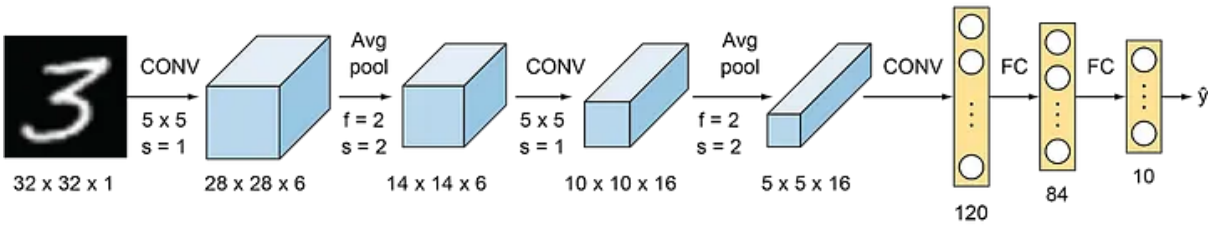
1. LeNet-5

This is also known as the Classic Neural Network that was designed by Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner for handwritten and machine-printed character recognition in 1990's which they called LeNet-5. The architecture was designed to identify handwritten digits in the MNIST data-set. The architecture is pretty straightforward and simple to understand. The input images were gray scale with dimension of 32*32*1 followed by two pairs of Convolution layer with stride 2 and Average pooling layer with stride 1. Finally, fully connected layers with Softmax activation in the output layer. Traditionally, this network had 60,000 parameters in total. Refer to the original paper.

- Paper link: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf
- Notebook: link

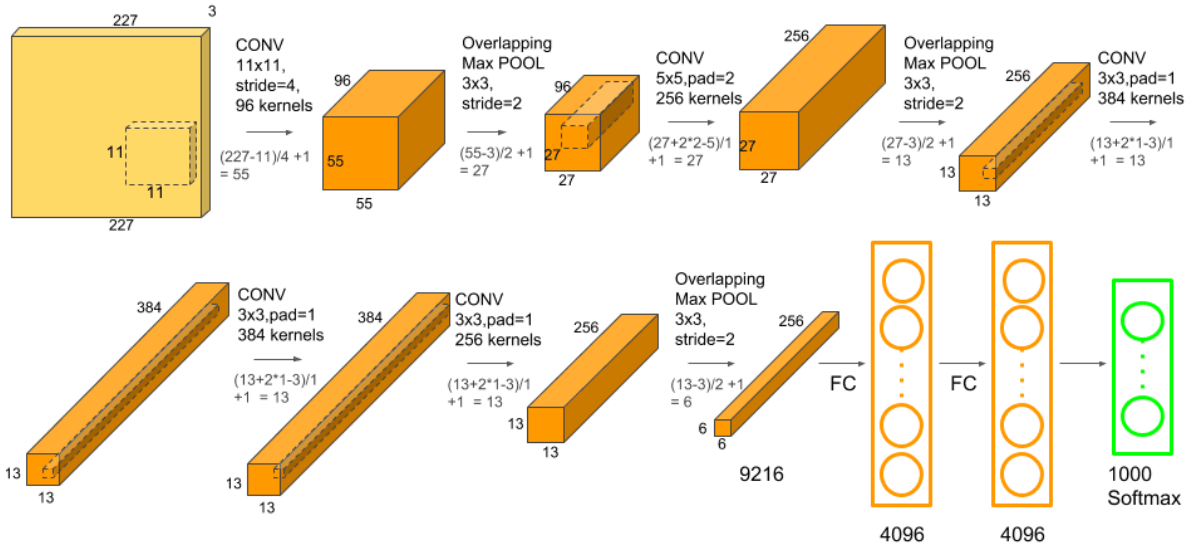


Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

2. AlexNet

AlexNet was born out of the need to improve the results of the ImageNet challenge. This was one of the first Deep convolutional networks to achieve considerable accuracy on the 2012 ImageNet LSVRC-2012 challenge with an accuracy of 84.7% as compared to the second-best with an accuracy of 73.8%. The idea of spatial correlation in an image frame was explored using convolutional layers and receptive fields.

