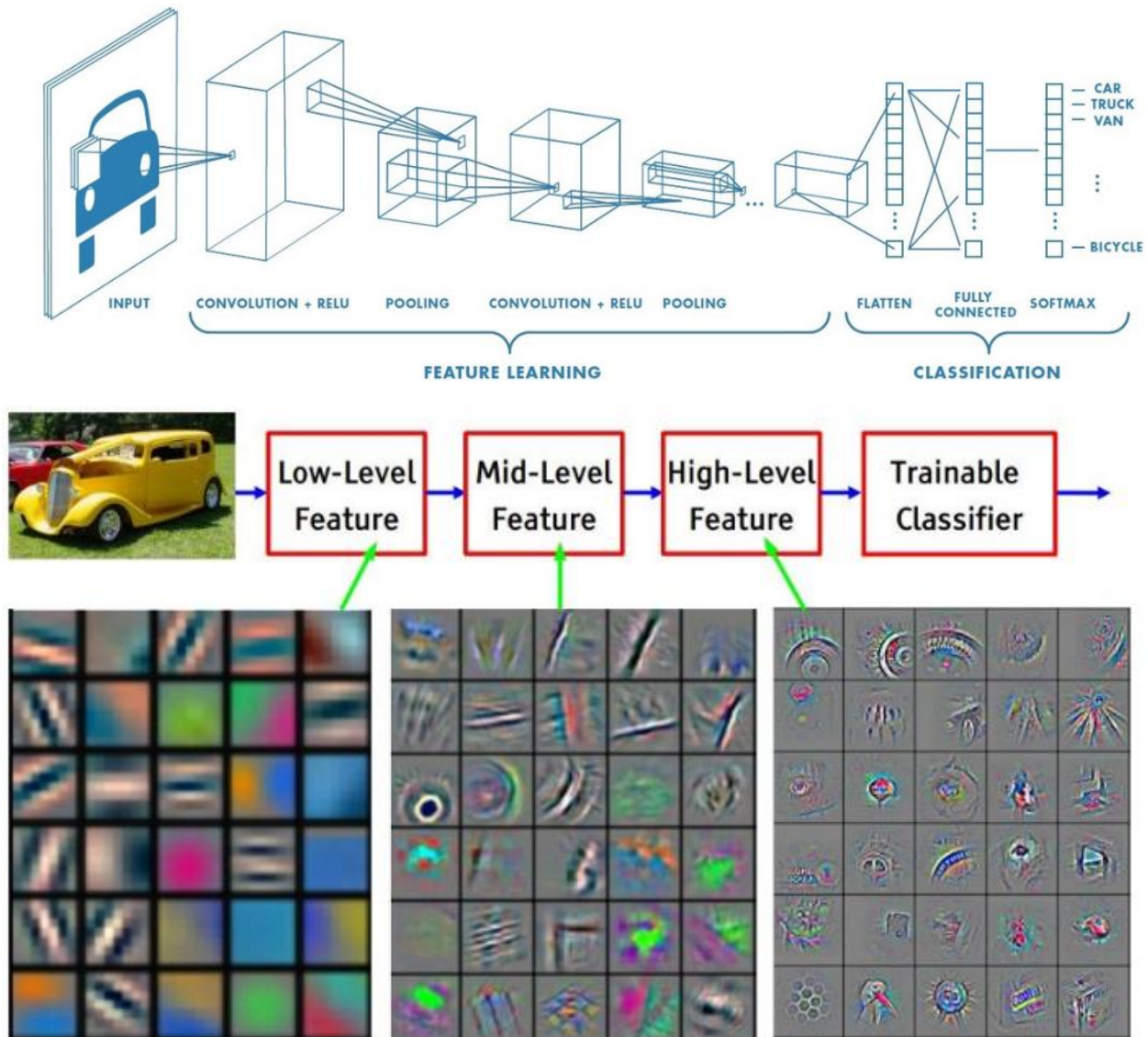


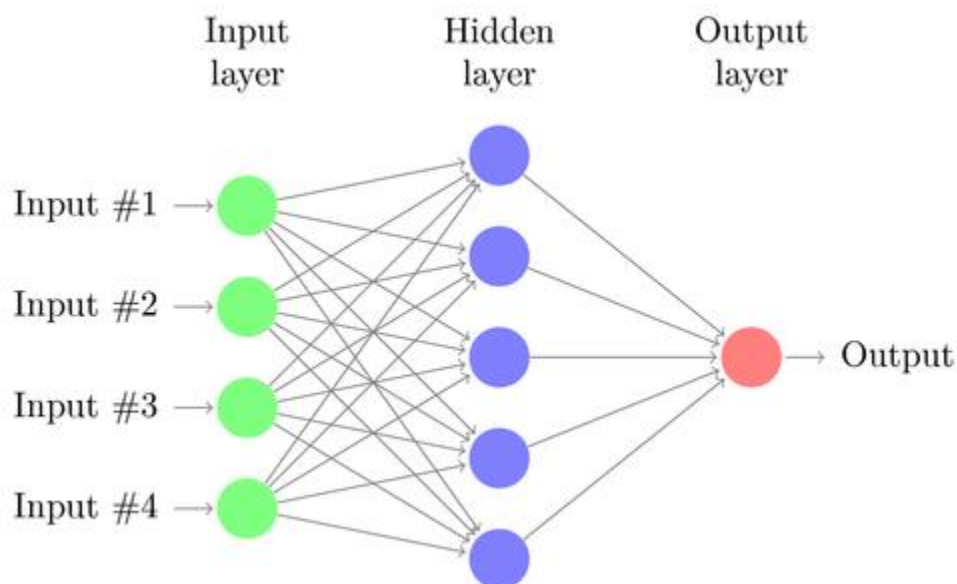
Convolutional Neural Network



Different features recognised at different layers

CNN do Representation Learning (or Feature Learning). Representation Learning is a technique that allows a system to automatically find relevant features for a given task. Replaces manual feature engineering.

Why not Traditional NN?





There are several drawbacks of MLP's, especially when it comes to image. MLPs use one perceptron for each input (e.g. pixel in an image, multiplied by 3 in RGB case). The amount of weights rapidly becomes unmanageable for large images. For a 224 x 224-pixel image with 3 color channels there are around 150,000 weights that must be trained! As a result, difficulties arise whilst training and overfitting can occur.

Another main problem is that spatial information is lost when the image is flattened into an MLP. Nodes that are close together are important because they help to define the features of an image. We thus need a way to leverage the spatial correlation of the image features (pixels) in such a way that we can see the cat in our picture no matter where it may appear.

Convolution Operation


Edge detection



 \ast

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
 $=$


Kernel

Sharpen


 \ast

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$
 $=$


If you are wondering how the different features are learned by the network, and whether it is possible that the network will learn the same features (having 10 nose filters would be kind of redundant), this is highly unlikely to happen. When building the network, we randomly specify values for the filters, which then continuously update themselves as the network is trained. It is very very unlikely that two filters that are the same will be produced unless the number of chosen filters is extremely large.




