Case study: Architecture of LeNet

LeNet refers to LeNet-5 and is a simple convolutional neural network.

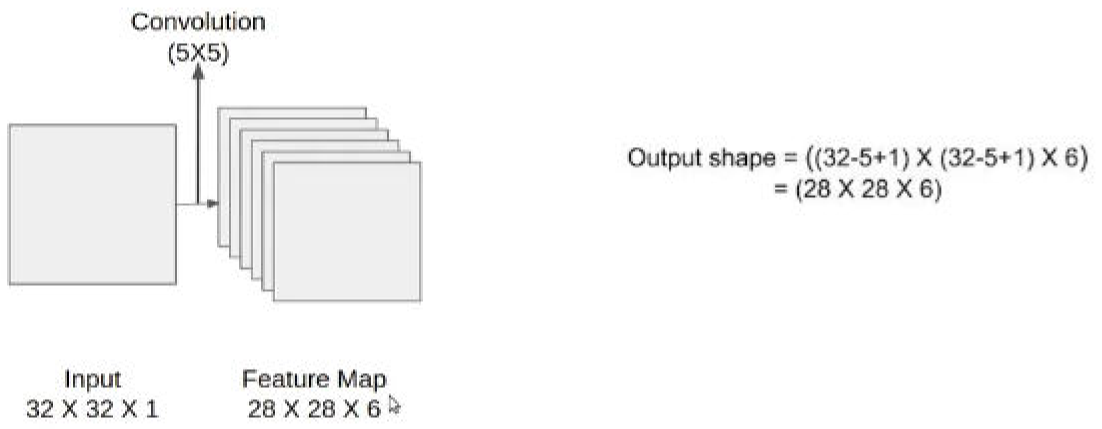
What is Lenet5?

Lenet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition. They used this architecture for recognizing the handwritten and machine-printed characters.

The Architecture of the Model

Let's understand the architecture of Lenet-5. The network has 5 layers with learnable parameters and hence named Lenet-5. It has three sets of convolution layers with a combination of average pooling. After the convolution and average pooling layers, we have two fully connected layers. At last, a Softmax classifier which classifies the images into respective class.

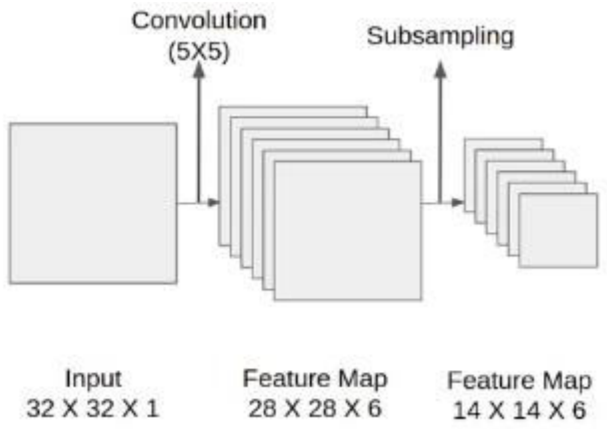
The input to this model is a 32 X 32 gray scale image hence the number of channels is one.



H1=32-5+1=28

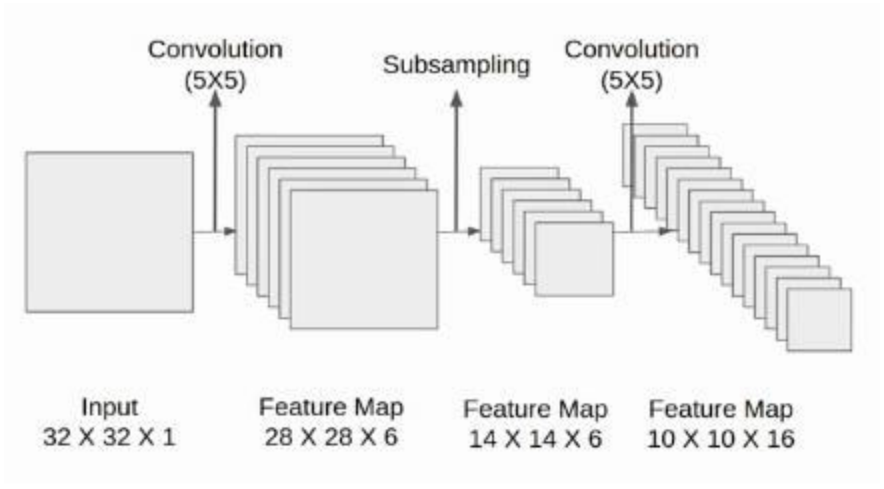
W1=32-5+1=28

We then apply the first convolution operation with the filter size 5X5 and we have 6 such filters. As a result, we get a feature map of size 28X28X6. Here the number of channels is equal to the number of filters applied.