The network consists of 5 Convolutional (CONV) layers and 3 Fully Connected (FC) layers. The activation used is the Rectified Linear Unit (ReLU). The structural details of each layer in the network can be found in the table below.



Parameters table:

AlexNet Network - Structural Details													
Input			Output			Layer	Stride	Pad	Kernel size		in	out	# of Param
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
Total													62,378,344

The network has a total of 62 million trainable variables

The input to the network is a batch of RGB images of size 227x227x3 and outputs a 1000x1 probability vector one corresponding to each class.

- Paper link: Link
- Notebook: link

Some general ideas:

- Data augmentation is carried out to reduce over-fitting. This Data augmentation includes mirroring and cropping the images to increase the variation in the training data-set. The network uses an overlapped max-pooling layer after the first, second, and fifth CONV layers. Overlapped maxpool layers are simply maxpool layers with strides less than the window size. 3x3 maxpool layer is used with a stride of 2 hence creating overlapped receptive fields. This overlapping improved the top-1 and top-5 errors by 0.4% and 0.3%, respectively.

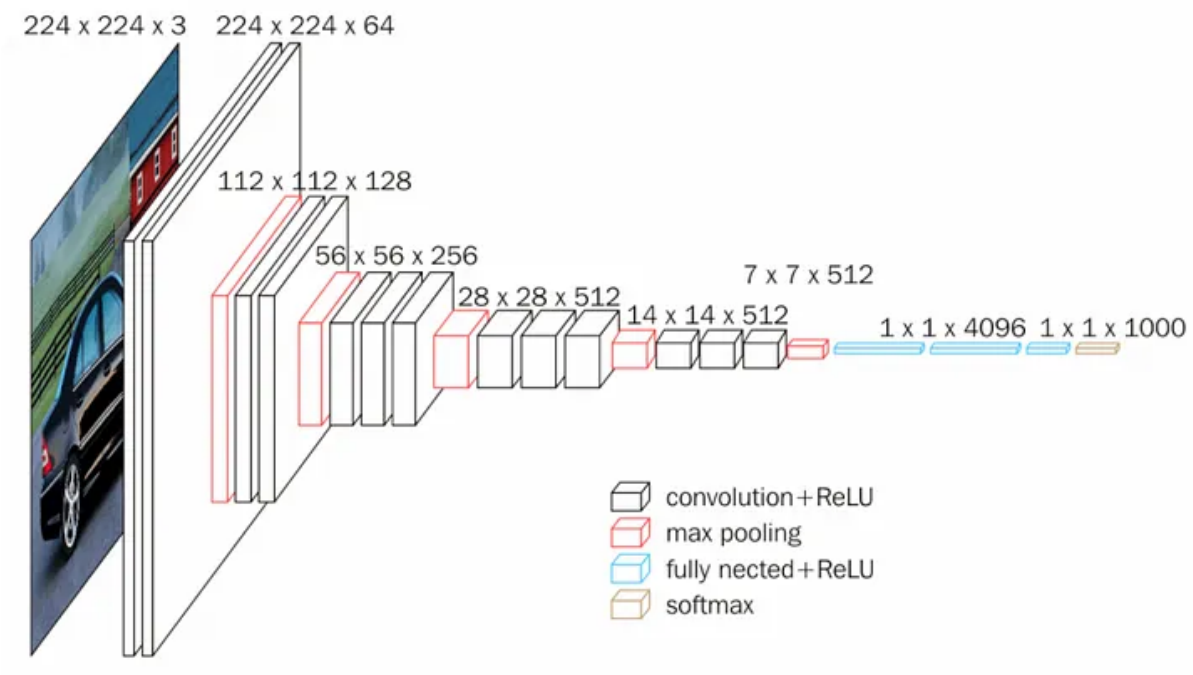
- Before AlexNet, the most commonly used activation functions were sigmoid and tanh. Due to the saturated nature of these functions, they suffer from the Vanishing Gradient (VG) problem and make it difficult for the network to train. AlexNet uses the ReLU activation function which doesn't suffer from the VG problem. The original paper showed that the network with ReLU achieved a 25% error rate about 6 times faster than the same network with tanh non-linearity.

- Although ReLU helps with the vanishing gradient problem, due to its unbounded nature, the learned variables can become unnecessarily high. To prevent this, AlexNet introduced Local Response Normalization (LRN). The idea behind LRN is to carry out a normalization in a neighborhood of pixels amplifying the excited neuron while dampening the surrounding neurons at the same time.