10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

 \ast

1	0	-1
1	0	-1
1	0	-1

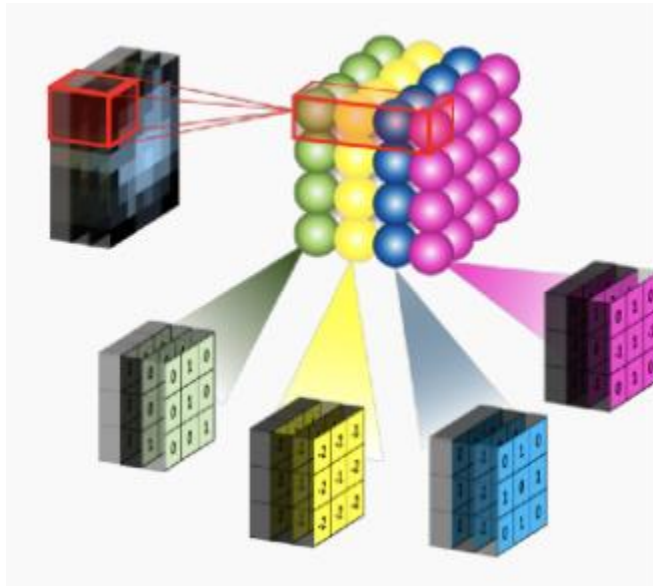
 $=$

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

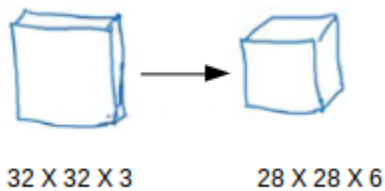

 \ast

 $=$


Note: Higher pixel values represent the brighter portion of the image and the lower pixel values represent the darker portions. This is how we can detect a vertical edge in an image.

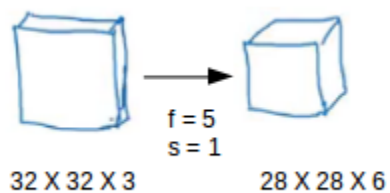
Because we have multiple filters, we end up with a 3D output: one 2D feature map per filter



Why convolutions?



If I was to create a fully connected NN with the above, the number of parameters in the first layer itself be ~ 14 million. But if I have a conv op as below



then number of parameters will be 456. And the reason that conv net has a relatively small parameter is that because it has **parameter sharing**.

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

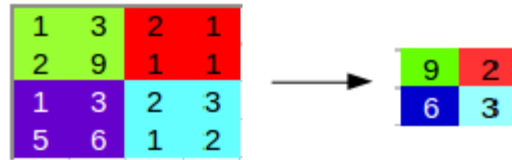
4		

Convolved
Feature

The second is **sparsity of connection**. ie if I take a grid in the output it only depend on a particular 3x3 part of the input image. So only those 9 pixels affect that part of the output and no other pixel affects at that part of the output.

Pooling layer

Other than convolutional layers, CNN also have pooling layers to reduce the size of representation and to speed up the computations. One interesting thing about pooling is it has a set of hyperparameters to learn, but no parameters to learn. There is actually nothing for Gradient descent to learn. Ones you fix the window size and stride, it is just a fixed computation and GD does not change anything.