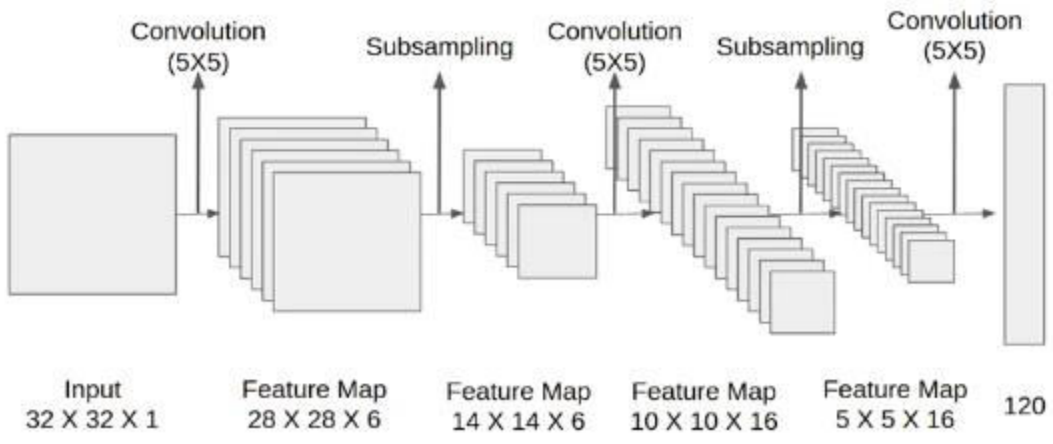


2x2 window (28-2)/2+1=14

After the first convolution operation, we apply the average pooling and the size of the feature map is reduced by half. Note that, the number of channels is intact.



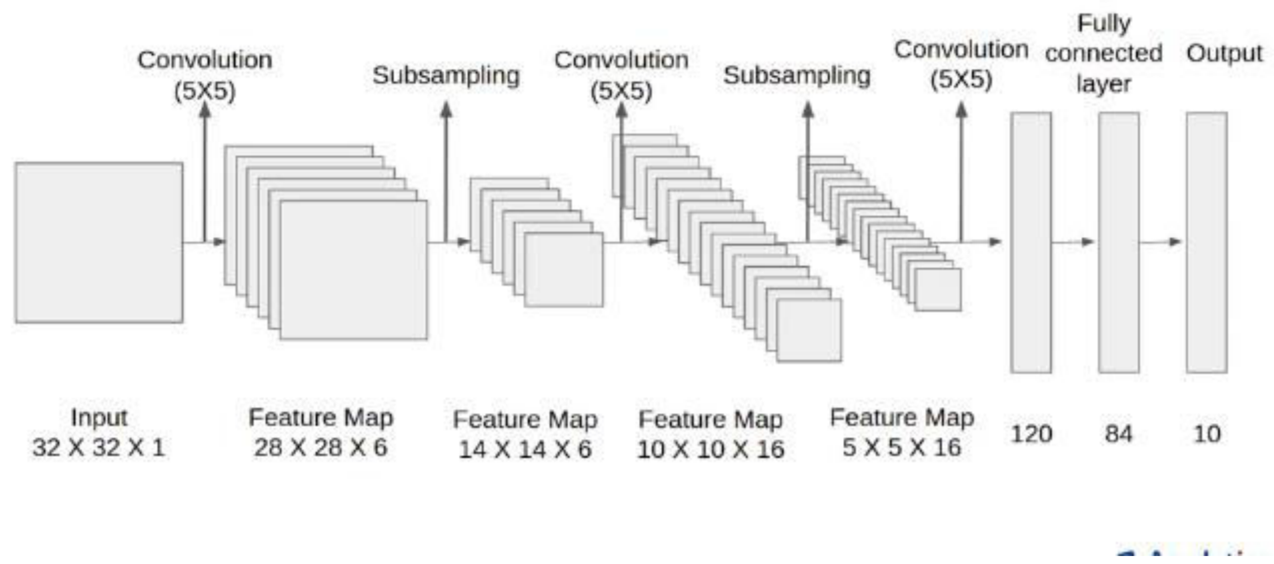
Next, we have a convolution layer with sixteen filters of size 5X5. Again the feature map changed it is 10X10X16. The output size is calculated in a similar manner. After this, we again applied an average pooling or subsampling layer, which again reduce the size of the feature map by half i.e 5X5X16. (10-2)/2+1=5