- AlexNet also addresses the over-fitting problem by using drop-out layers where a connection is dropped during training with a probability of p=0.5. Although this avoids the network from over-fitting by helping it escape from bad local minima, the number of iterations required for convergence is doubled too.

3. VGGNet

VGGNet was born out of the need to reduce the # of parameters in the CONV layers and improve on training time. There are multiple variants of VGGNet (VGG16, VGG19, etc.) which differ only in the total number of layers in the network. The structural details of a VGG16 network have been shown below.

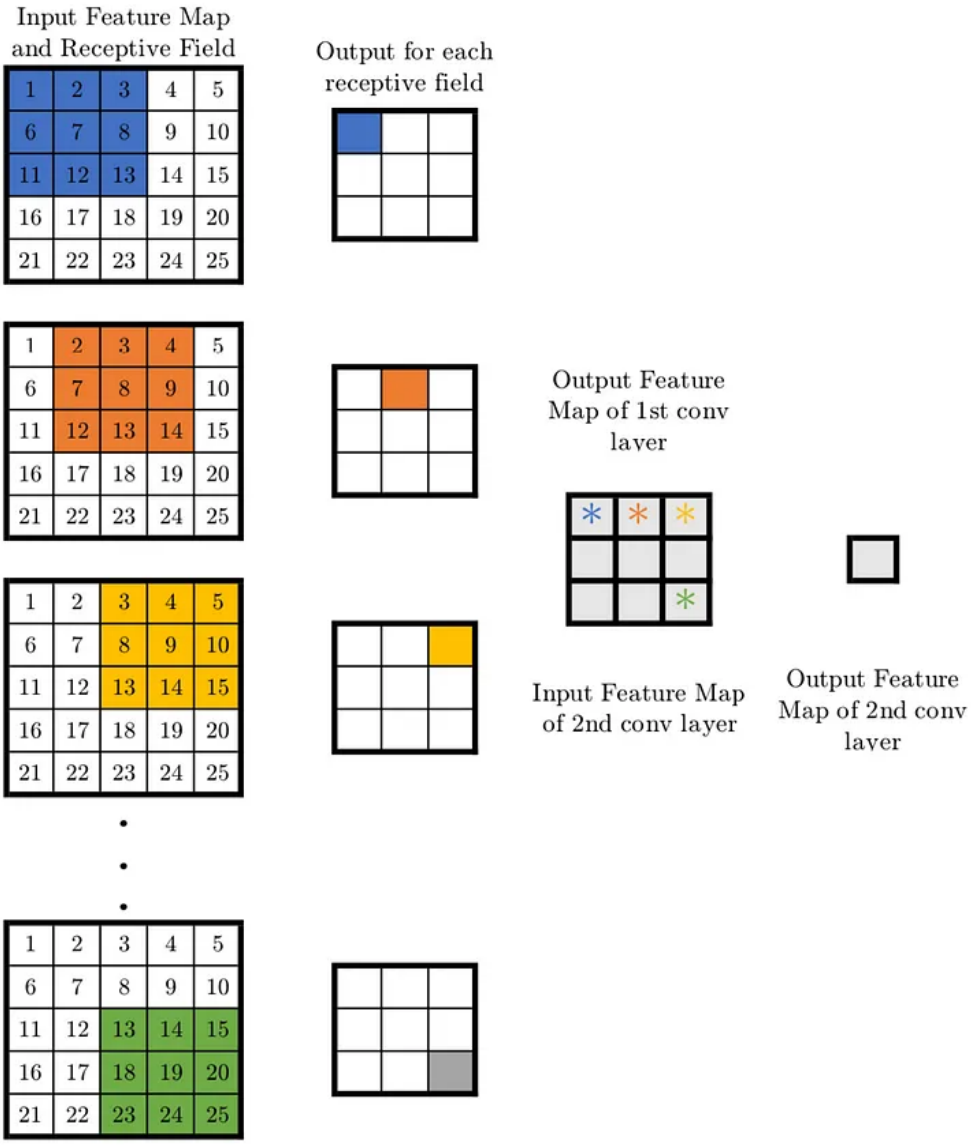


Parameters table:

VGG16 - Structural Details													
#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
Total													138,423,208

VGG16 has a total of 138 million parameters. The important point to note here is that all the conv kernels are of size 3x3 and maxpool kernels are of size 2x2 with a stride of two. The idea behind having fixed size kernels is that all the variable size convolutional kernels used in Alexnet (11x11, 5x5, 3x3) can be replicated by making use of multiple 3x3 kernels as building blocks. The replication is in terms of the receptive field covered by the kernels.

