Convolution Filters, Padding and Pooling

WHAT IS A CONVOLUTION?

- A convolution is an operation that changes a function into something else. We do convolutions so that we can transform the original function into a form to get more information.
- Convolutions have been used for a long time in image processing to blur and sharpen images, and perform other operations, such as, enhance edges and emboss.

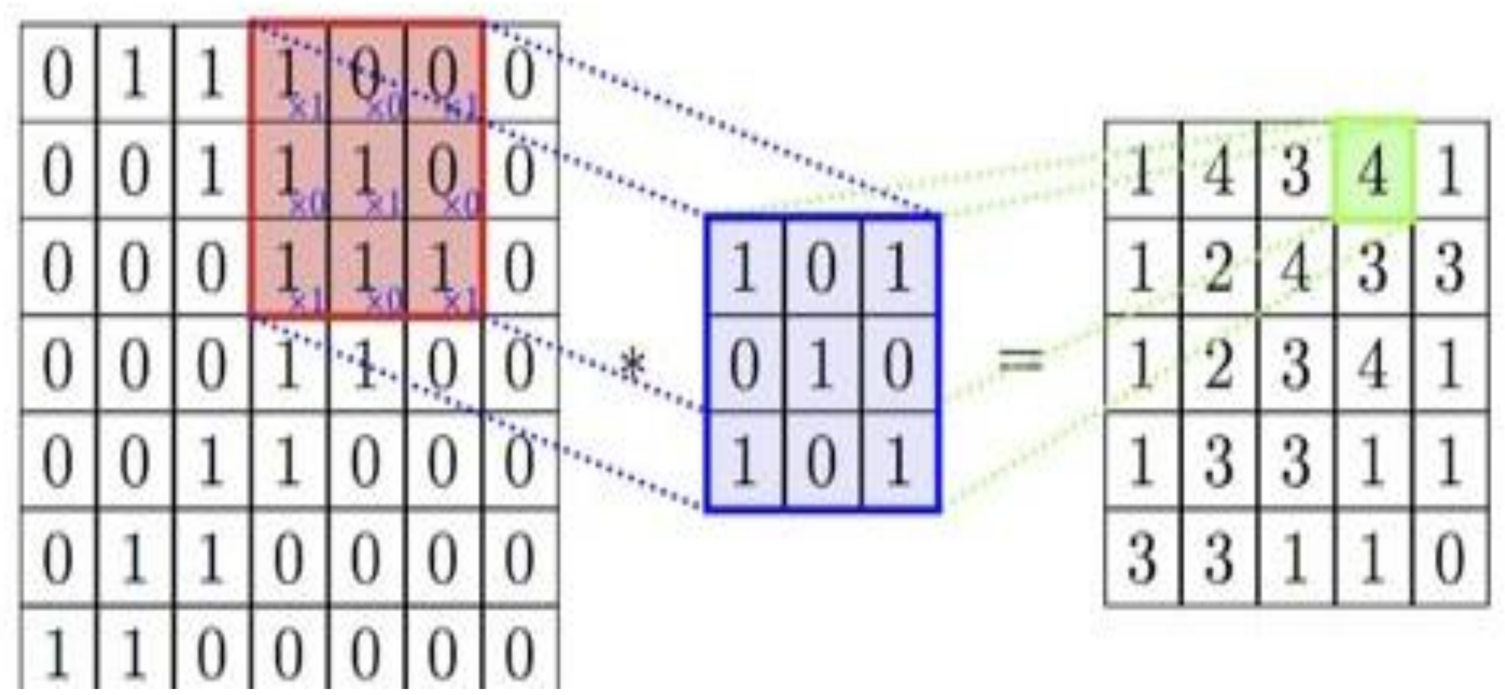
The three elements that enter into the convolution operation:

Input image
Feature detector
Feature map

A convolution operation is an element wise matrix multiplication operation. Where one of the matrices is the image, and the other is the filter or kernel that turns the image into something else. The output of this is the final convoluted image.

Example

If the image is larger than the size of the filter, we slide the filter to the various parts of the image and perform the convolution operation. Each time we do that, we generate a new pixel in output image.