Then we have a final convolution layer of size 5X5 with 120 filters. As shown in the above image the feature map size 1X1X120. After which flatten result is 120 values.

After these convolution layers, we have a fully connected layer with eighty-four neurons. At last, we have an output layer with ten neurons since the data have ten classes.

Here is the final architecture of the Lenet-5 model.



Architecture Details

The first layer is the input layer with feature map size 32X32X1.

Then we have the first convolution layer with 6 filters of size 5X5 and stride is 1. The activation function used at his layer is tanh. The output feature map is 28X28X6.

Next, we have an average pooling layer with filter size 2X2 and stride 1. The resulting feature map is 14X14X6. Since the pooling layer doesn’t affect the number of channels.

After this comes the second convolution layer with 16 filters of 5X5 and stride 1. Also, the activation function is tanh. Now the output size is 10X10X16.

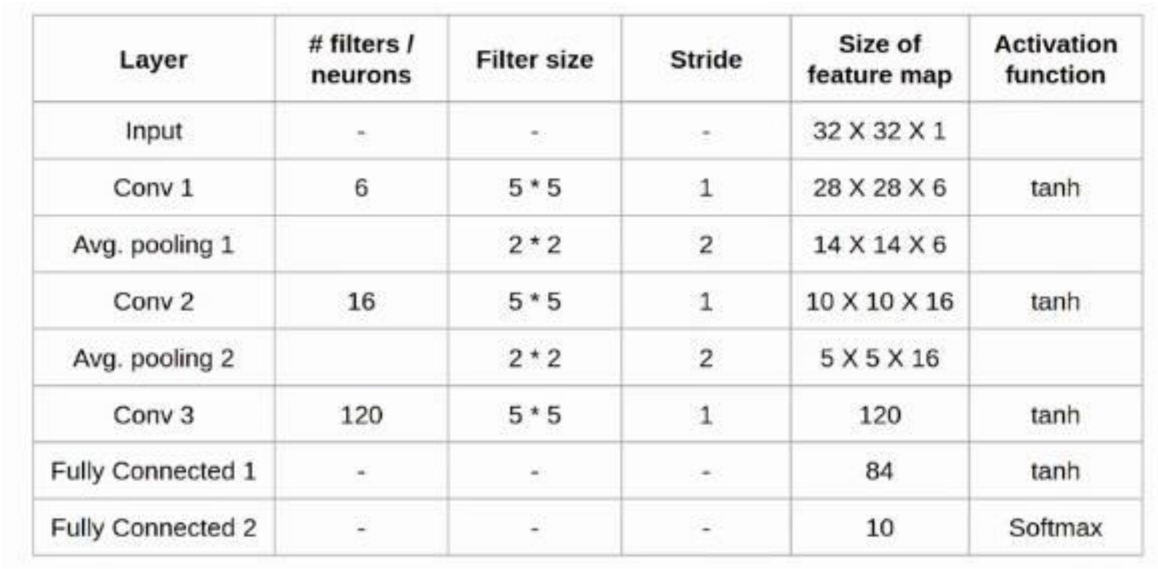
Again comes the other average pooling layer of 2X2 with stride 2. As a result, the size of the feature map reduced to 5X5X16.

The final pooling layer has 120 filters of 5X5 with stride 1 and activation function tanh. Now the output size is 120.

The next is a fully connected layer with 84 neurons that result in the output to 84 values and the activation function used here is again tanh.

The last layer is the output layer with 10 neurons and Softmax function. The Softmax gives the probability that a data point belongs to a particular class. The highest value is then predicted.

This is the entire architecture of the Lenet-5 model. The number of trainable parameters of this architecture is around sixty thousand.



