Convolutional Neural Networks
State of the Art Model

Dr. Mongkol Ekpanyapong

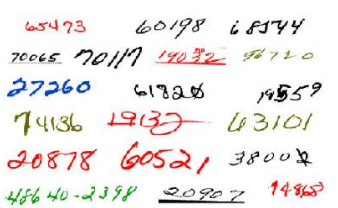
Timeline

1999	2012	2013	2014	2015	2016	2017	2018	2020
LeNet	AlexNet	ZFNet	GoogLeNet VGG	RESNet	DenseNet ShuffleNet	SqueezeExiteNet MobileNet	ResNeXt	Transformer

LeNet

- This is a pioneering seven-level convolutional network designed by Lecun in 1998
- It was developed for handwritten digit recognition for US zip codes
- It is also used by several banks for handwritten number classification on cheques
- After LeNet, researchers focused on ImageNet Challenge

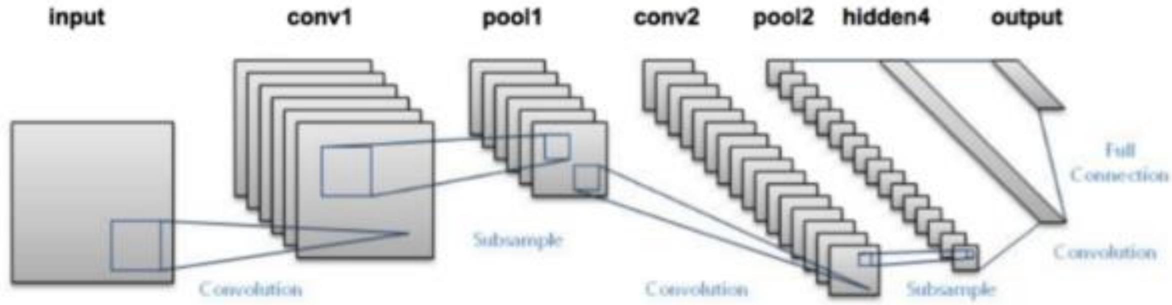
Zipcode Example



Examples of handwritten postal codes drawn from a database available from the US Postal service

LeNet

- Below is LeNet Architecture



Two conv. layers with max pol

First conv. layer has 6 filters size 5x5

Second conv. Layer has 16 filters size 5x5

After 2nd average pool, we flatten into 120 neurons and connection to first FC of 84 nodes and 10 nodes respectively

LeNet achieve 99.3% accuracy on MNIST

LeNet

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1 × 1	5 × 5	1	tanh
S4	Avg Pooling	16	5 × 5	2 × 2	2	tanh
C3	Convolution	16	10 × 10	5 × 5	1	tanh
S2	Avg Pooling	6	14 × 14	2 × 2	2	tanh
C1	Convolution	6	28 × 28	5 × 5	1	tanh
In	Input	1	32 × 32	–	–	–

Link to Lecun website

http://yann.lecun.com/exdb/lenet/

Try this:

- Implement LeNet using Keras

AlexNet

- AlexNet was introduced in 2012 by Alex Krizhevsky, Iiya Sutskever and Geoffrey Hinton from University of Toronto

- In 2012, AlexNet outperformed all the prior competitors and won ILSRVC by reducing top-5 error to 15.3% (84.7% accuracy), compared to the runner-up with 26%

- It has more filters per layer and deeper than LeNet

AlexNet

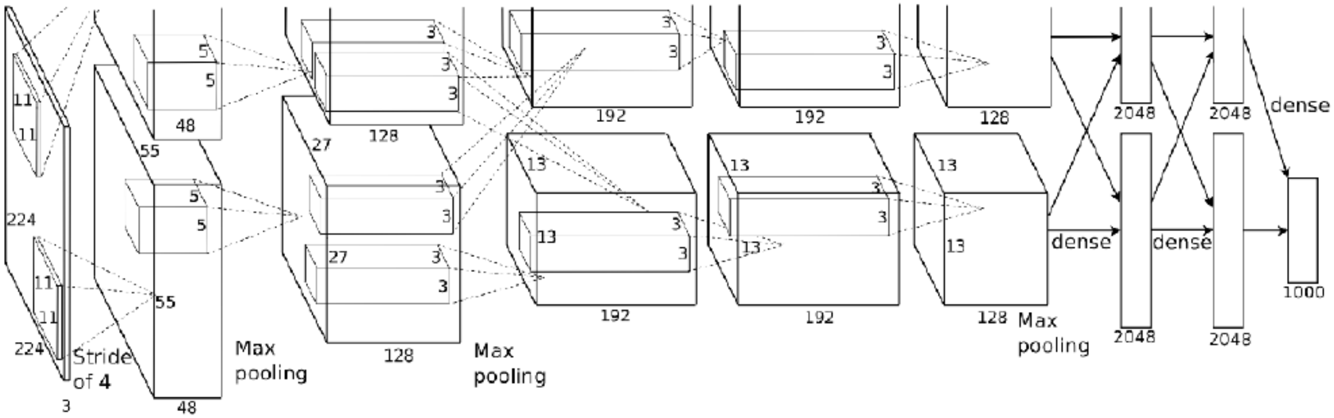
- It contains 8 layers with first five being Convolutional Layers and the last 3 being FC layers
- It has over 60 Million parameters and was trained on two GPUs for over a week
- AlexNet introduces the use of stacked convolution instead of alternative convolution pooling
- A stack of small convolutions is better because it introduces more non-linearities and fewer parameters

AlexNet

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13 × 13	3 × 3	1	SAME	ReLU
C6	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
C5	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
S4	Max Pooling	256	13 × 13	3 × 3	2	VALID	–
C3	Convolution	256	27 × 27	5 × 5	1	SAME	ReLU
S2	Max Pooling	96	27 × 27	3 × 3	2	VALID	–
C1	Convolution	96	55 × 55	11 × 11	4	SAME	ReLU
In	Input	3 (RGB)	224 × 224	–	–	–	–

Image from Hands-on Machine Learning with SciKit-Learn and TensorFlow

AlexNet Architecture



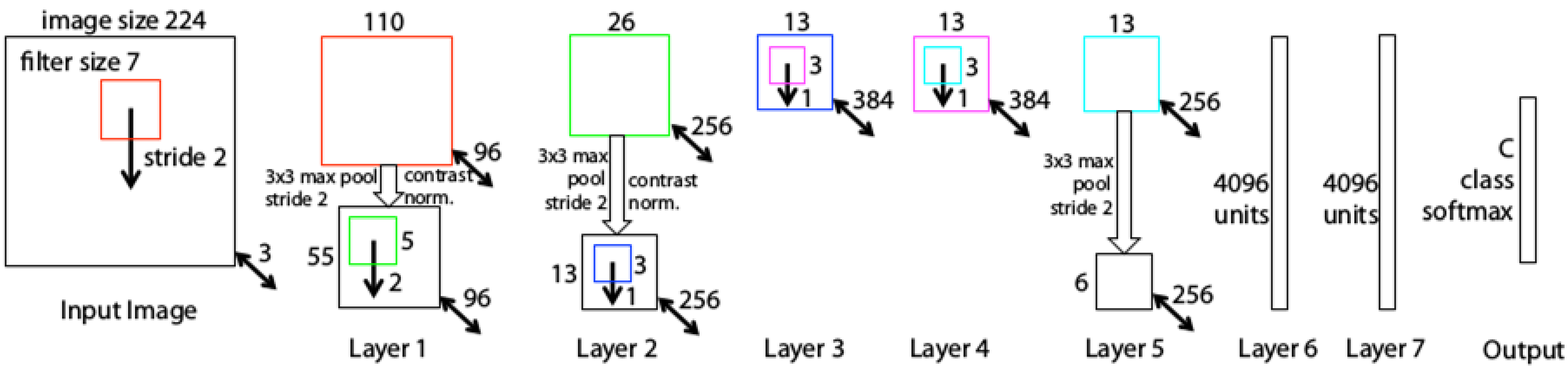
Homework

- Implement Alexnet on your preferred dataset

ZFNet

- ZFNet is the winner of ILSVRC2013
- It is expanding the size of middle convolution layers and making the stride and filter size on the first layer smaller
- It goes from 11x11 stride 4 in AlexNet to 7x7 stride 2 in ZFNet
- The intuition is that a smaller filter size helps to retain original pixel information

ZFNet



ZF Net Architecture

VGGNet

- It is developed by Oxford Visual Geometry Group (VGG for short)
- It uses only 3x3 convolutional layers stacked on top of each other
- At the end, two fully connected layers with 4096 nodes are followed by a softmax layer
- Trained on 4 Nvidia Titan GPUs for three weeks

VGG Net

- It achieved 92.7% top-5 accuracy (1,000 classes) (7.3% error rate)

- VGG16 has 13 Conv. Layers with 3FC layers

- VGG19 has 16 Conv. Layers with 3 FC layers

VGGNet

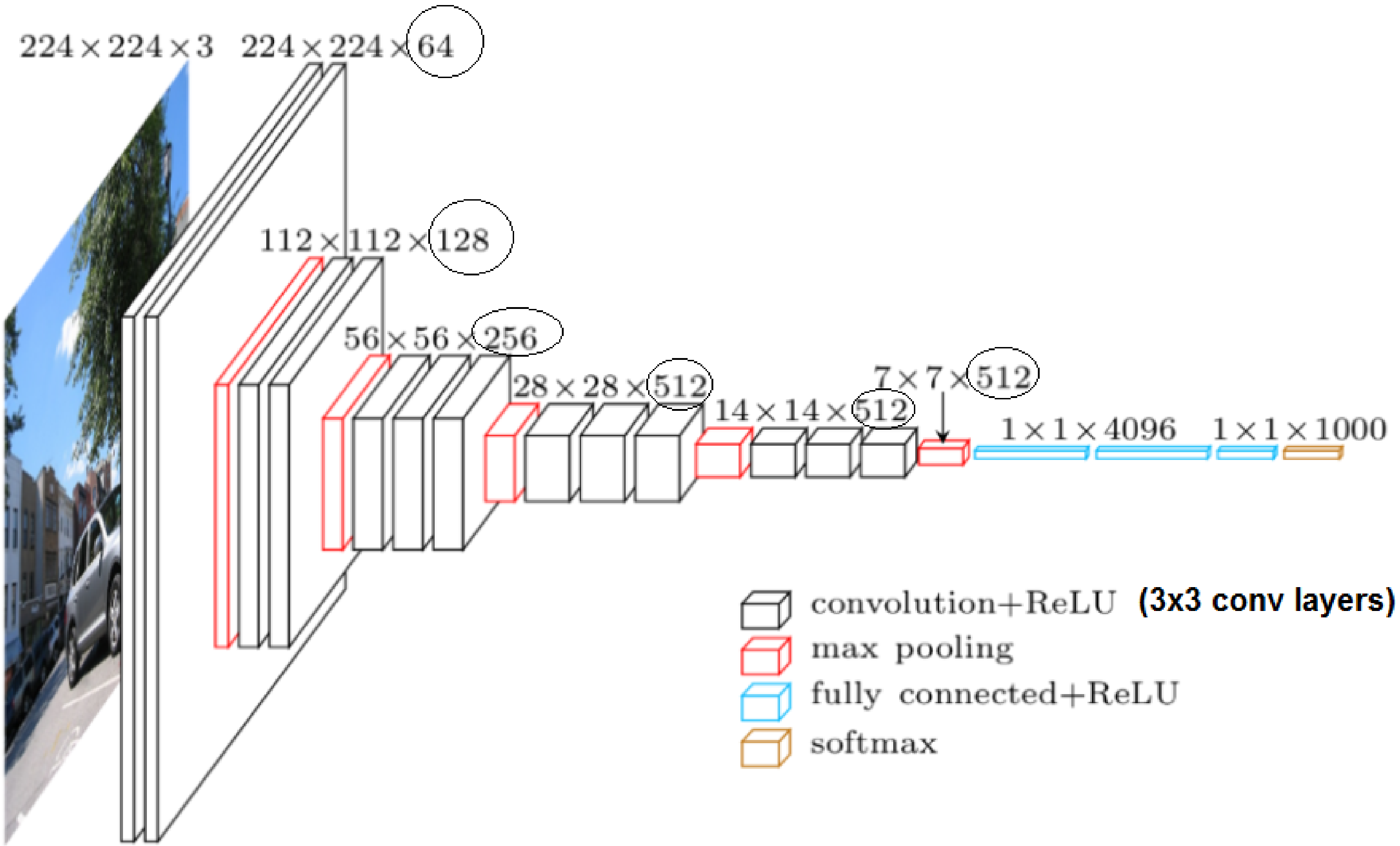
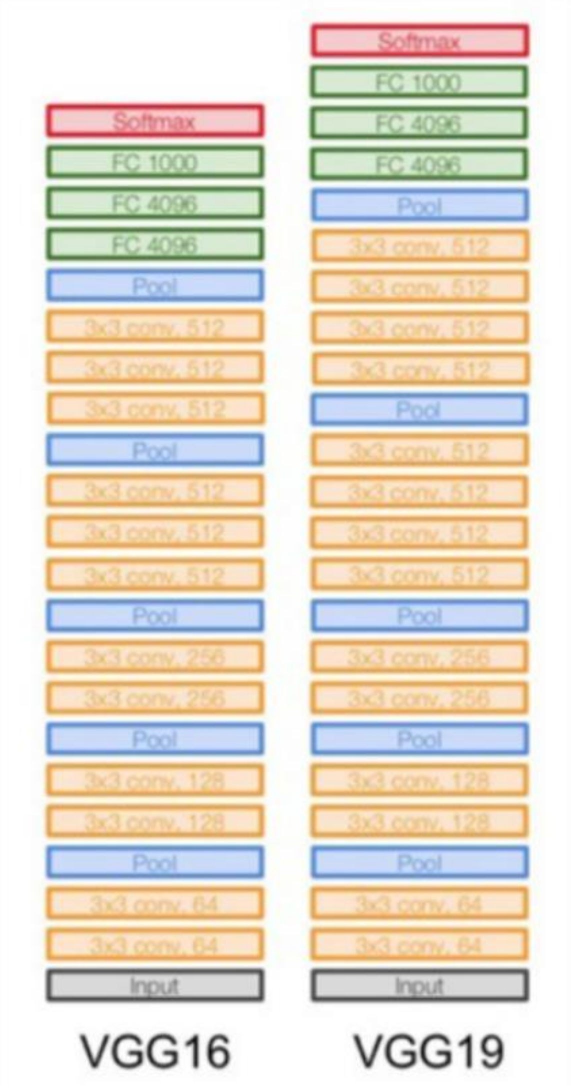


Image from Hands-on Transfer Learning with Python

VGG Net

- VGG follows classical CNN approach

- Multiple FC layers then output with a softmax layer



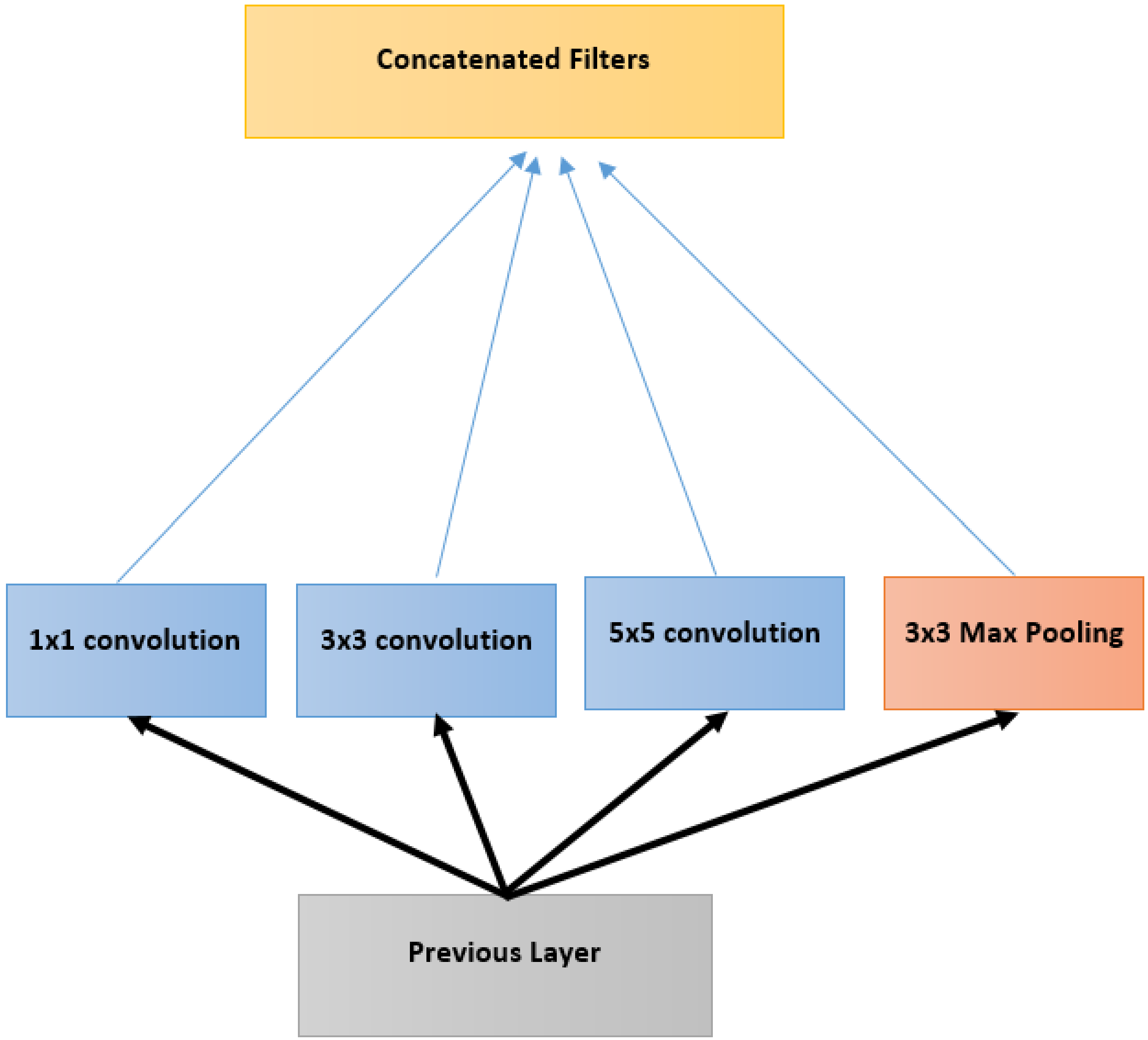
GoogLeNet

- ILSVRC 2014 winner is GoogLeNet from Google
- It achieved a top-5 error rate of 6.67% (close to human performance 93.3% accuracy)
- The runner up was VGGNet
- Google introduced a new component called the inception layer
- 22 layers in total
- Uses 12x fewer parameters than AlexNet
- Trained on 2-3 high-end GPUs for 1 week