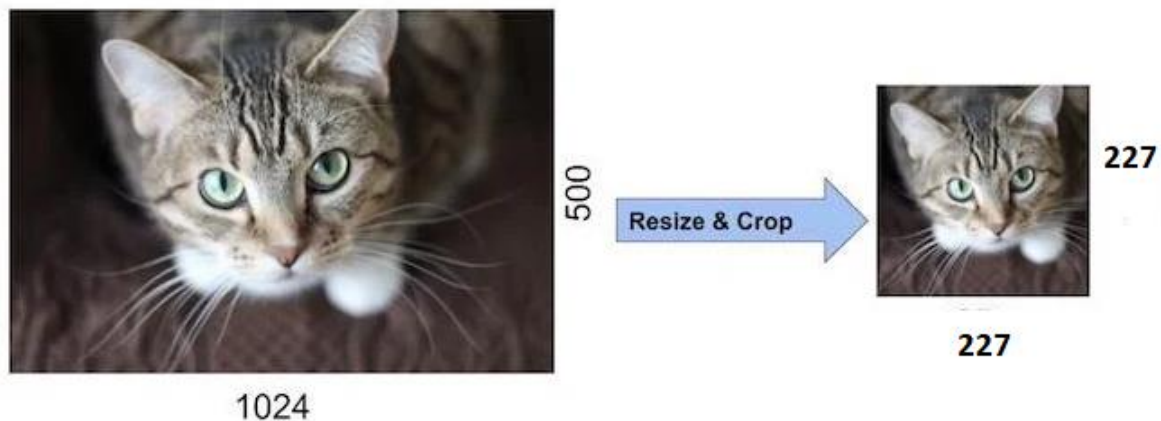


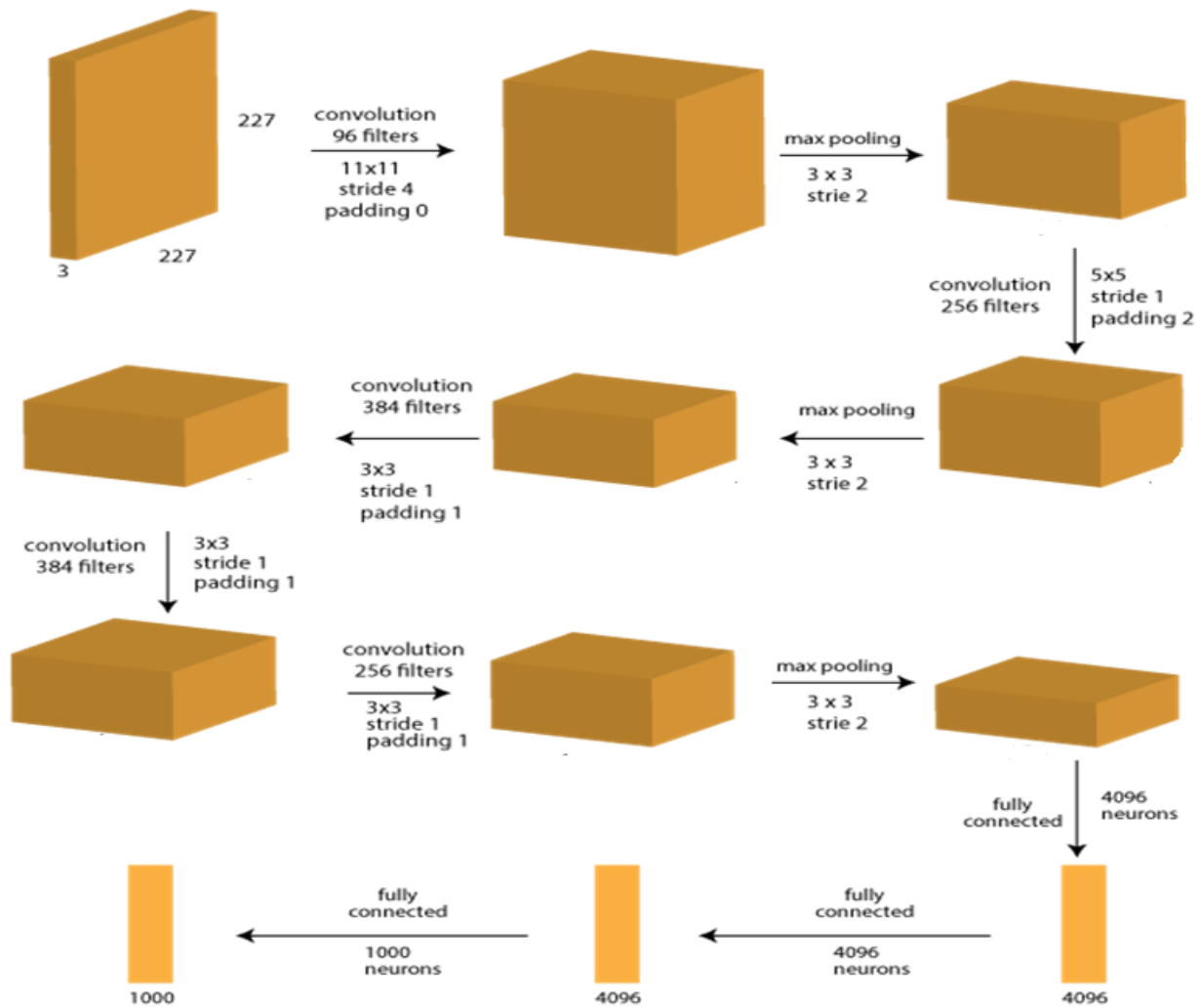
Common Pattern in a CNN

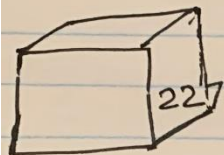
1. As we go deeper in a CNN, usually n_h and n_w will decrease, whereas the number of channel will increase.
2. In a CNN we have CONV layer followed by POOL layer, and then followed by one or more CONV+POOL layer.
3. At the end we have a fully connected layers and then followed by maybe a softmax.
4. A lot of work in designing CNN is selecting hyperparameters like number of filters, filter size, stride, padding.

AlexNet

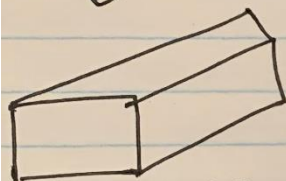
AlexNet was the winning entry in ILSVRC 2012. It solves the problem of image classification where the input is an image of one of 1000 different classes (e.g. cats, dogs etc.) and the output is a vector of 1000 numbers. The i th element of the output vector is interpreted as the probability that the input image belongs to the i th class. Therefore, the sum of all elements of the output vector is 1.







