Advanced Convolutional Neural Network

Rohit Kumar Kaliyar

CNN Architectures

CNN hyperparameters: num of filters, filter size, max-pooling, stride

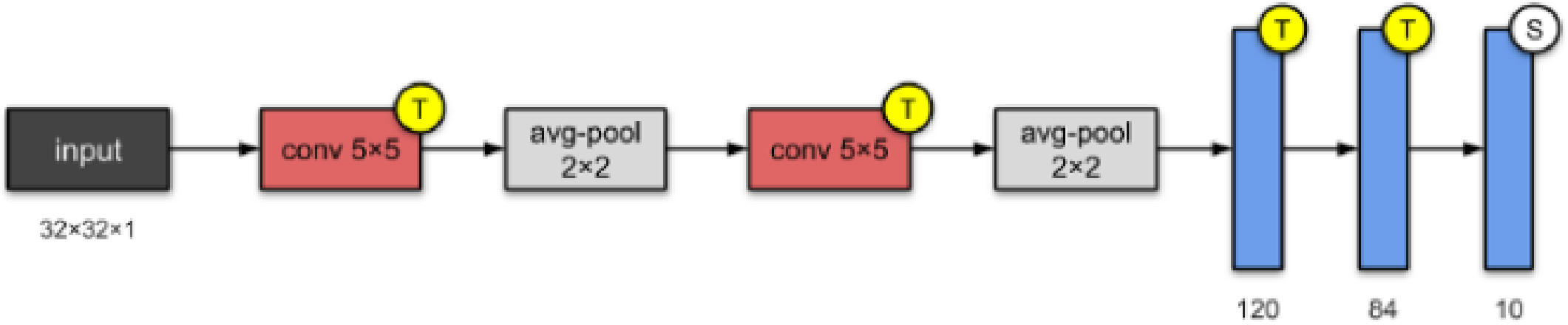
One can change the values of hyperparameters and get different architectures.

But what values to choose for which hyperparameter in which application?

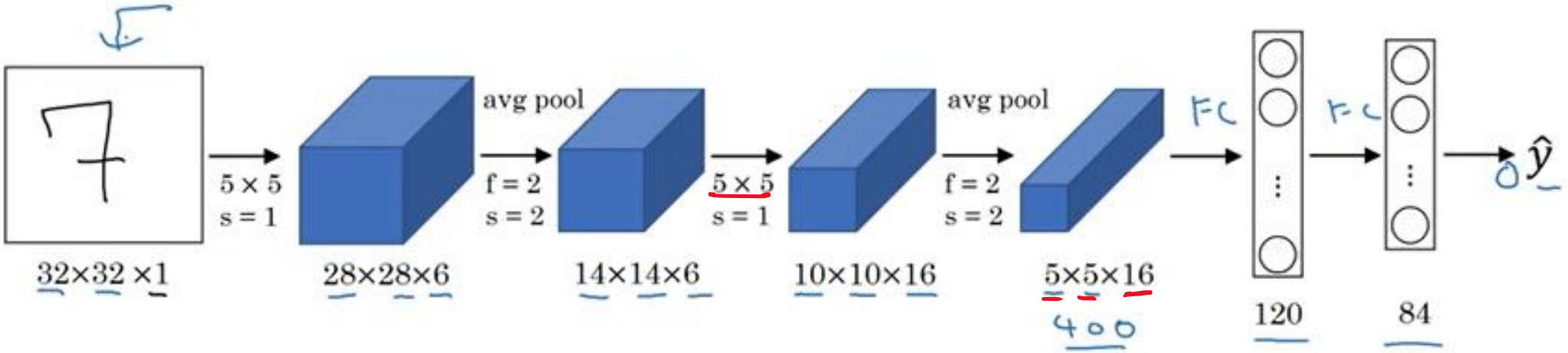
Go for some standard architectures.

LeNet-5

- Proposed in 1998
- Simplest architecture
- 2 conv layers, 3 FC layers -> total 5 layers
- 60,000 parameters



LeNet - 5



Calculating number of parameters:

1st conv layer:	**6 x (5x5x1 + 1)**	**= 156**
2nd conv layer:	**16 x (5x5x6 + 1)**	**= 2,416**
1st FC layer:	**(400 x 120) + 120**	**= 48,120**
2nd FC layer:	**(120 x 84) + 84**	**= 10,164**
3rd FC layer:	**(84 x 10) + 10**	**= 850**