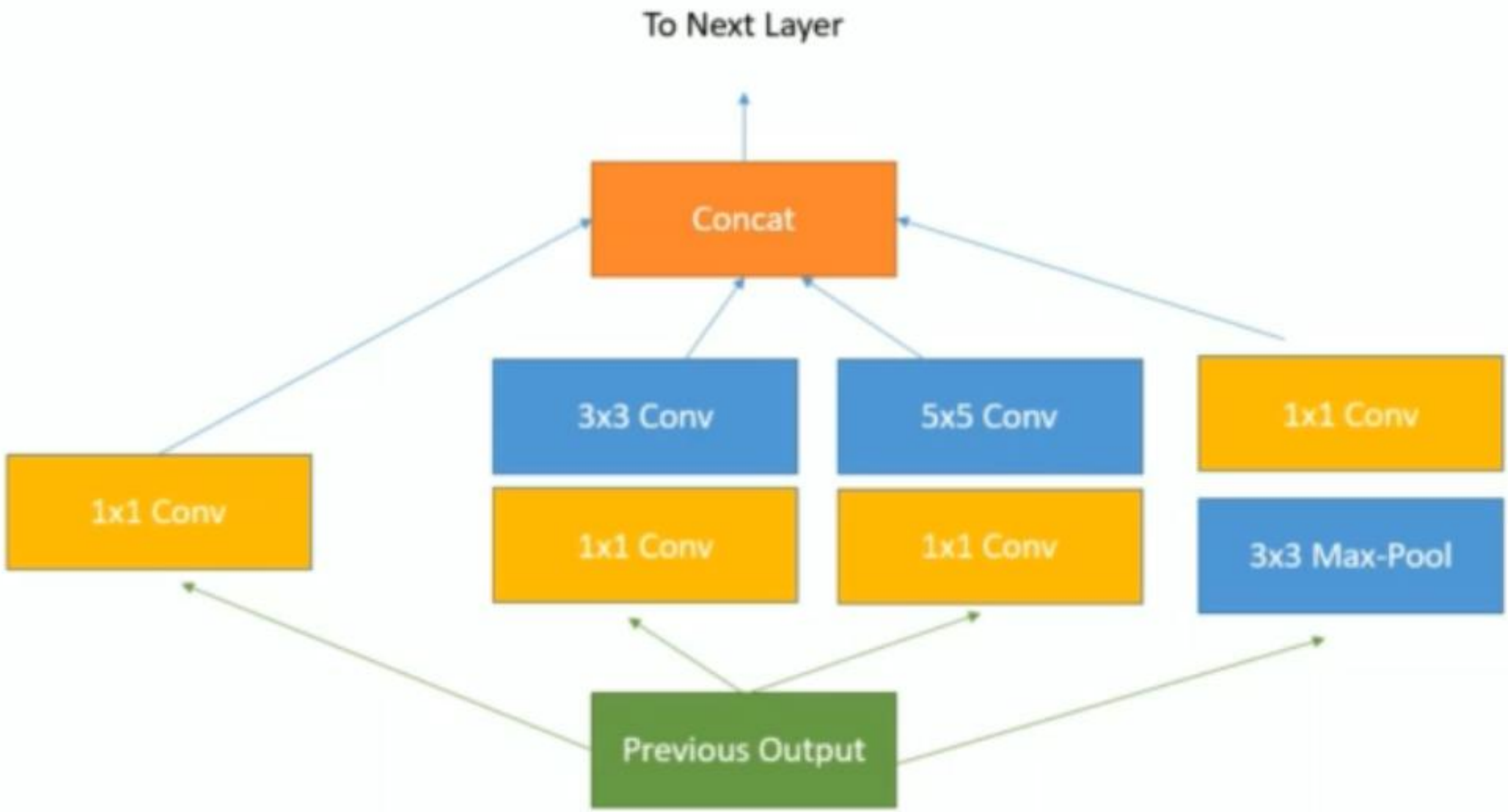
Inception Layer

- It has many sequential network operating in parallel starting from 1x1, 3x3 to 5x5
- The outputs are concatenated to produce the next layer
- The 1x1 convolution reduces the depth of the convolution layers

Network in Network

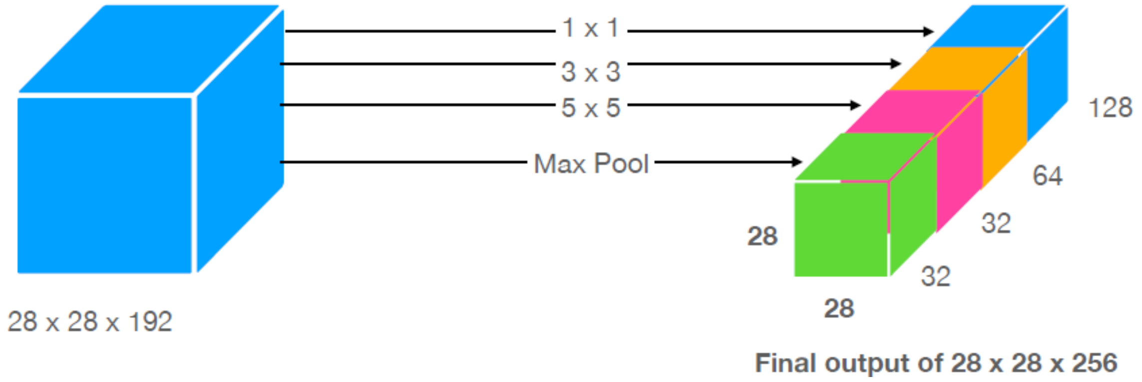


Using 1x1 for dimensionality reduction



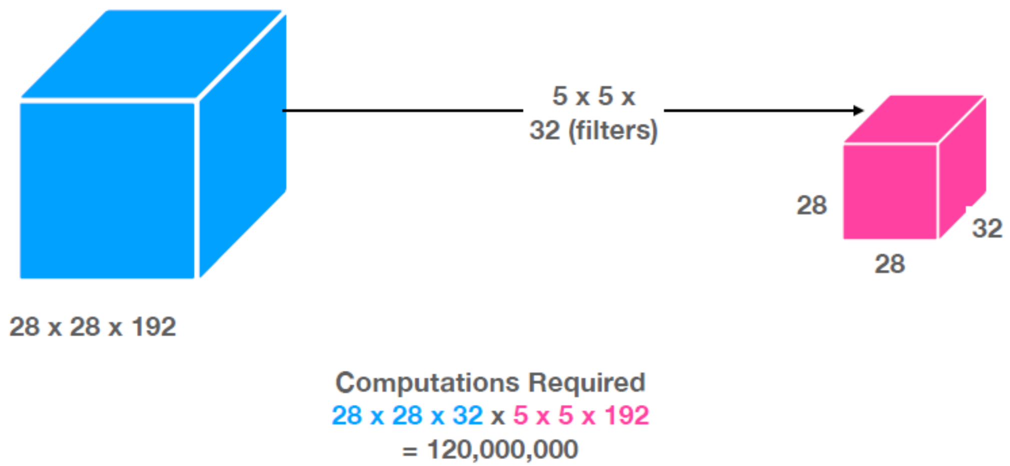
Inception Network

- It allows to use several convolution filters
- Use ‘same’ padding (zero) and stride of 1
- Use Max pooling with stride of 1

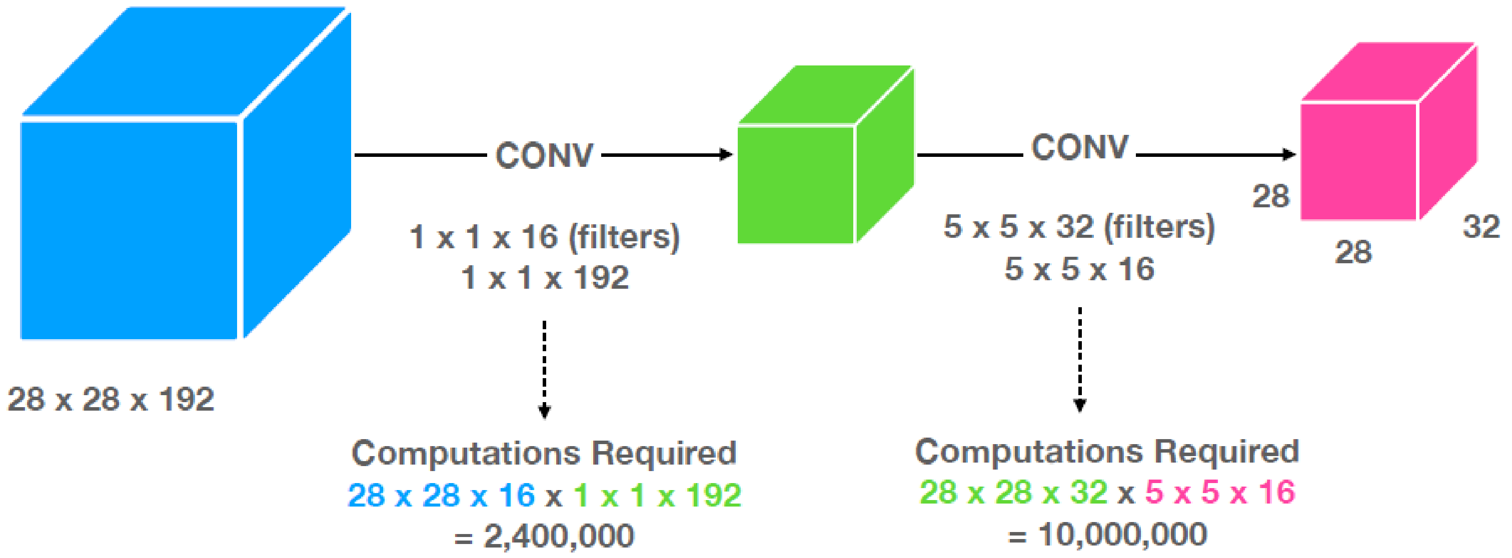


Inception Network's advantage

- It helps reduce the number of computation (number of calculation)

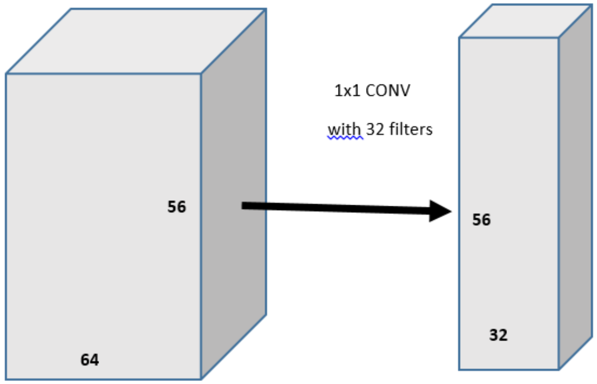


With Inception Layer

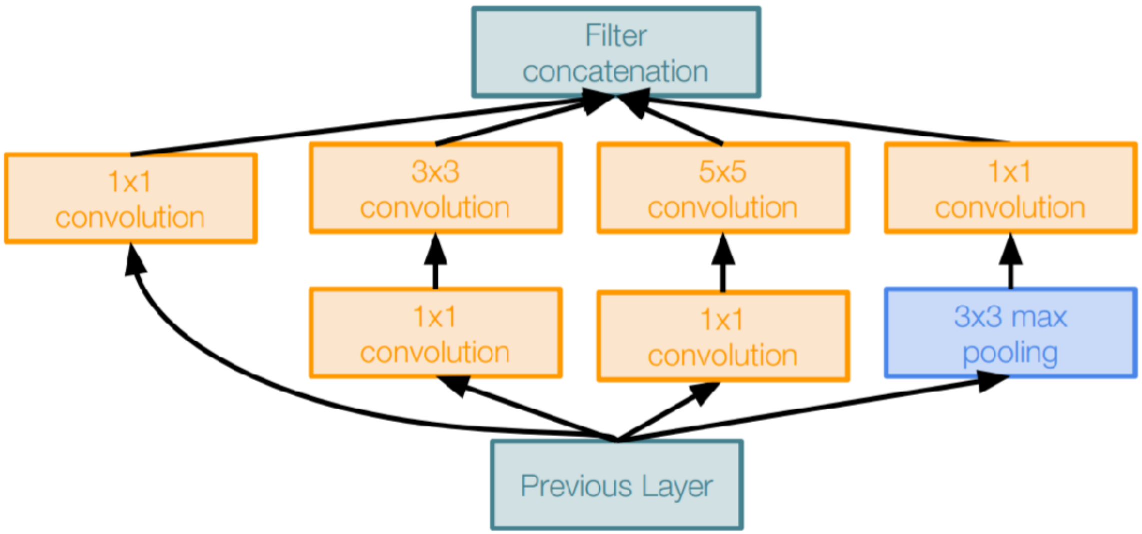


- The new computation is only 12.4M calculation comparing with 120M previously

Dimension Reduction



Dimension reduction by 1x1 Conv



Inception module with dimension reduction

Inception Block

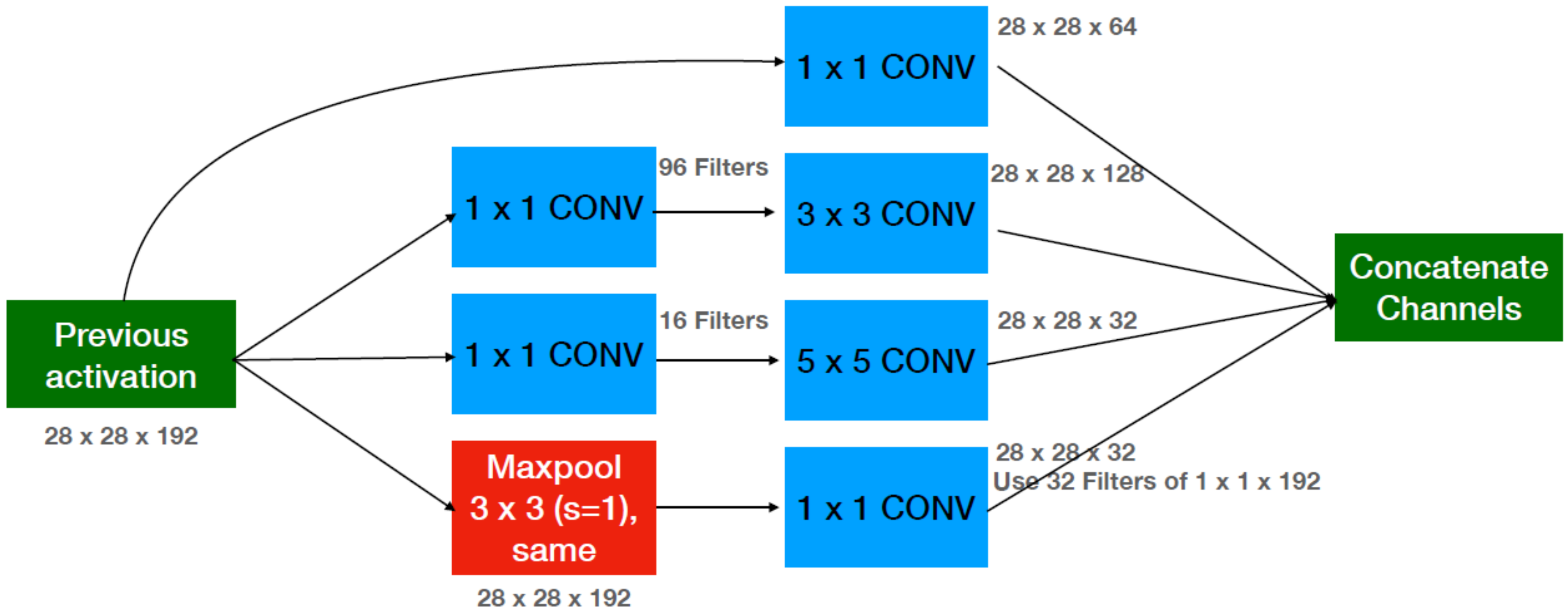
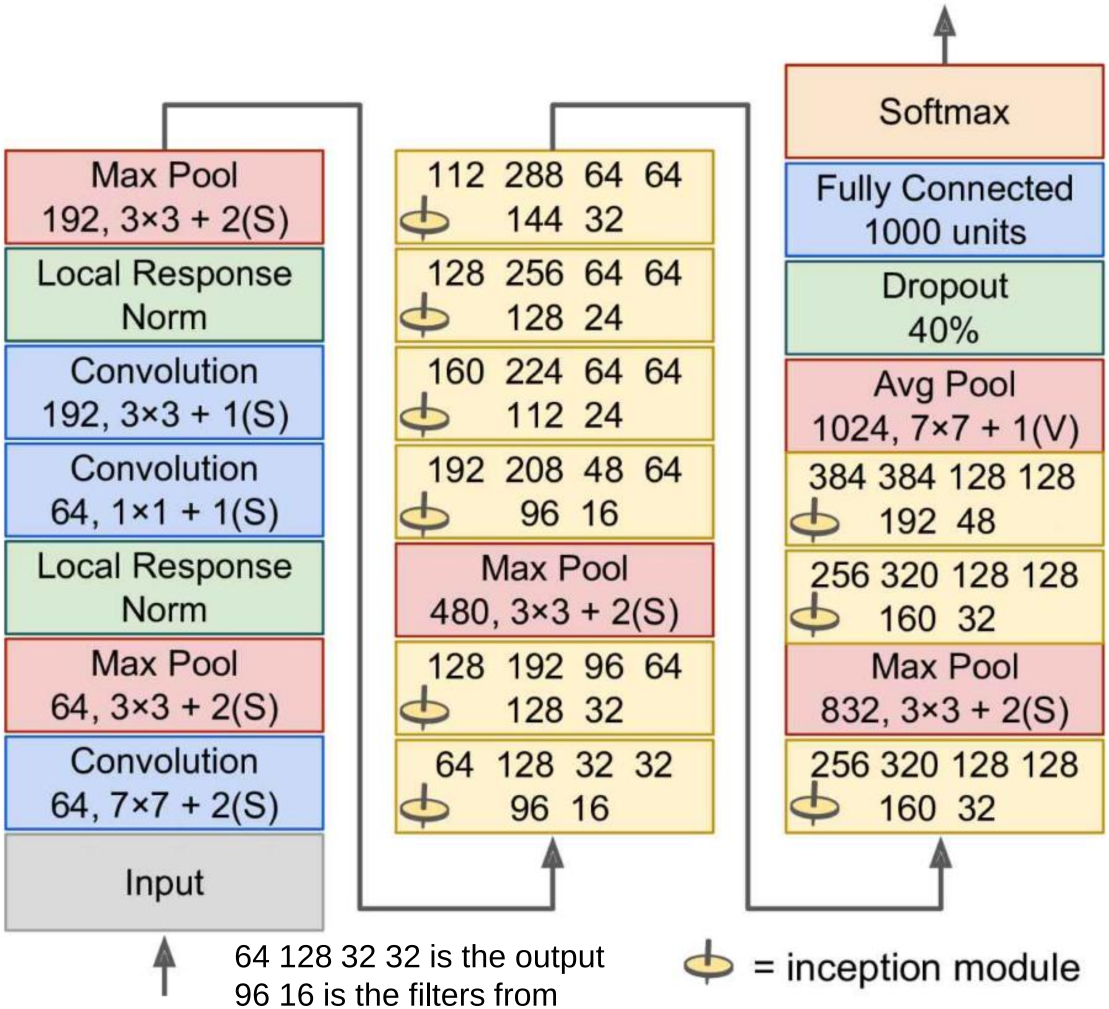