227

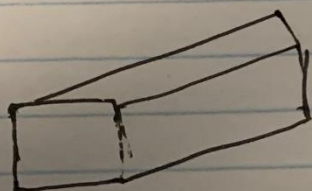


55x55x96



POOL

3x3, S=2



27x27x96

filter = 96

11x11

Stride = 4

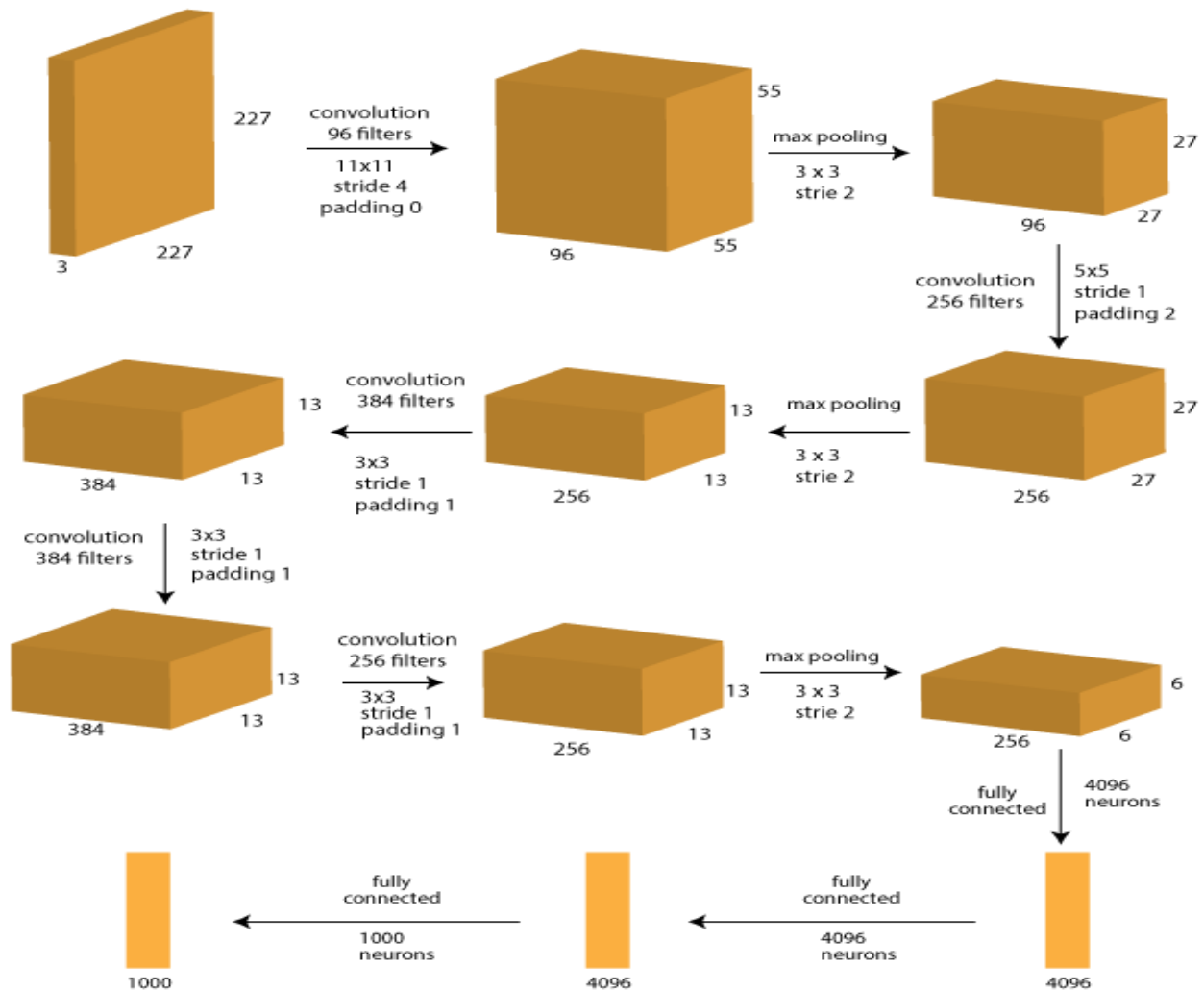
Padding = 0

$$\left\lfloor \frac{n + 2p - s}{s} + 1 \right\rfloor$$

$$\left\lfloor \frac{227 + 2 \times 0 - 11}{4} + 1 \right\rfloor = 55$$

$$\left\lfloor \frac{55 + 2 \times 0 - 3}{2} + 1 \right\rfloor = 27$$

Nilroy



In AlexNet, the input is an image of size $227 \times 227 \times 3$. After Conv-1, the size of changes to $55 \times 55 \times 96$ which is transformed to $27 \times 27 \times 96$ after MaxPool-1. After Conv-2, the size changes to $27 \times 27 \times 256$ and following MaxPool-2 it changes to $13 \times 13 \times 256$. Conv-3 transforms it to a size of $13 \times 13 \times 384$, while Conv-4 preserves the size and Conv-5 changes the size back go $27 \times 27 \times 256$. Finally, MaxPool-3 reduces the size to $6 \times 6 \times 256$. This image feeds into FC-1 which transforms it into a vector of size 4096×1 . The size remains unchanged through FC-2, and finally, we get the output of size 1000×1 after FC-3.

Number of Parameters of a Conv Layer

In a CNN, each layer has two kinds of parameters : **weights** and **biases**. The total number of parameters is just the sum of all weights and biases.

