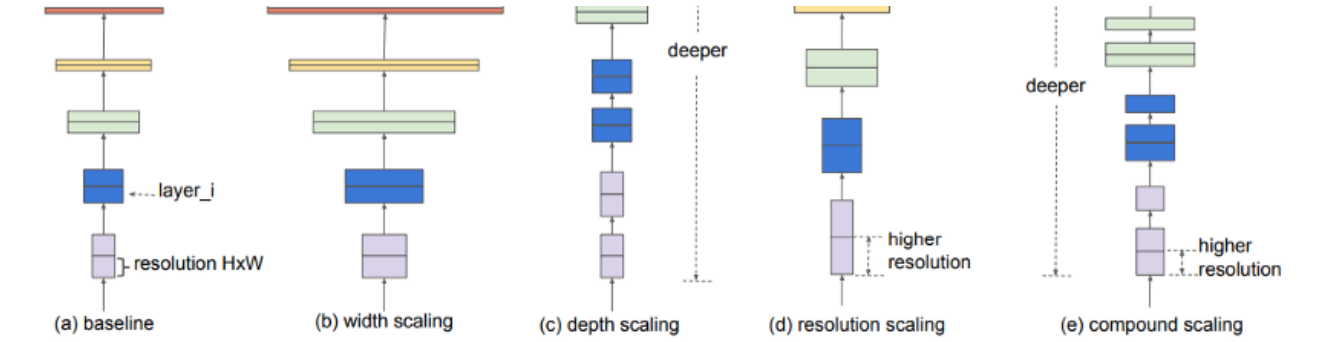


Photo by Joel Filipe on Unsplash

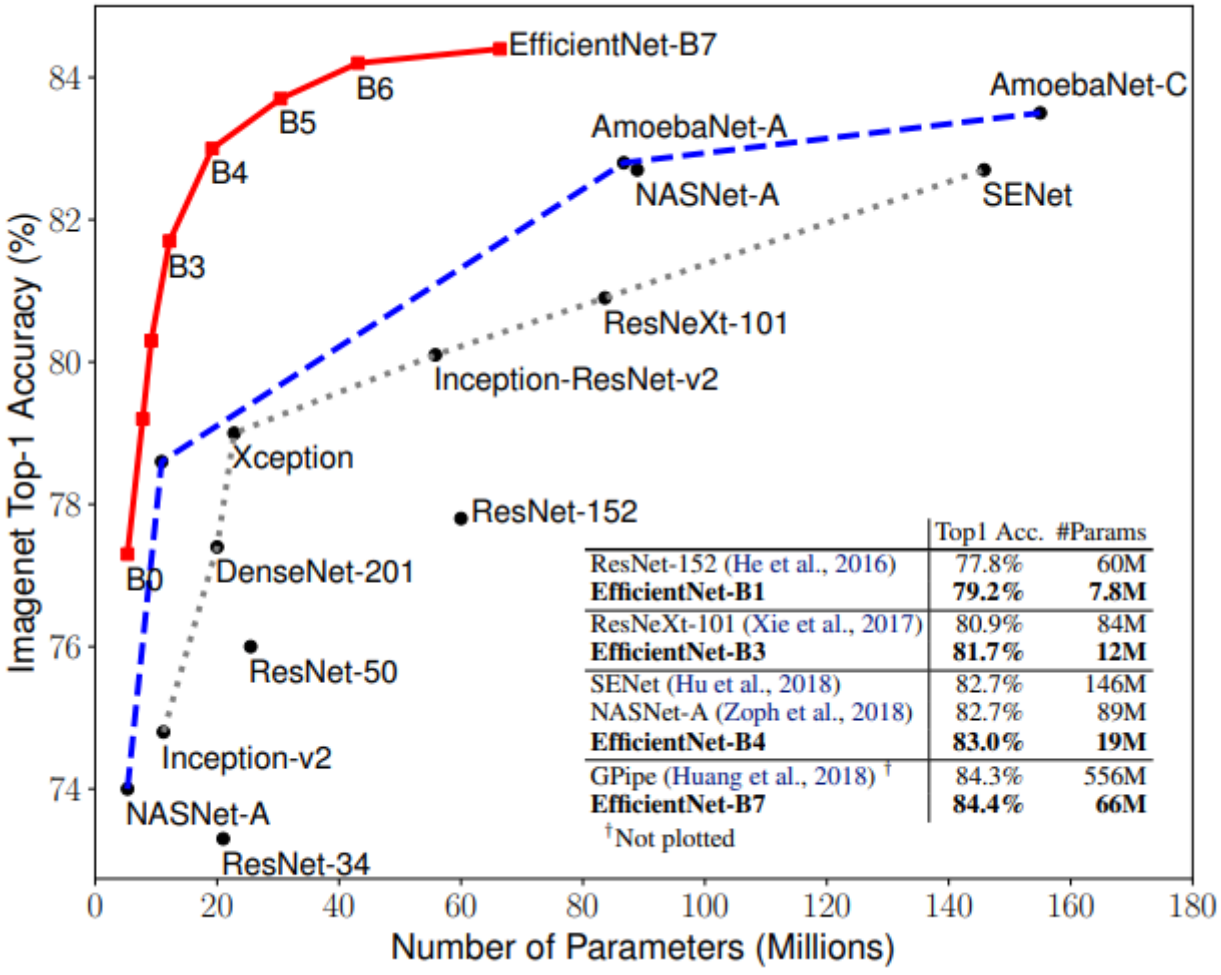
I was scrolling through notebooks in a Kaggle competition and found almost everyone was using EfficientNet as their backbone which I had not heard about till then. It is introduced by Google AI in this paper and they tried to propose a method that is more efficient as suggested by its name while improving the state of the art results. Generally, the models are made too wide, deep, or with a very high resolution. Increasing these characteristics helps the model initially but it quickly saturates and the model made just has more parameters and is therefore not efficient. In EfficientNet they are scaled in a more principled way i.e. gradually everything is increased.