Let's consider the following example. Say we have an input layer of size 5x5x1. Implementing a conv layer with a kernel size of 5x5 and stride one will result in an output feature map of 1x1. The same output feature map can be obtained by implementing two 3x3 conv layers with a stride of 1 as shown below



look at the number of variables needed to be trained. For a 5x5 conv layer filter, the number of variables is 25. On the other hand, two conv layers of kernel size 3x3 have a total of 3x3x2=18 variables (a reduction of 28%).

Similarly, the effect of one 7x7 (11x11) conv layer can be achieved by implementing three (five) 3x3 conv layers with a stride of one. This reduces the number of trainable variables by 44.9% (62.8%). A reduced number of trainable variables means faster learning and more robust to over-fitting.

- Paper link: Link
- Notebook: link

4. ResNet

Neural Networks are notorious for not being able to find a simpler mapping when it exists.

- For example, say we have a fully connected multi-layer perceptron network and we want to train it on a data-set where the input equals the output. The simplest solution to this problem is having all weights equaling one and all biases zeros for all the hidden layers. But when such a network is trained using back-propagation, a rather complex mapping is learned where the weights and biases have a wide range of values.

- Another example is adding more layers to an existing neural network. Say we have a network f(x) that has achieved an accuracy of n% on a data-set. Now adding more layers to this network g(f(x)) should have at least an accuracy of n% i.e. in the worst case g(.) should be an identical mapping yielding the same accuracy as that of f(x) if not more. But unfortunately, that is not the case. Experiments have shown that the accuracy decreases by adding more layers to the network.

- The issues mentioned above happens because of the vanishing gradient problem. As we make the CNN deeper, the derivative when back-propagating to the initial layers becomes almost insignificant in value.

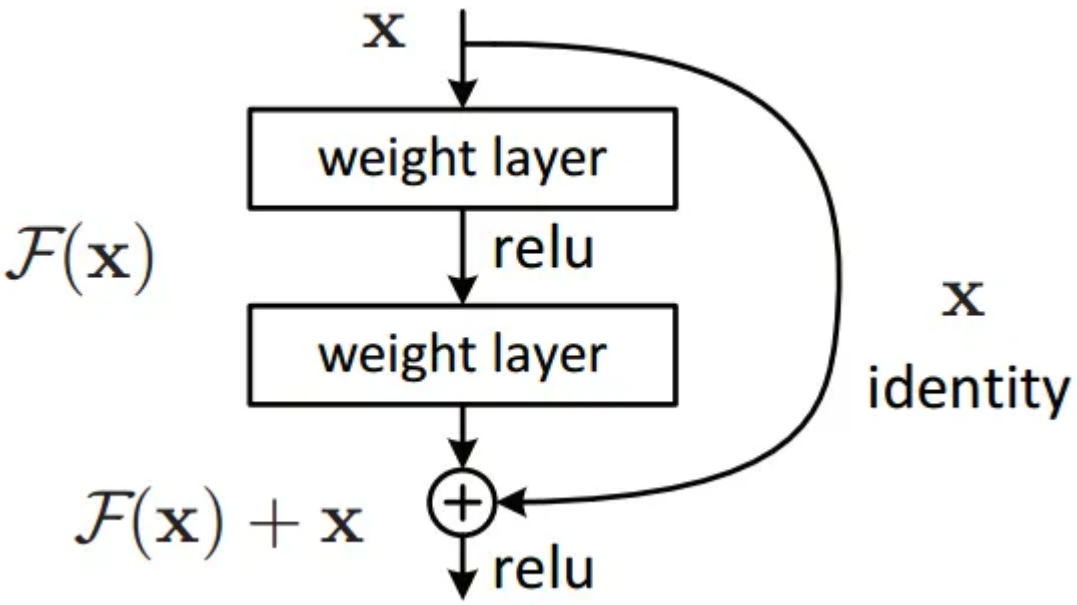
ResNet addresses this network by introducing two types of ‘shortcut connections’: Identity shortcut and Projection shortcut.