Let's define,

W_c = Number of weights of the Conv Layer.

B_c = Number of biases of the Conv Layer.

P_c = Number of parameters of the Conv Layer.

K = Size (width) of kernels used in the Conv Layer.

N = Number of kernels.

C = Number of channels of the input image.

$$W_c = K^2 \times C \times N$$

$$B_c = N$$

$$P_c = W_c + B_c$$

In a Conv Layer, the depth of every kernel is always equal to the number of channels in the input image. So every kernel has $K^2 \times C$ parameters, and there are N such kernels. That's how we come up with the above formula.

Example: In AlexNet, at the first Conv Layer, the number of channels (C) of the input image is 3, the kernel size (K) is 11, the number of kernels (N) is 96. So the number of parameters is given by

$$W_c = 11^2 \times 3 \times 96 = 34,848$$

$$B_c = 96$$

$$P_c = 34,848 + 96 = 34,944$$

Readers can verify the number of parameters for Conv-2, Conv-3, Conv-4, Conv-5 are 614656 , 885120, 1327488 and 884992 respectively. The total number of parameters for the Conv Layers is therefore 3,747,200. Think this is a large number? Well, wait until we see the fully connected layers. One of the benefits of the Conv Layers is that weights are shared and therefore we have fewer parameters than we would have in case of a fully connected layer.