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.

Did not understand what going on? Don't worry you will once you see the architecture. But first, let's see the results they got with this.



	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

Model Size Vs ImageNet accuracy

With considerably fewer numbers of parameters, the family of models are efficient and also provide better results. So now we have seen why these might become the standard pre-trained model but something's missing. I remember an article by Raimi Karim

where he showed the architectures of pre-trained models and that helped me a lot in understanding them and creating similar architectures.

Illustrated: 10 CNN Architectures

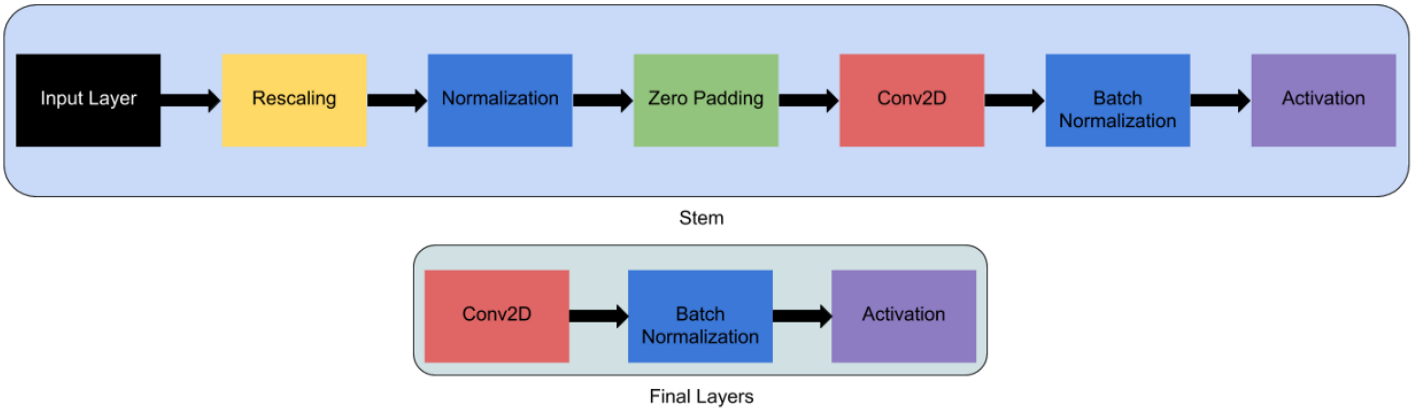
A compiled visualization of the common convolutional neural networks

towardsdatascience.com

As I could not find one like this on the net, I decided to understand it and create one for all of you.

Common Things In All

The first thing is any network is its stem after which all the experimenting with the architecture starts which is common in all the eight models and the final layers.