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

To summarize… The network has

- **5 layers** with learnable parameters.
- The input to the model is a gray scale image.
- It has **3 convolution layers, two average pooling layers, and two fully connected layers** with a softmax classifier.
- The number of trainable parameters is 60000.

$$\boxed{\begin{aligned} W_2 &= W_1 - F + 1 \\ H_2 &= H_1 - F + 1 \end{aligned}}$$

$$\boxed{\begin{aligned} W_2 &= W_1 - F + 2P + 1 \\ H_2 &= H_1 - F + 2P + 1 \end{aligned}}$$

$$\boxed{\begin{aligned} W_2 &= \frac{W_1 - F + 2P}{S} + 1 \\ H_2 &= \frac{H_1 - F + 2P}{S} + 1 \end{aligned}}$$

Architecture of AlexNet

What is AlexNet?

Alex Net is the name given to a **Convolutional Neural Network Architecture** that won the LSVRC competition in **2012**.

LSVRC **(**Large Scale Visual Recognition Challenge) is competition where research teams evaluate their algorithms on a huge dataset of labeled images (**ImageNet**) and compete to achieve high accuracy on several visual recognition tasks.

The **Alex Net contains 8 layers** with weights; **5 convolutional layers 3 fully connected layers**.

The Alexnet has eight layers with learnable parameters. The model consists of five layers with a combination of max pooling followed by 3 fully connected layers and they use Relu activation in each of these layers except the output layer.

They found out that using the relu as an activation function accelerated the speed of the training process by almost six times. They also used the dropout layers that prevented their model from overfitting. Further, the model is trained on the Imagenet dataset. The Imagenet dataset has almost 14 million images across a thousand classes.

Since Alexnet is a deep architecture, the authors introduced **padding** to prevent the size of the feature maps from reducing drastically. The input to this model is the images of size 227X227X3.

Then we apply the first convolution layer with 96 filters of size 11X11 with stride 4. The activation function used in this layer is relu. The output feature map is 55X55X96.

In case, you are unaware of how to calculate the output size of a convolution layer

output= ((Input-filter size)/ stride)+1

Also, the number of filters becomes the channel in the output feature map.

Next, we have the first Maxpooling layer, of size 3X3 and stride 2. Then we get the resulting feature map with the size 27X27X96.

After this, we apply the second convolution operation. This time the filter size is reduced to 5X5 and we have 256 such filters. The stride is 1 and padding 2. The activation function used is again relu. Now the output size we get is 27X27X256.

Again we applied a max-pooling layer of size 3X3 with stride 2. The resulting feature map is of shape 13X13X256.

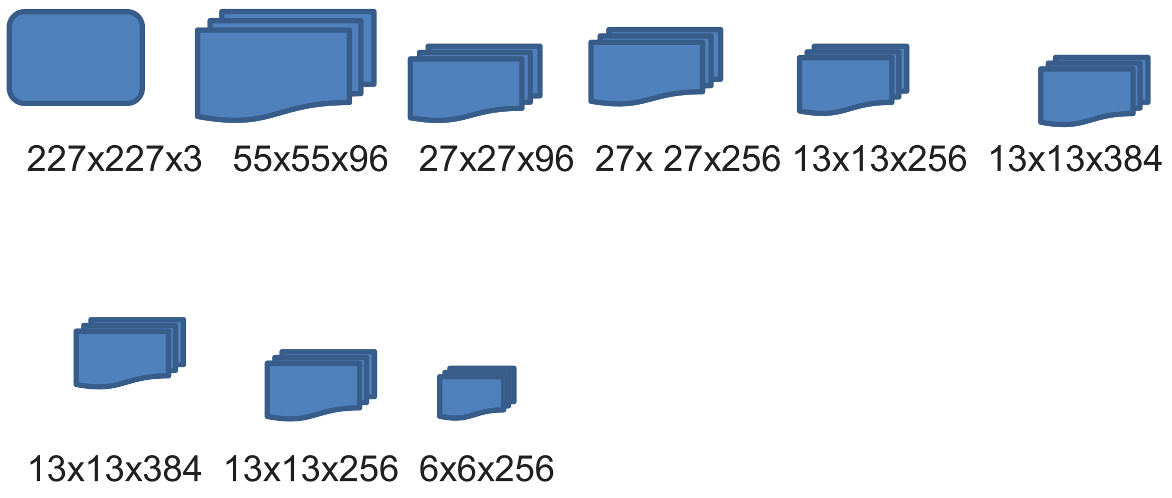
Now we apply the third convolution operation with 384 filters of size 3X3 stride 1 and also padding 1. Again the activation function used is relu. The output feature map is of shape 13X13X384.