Adding all up: 156 + 2416 + 48120 + 10164 + 850 = 61706

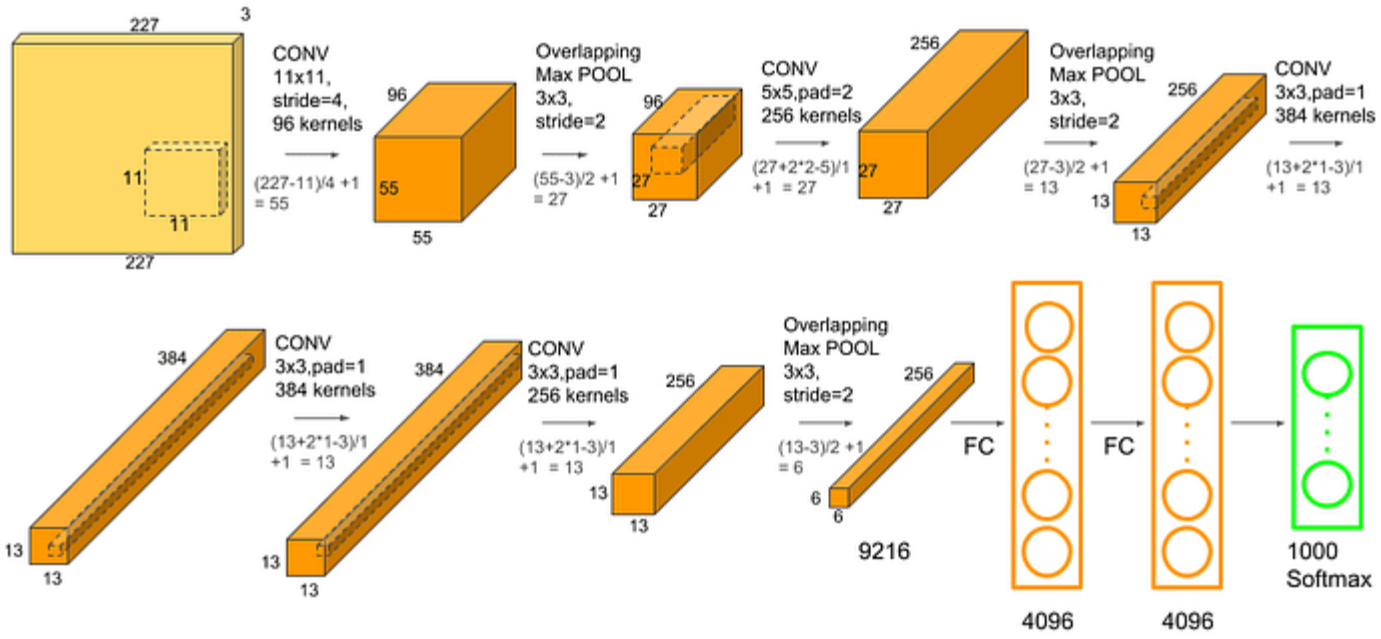
LeNet-5

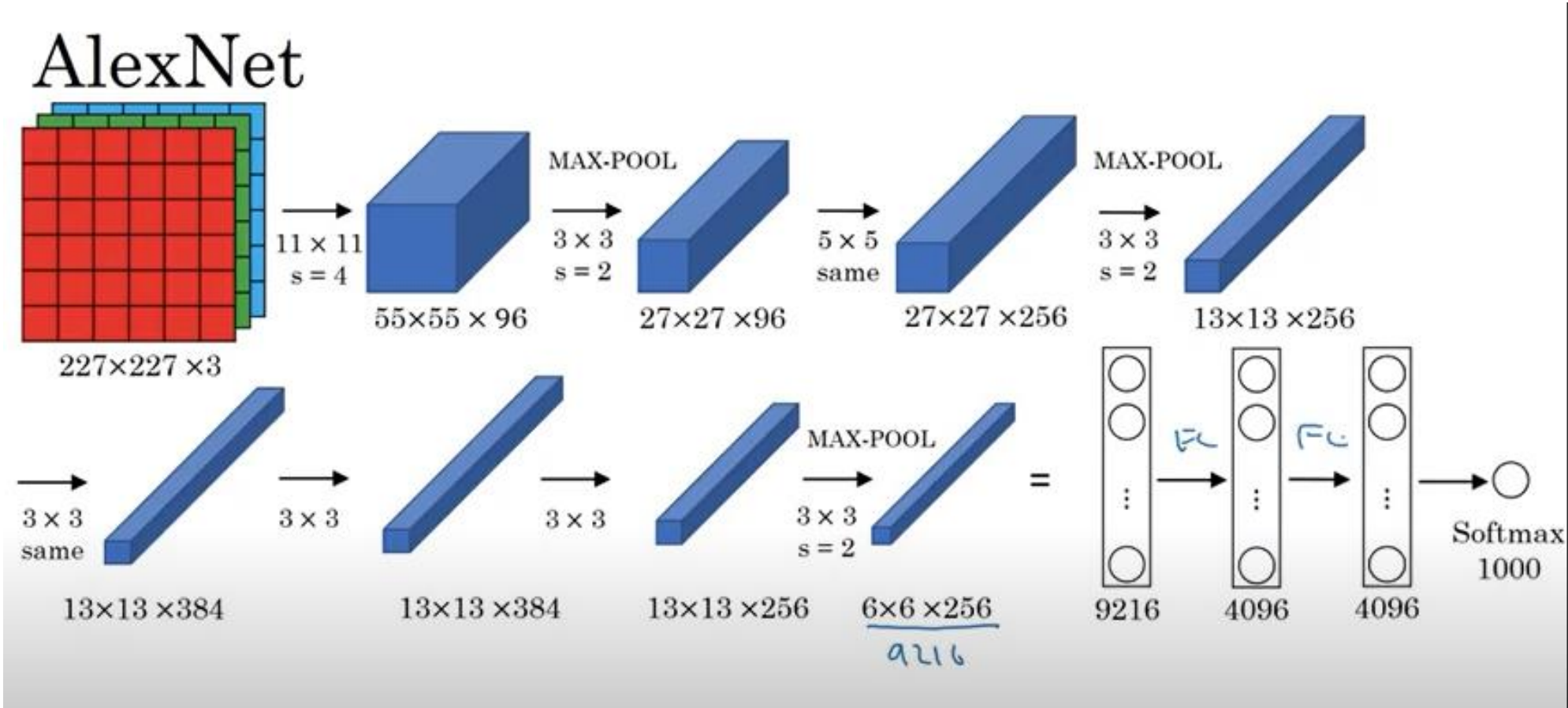
Key points:

- As we go from left to right in LeNet-5, the num of dimension decreases from 32 to 5, while the num of channel increases, from 1 to 16.
- The total number of parameters in LeNet-5 are: 60,000
- Layers flow: Conv -> Pool -> Conv -> Pool -> FC -> FC -> Output
- Non-linearity (activation functions) is applied after pooling
- Activation functions: Sigmoid/tanh

AlexNet

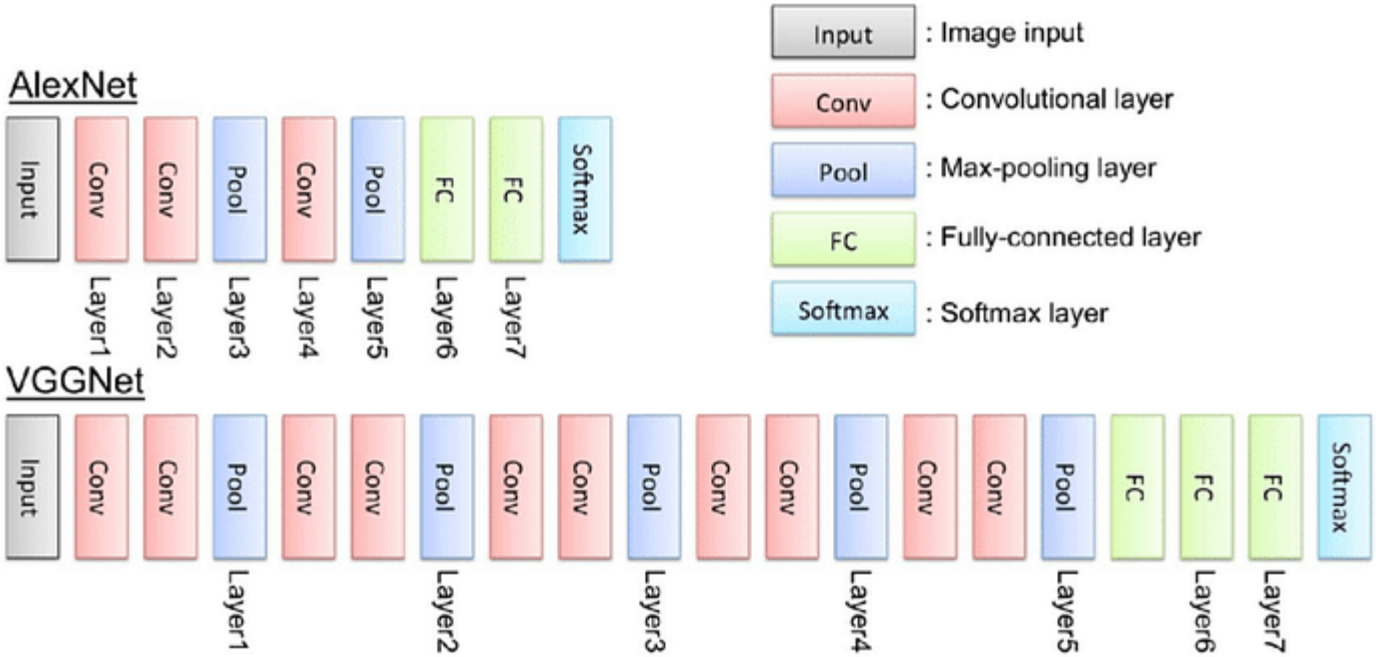
- Proposed in 2012
- 5 conv layers, 3 FC layers -> total 8 layers
- 60 million parameters

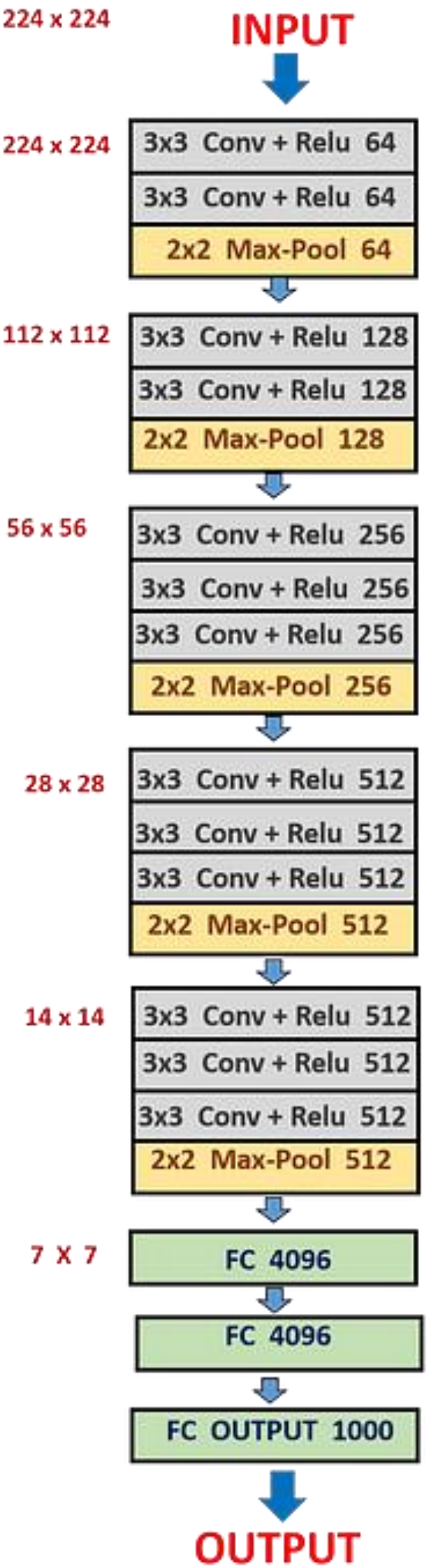



AlexNet
227×227 ×3
11 × 11
s = 4
55×55 × 96
MAX-POOL
3 × 3
s = 2
27×27 ×96
5 × 5
same
27×27 ×256
MAX-POOL
3 × 3
s = 2
13×13 ×256
3 × 3
same
13×13 ×384
3 × 3
13×13 ×384
3 × 3
13×13 ×256
MAX-POOL
3 × 3
s = 2
6×6 ×256
9216
=
FC
FC
9216
4096
4096
Softmax
1000