After this each of them contains 7 blocks. These blocks further have a varying number of sub-blocks whose number is increased as we move from EfficientNetB0 to EfficientNetB7. To have a look at the layers of the models in Colab write this code:

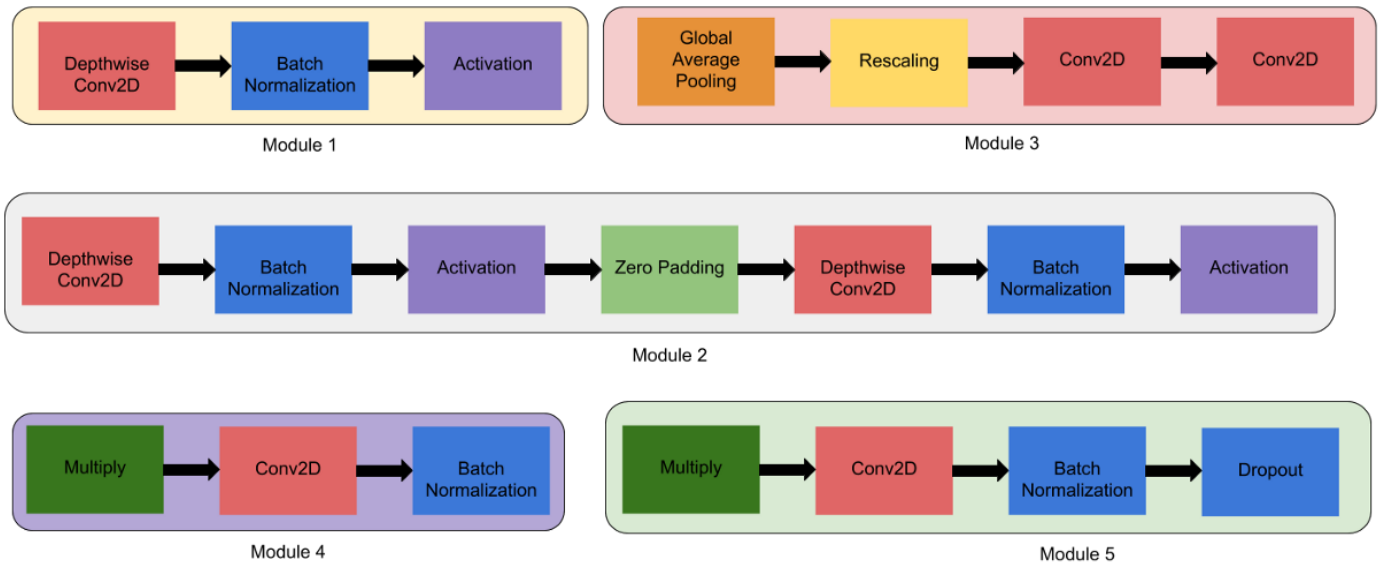
```
!pip install tf-nightly-gpu

import tensorflow as tf

IMG_SHAPE = (224, 224, 3)
model0 = tf.keras.applications.EfficientNetB0(input_shape=IMG_SHAPE,
include_top=False, weights="imagenet")

tf.keras.utils.plot_model(model0) # to draw and visualize
model0.summary() # to see the list of layers and parameters
```

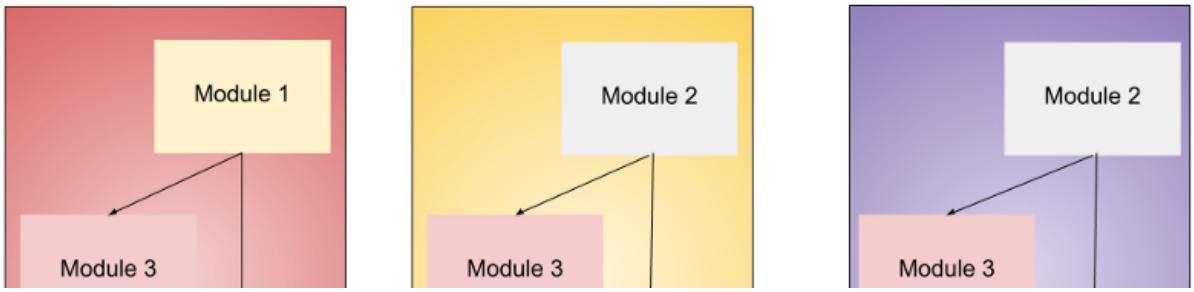
If you count the total number of layers in EfficientNet-B0 the total is 237 and in EfficientNet-B7 the total comes out to 813!! But don't worry all these layers can be made from 5 modules shown below and the stem above.