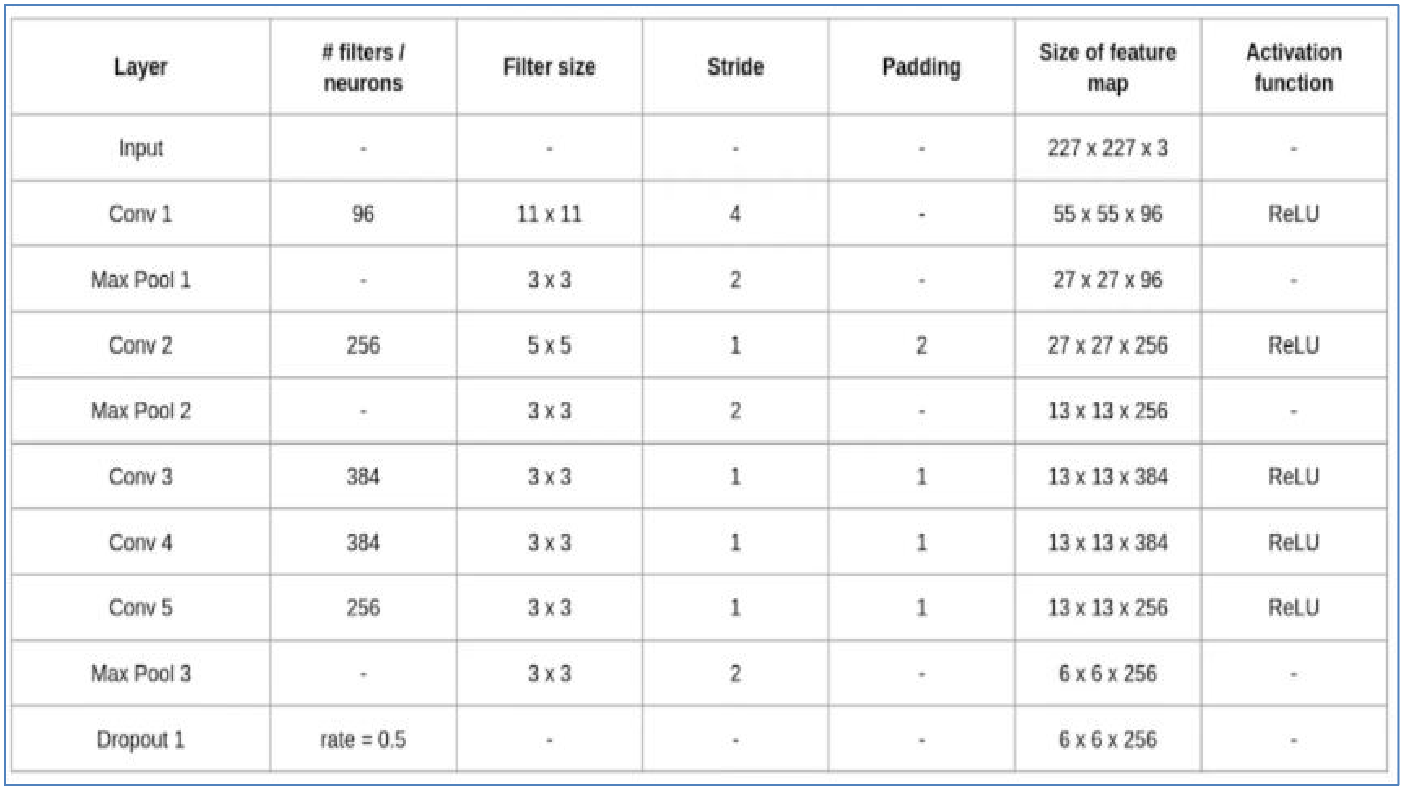
Then we have the fourth convolution operation with 384 filters of size 3X3. The stride along with the padding is 1. On top of that activation function used is relu. Now the output size remains unchanged i.e 13X13X384.