Image from Hands-on Machine Learning with SciKit-Learn and TensorFlow

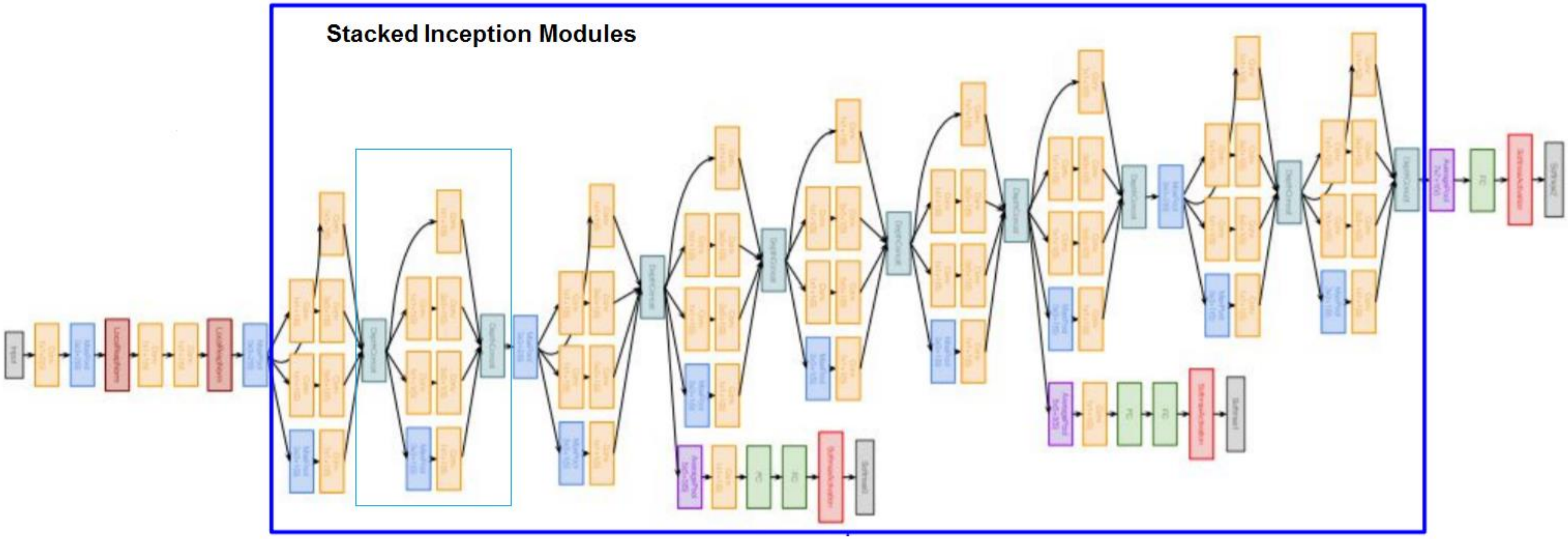
GoogleNet



64 128 32 32 is the output
96 16 is the filters from
previous inception block

Image from Hands-on Machine Learning with SciKit-Learn and TensorFlow

Complete GoogleNet



History of Inception Network

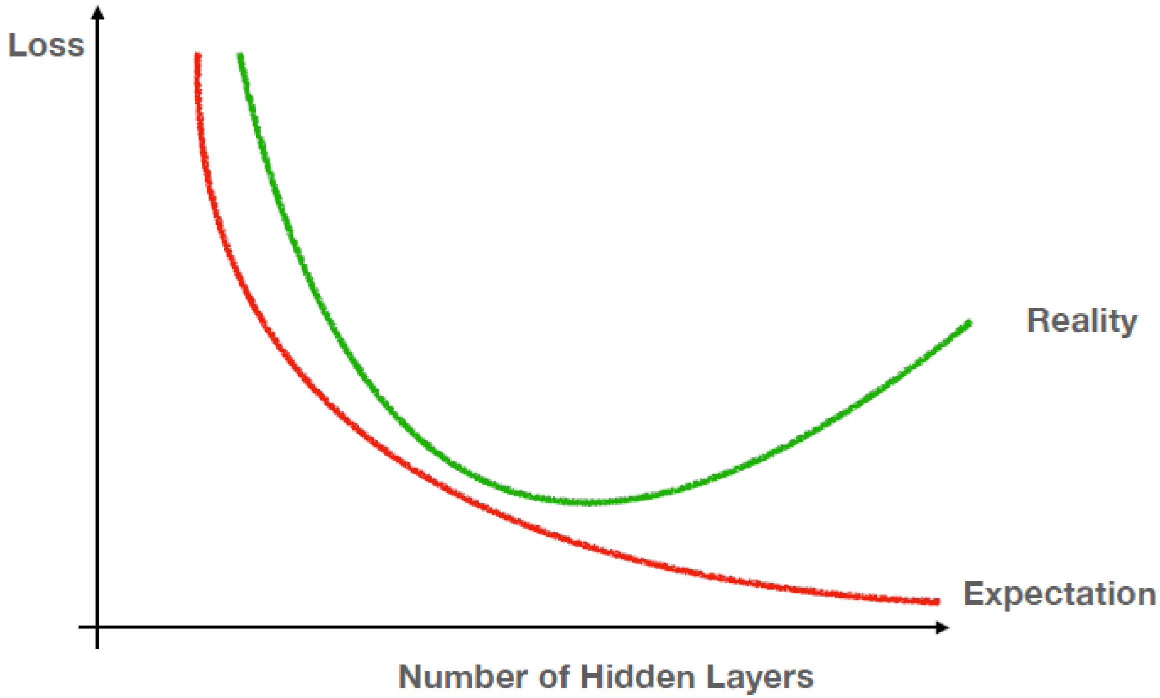
- **InceptionV1** - 2014 - Network in Network
  - https://arxiv.org/pdf/1312.4400v3.pdf
- **InceptionV2**- Going Deeper with Convolutions - 2014
  - https://arxiv.org/pdf/1409.4842v1.pdf
- **InceptionV3** - Rethinking the inception architecture for computer vision - 2015
  - https://arxiv.org/pdf/1512.00567v3.pdf
- **InceptionV4** - v4: Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, Szegedy et al. (2016)
  - https://arxiv.org/pdf/1602.07261v2.pdf

Residual Neural Network

- In ILSVRC 2015, a novel CNN architecture with skip connections and batch normalization was introduced by Microsoft Research Asia called Residual Neural Network (ResNet)
- They are successful to train a network with 152 layers (8 times deeper than VGG)
- It achieves a top-5 error rate of 3.57% (96.43% accuracy)
- Address Vanishing Gradient Problem

Classical CNN Performance on Deep Layers

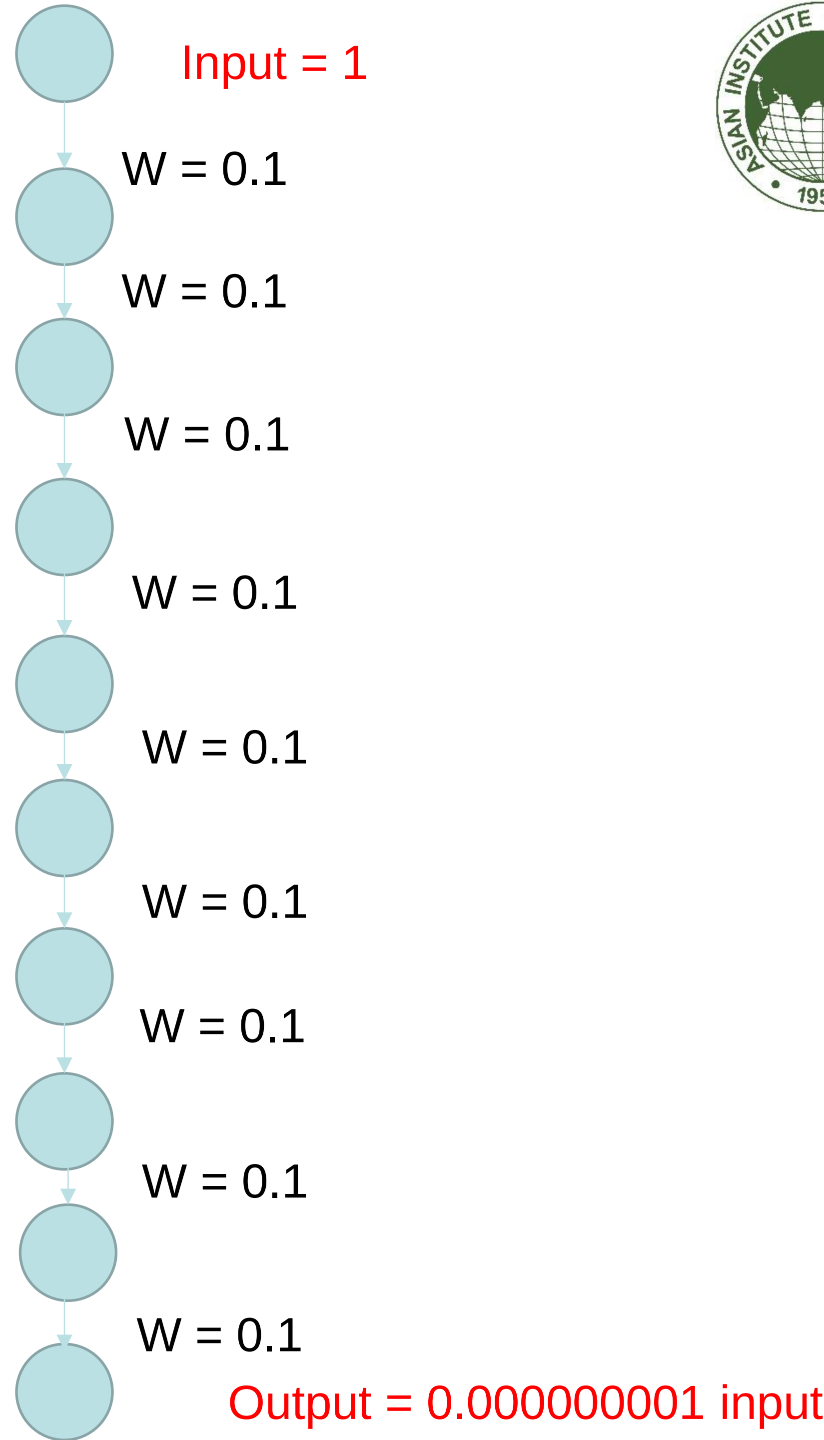
- A Classical CNN has lots of problems training if layers are very deep



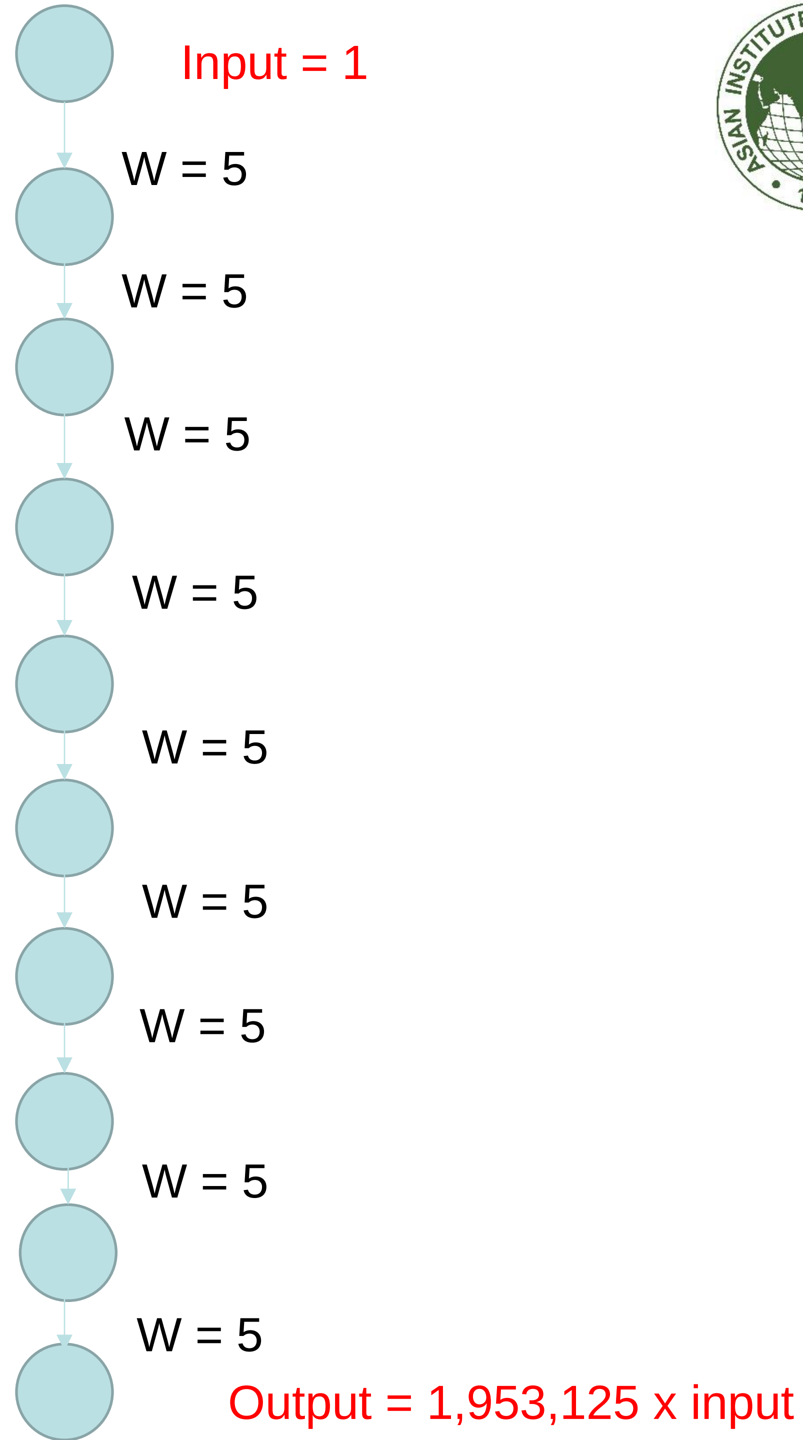
Exploding and Vanishing Gradients problem

- In deep network with N layers, N derivatives must be multiplied together
- If derivatives are large, gradient could explode
- If derivatives are small, gradient could vanish

Vanishing gradient

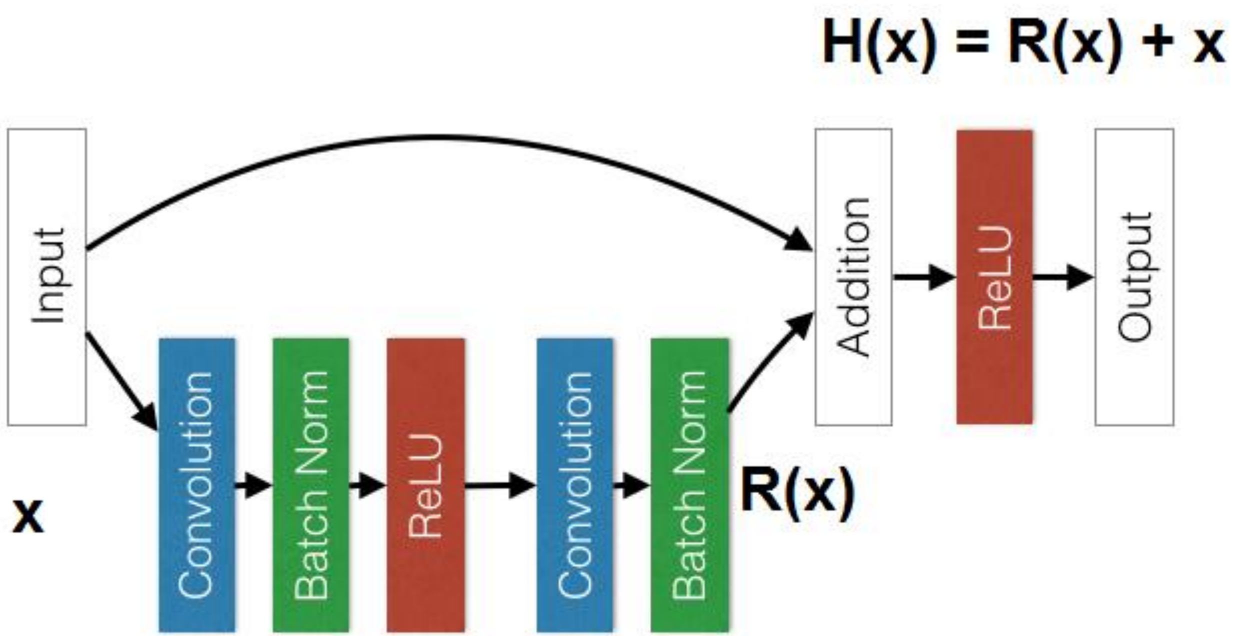


Exploding gradient



ResNet

- The convolutional stacked layers is called the residual R(x)
- They define a desired underlying mapping H(x) = R(x) + x