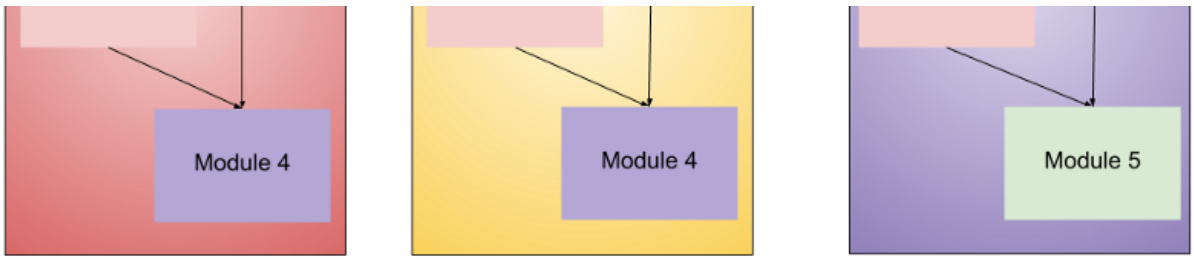


5 modules we will use to make the architecture.

- **Module 1** — This is used as a starting point for the sub-blocks.
- **Module 2** — This is used as a starting point for the first sub-block of all the 7 main blocks except the 1st one.
- **Module 3** — This is connected as a skip connection to all the sub-blocks.
- **Module 4** — This is used for combining the skip connection in the first sub-blocks.
- **Module 5** — Each sub-block is connected to its previous sub-block in a skip connection and they are combined using this module.

These modules are further combined to form sub-blocks which will be used in a certain way in the blocks.



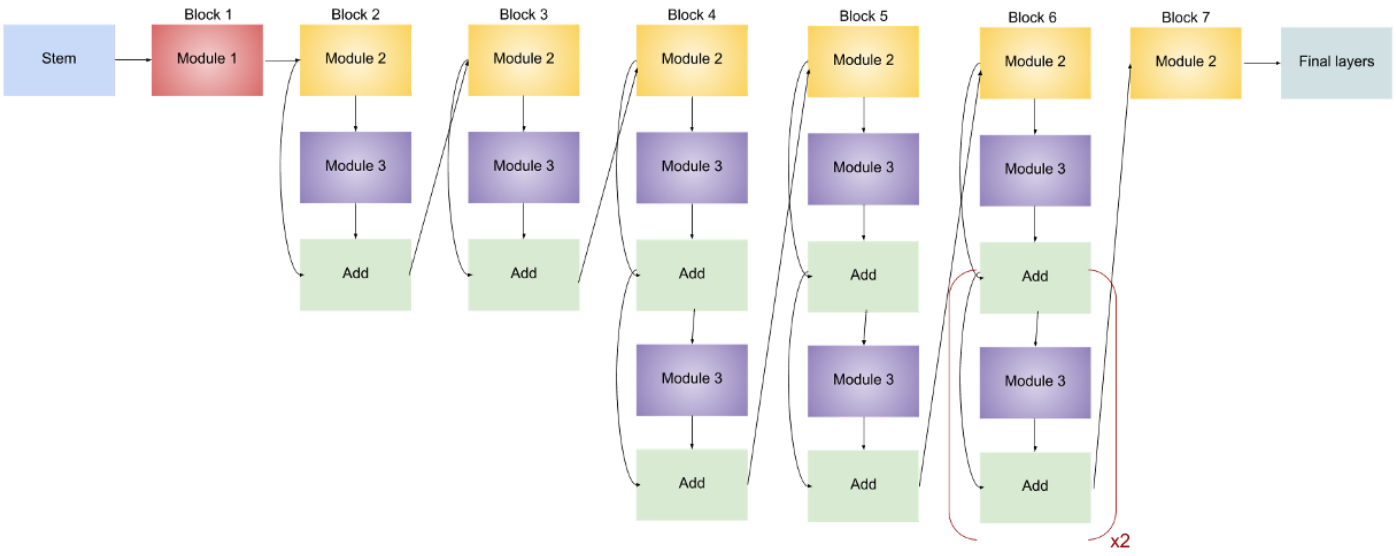


Sub-block 1 Sub-block 2 Sub-block 3

- **Sub-block 1** — This is used only used as the first sub-block in the first block.
- **Sub-block 2** — This is used as the first sub-block in all the other blocks.
- **Sub-block 3** — This is used for any sub-block except the first one in all the blocks.