After this, we have the final convolution layer of size 3X3 with 256 such filters. The stride and padding are set to one also the activation function is relu. The resulting feature map is of shape 13X13X256.

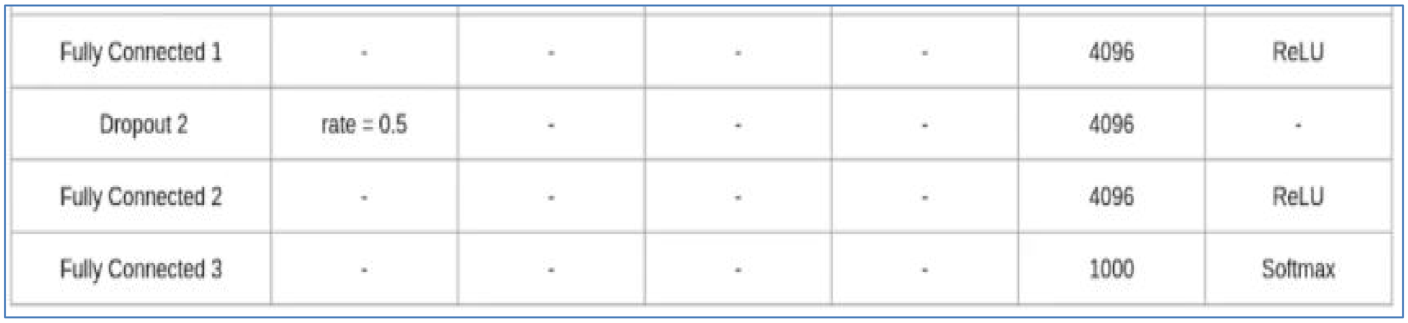
So if you look at the architecture till now, the number of filters is increasing as we are going deeper. Hence it is extracting more features as we move deeper into the architecture. Also, the filter size is reducing, which means the initial filter was larger and as we go ahead the filter size is decreasing, resulting in a decrease in the feature map shape.

Next, we apply the third max-pooling layer of size 3X3 and stride 2. Resulting in the feature map of the shape 6X6X256.





Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU
Max Pool 3	-	3 x 3	2	-	6 x 6 x 256	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-



Fully Connected 1	-	-	-	-	4096	ReLU
Dropout 2	rate = 0.5	-	-	-	4096	-
Fully Connected 2	-	-	-	-	4096	ReLU
Fully Connected 3	-	-	-	-	1000	Softmax

After this, we have our first dropout layer. The drop-out rate is set to be 0.5.

Then we have the first fully connected layer with a relu activation function. The size of the output is 4096. Next comes another dropout layer with the dropout rate fixed at 0.5.

This followed by a second fully connected layer with 4096 neurons and relu activation.

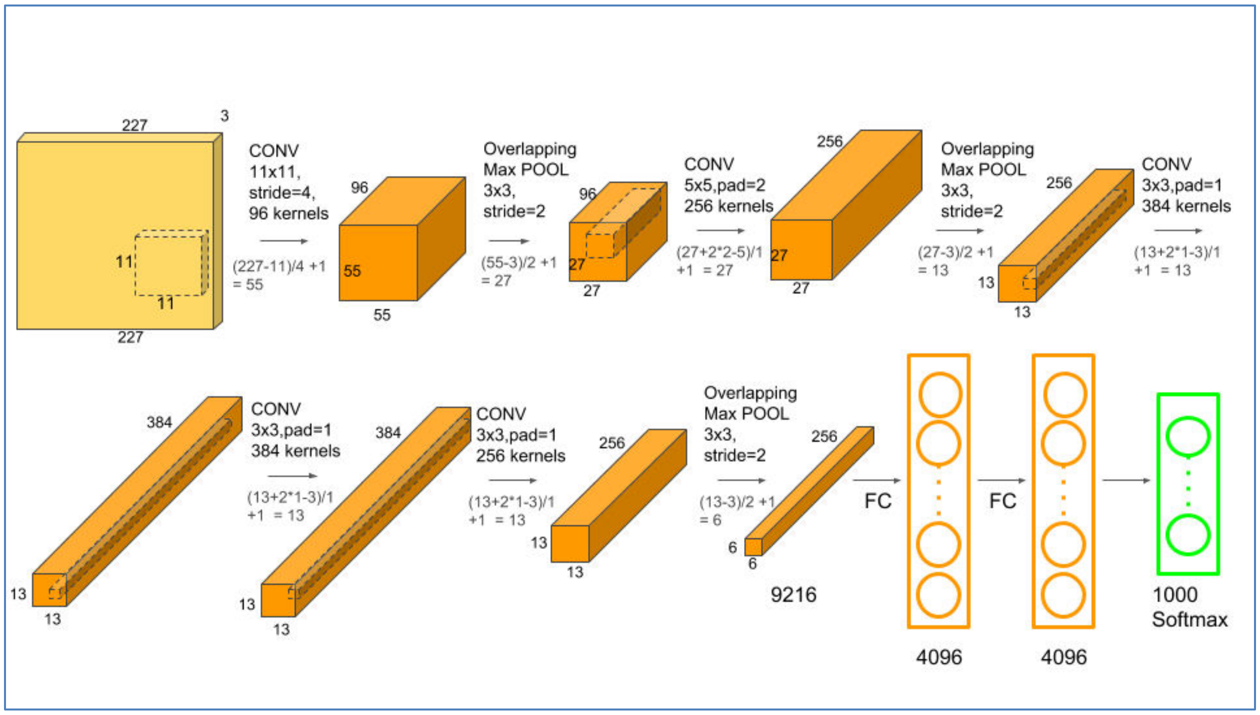
Finally, we have the last fully connected layer or output layer with 1000 neurons as we have 1000 classes in the data set. The activation function used at this layer is Softmax.