ResNet

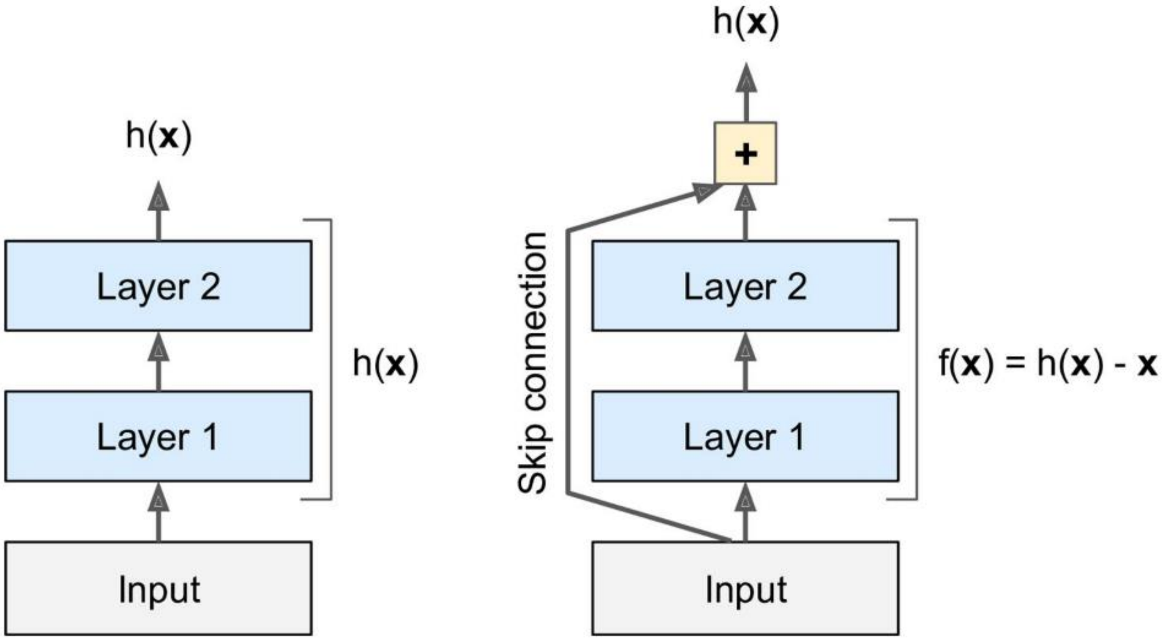


Image from Hands-on Machine Learning with SciKit-Learn and TensorFlow

ResNet

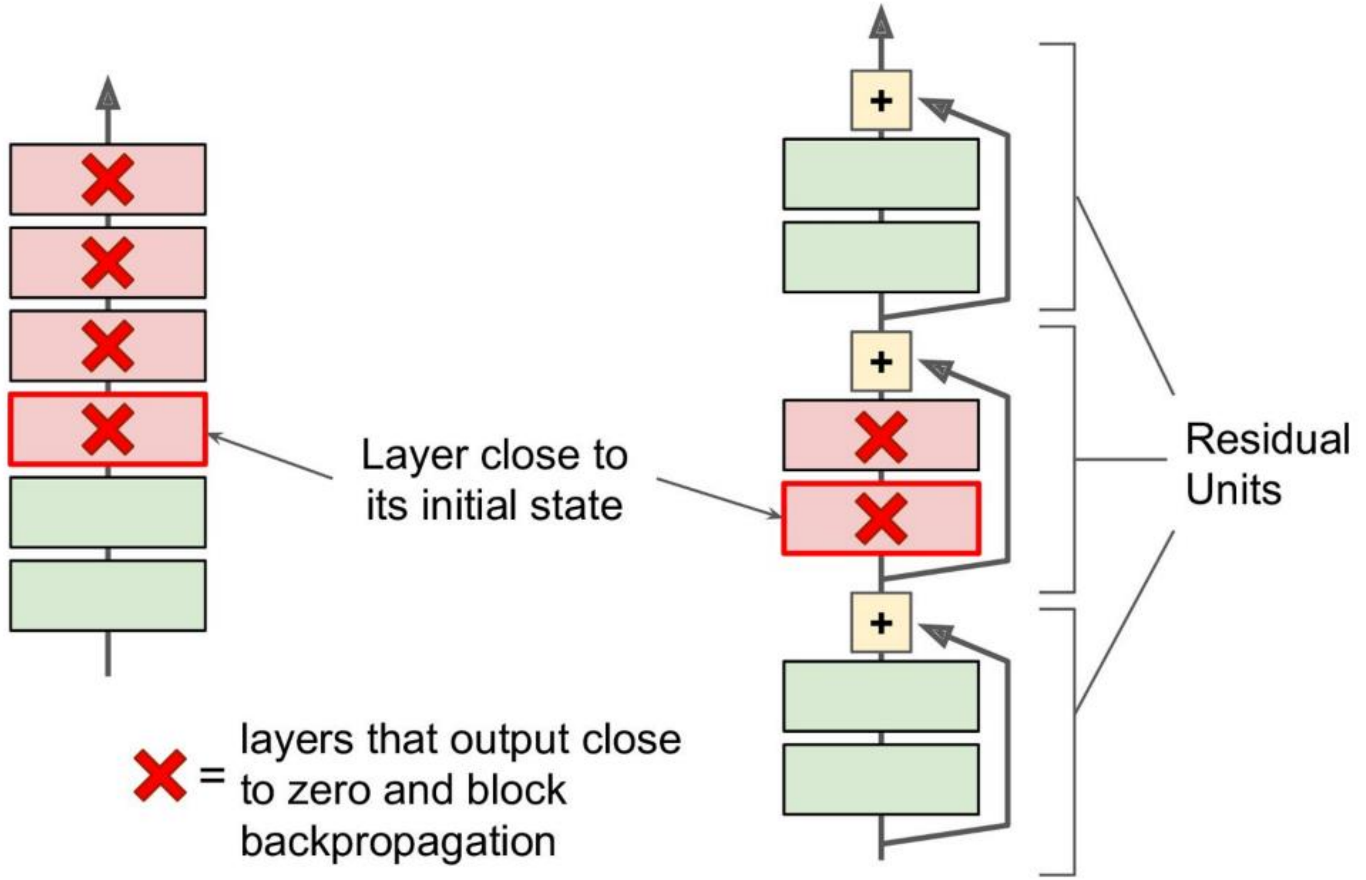
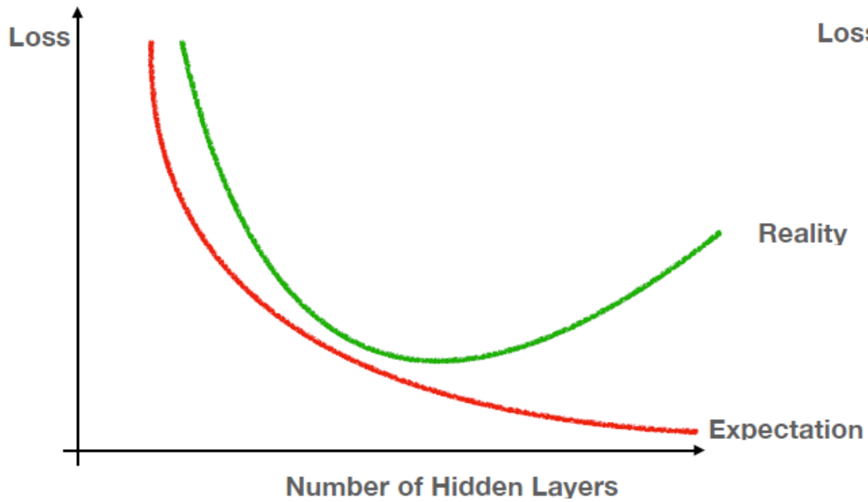
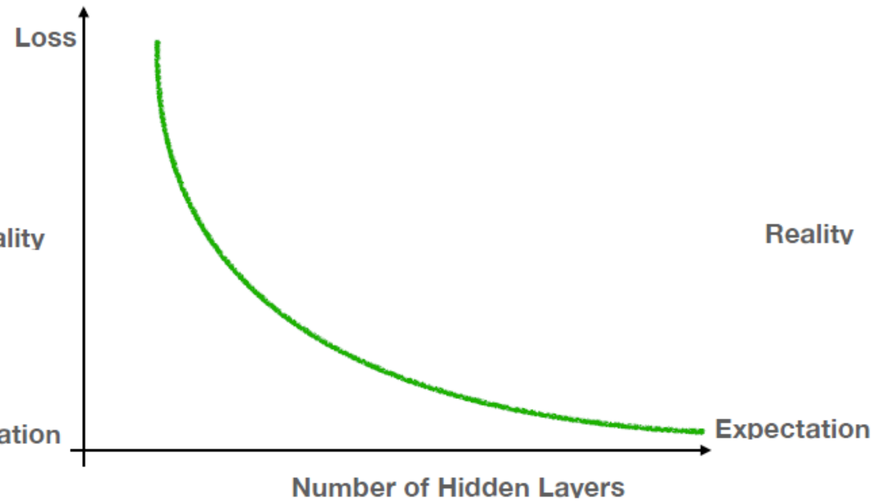


Image from Hands-on Machine Learning with SciKit-Learn and TensorFlow

ResNet Performance





ResNet

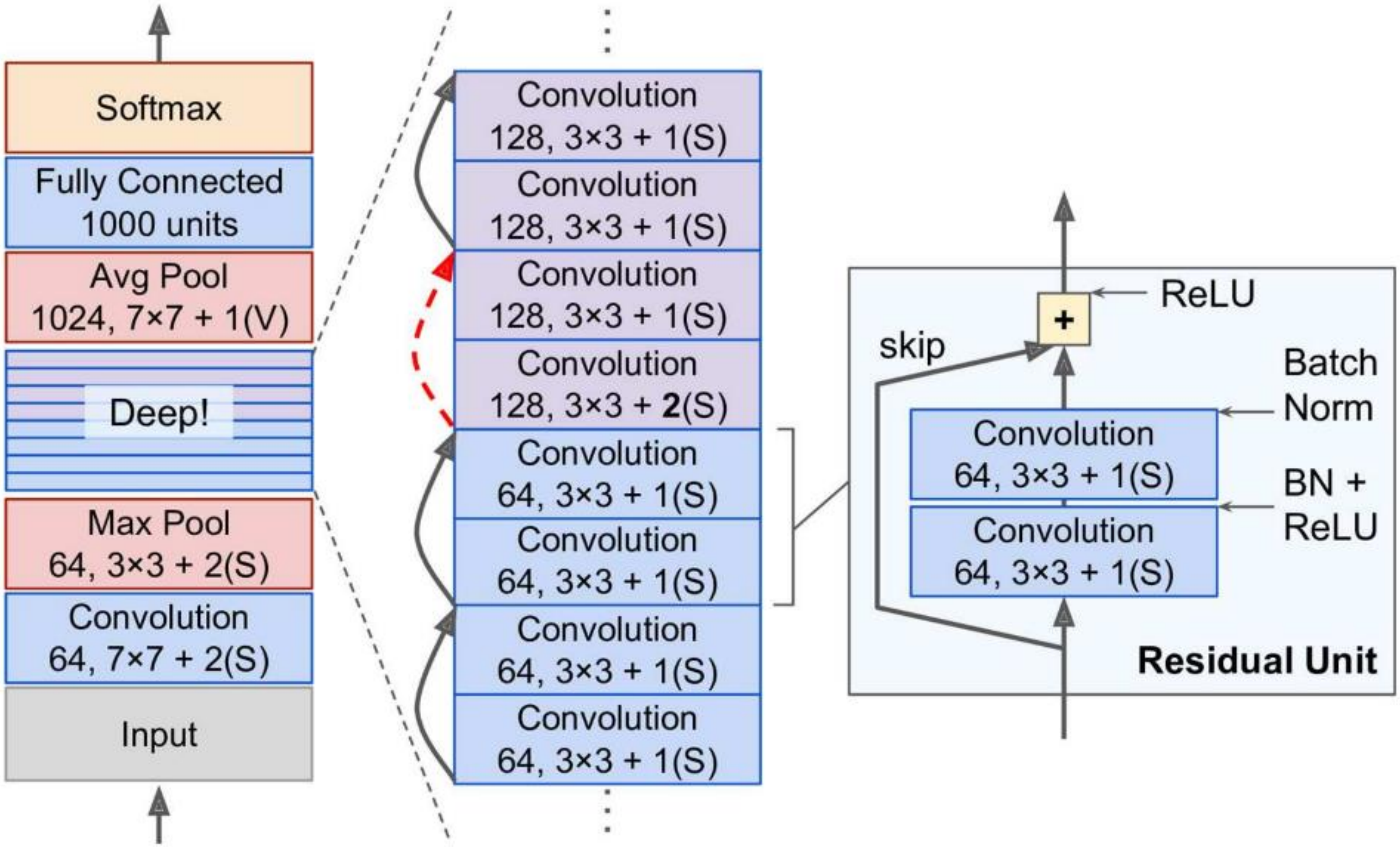
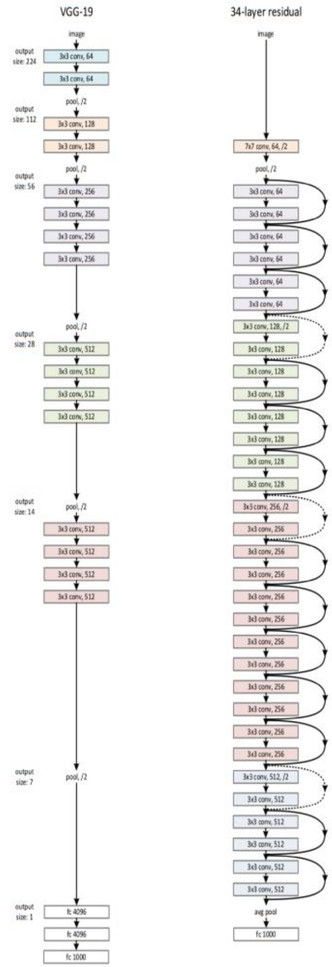


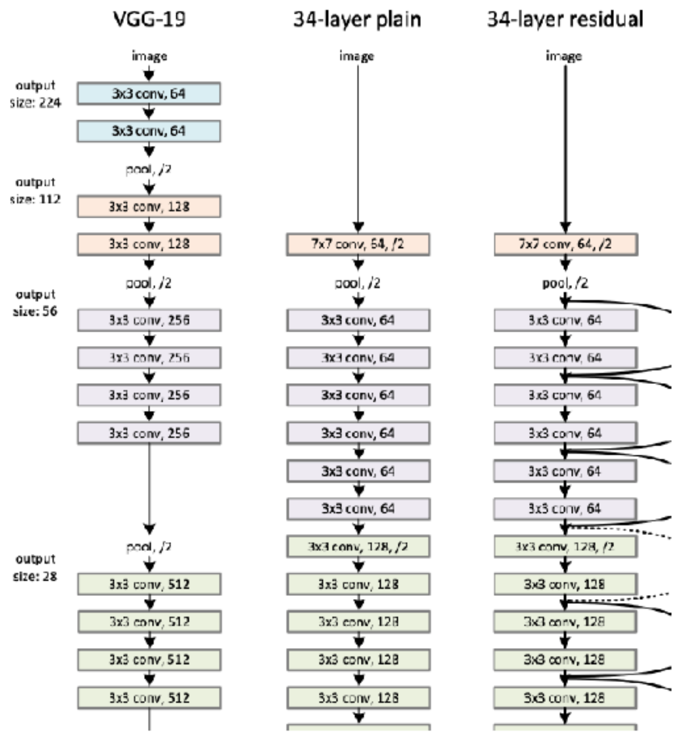
Image from Hands-on Machine Learning with SciKit-Learn and TensorFlow

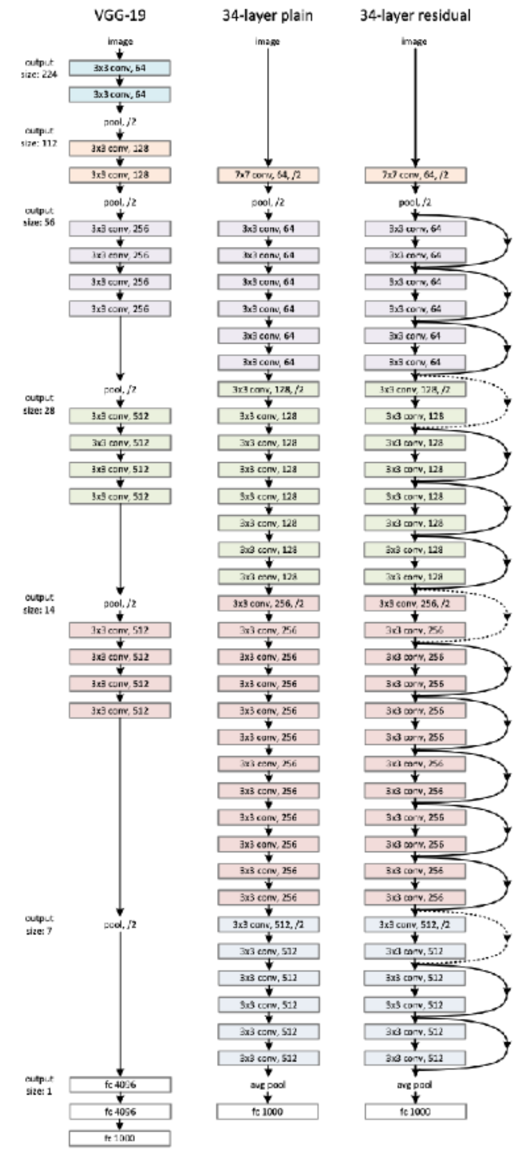
Comparing with VGG



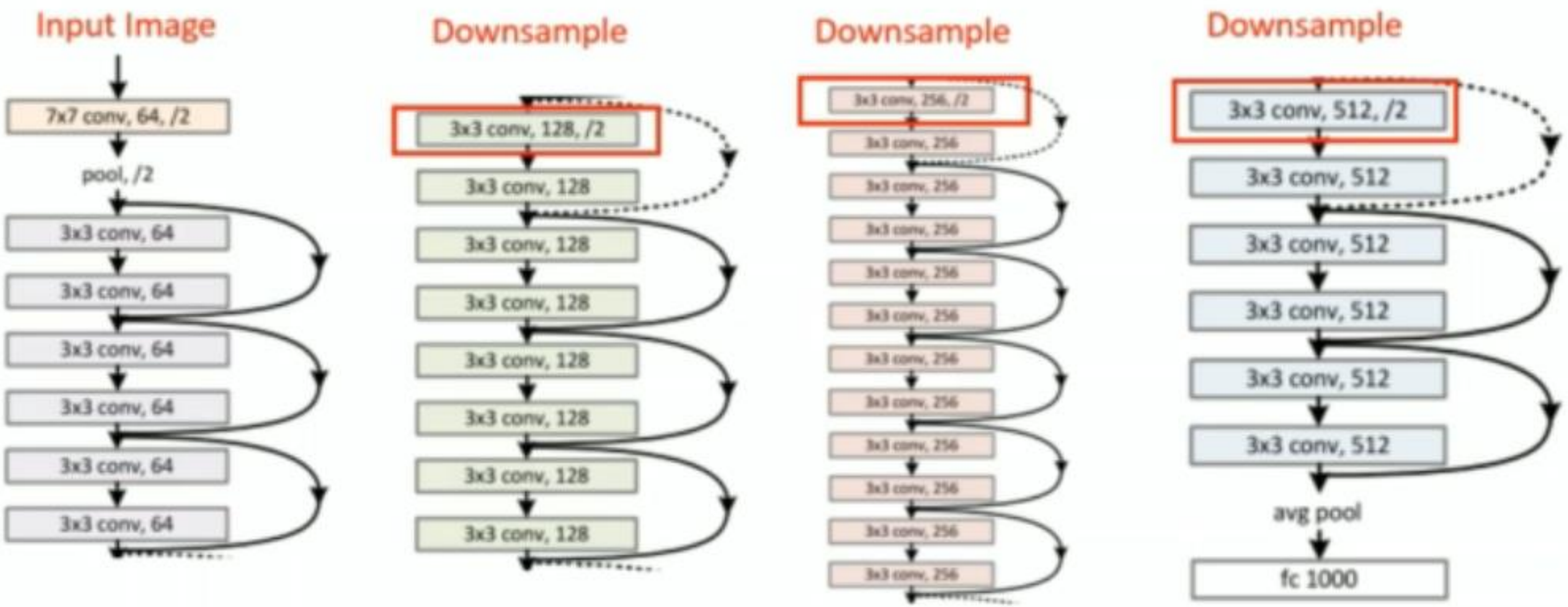
ResNet34 and ResNet50 Architectures

- ResNet34 and ResNet50 features several consecutive Conv layers of 3x3 with feature maps (64, 128, 256, 512) with bypasses every 2 Convolutions.
- Their output dimensions remain constant (padding same, stride =1)
- ResNet was built by several stacked residual units and developed with many different numbers of layers: 18, 34, 50, 101, 152, and 1202.