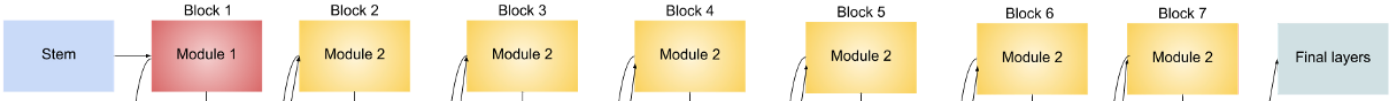
Till now we have specified everything that will be combined to create the EfficientNet models so let's get started.

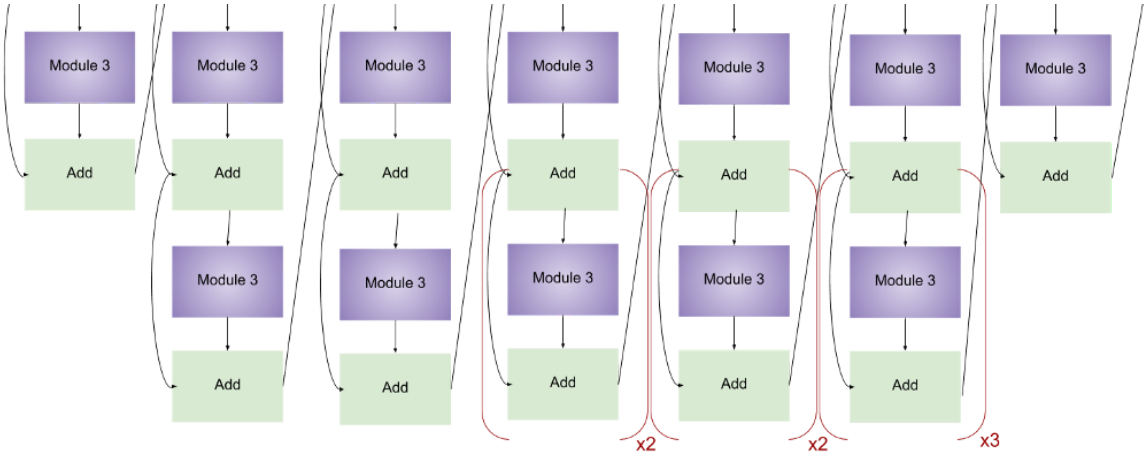
EfficientNet-B0



Architecture for EfficientNet-B0. (x2 means that modules inside the bracket are repeated twice)