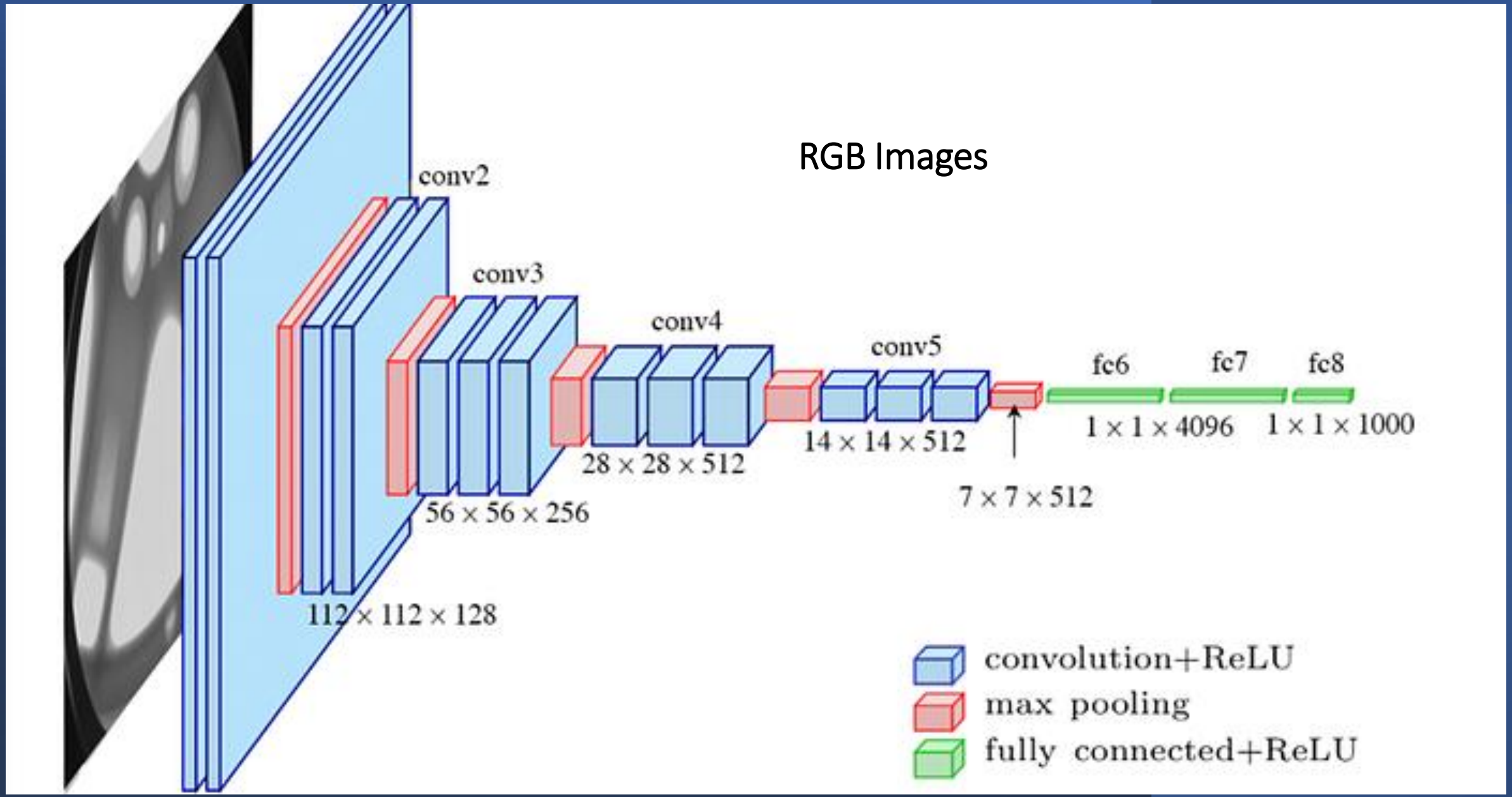
VGG16

- Proposed in 2014 by Visual Geometry Group (VGG)
- 13 conv layers, 3 FC layers -> total 16 layers
- 138 million parameters

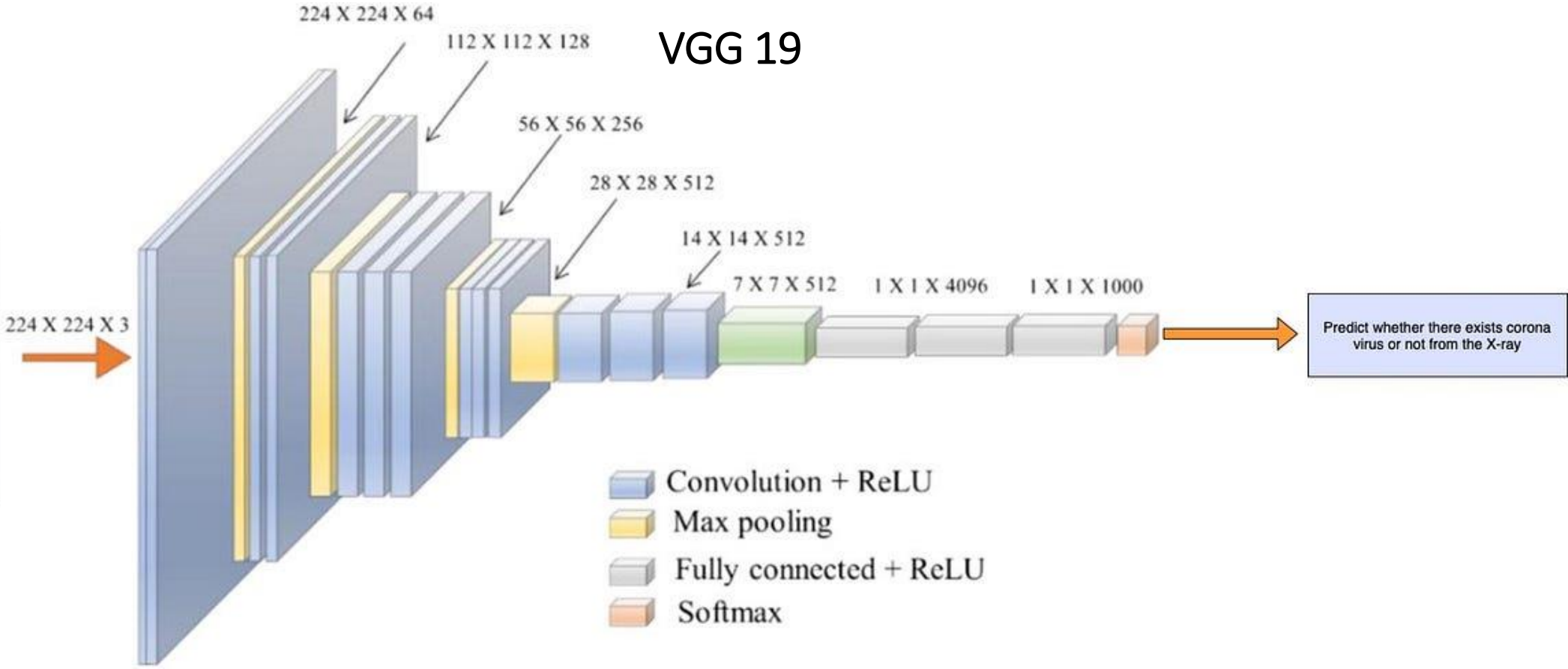



224 x 224
INPUT
224 x 224
3x3 Conv + Relu 64
3x3 Conv + Relu 64
2x2 Max-Pool 64
112 x 112
3x3 Conv + Relu 128
3x3 Conv + Relu 128
2x2 Max-Pool 128
56 x 56
3x3 Conv + Relu 256
3x3 Conv + Relu 256
3x3 Conv + Relu 256
2x2 Max-Pool 256
28 x 28
3x3 Conv + Relu 512
3x3 Conv + Relu 512
3x3 Conv + Relu 512
2x2 Max-Pool 512
14 x 14
3x3 Conv + Relu 512
3x3 Conv + Relu 512
3x3 Conv + Relu 512
2x2 Max-Pool 512
7 X 7
FC 4096
FC 4096
FC OUTPUT 1000
OUTPUT
VGG-16

VGG-16


RGB Images
conv2
conv3
conv4
conv5
fc6
fc7
fc8
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU

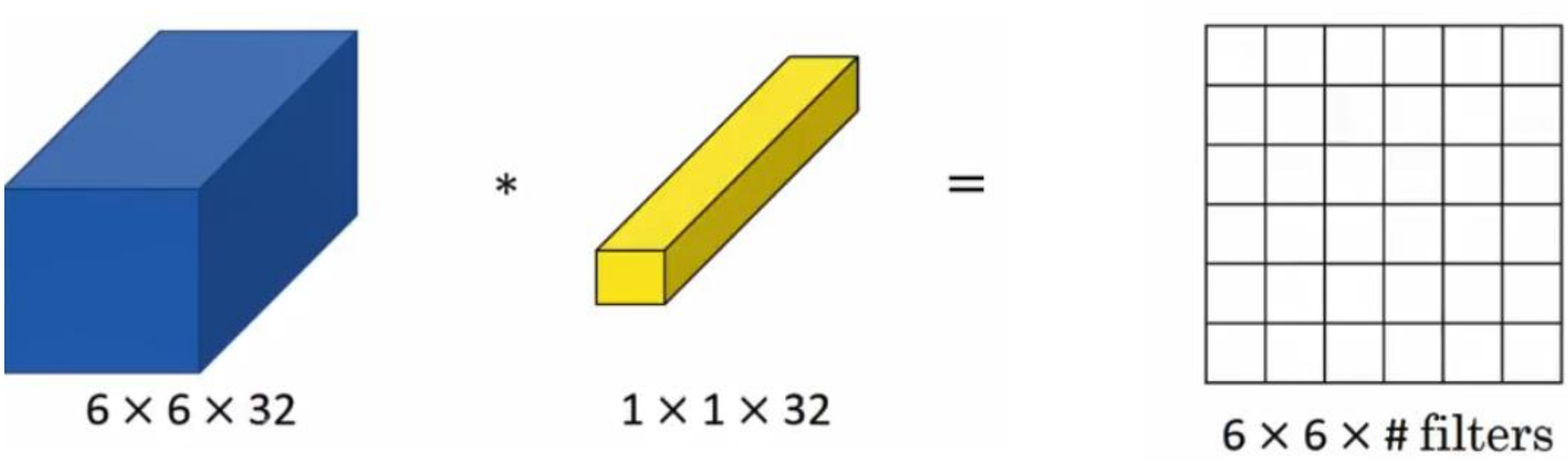
VGG 19

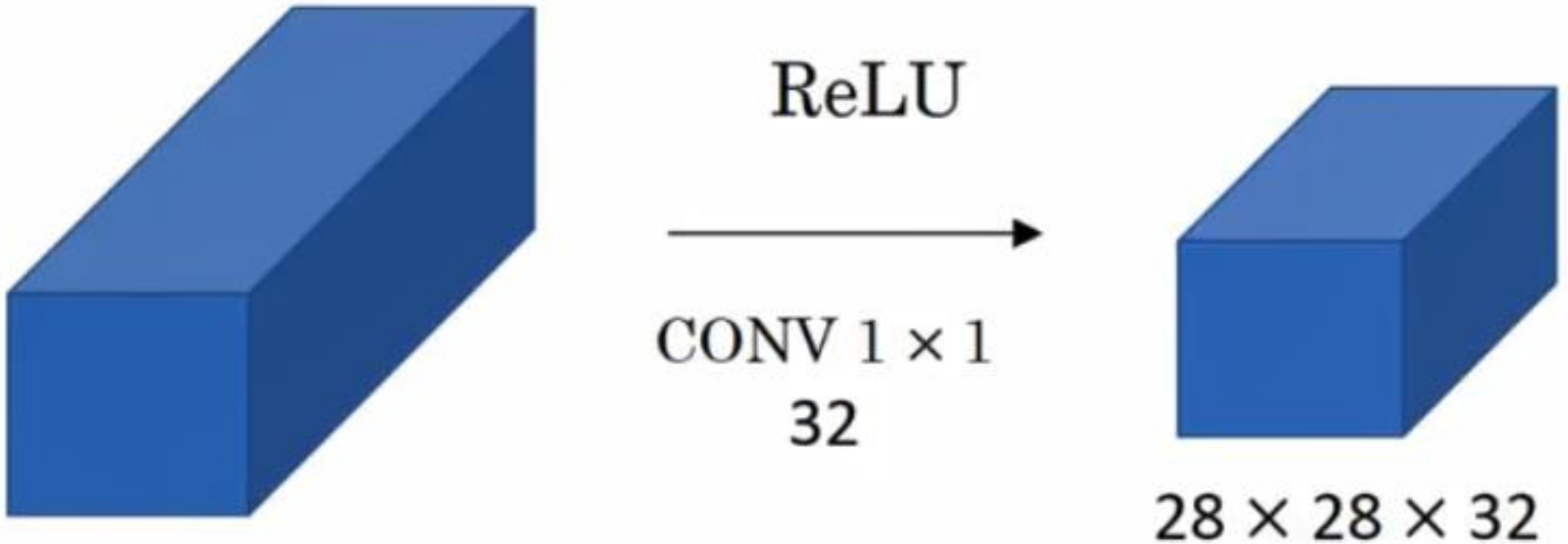

224 X 224 X 64
112 X 112 X 128
56 X 56 X 256
28 X 28 X 512
14 X 14 X 512
7 X 7 X 512
1 X 1 X 4096
1 X 1 X 1000
224 X 224 X 3
Predict whether there exists corona virus or not from the X-ray
Convolution + ReLU
Max pooling
Fully connected + ReLU
Softmax

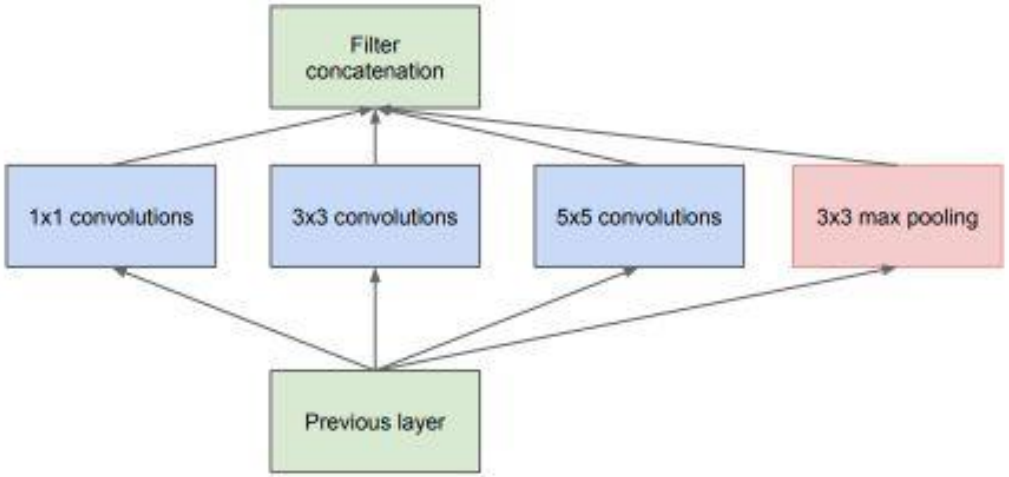
InceptionNet

- Proposed in 2014
- 22-layer architecture
- Uses all 1x1, 3x3, and 5x5 filters and pooling layer all at once and stacks the output
- 1x1 convolutions are used for dimensionality reduction
- Use of inception module
- 4 million parameters

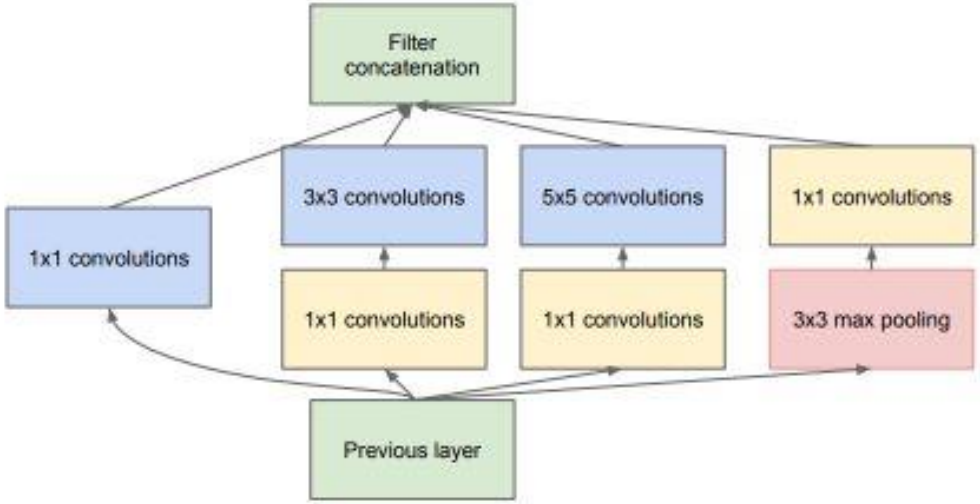
Innovative use of 1X1 Convolution Filter