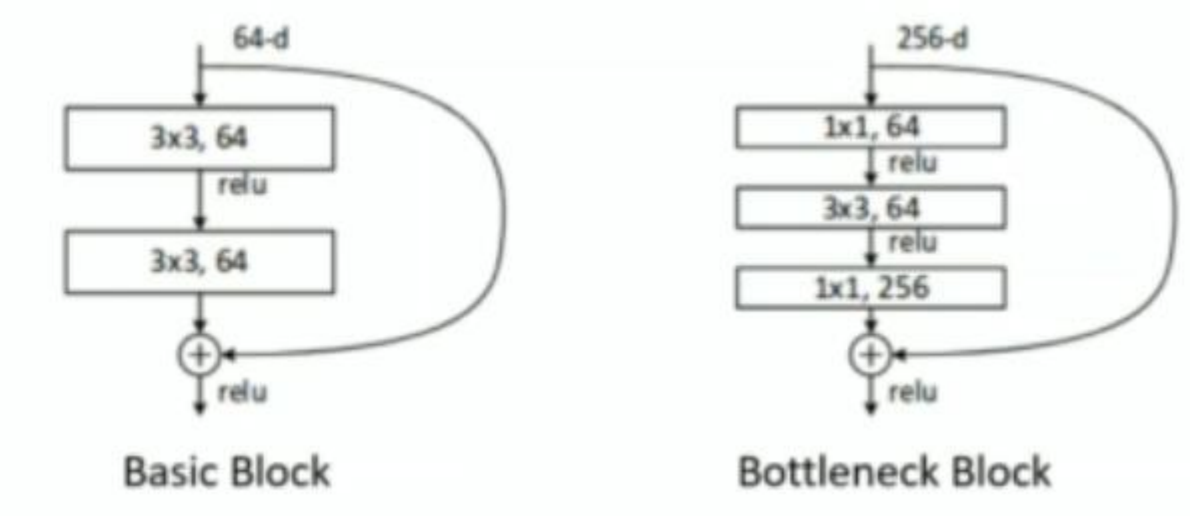


ResNet 34 layers



The dotted line means bottle neck layer (need to adjust input size)

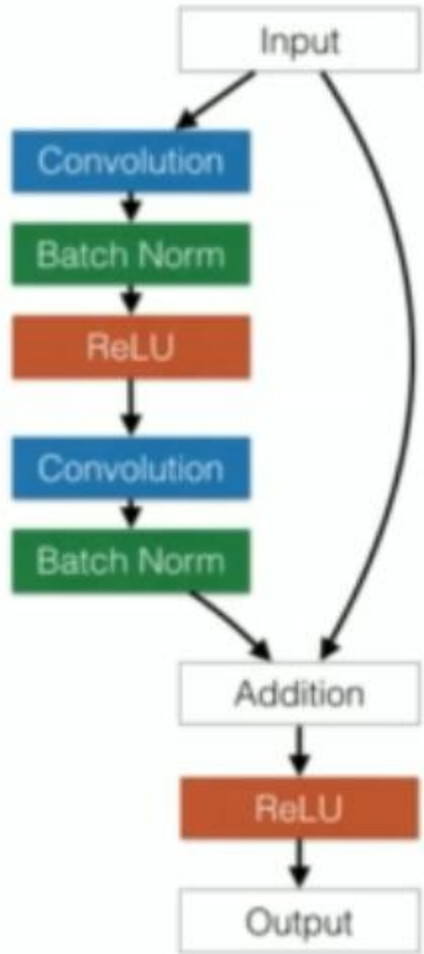
Residual Block



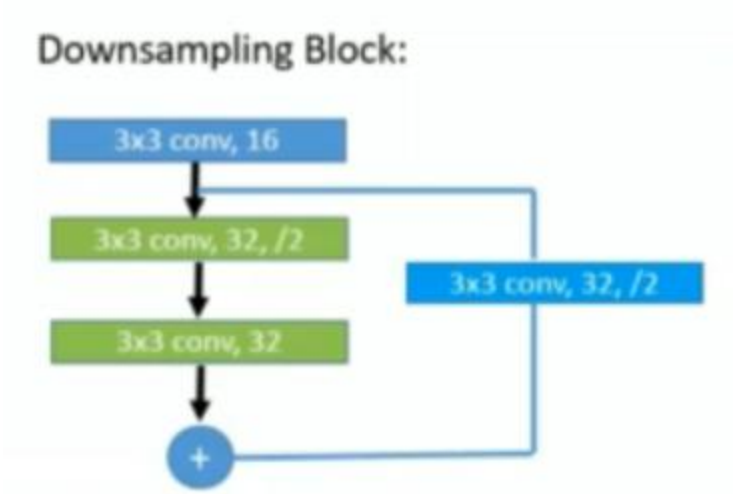
Basic block is used in small network like ResNet 18, 34

Bottlenect block is used is ResNet 50, 101, 152

Inside 3x3 Convolution



Downsampling



Down sampling is achieved by convolution with stride of 2

ResNet: Downsampling

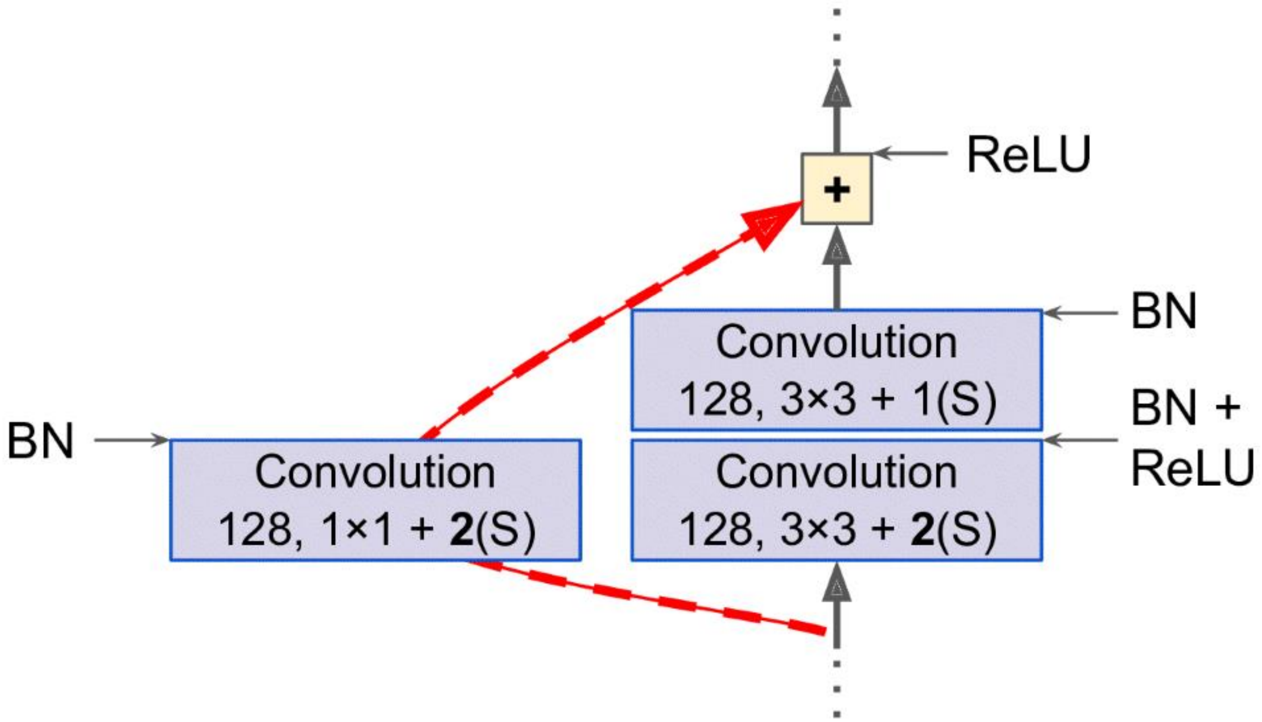


Image from Hands-on Machine Learning with SciKit-Learn and TensorFlow

Stochastic Depth

- With ResNet, now we can go 100 layers depth

- Can we go 1,000 layers depth?

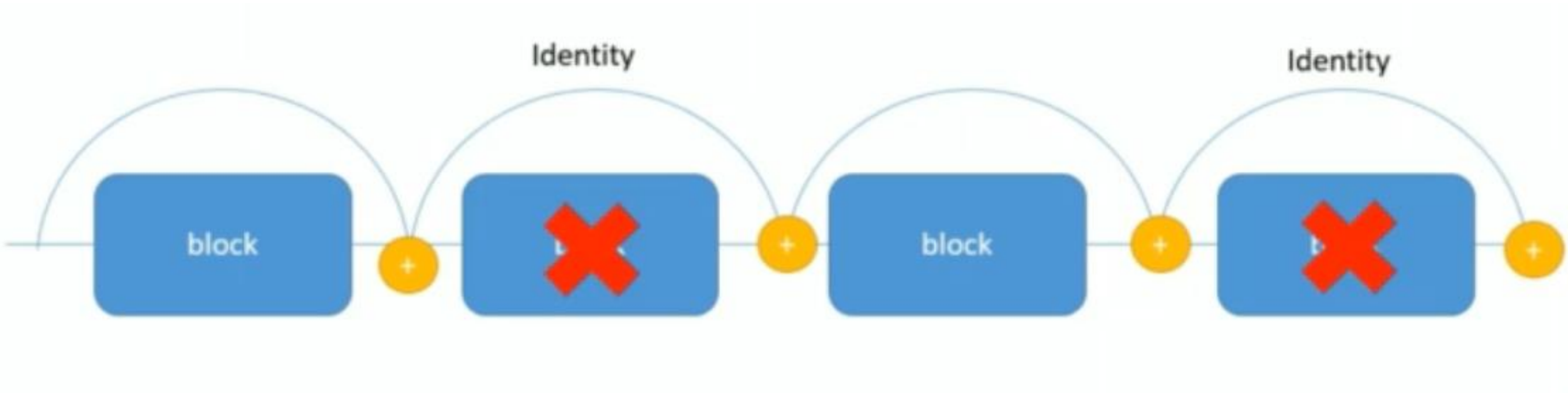
Can we go 1k layers depth?

- With ResNet alone, No

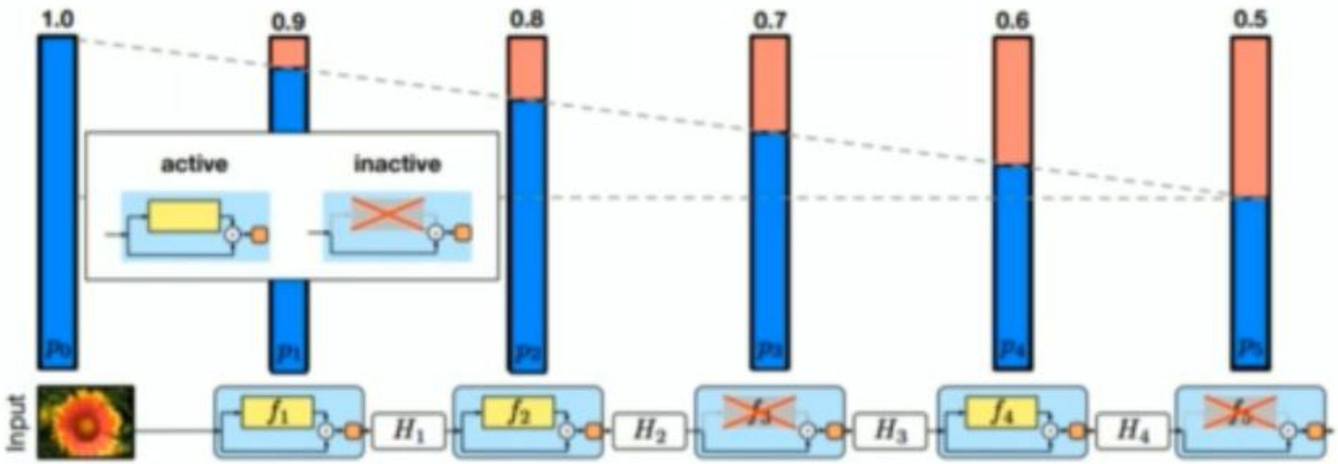
- With ResNet and Stochastic Depth Regularization, Yes

Stochastic Depth Regularization

Randomly dropping entire blocks during training



Survival probability

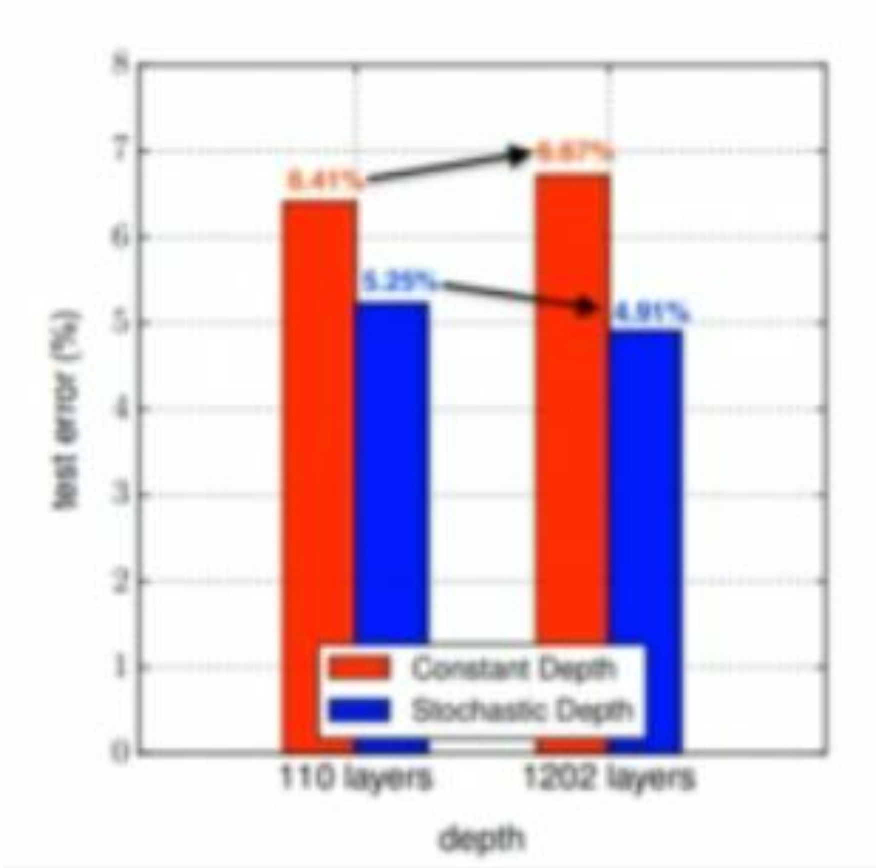


$$p_\ell = 1 - \frac{\ell}{L}(1 - p_L)$$

p_L is 0.5, L is the total number of block

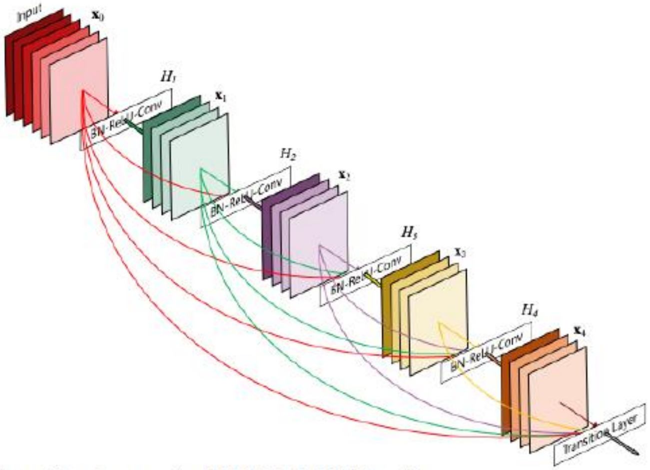
l is current block

Test Error Rate



DenseNet

- Proposed by Cornwell U., Tsinghua U., and Facebook AI Research in 2016
- DenseNet got Best CVPR Paper Award in 2017