Number of Parameters of a MaxPool Layer

There are no parameters associated with a MaxPool layer. The pool size, stride, and padding are hyperparameters.

Number of Parameters of a Fully Connected (FC) Layer

There are two kinds of fully connected layers in a CNN. The first FC layer is connected to the last Conv Layer, while later FC layers are connected to other FC layers. Let's consider each case separately.

Case 1: Number of Parameters of a Fully Connected (FC) Layer connected to a Conv Layer

Let's define,

W_{cf} = Number of weights of a FC Layer which is connected to a Conv Layer.

B_{cf} = Number of biases of a FC Layer which is connected to a Conv Layer.

O = Size (width) of the output image of the previous Conv Layer.

N = Number of kernels in the previous Conv Layer.

F = Number of neurons in the FC Layer.

$$W_{cf} = O^2 \times N \times F$$

$$B_{cf} = F$$

$$P_{cf} = W_{cf} + B_{cf}$$

Example: The first fully connected layer of AlexNet is connected to a Conv Layer. For this layer, $O = 6$, $N = 256$ and $F = 4096$. Therefore,

$$W_{cf} = 6^2 \times 256 \times 4096 = 37,748,736$$

$$B_{cf} = 4096$$

$$P_{cf} = W_{cf} + B_{cf} = 37,752,832$$

That's an order of magnitude more than the total number of parameters of all the Conv Layers combined!

Case 2: Number of Parameters of a Fully Connected (FC) Layer connected to a FC Layer

Let's define,

W_{ff} = Number of weights of a FC Layer which is connected to an FC Layer.

B_{ff} = Number of biases of a FC Layer which is connected to an FC Layer.

P_{ff} = Number of parameters of a FC Layer which is connected to an FC Layer.

F = Number of neurons in the FC Layer.

F_{-1} = Number of neurons in the previous FC Layer.

$$W_{ff} = F_{-1} \times F$$

$$B_{ff} = F$$

$$P_{ff} = W_{ff} + B_{ff}$$

In the above equation, $F_{-1} \times F$ is the total number of connection weights from neurons of the previous FC Layer to the neurons of the current FC Layer. The total number of biases is the same as the number of neurons (F).

Example: The last fully connected layer of AlexNet is connected to an FC Layer. For this layer, $F_{-1} = 4096$ and $F = 1000$. Therefore,