ReLU
CONV 1 × 1
32
28 × 28 × 192
28 × 28 × 32

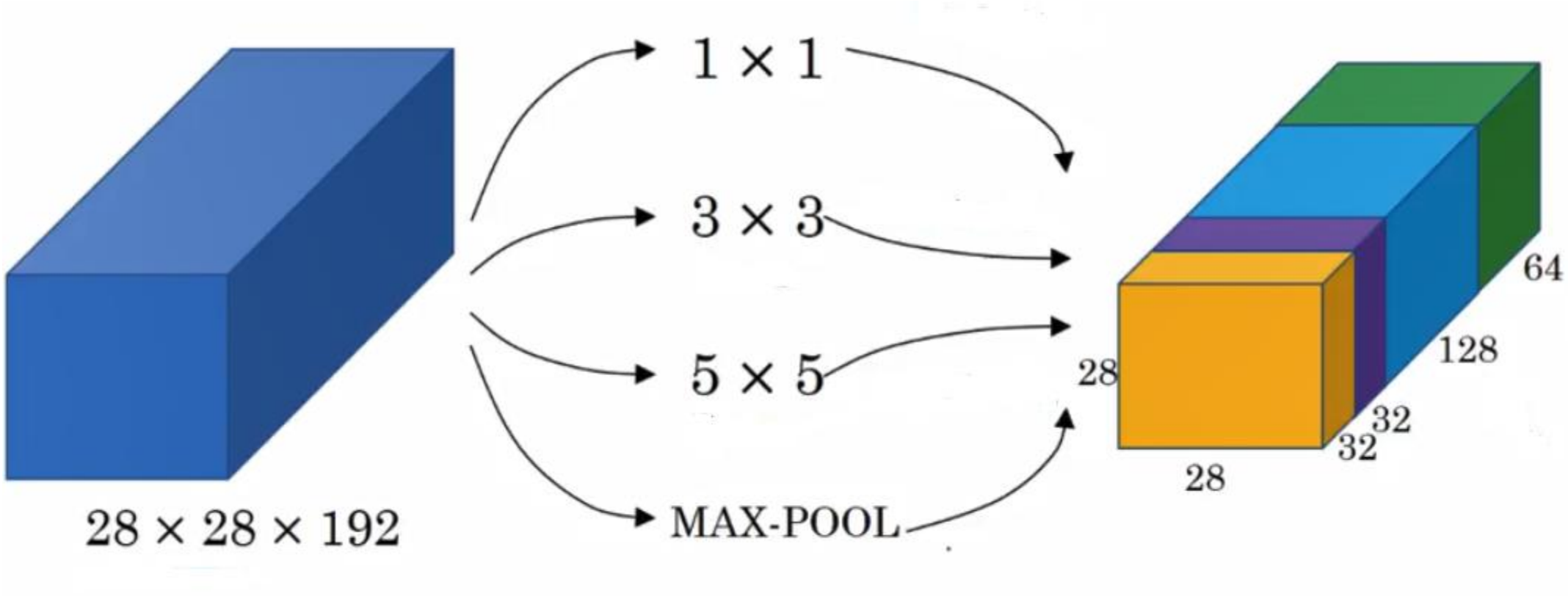


(a) Inception module, naïve version

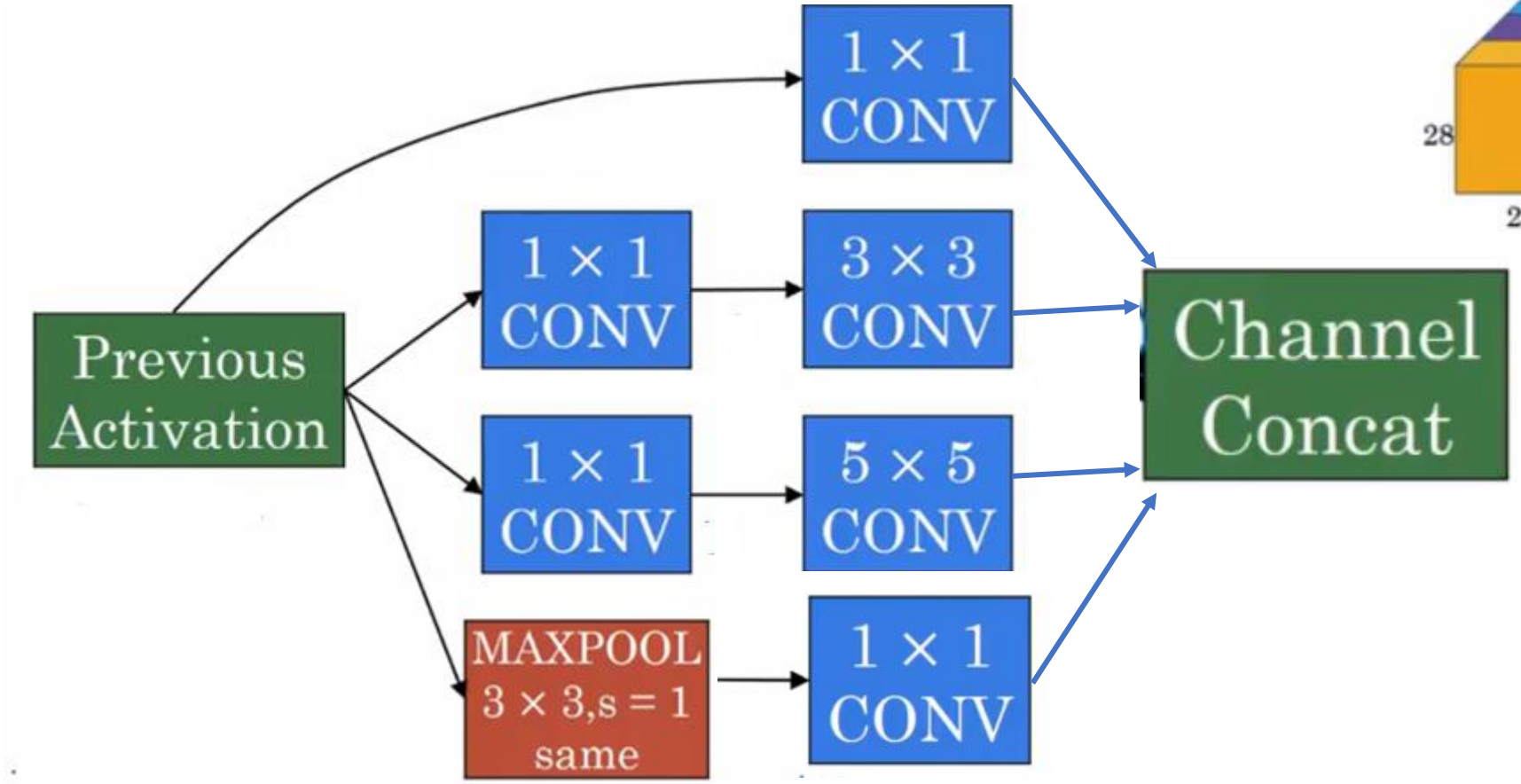


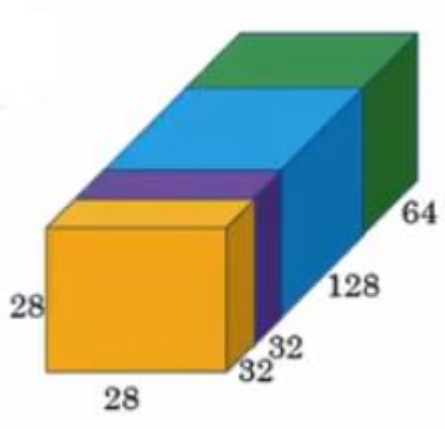
(b) Inception module with dimension reductions