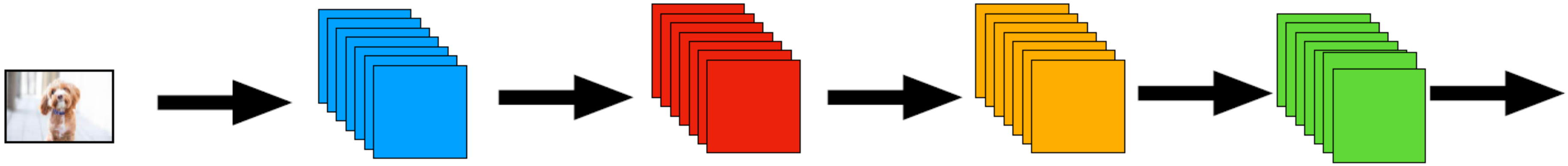


https://arxiv.org/pdf/1608.06993.pdf

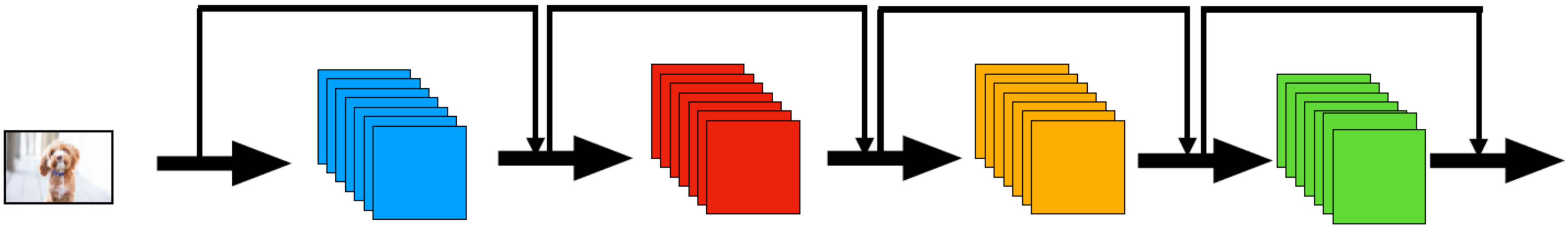
Vanishing Gradient Problem

- DenseNet solves the vanishing gradient problem with the concept of “collective knowledge”

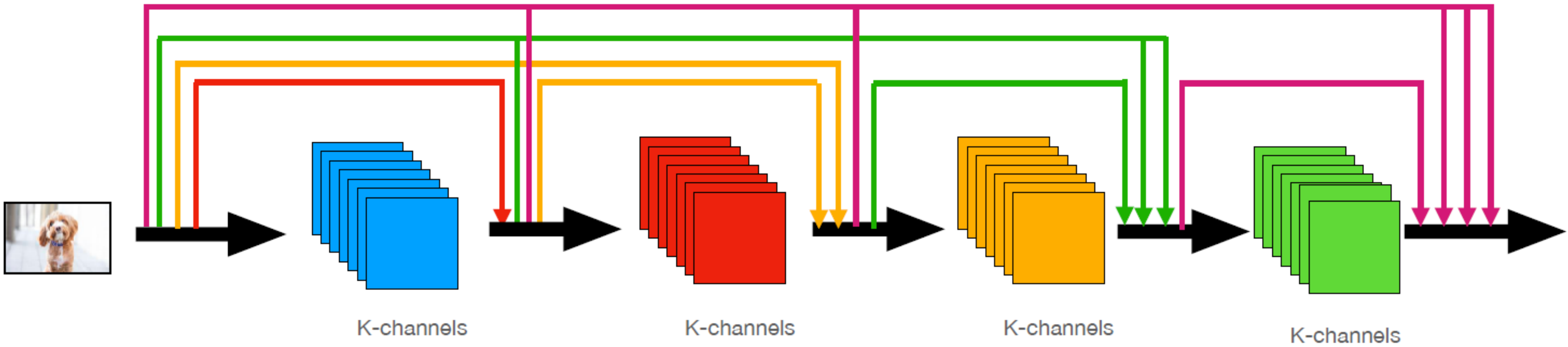
Classical CNN



ResNet Architecture



DenseNet Architecture



- Each layer receives info from all previous layer

DenseNet Architecture

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Table 1: DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

DenseNet Performance

Model	top-1	top-5
DenseNet-121	25.02 / 23.61	7.71 / 6.66
DenseNet-169	23.80 / 22.08	6.85 / 5.92
DenseNet-201	22.58 / 21.46	6.34 / 5.54
DenseNet-264	22.15 / 20.80	6.12 / 5.29

Table 3: The top-1 and top-5 error rates on the ImageNet validation set, with single-crop / 10-crop testing.

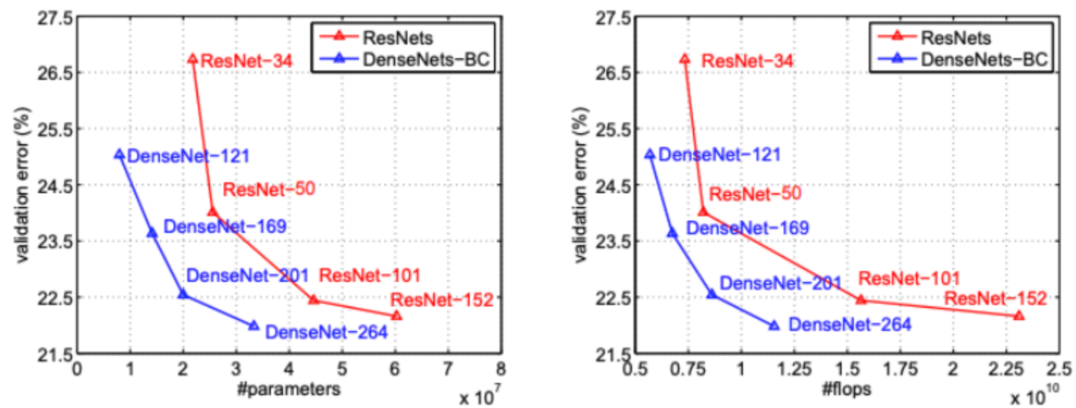
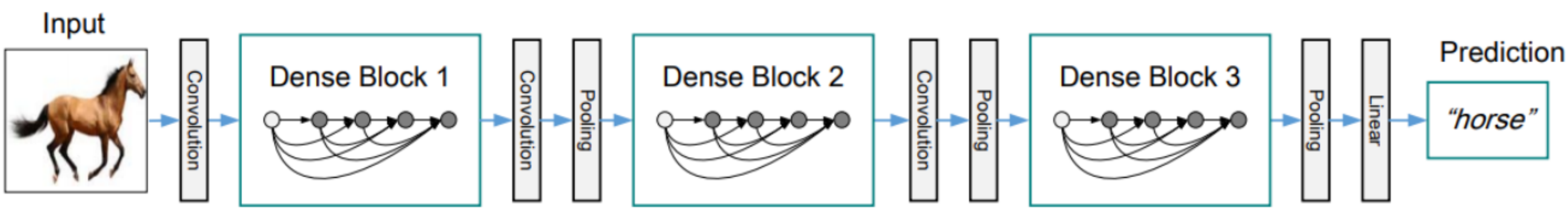


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

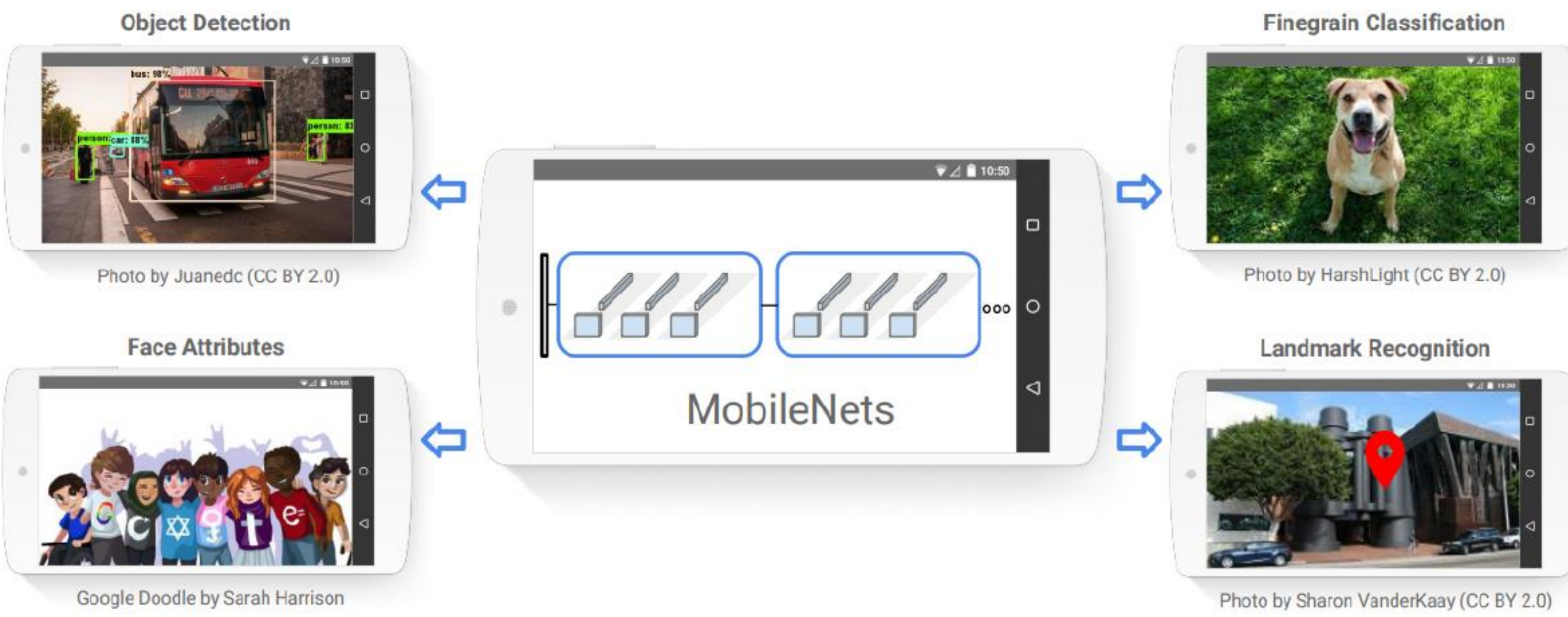
Multiple Dense Blocks



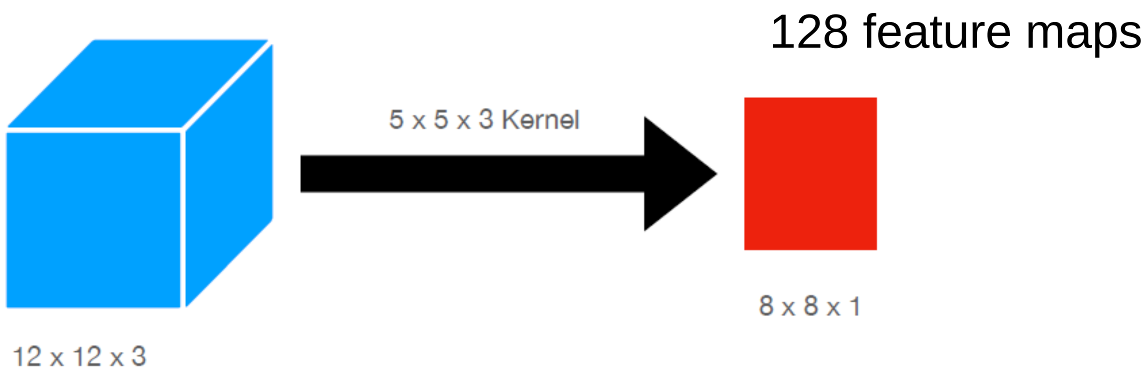
MobileNet

- MobileNet was developed by Google in 2017
- The goal is to make an efficient light-weight for embedded devices and mobile phone
- Relatively slow on inference

MobileNet Application



Convolution Operations in CNNs



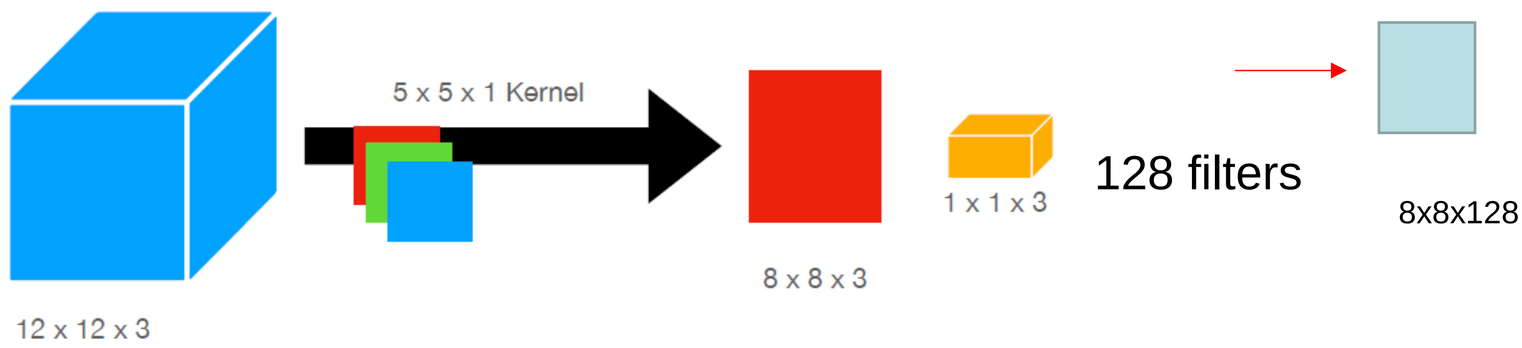
Stride of 1, we have 5 x 5 x 3 = 75

Output = 8x8 = 64

If we have 128 filters:

75 x 64 x 128 = 614,400 operations

Depth-wise and point-wise Convolutions



5x5x3x8x8 (output) = 4,800

8x8x3x128 = 24,576

Total = 4,800 + 24,576 = 29,376 operations

Results in 20x less operation

Two additional Hyper parameter

- Width Multiplier : reduce the depth (number of filter)

- Resolution Multiplier: reduce image size

Accuracy

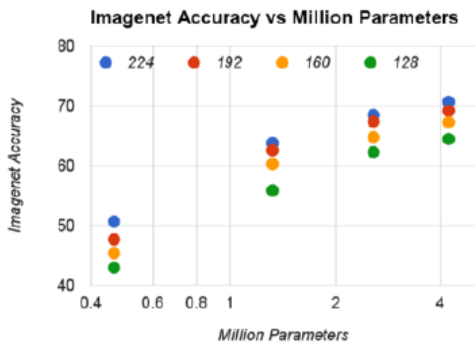


Figure 5. This figure shows the trade off between the number of parameters and accuracy on the ImageNet benchmark. The colors encode input resolutions. The number of parameters do not vary based on the input resolution.

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Parameters
1.0 MobileNet-224	70.6%	4.2
GoogleNet	69.8%	6.8
VGG 16	71.5%	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Parameters
0.50 MobileNet-160	60.2%	1.32
Squeezenet	57.5%	1.25
AlexNet	57.2%	60

Table 10. MobileNet for Stanford Dogs

Model	Top-1 Accuracy	Million Parameters
Inception V3 [18]	84%	23.2
1.0 MobileNet-224	83.3%	3.3
0.75 MobileNet-224	81.9%	1.9
1.0 MobileNet-192	81.9%	3.3
0.75 MobileNet-192	80.5%	1.9

SqueezeNet

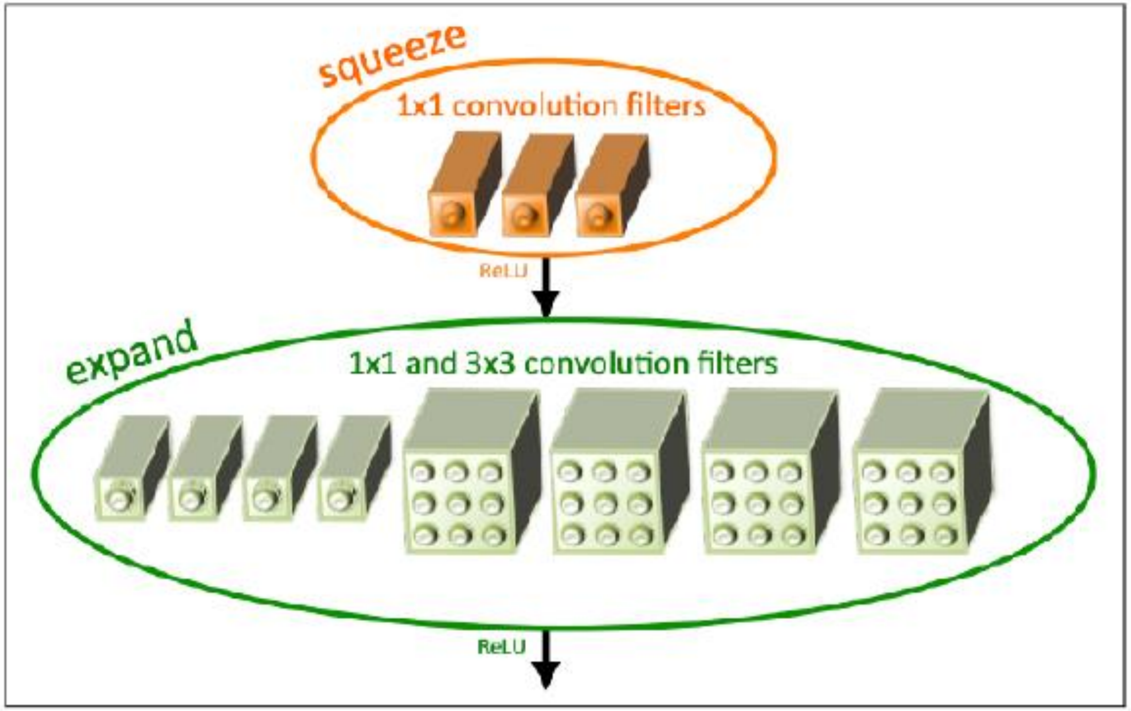
- Introduced in 2016 by UC. Berkeley, Stanford and DeepScale
- The goal is to make a highly accurate CNN with smaller size
  - Less communication
  - Smaller size
  - Faster
- It had 50x less parameters than AlexNet, but 3X faster

SqueezeNet Feature

- Replace 3x3 Filters with 1x1

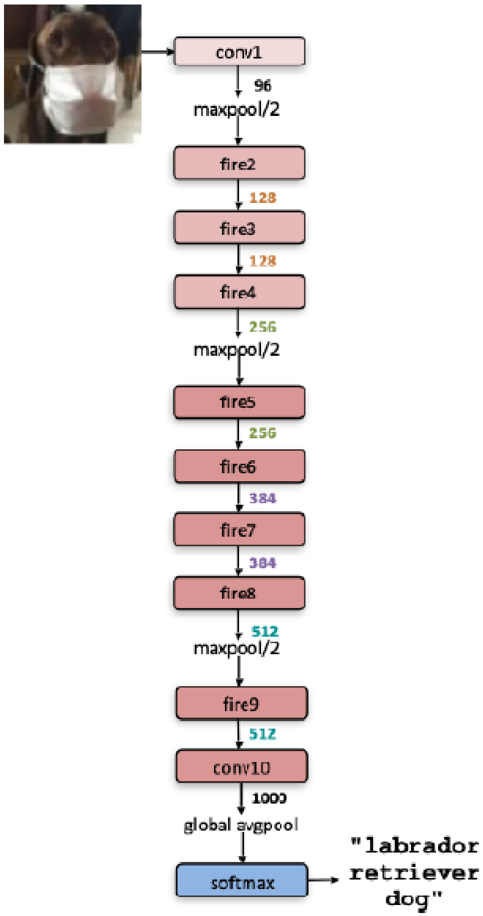
- Downsample later in the network so that convolution layers have larger activation maps