$$W_{ff} = 4096 \times 1000 = 4,096,000$$

$$B_{ff} = 1,000$$

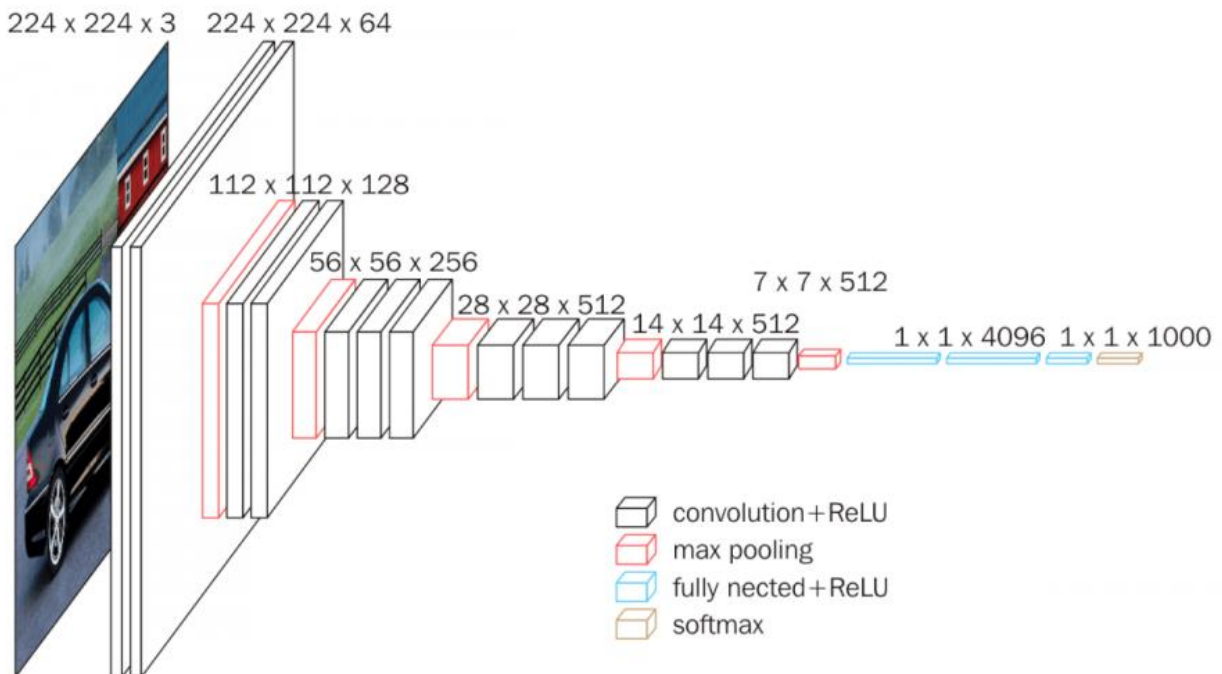
$$P_{ff} = W_{ff} + B_{ff} = 4,097,000$$

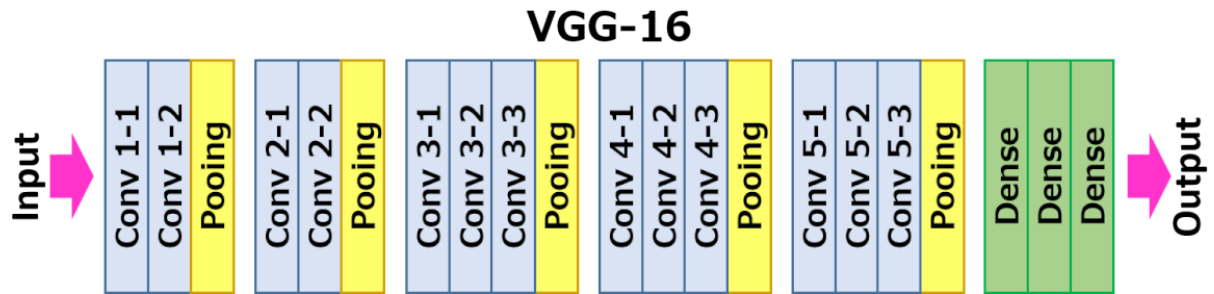
Layer Name	Tensor Size	Weights	Biases	Parameters
Input Image	227x227x3	0	0	0
Conv-1	55x55x96	34,848	96	34,944
MaxPool-1	27x27x96	0	0	0
Conv-2	27x27x256	614,400	256	614,656
MaxPool-2	13x13x256	0	0	0
Conv-3	13x13x384	884,736	384	885,120
Conv-4	13x13x384	1,327,104	384	1,327,488
Conv-5	13x13x256	884,736	256	884,992
MaxPool-3	6x6x256	0	0	0
FC-1	4096×1	37,748,736	4,096	37,752,832
FC-2	4096×1	16,777,216	4,096	16,781,312
FC-3	1000×1	4,096,000	1,000	4,097,000
Output	1000×1	0	0	0
Total				62,378,344

LeNet-5 Architecture

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

VGG16





Conv 1 has number of filters as 64 while Conv 2 has 128 filters, Conv 3 has 256 filters while Conv 4 and Conv 5 has 512 filters.