This is the architecture of the Alexnet model. It has a total of 62.3 million learnable parameters.

To summarize the architecture

It has 8 layers with learnable parameters.

- The input to the Model is RGB images.
- It has 5 convolution layers with a combination of max-pooling layers.
- Then it has 3 fully connected layers.
- The activation function used in all layers is Relu.
- It used two Dropout layers.
- The activation function used in the output layer is Softmax.
- The total number of parameters in this architecture is 62.3 million.



VGG 16 Architecture

VGG stands for *Visual Geometry Group* (a group of researchers at Oxford who developed this architecture). The VGG architecture consists of blocks, where each block is composed of 2D Convolution and Max Pooling layers. VGGNet comes in two flavors, VGG16 and VGG19, where 16 and 19 are the number of layers in each of them respectively.

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another.

The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3. The convolution stride is fixed to 1 pixel; the padding is 1-pixel for 3×3 conv. Layers.

Start with initializing the model by specifying that the model is a sequential model. After initializing the model add

→ 2 x convolution layer of 64 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

→ 2 x convolution layer of 128 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

→ 3 x convolution layer of 256 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

→ 3 x convolution layer of 512 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

→ 3 x convolution layer of 512 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

Add relu(Rectified Linear Unit) activation to each layers so that all the negative values are not passed to the next layer.

After creating the entire convolution, pass the data to the dense layer so for that flatten the vector which comes out of the convolutions and add