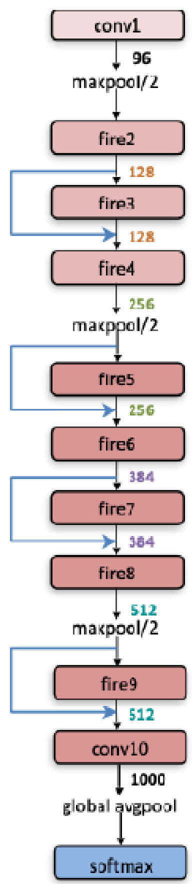
Fire Module

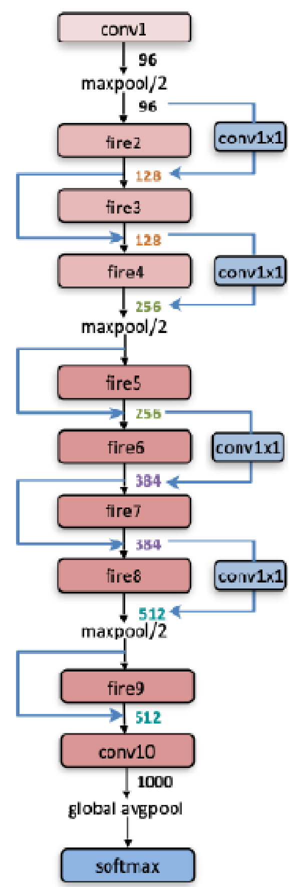

squeeze
1x1 convolution filters
ReLU
expand
1x1 and 3x3 convolution filters
ReLU

SqueezeNet

-SqueezeNet - SqueezeNet with simplebypass - SqueezeNet with complex bypass



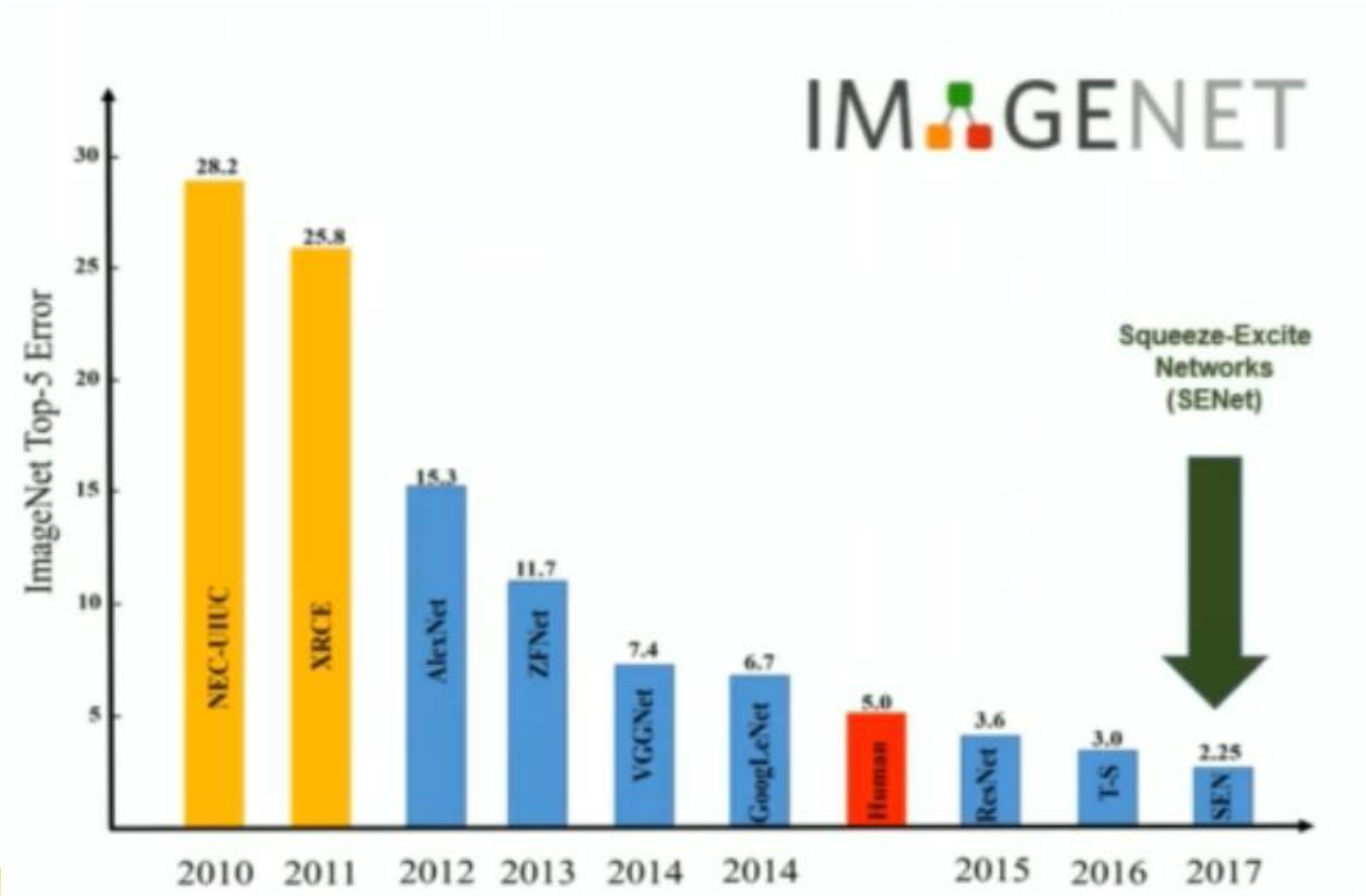




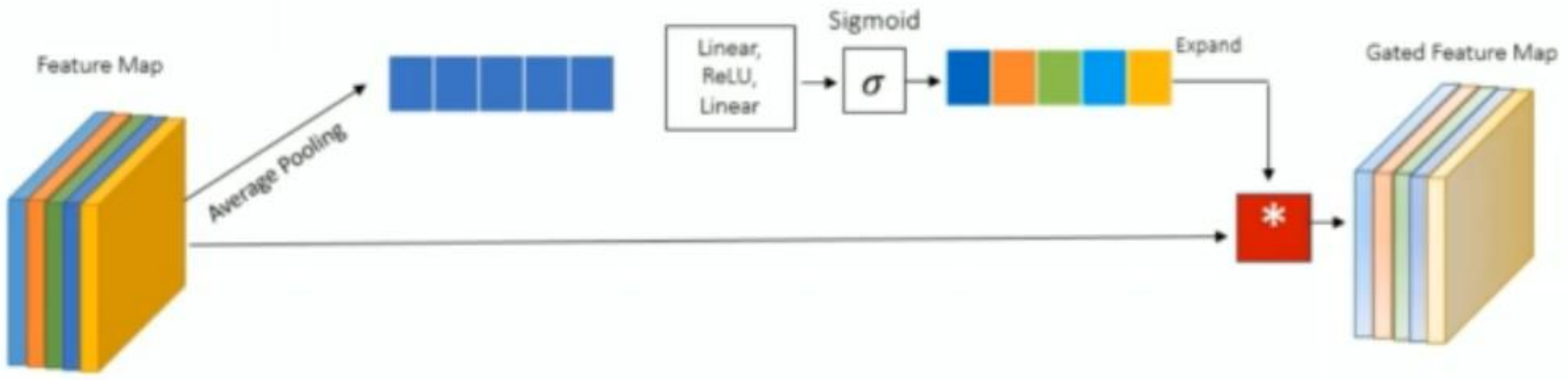
Performance

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

SqueezeExiteNet (SENet)

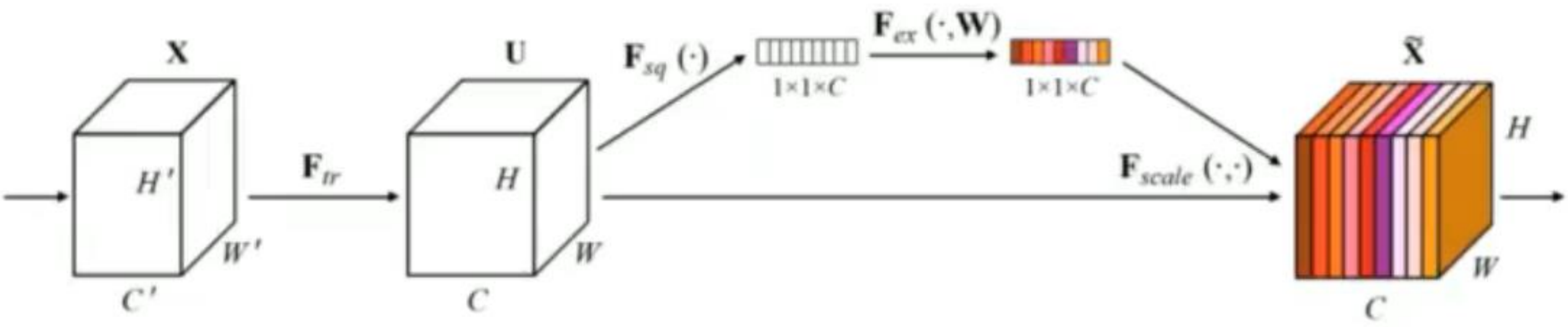


SE Net



Feature Map
Average Pooling
Linear,
ReLU,
Linear
Sigmoid
σ
Expand
*
Gated Feature Map

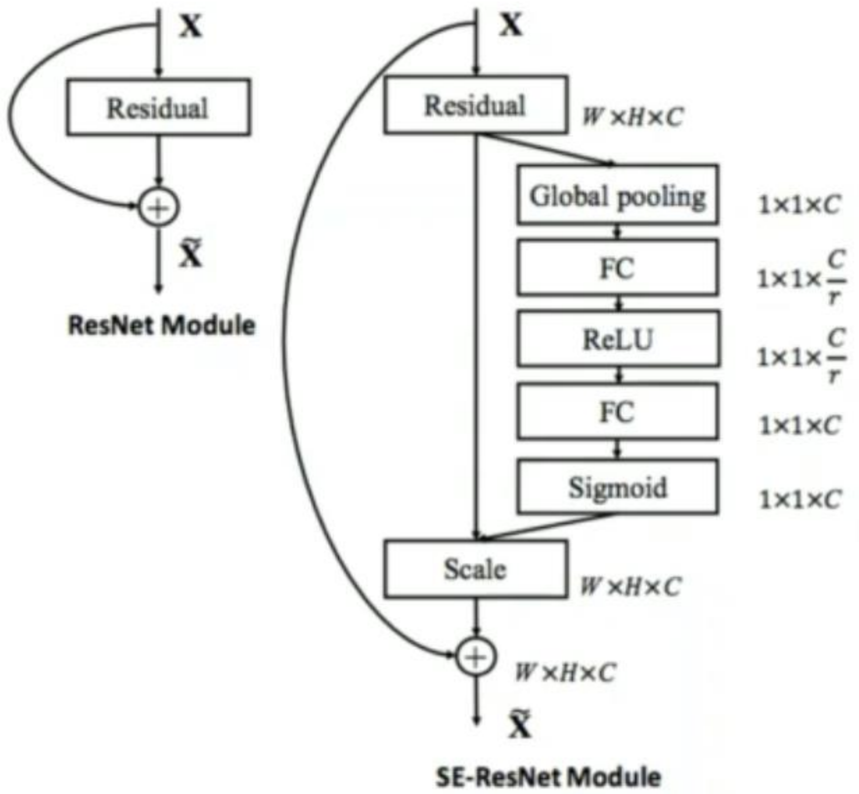
SE Net



Usually, a network weights each of its channels equally

Hence, we should provide content aware mechanism for each feature map

SE Net Implementation



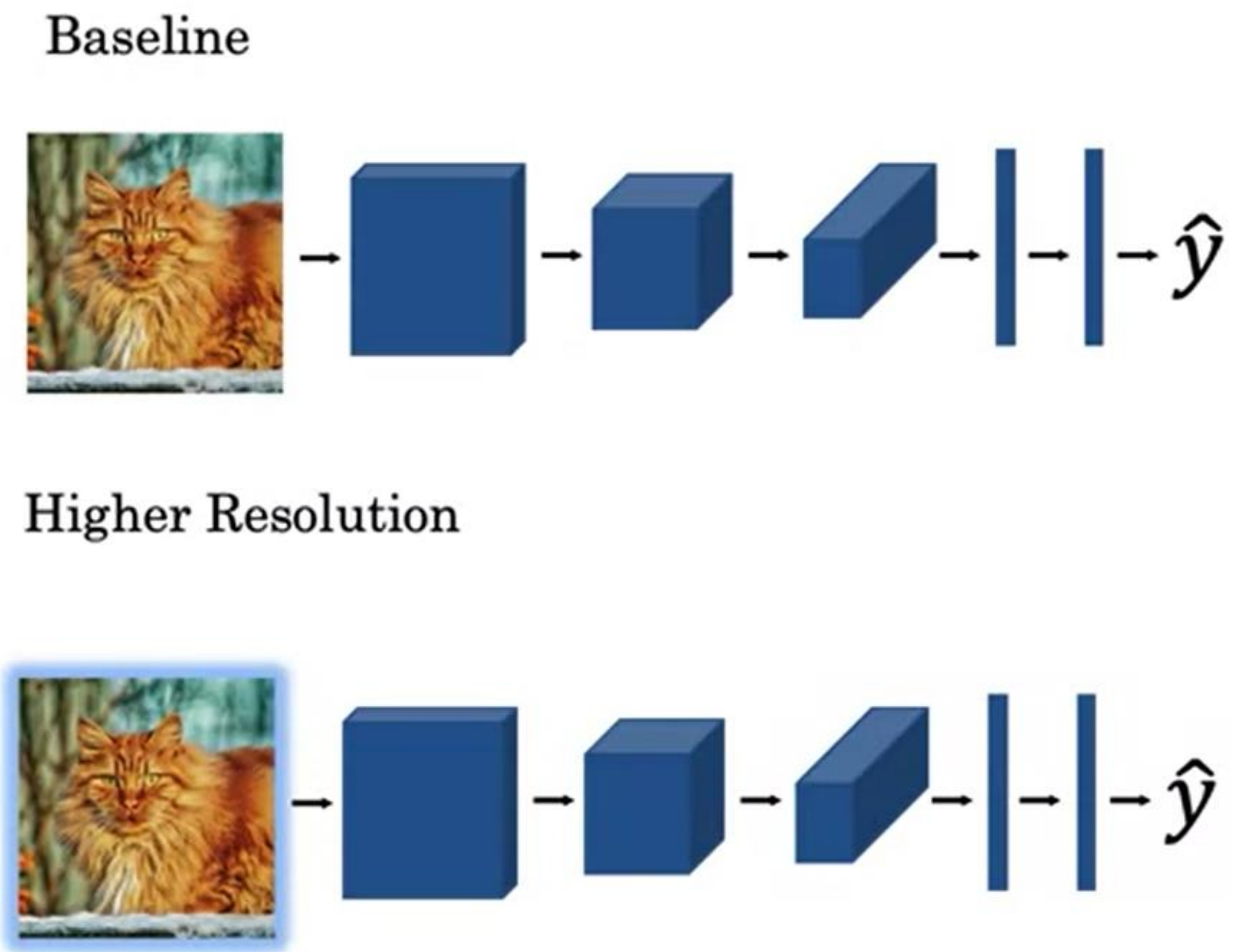
r is the reduction factor

C is the number of channel

EfficientNet

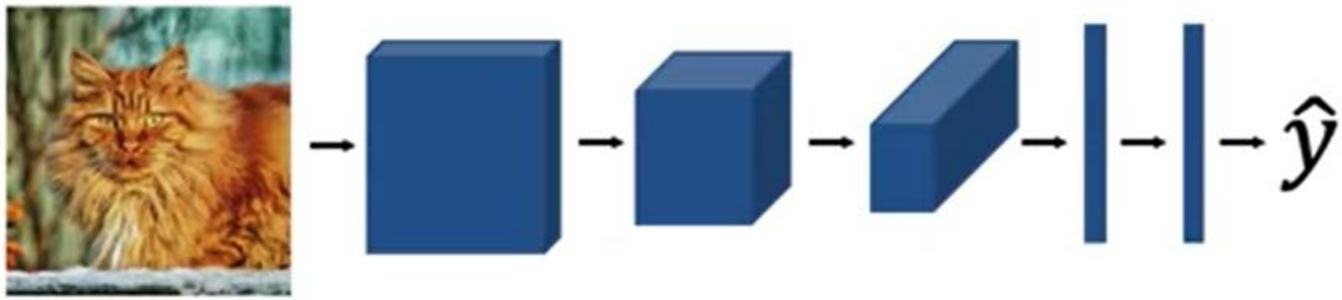
- Introduced by researchers from Google in 2019 [Tan and Le EfficientNet: Rethinking Model Scaling for CNN]
- CNN architectures is designed from lower layer and then scale up e.g., from ResNet 18 to ResNet-200