There are multiple versions of ResNetXX architectures where ‘XX’ denotes the number of layers. The most commonly used ones are ResNet50 and ResNet101. Since the vanishing gradient problem was taken care of (more about it in the How part), CNN started to get deeper and deeper. Below we present the structural details of ResNet18

ResNet18 - Structural Details														
#	Input Image			output			Layer	Stride	Pad	Kernel		in	out	Param
1	227	227	3	112	112	64	conv1	2	1	7	7	3	64	9472
	112	112	64	56	56	64	maxpool	2	0.5	3	3	64	64	0
2	56	56	64	56	56	64	conv2-1	1	1	3	3	64	64	36928
3	56	56	64	56	56	64	conv2-2	1	1	3	3	64	64	36928
4	56	56	64	56	56	64	conv2-3	1	1	3	3	64	64	36928
5	56	56	64	56	56	64	conv2-4	1	1	3	3	64	64	36928
6	56	56	64	28	28	128	conv3-1	2	0.5	3	3	64	128	73856
7	28	28	128	28	28	128	conv3-2	1	1	3	3	128	128	147584
8	28	28	128	28	28	128	conv3-3	1	1	3	3	128	128	147584
9	28	28	128	28	28	128	conv3-4	1	1	3	3	128	128	147584
10	28	28	128	14	14	256	conv4-1	2	0.5	3	3	128	256	295168
11	14	14	256	14	14	256	conv4-2	1	1	3	3	256	256	590080
12	14	14	256	14	14	256	conv4-3	1	1	3	3	256	256	590080
13	14	14	256	14	14	256	conv4-4	1	1	3	3	256	256	590080
14	14	14	256	7	7	512	conv5-1	2	0.5	3	3	256	512	1180160
15	7	7	512	7	7	512	conv5-2	1	1	3	3	512	512	2359808
16	7	7	512	7	7	512	conv5-3	1	1	3	3	512	512	2359808
17	7	7	512	7	7	512	conv5-4	1	1	3	3	512	512	2359808
	7	7	512	1	1	512	avg pool	7	0	7	7	512	512	0
18	1	1	512	1	1	1000	fc					512	1000	513000
Total														11,511,784

Resnet18 has around 11 million trainable parameters. It consists of CONV layers with filters of size 3x3 (just like VGGNet). Only two pooling layers are used throughout the network one at the beginning and the other at the end of the network. Identity connections are between every two CONV layers. The solid arrows show identity shortcuts where the dimension of the input and output is the same, while the dotted ones present the projection connections where the dimensions differ.

As mentioned earlier, ResNet architecture makes use of shortcut connections to solve the vanishing gradient problem. The basic building block of ResNet is a Residual block that is repeated throughout the network.



Instead of learning the mapping from x →F(x), the network learns the mapping from x → F(x)+G(x). When the dimension of the input x and output F(x) is the same, the function G(x) = x is an identity function and the shortcut connection is called Identity connection. The identical mapping is learned by zeroing out the weights in the intermediate layer during training since it's easier to zero out the weights than push them to one.