→ 1 x Dense layer of 4096 units, relu activation

→ 1 x Dense layer of 4096 units, relu activation

→ 1 x Dense Softmax layer of 2 units

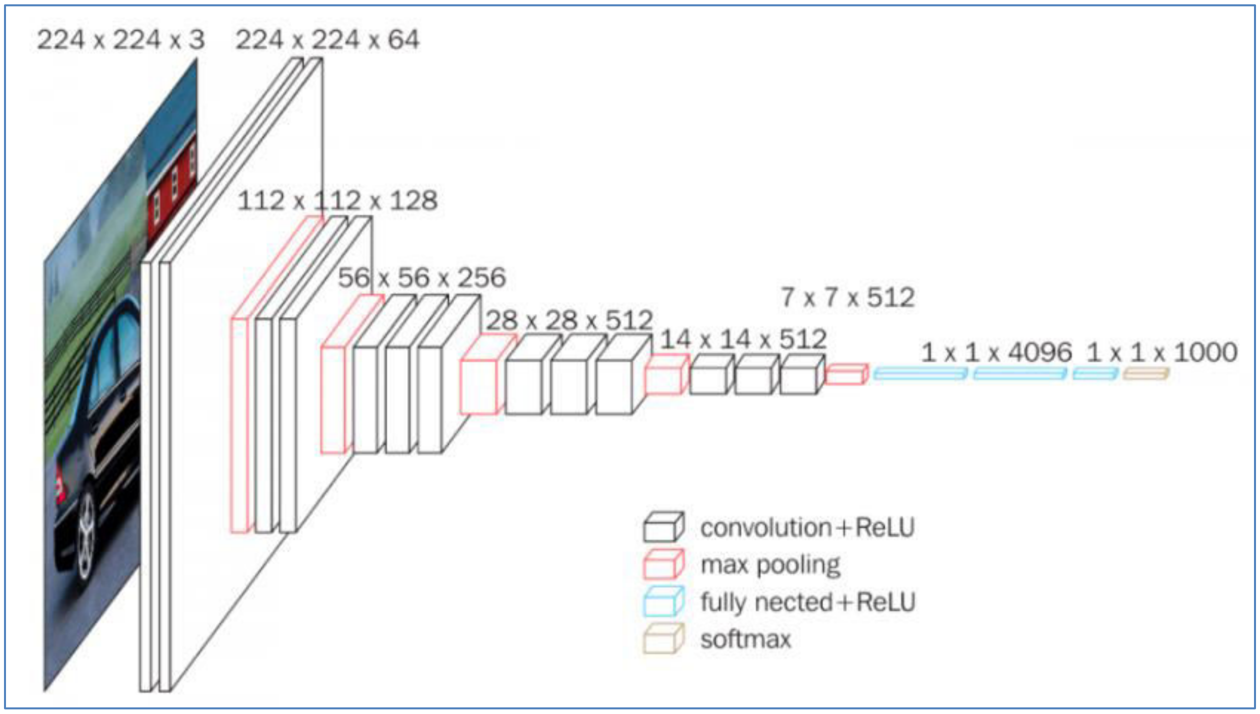
The softmax layer will output the value between 0 and 1 based on the confidence of the model that which class the images belongs to.

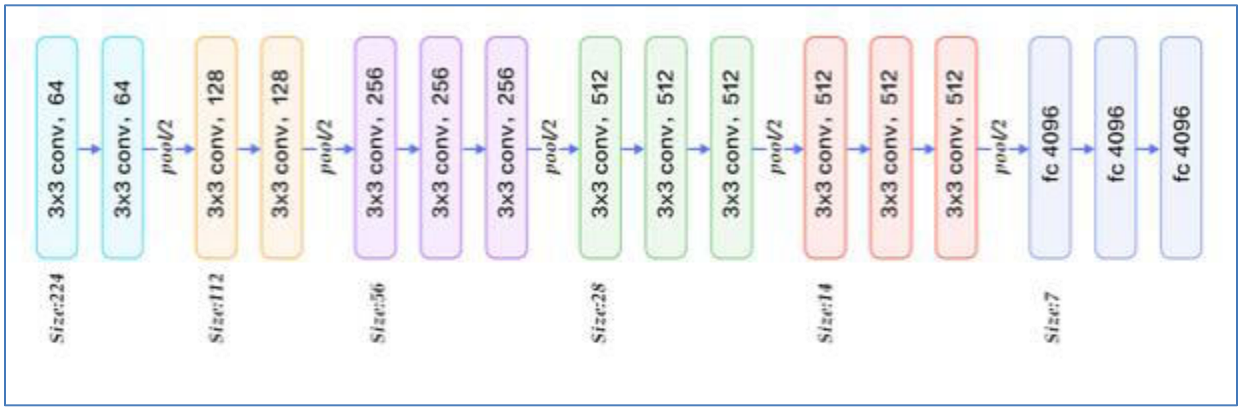
Conv 3x3 filter,s=1,p=1

Maxpool window 2x2 s=2

14x14x512

(14-2)/2+1=7





16 layers of VGG16

1.Convolution using 64 filters

2.Convolution using 64 filters + Max pooling

3 Convolution using 128 filters

4.Convolution using 128 filters + Max pooling

5. Convolution using 256 filters

6. Convolution using 256 filters

7. Convolution using 256 filters + Max pooling

8. Convolution using 512 filters

9. Convolution using 512 filters
10. Convolution using 512 filters+Max pooling
11. Convolution using 512 filters
12. Convolution using 512 filters
13. Convolution using 512 filters+Max pooling
14. Fully connected with 4096 nodes
15. Fully connected with 4096 nodes
16. Output layer with Softmax activation with 1000 nodes.