Efficient Net

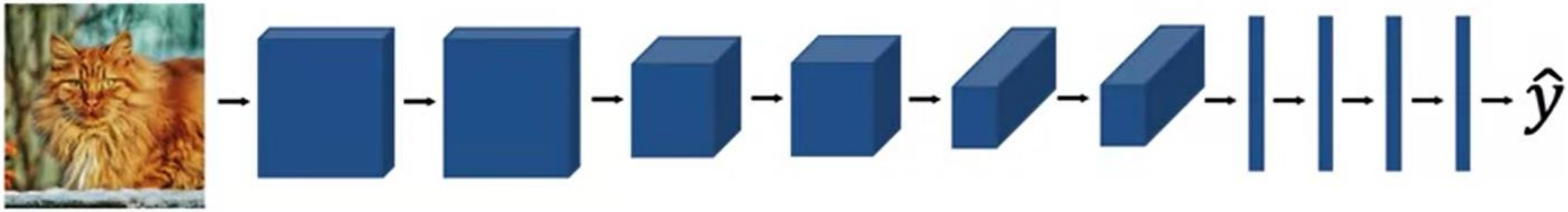


Efficient Net

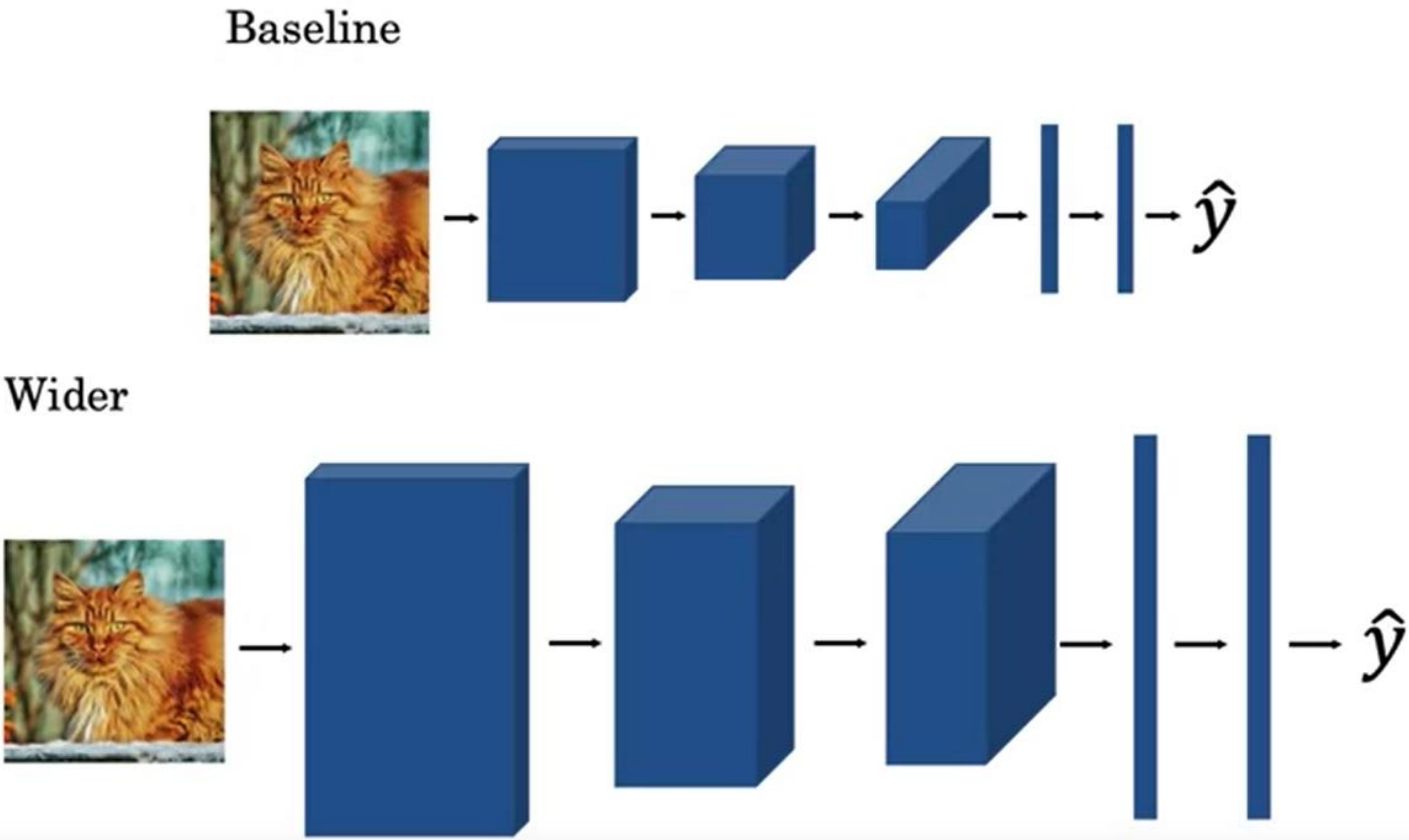
Baseline



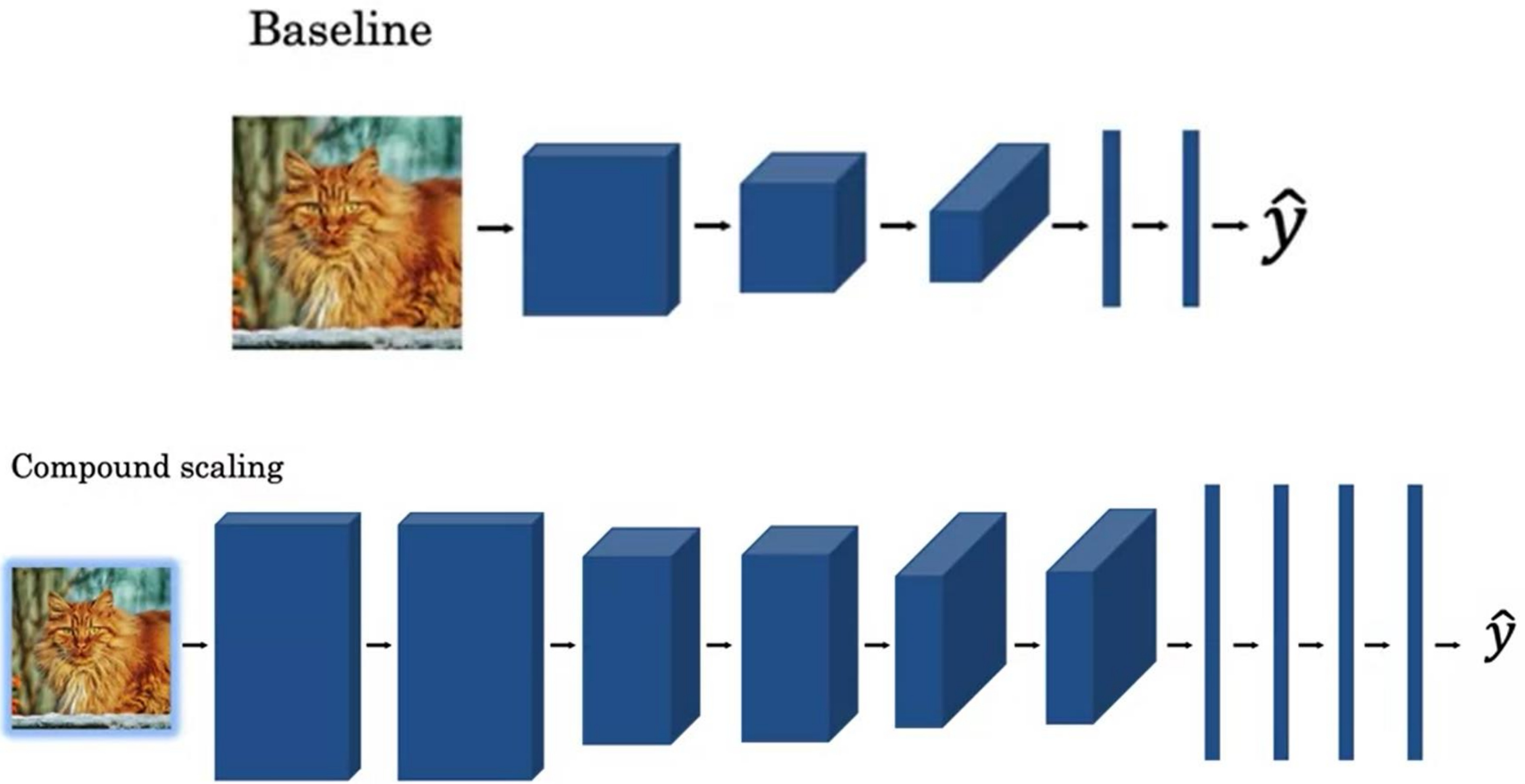
Deeper



Efficient Net



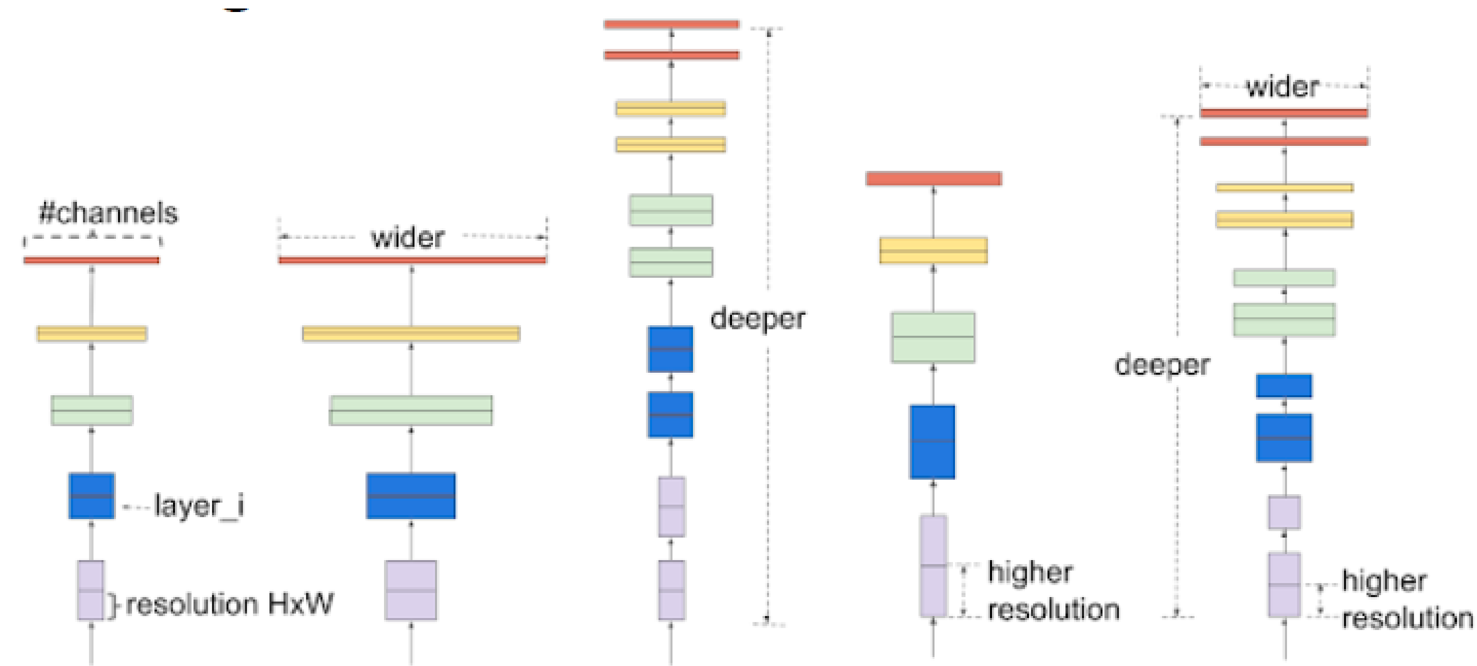
Efficient Net



EfficientNet Scaling Method

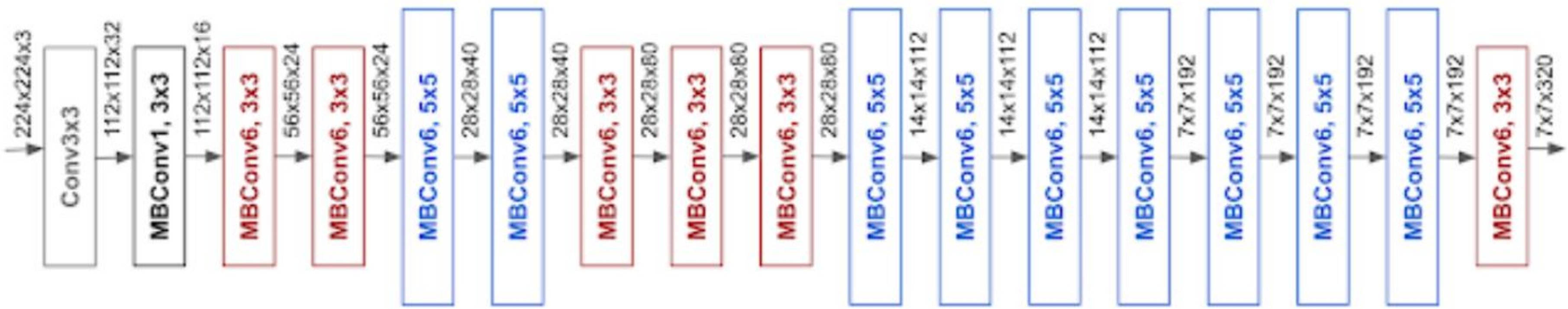
- Google uses new AutoML to design a better way for scaling
- The method uniformly scale each dimension (width, depth, resolution)
- It provides better accuracy with 10x better efficiency

Scaling Method

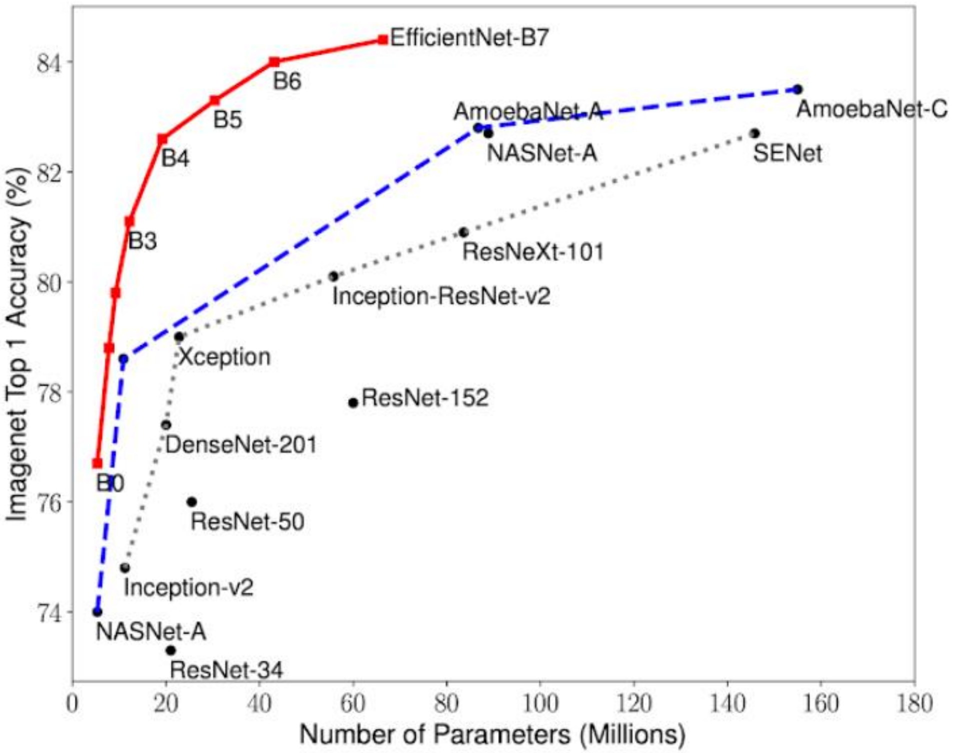


(a) baseline (b) width scaling (c) depth scaling (d) resolution scaling (e) compound scaling

EfficientNet Architecture

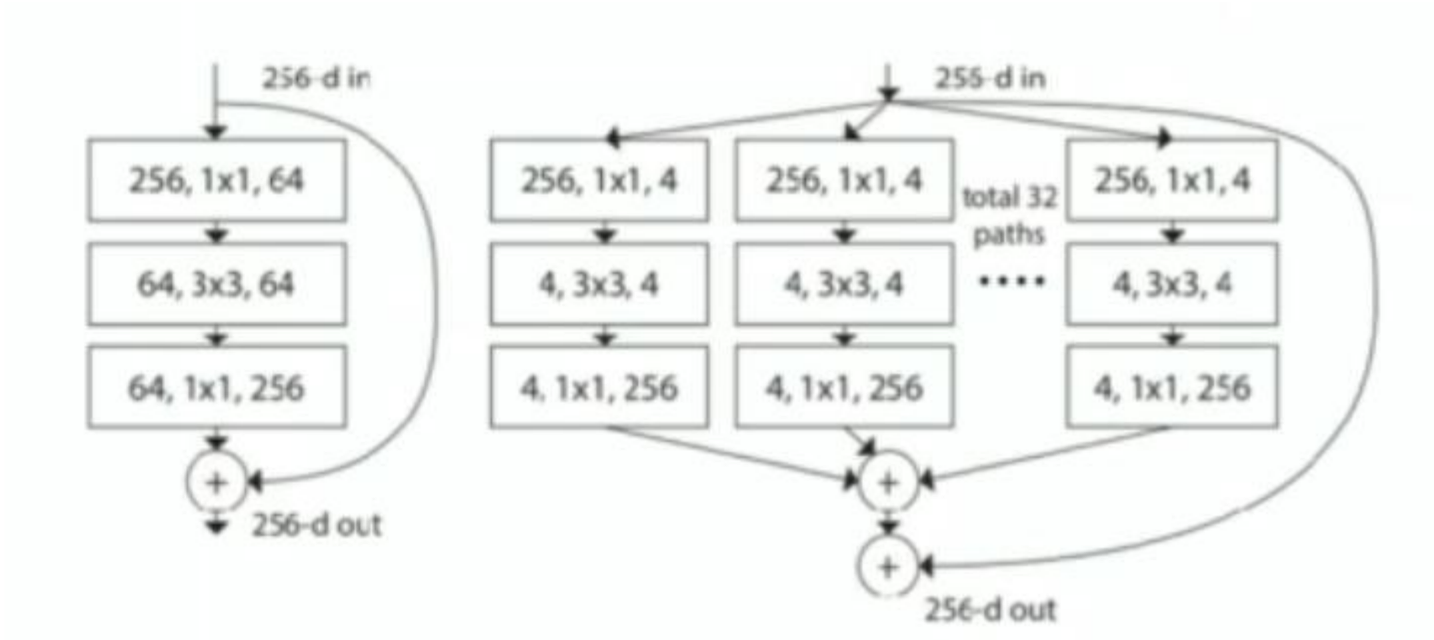


Performance



ResNext

- Paper titled “Aggregated Residual Tranformations for Deep Neural Networks”



Capsule Network

- Address limitation of CNN that it lacks information about orientation (rotation), lighting condition, and color

- It tried to address those problems by introducing a new network

- TensorFlow implementation is available at:

https://github.com/naturomics/CapsNet-Tensorflow.

Pytorch/Keras Available Framework

PyTorch

2. AlexNet
3. VGG
4. ResNet
5. SqueezeNet
6. DenseNet
7. Inception v3
8. GoogLeNet
9. ShuffleNet v2
10. MobileNet v2
11. ResNeXt
12. Wide ResNet
13. MNASNet

Keras

1. Xception
2. VGG16 19
3. ResNet50
4. ResNet152V2
5. InceptionV3
6. MobileNetV2
7. DenseNet
8. NASNetMobile
9. NASNetLarge
10. EfficientNet

Success of Deep Learning

- How to train deep network (many layers)
- Faster machines (GPUs)
- Larger Dataset

Homework

- Implement AlexNet from Scratch using Keras on CIFAR 100

Questions?