Inception Network

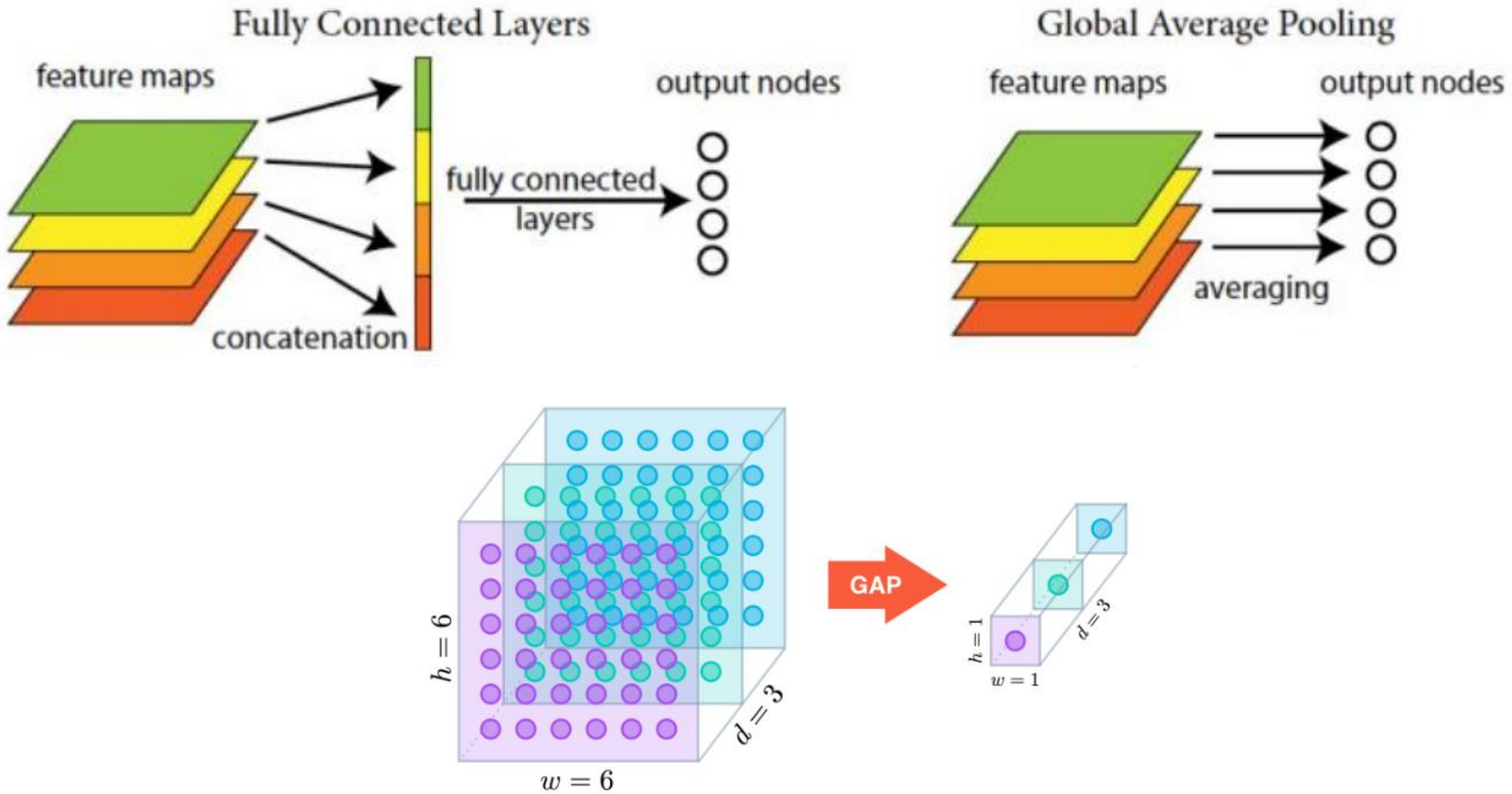


Single Unit of Inception Network





GoogLe Net: Global Average Pooling

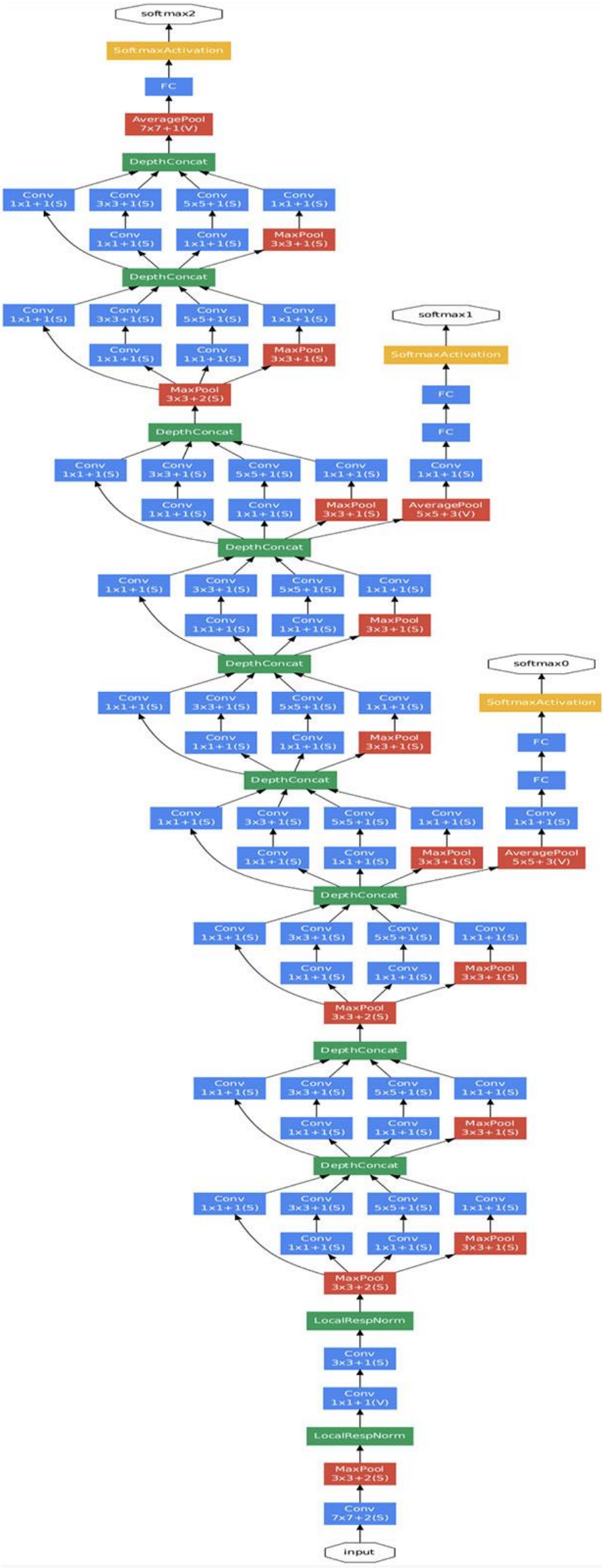


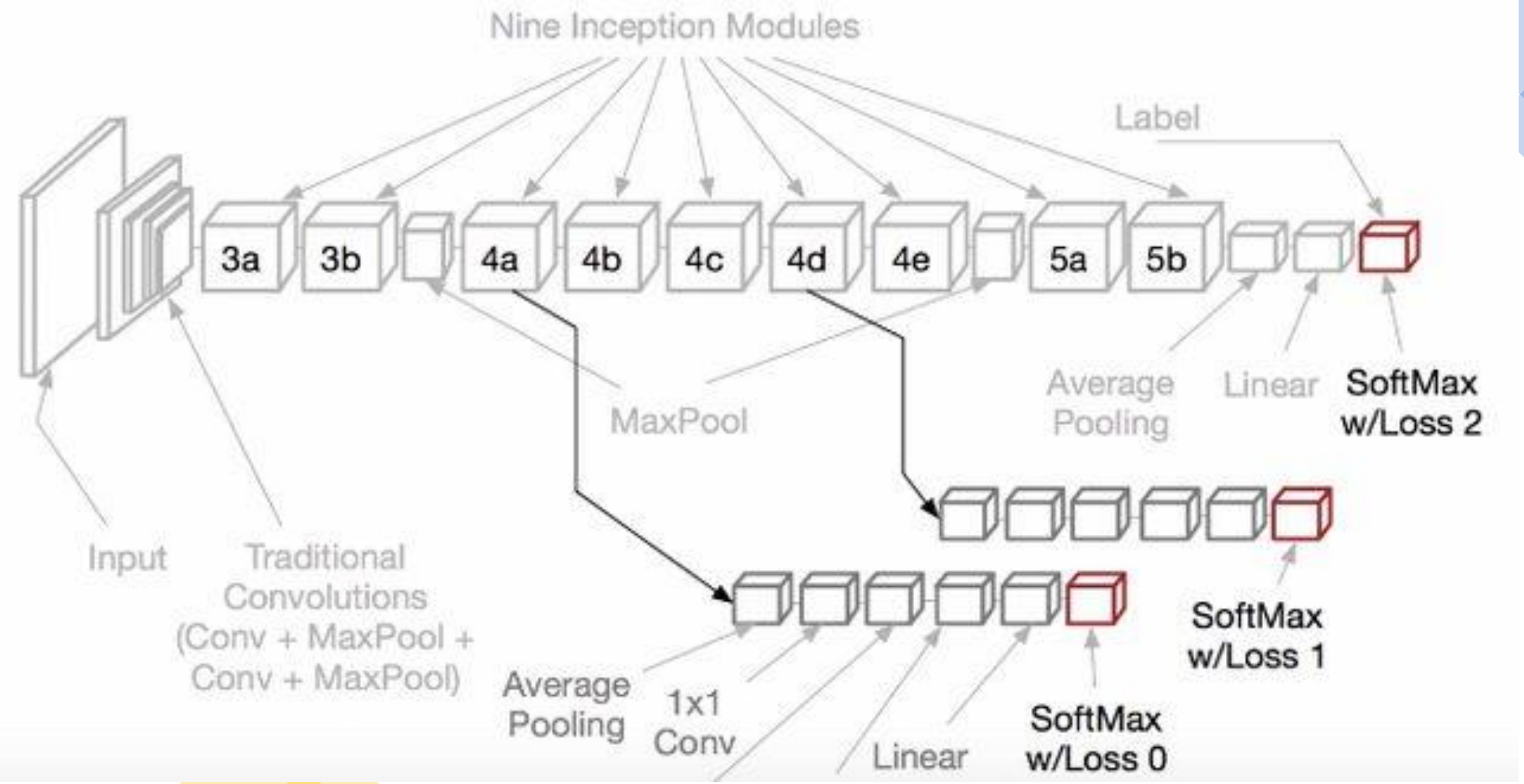
Source: https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/