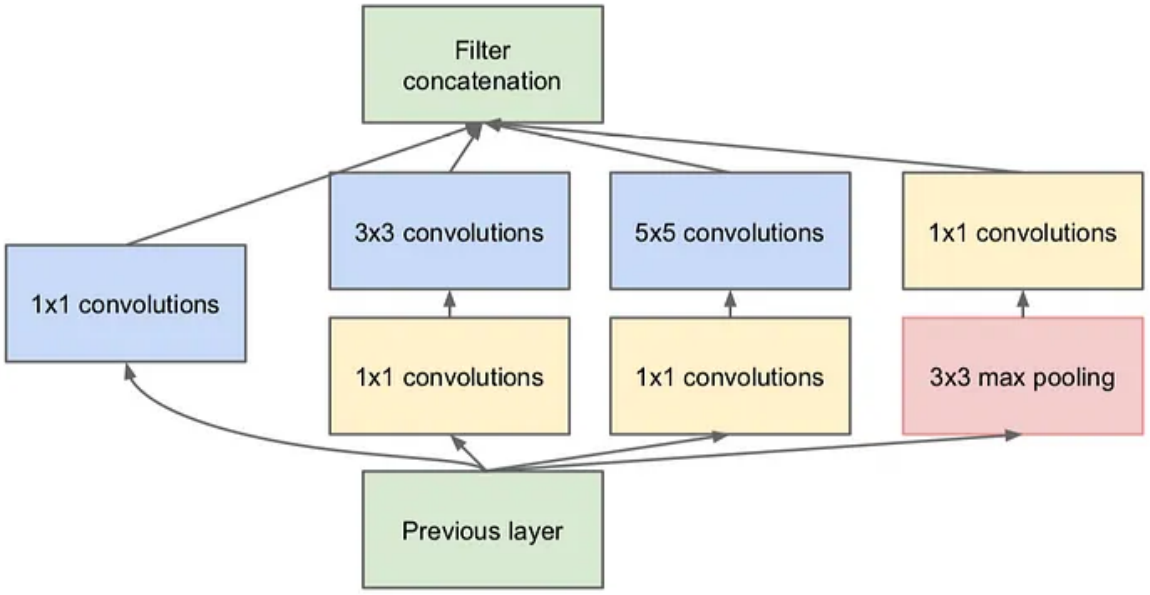
For the case when the dimensions of F(x) differ from x (due to stride length>1 in the CONV layers in between), the Projection connection is implemented rather than the Identity connection. The function G(x) changes the dimensions of input x to that of output F(x). Two kinds of mapping were considered in the original paper.

- Paper link: Link
- Notebook: link

5. Inception

In an image classification task, the size of the salient feature can considerably vary within the image frame. Hence, deciding on a fixed kernel size is rather difficult. Lager kernels are preferred for more global features that are distributed over a large area of the image, on the other hand, smaller kernels provide good results in detecting area-specific features that are distributed across the image frame. For effective recognition of such a variable-sized feature, we need kernels of different sizes. That is what Inception does. Instead of simply going deeper in terms of the number of layers, it goes wider. Multiple kernels of different sizes are implemented within the same layer.

The Inception network architecture consists of several inception modules of the following structure



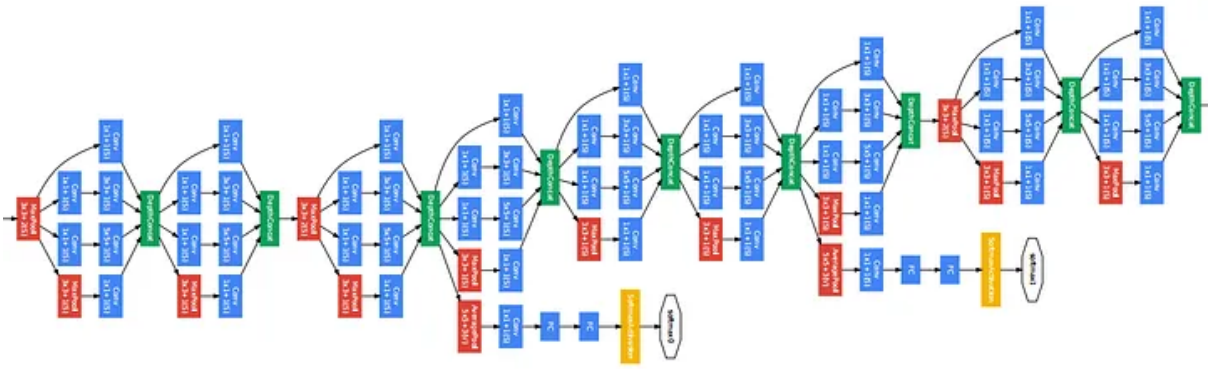
Each inception module consists of four operations in parallel

- 1x1 conv layer
- 3x3 conv layer
- 5x5 conv layer
- max pooling

The 1x1 conv blocks shown in yellow are used for depth reduction. The results from the four parallel operations are then concatenated depth-wise to form the Filter Concatenation block (in green). There is multiple version of Inception, the simplest one being the GoogLeNet.