EfficientNet-B1



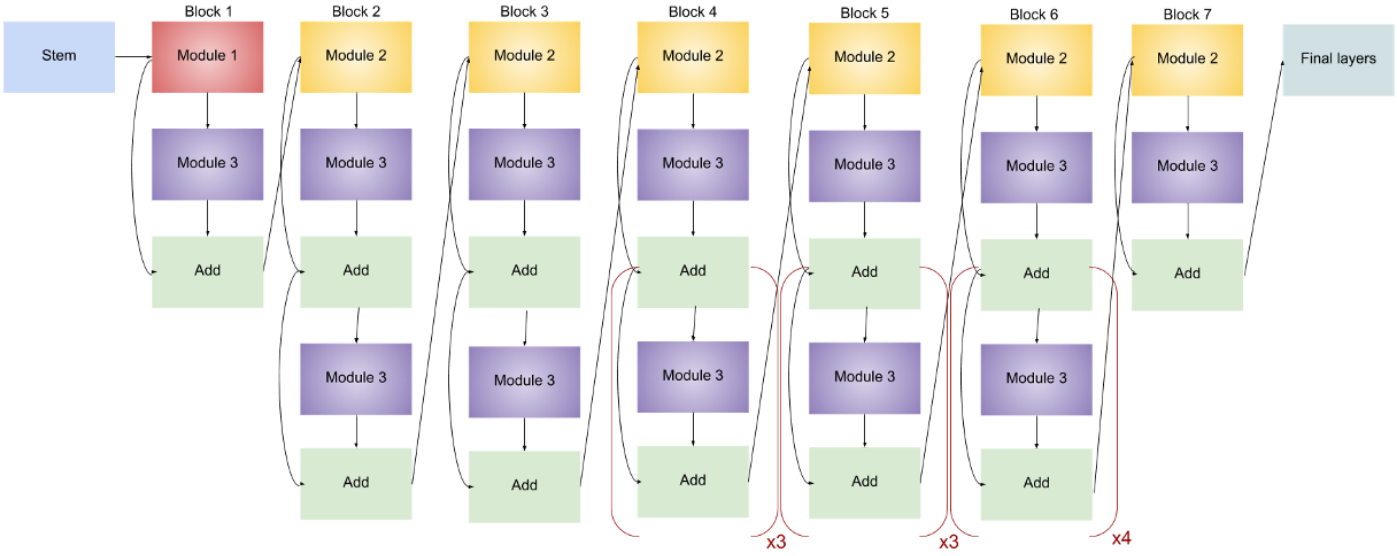


Architecture for EfficientNet-B1

EfficientNet-B2

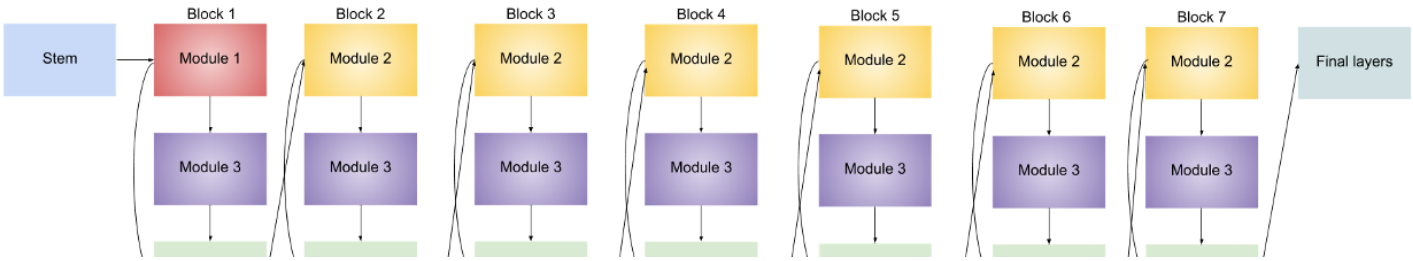
Its architecture is the same as the above model the only difference between them is that the number of feature maps (channels) is varied that increases the number of parameters.