GoogLe Net: Global Average Pooling

Global Average Pooling as replacement to FC layers:

- An alternative is to use spatial average of feature maps.
- Huge reduction the number of parameters as compared to the Fully Connected layer.
- Stronger local modelling using the micro network.
- It is itself a structural regularizer and hence doesn’t need dropout.


softmax2
SoftmaxActivation
FC
AveragePool 7x7+1(V)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
MaxPool 3x3+2(S)
softmax1
SoftmaxActivation
FC
FC
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
AveragePool 5x5+3(V)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
DepthConcat
softmax0
SoftmaxActivation
FC
FC
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
AveragePool 5x5+3(V)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
MaxPool 3x3+2(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
MaxPool 3x3+2(S)
LocalRespNorm
Conv 3x3+1(S)
Conv 1x1+1(V)
LocalRespNorm
MaxPool 3x3+2(S)
Conv 7x7+2(S)
input


Nine Inception Modules
Label
3a
3b
4a
4b
4c
4d
4e
5a
5b
MaxPool
Average Pooling
Linear
SoftMax w/Loss 2
Input
Traditional Convolutions (Conv + MaxPool + Conv + MaxPool)
Average Pooling
1x1 Conv
Linear
SoftMax w/Loss 0
SoftMax w/Loss 1

Residual Neural Network (ResNet-50)

- Proposed in 2015 by Microsoft
- Total 50 layers
- Trained on ImageNet database
- 25 million+ parameters
- Used skip connections (residual blocks) and batch normalization

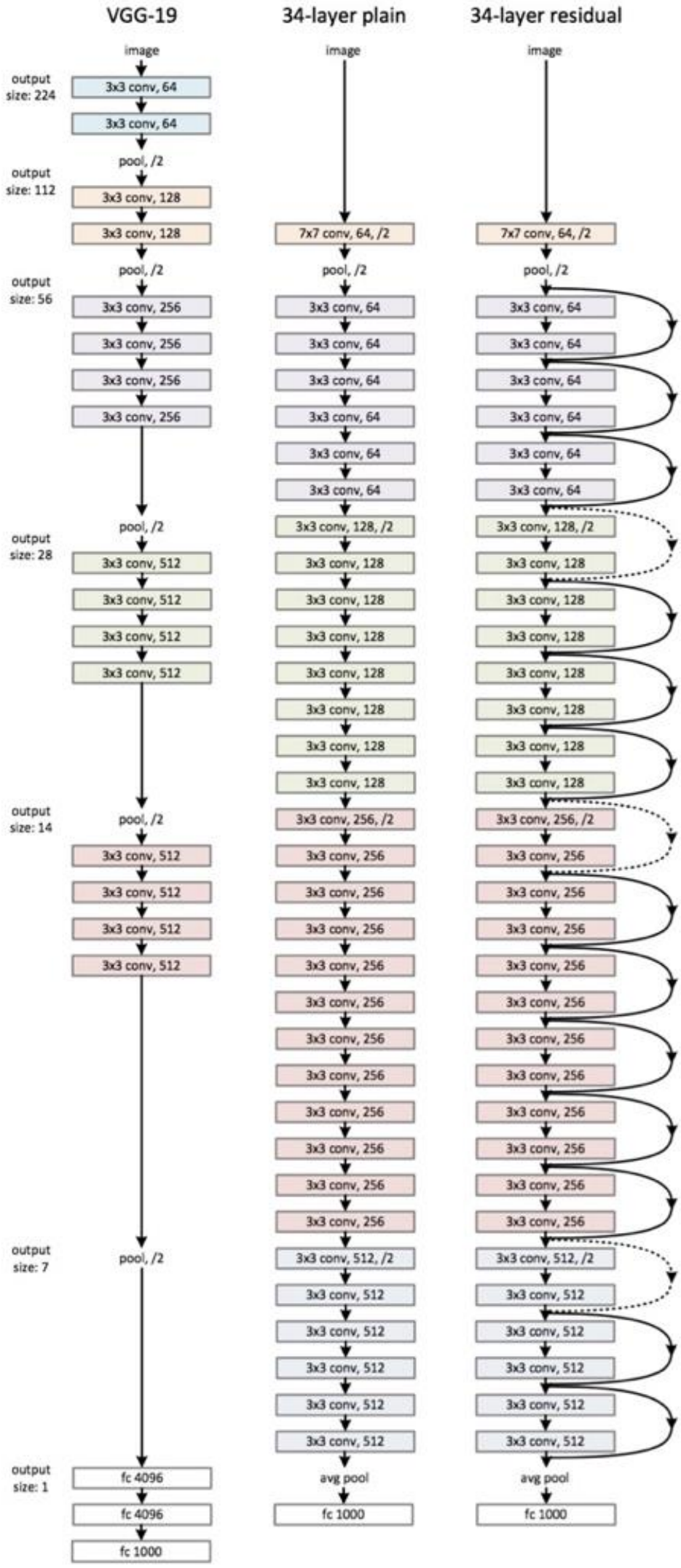


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

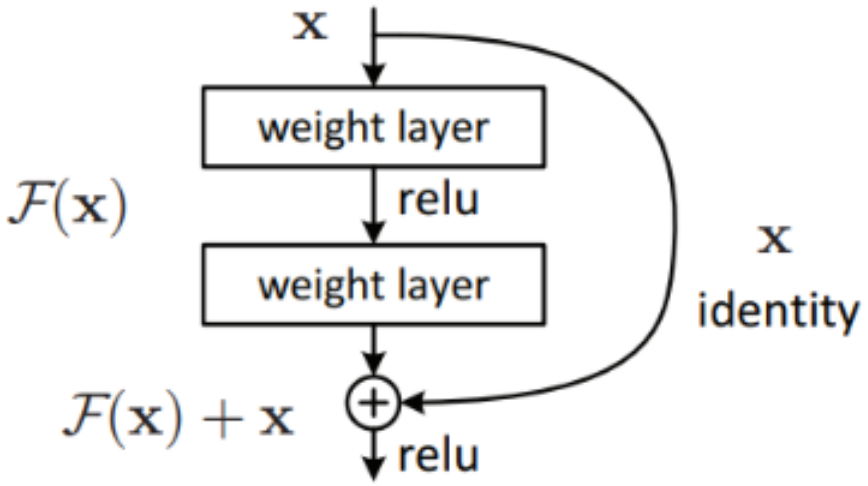
Problems with deep neural networks

- After the first CNN-based architecture AlexNet was proposed, every consecutive winning architecture uses more layers in a deep neural network to lower the error rate. But when we add more layers, a typical deep learning issue known as the Vanishing/Exploding gradient arises.
- Neural networks are universal function approximators and the accuracy increases with increasing the number of layers. But there is a limit to the number of layers added that results in an accuracy improvement. So, if neural networks were universal function approximators, then they should have been able to learn any simple or complex function. But it turns out that, due to problems like vanishing gradients and the curse of dimensionality, if we have sufficiently deep networks, it may not be able to learn simple functions like an identity function which is clearly undesirable.
- To overcome these, ResNet is used which makes use of “SKIP CONNECTION”.