GoogLeNet - Structural Details																
	Input Image			output			Layer	Input Layer	Stride	Pad	Kernel		in	out	Param
	227	227	3	112	112	64	conv1	input	2	1	7	7	3	64	9472
	112	112	64	56	56	64	maxpool1	conv1	2	0.5	3	3	64	64	0
	56	56	64	56	56	64	conv1x1	maxpool1	1	0	1	1	64	64	4160
	56	56	64	56	56	192	conv2-1		1	1	3	3	64	192	110784
	56	56	192	28	28	192	maxpool2		2	0.5	3	3	192	192	0
inception (3a)	28	28	192	28	28	96	conv1x1a	maxpool2	1	0	1	1	192	96	18528
	28	28	96	28	28	16	conv1x1b	maxpool2	1	0	1	1	192	16	3088
	28	28	192	28	28	192	maxpool-a	maxpool2	1	1	3	3	192	192	0
	28	28	192	28	28	64	conv1x1c	maxpool2	1	0	1	1	192	64	12352
	28	28	96	28	28	128	conv3-3	conv1x1a	1	1	3	3	96	128	110720
	28	28	16	28	28	32	conv5x5	conv1x1b	1	2	5	5	16	32	12832
	28	28	192	28	28	32	conv1x1d	maxpool-a	1	0	1	1	192	32	6176
				28	28	256	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (3b)	28	28	256	28	28	128	conv1x1a	depth-concat	1	0	1	1	256	128	32896
	28	28	128	28	28	32	conv1x1b	depth-concat	1	0	1	1	256	32	8224
	28	28	192	28	28	256	maxpool-a	depth-concat	1	1	3	3	256	256	0
	28	28	192	28	28	128	conv1x1c	depth-concat	1	0	1	1	256	128	32896
	28	28	96	28	28	192	conv3-3	conv1x1a	1	1	3	3	128	192	221376
	28	28	16	28	28	96	conv5x5	conv1x1b	1	2	5	5	32	96	76896
	28	28	192	28	28	64	conv1x1d	maxpool-a	1	0	1	1	256	64	16448
				28	28	480	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
	28	28	480	14	14	480	maxpool3	depth-concat	2	0.5	3	3	480	480	0
inception (4a)	14	14	480	14	14	96	conv1x1a	maxpool3	1	0	1	1	480	96	46176
	14	14	480	14	14	16	conv1x1b	maxpool3	1	0	1	1	480	16	7696
	14	14	480	14	14	480	maxpool-a	maxpool3	1	1	3	3	480	480	0
	14	14	480	14	14	192	conv1x1c	maxpool3	1	0	1	1	480	192	92352
	14	14	96	14	14	208	conv3-3	conv1x1a	1	1	3	3	96	208	179920
	14	14	16	14	14	48	conv5x5	conv1x1b	1	2	5	5	16	48	19248
	14	14	192	14	14	64	conv1x1d	maxpool-a	1	0	1	1	480	64	30784
				14	14	512	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (4b)	14	14	512	14	14	112	conv1x1a	depth-concat	1	0	1	1	512	112	57456
	14	14	512	14	14	24	conv1x1b	depth-concat	1	0	1	1	64	24	1560
	14	14	512	14	14	64	maxpool-a	depth-concat	1	1	3	3	64	64	0
	14	14	512	14	14	160	conv1x1c	depth-concat	1	0	1	1	64	160	10400
	14	14	96	14	14	224	conv3-3	conv1x1a	1	1	3	3	112	224	226016
	14	14	16	14	14	64	conv5x5	conv1x1b	1	2	5	5	24	64	38464
	14	14	160	14	14	64	conv1x1d	maxpool-a	1	0	1	1	64	64	4160
				14	14	512	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (4c)	14	14	512	14	14	128	conv1x1a	depth-concat	1	0	1	1	512	128	65664
	14	14	512	14	14	24	conv1x1b	depth-concat	1	0	1	1	64	24	1560
	14	14	512	14	14	64	maxpool-a	depth-concat	1	1	3	3	64	64	0
	14	14	512	14	14	128	conv1x1c	depth-concat	1	0	1	1	64	128	8320
	14	14	96	14	14	256	conv3-3	conv1x1a	1	1	3	3	128	256	295168
	14	14	16	14	14	64	conv5x5	conv1x1b	1	2	5	5	24	64	38464
	14	14	128	14	14	64	conv1x1d	maxpool-a	1	0	1	1	64	64	4160
				14	14	512	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (4d)	14	14	512	14	14	144	conv1x1a	depth-concat	1	0	1	1	512	144	73872
	14	14	512	14	14	32	conv1x1b	depth-concat	1	0	1	1	64	32	2080