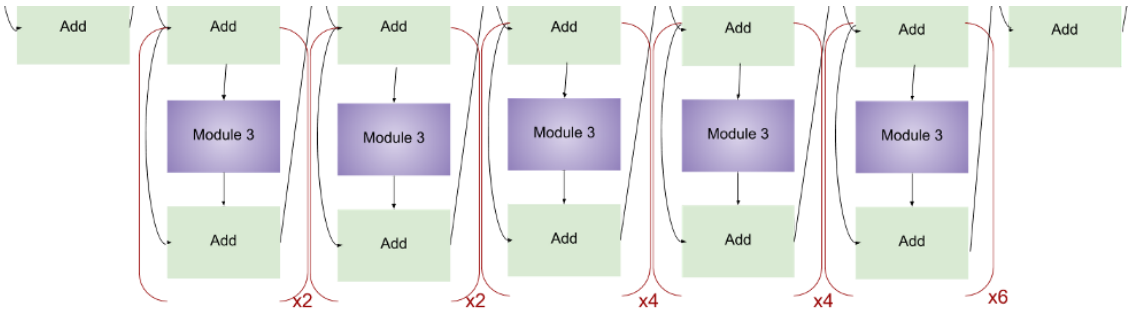
EfficientNet-B3



Architecture for EfficientNet-B3

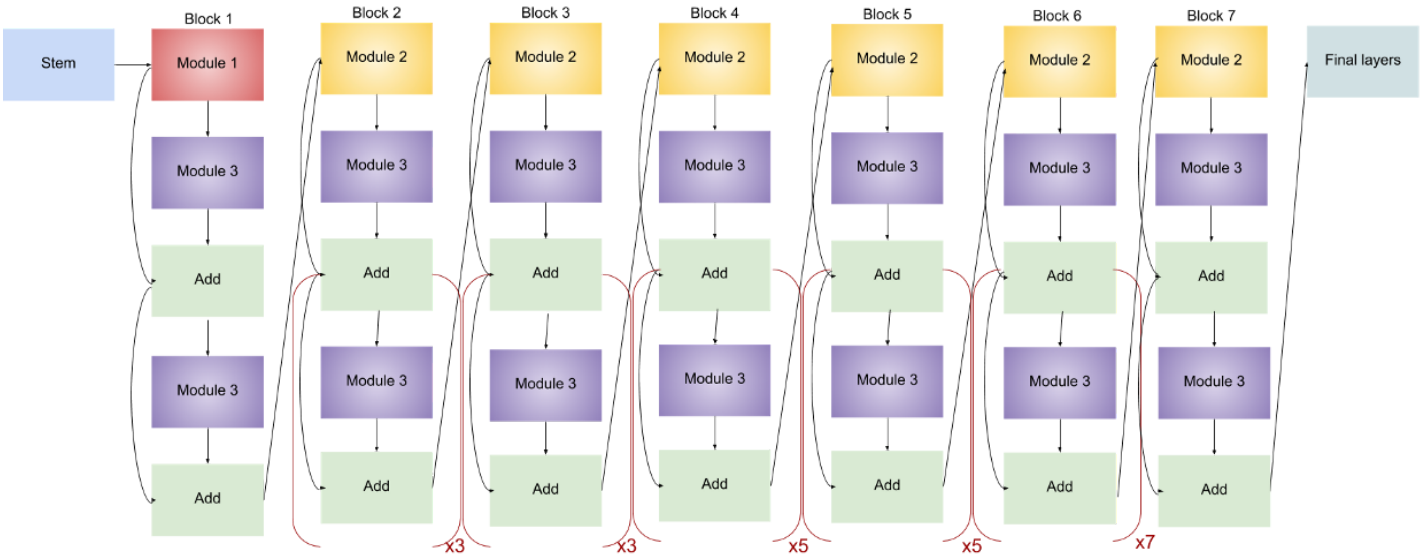
EfficientNet-B4





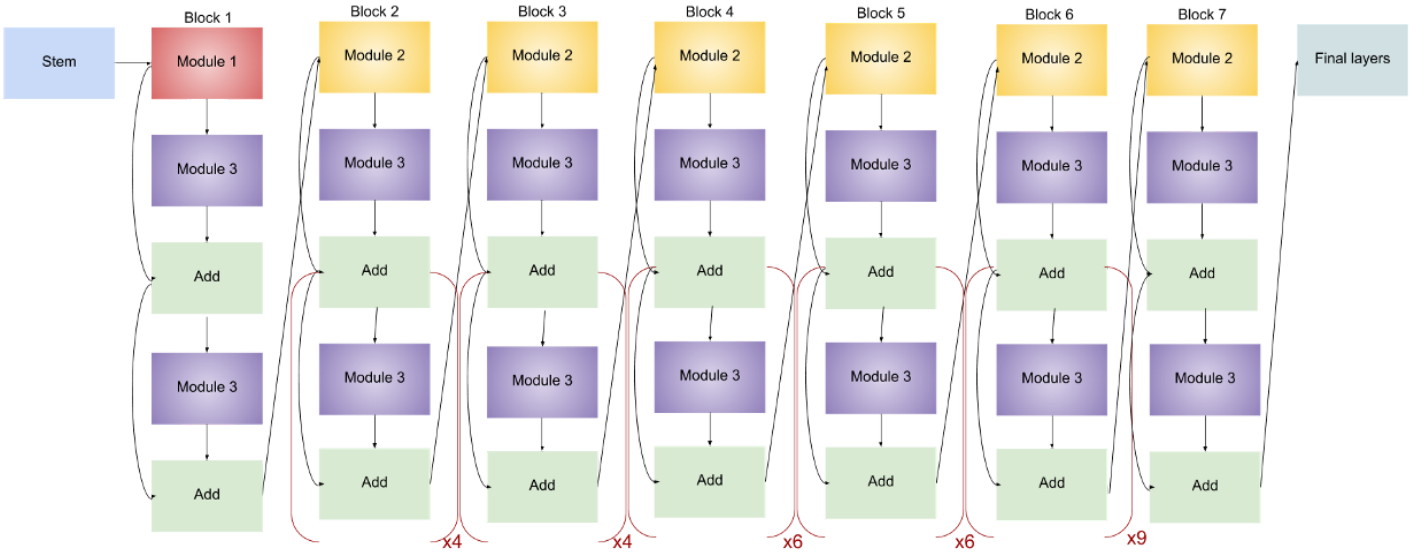
Architecture for EfficientNet-B4

EfficientNet-B5



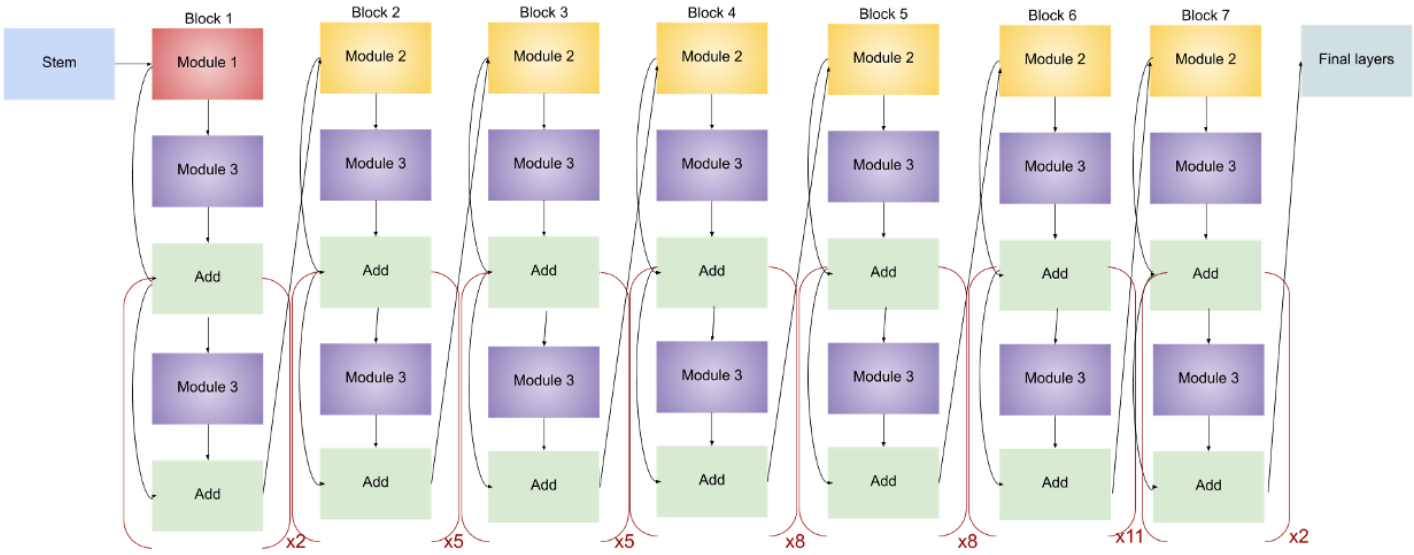
Architecture of EfficientNet-B5

EfficientNet-B6



Architecture of EfficientNet-B6

EfficientNet-B7



Architecture of EfficientNet-B7

It's easy to see the difference among all the models and they gradually increased the number of sub-blocks. If you understood the architectures I will encourage you to take any model and print its summary and have a go through it to know it more thoroughly. The table shown below denotes the kernel size for convolution operations along with the resolution, channels, and layers in EfficientNet-B0.

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

Kernel Size, resolution, channels, and no. of layers information.

This table was included in the original paper. The resolution remains the same as for the whole family. I don't know about whether the kernel size changes or remains the same so if anyone knows leave a reply. The number of layers is already shown above in the figures. The number of channels varies and it is calculated from the information seen from each model's summary and is presented below (*If you are using Mobile device then you will need to view it in landscape mode*.)

Stage	B1	B2	B3	B4	B5	B6	B7
1	32	32	40	48	48	56	64
2	16	16	24	24	24	32	32
3	24	24	32	32	40	40	48
4	40	48	48	56	64	72	80
5	80	88	96	112	128	144	160
6	112	120	136	160	176	200	224
7	192	208	232	272	304	344	384
8	320	352	384	448	512	576	640
9	1280	1408	1536	1792	2048	2304	2560

Medium does not has any format to make tables, so if you want to create tables like the one above you create ASCII tables from this site.

Before ending I have attached another image again from its research paper which shows its performance against the other state of the art techniques and also reduces the number of parameters and the number of FLOPS required.

Source

If you want to create advanced CNN architectures like this or had problems understanding any of the layers or terms used don't worry, I have written an article that will solve these problems.

Beyond the Standard CNN in Tensorflow 2

Generate deeper models with complex architectures and learn about different layers which makes the model better.

towardsdatascience.com

Do you want to see how EfficientNet stacks up against models on a Kaggle challenge you can check this article.

EfficientNet should be the goto pre-trained model or...

Compare the time and accuracy of different pre-trained models and finally create an ensemble to boost results.

towardsdatascience.com