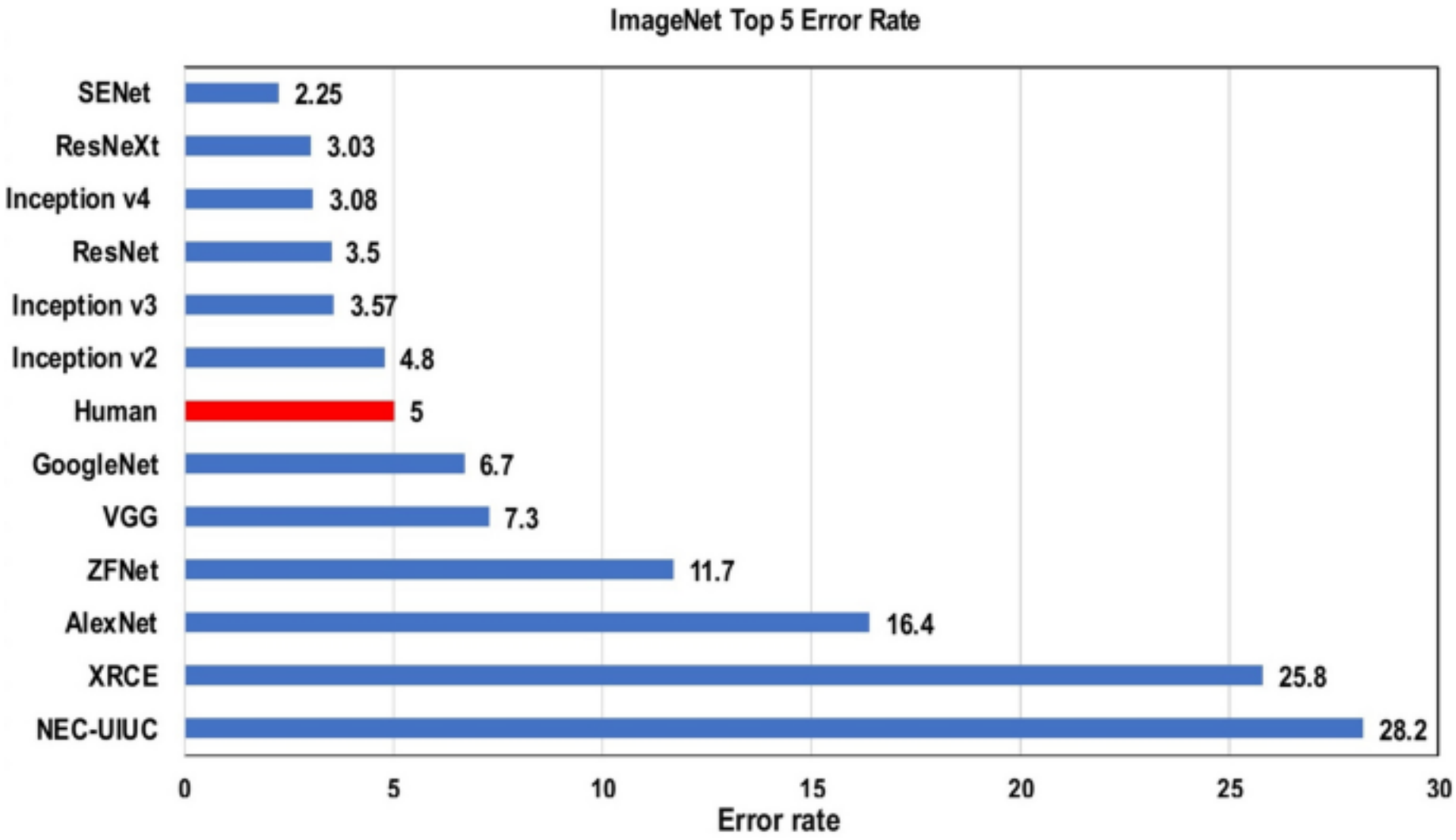
Residual block: A block with skip connections

Skip connection: Add the original input to the output of the convolution block

Skip connections perform identity mapping



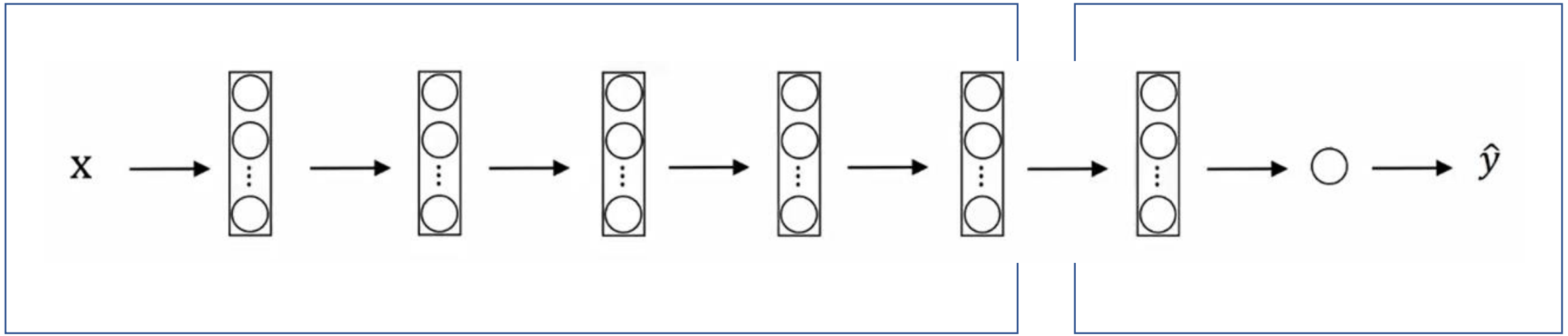
Top 5 error rate



Transfer Learning

- Sharing the Knowledge gained solving one problem and applying to a different but related problem
- Transfer Learning is next popular driver of deep learning after supervised learning
- The feature spaces of the source and target domain are different, e.g. the documents are written in two different languages.
- The marginal probability distributions of source and target domain are different, e.g. the documents discuss different topics.
- Simulation training is becoming a hot area within the sphere of Deep Learning. Few labs have also started using AR/VR Technologies to be integrated for making advance learning models for some of the critical problems area

Transfer Learning – Small Data



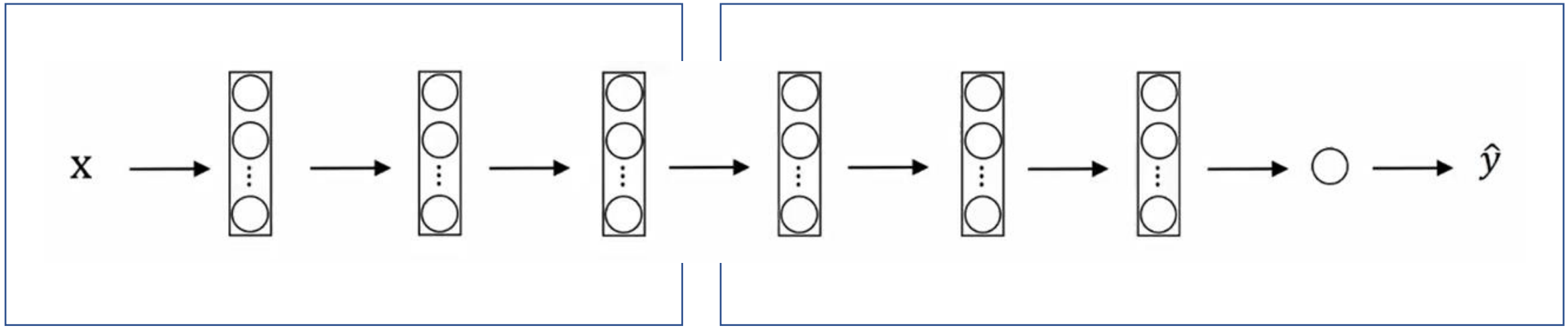
*Finding a Bengal Cat or British Cat where you have small dataset of these categories of cats

*You can download a cat classifier and train last few layers on your data specifically

*In Popular Deep Learning platforms, now you have good support for transfer learning

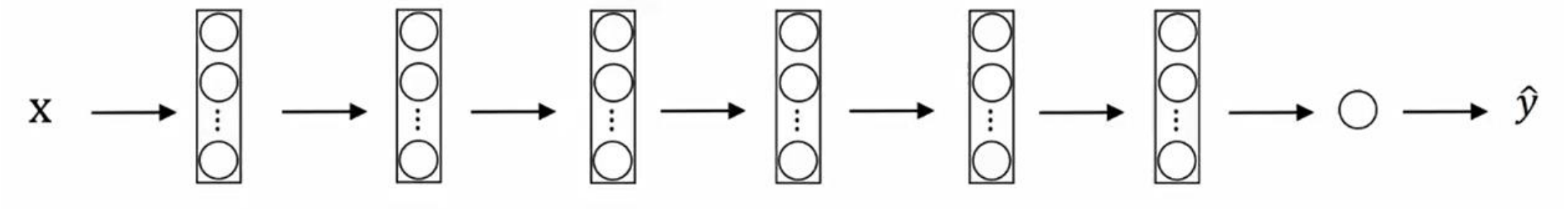
* Functions like Trainable_Parameters and Freezing specific layers are available

Transfer Learning-Mid Size Data



In this category we use initial set of layers from the open source model and use the trained weight values. For the remaining layers there are two ways to handle: a) either we can train the layers from start or b) we can start the weight optimization from the existing set of weights that we have received from the open source model

Transfer Learning- Enough Data



In scenarios, where we have enough data to train our new model, open source model may still be useful. We can use the pre-trained weights of the model from the same domain and consider that as a starting point for our model. It may be much easier and faster to adapt these model for new set of data that we want to train upon.