	14	14	512	14	14	64	maxpool-a	depth-concat	1	1	3	3	64	64	0
	14	14	512	14	14	112	conv1x1c	depth-concat	1	0	1	1	64	112	7280
	14	14	96	14	14	288	conv3-3	conv1x1a	1	1	3	3	144	288	373536
	14	14	16	14	14	64	conv5x5	conv1x1b	1	2	5	5	32	64	51264
	14	14	112	14	14	64	conv1x1d	maxpool-a	1	0	1	1	64	64	4160
				14	14	528	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (4e)	14	14	528	14	14	160	conv1x1a	depth-concat	1	0	1	1	528	160	84640
	14	14	528	14	14	32	conv1x1b	depth-concat	1	0	1	1	64	32	2080
	14	14	528	14	14	64	maxpool-a	depth-concat	1	1	3	3	64	64	0
	14	14	528	14	14	256	conv1x1c	depth-concat	1	0	1	1	64	256	16640
	14	14	96	14	14	320	conv3-3	conv1x1a	1	1	3	3	160	320	461120
	14	14	16	14	14	128	conv5x5	conv1x1b	1	2	5	5	32	128	102528
	14	14	256	14	14	128	conv1x1d	maxpool-a	1	0	1	1	64	128	8320
				14	14	832	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
	14	14	832	7	7	832	maxpool4	depth-concat	2	0.5	3	3	832	832	0
inception (5a)	7	7	832	7	7	160	conv1x1a	maxpool4	1	0	1	1	832	160	133280
	7	7	832	7	7	32	conv1x1b	maxpool4	1	0	1	1	832	32	26656
	7	7	832	7	7	832	maxpool-a	maxpool4	1	1	3	3	832	832	0
	7	7	832	7	7	256	conv1x1c	maxpool4	1	0	1	1	832	256	213248
	7	7	96	7	7	320	conv3-3	conv1x1a	1	1	3	3	160	320	461120
	7	7	16	7	7	128	conv5x5	conv1x1b	1	2	5	5	32	128	102528
	7	7	256	7	7	128	conv1x1d	maxpool-a	1	0	1	1	832	128	106624
				7	7	832	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (5b)	7	7	832	7	7	192	conv1x1a	depth-concat	1	0	1	1	832	192	159936
	7	7	832	7	7	48	conv1x1b	depth-concat	1	0	1	1	832	48	39984
	7	7	832	7	7	832	maxpool-a	depth-concat	1	1	3	3	832	832	0
	7	7	832	7	7	384	conv1x1c	depth-concat	1	0	1	1	832	384	319872
	7	7	96	7	7	384	conv3-3	conv1x1a	1	1	3	3	192	384	663936
	7	7	16	7	7	128	conv5x5	conv1x1b	1	2	5	5	48	128	153728
	7	7	384	7	7	128	conv1x1d	maxpool-a	1	0	1	1	128	128	16512
				7	7	1024	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
	7	7	1024	1	1	1024	avgpool	depth-concat	1	0	7	7	1024	1024	0
	1	1	1024	1	1	1000	fc	depth-concat	1	0	1	1	1024	1000	1025000
								Total							6,414,360

Inception increases the network space from which the best network is to be chosen via training. Each inception module can capture salient features at different levels. Global features are captured by the 5x5 conv layer, while the 3x3 conv layer is prone to capturing distributed features. The max-pooling operation is responsible for capturing low-level features that stand out in a neighborhood. At a given level, all of these features are extracted and concatenated before it is fed to the next layer. We leave for the network/training to decide what features hold the most values and weight accordingly. Say if the images in the data-set are rich in global features without too many low-level features, then the trained Inception network will have very small weights corresponding to the 3x3 conv kernel as compared to the 5x5 conv kernel.



- Paper link: Link
- Notebook: link

All Comparisons:

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B