Padding

- We have seen that convolving an input of 7 X 7 dimension with a 3 X 3 filter results in 5 X 5 output. We can generalize it and say that if the input is n X n and the filter size is f X f, then the output size will be (n-f+1) X (n-f+1)
- There are primarily two disadvantages here:
- Every time we apply a convolutional operation, the size of the image shrinks
- Pixels present in the corner of the image are used only a few number of times during convolution as compared to the central pixels. Hence, we do not focus too much on the corners since that can lead to information loss

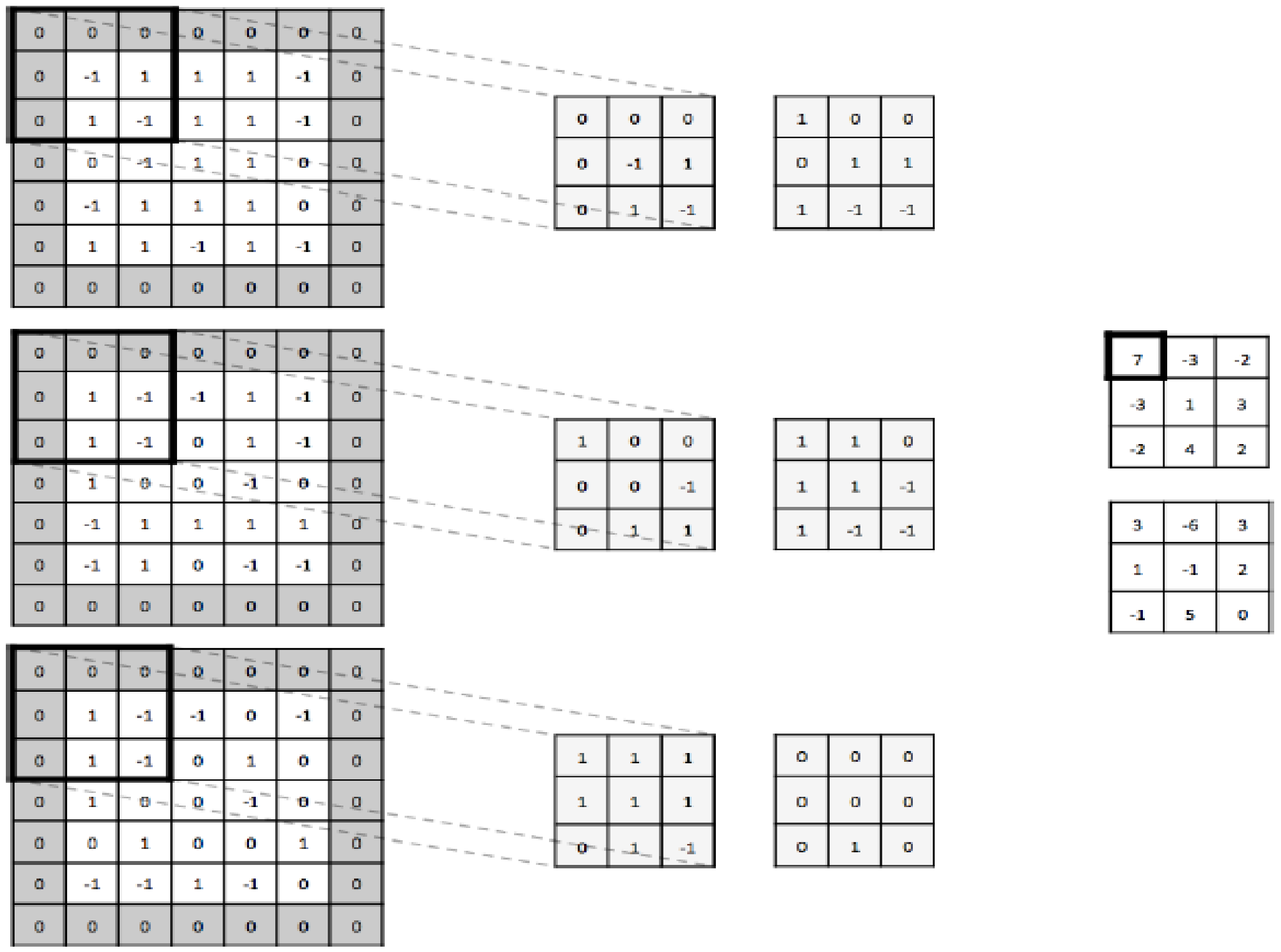
padding

- To overcome these issues, we can pad the image with an additional border, i.e., we add one pixel all around the edges. This means that the input will be an 9 X 9 matrix (instead of a 7 X 7 matrix). Applying convolution of 3 X 3 on it will result in a 7 X 7 matrix which is the original shape of the image.
- **Input:** n X n
- **Padding:** p
- **Filter size:** f X f
- **Output:** (n+2p-f+1) X (n+2p-f+1)

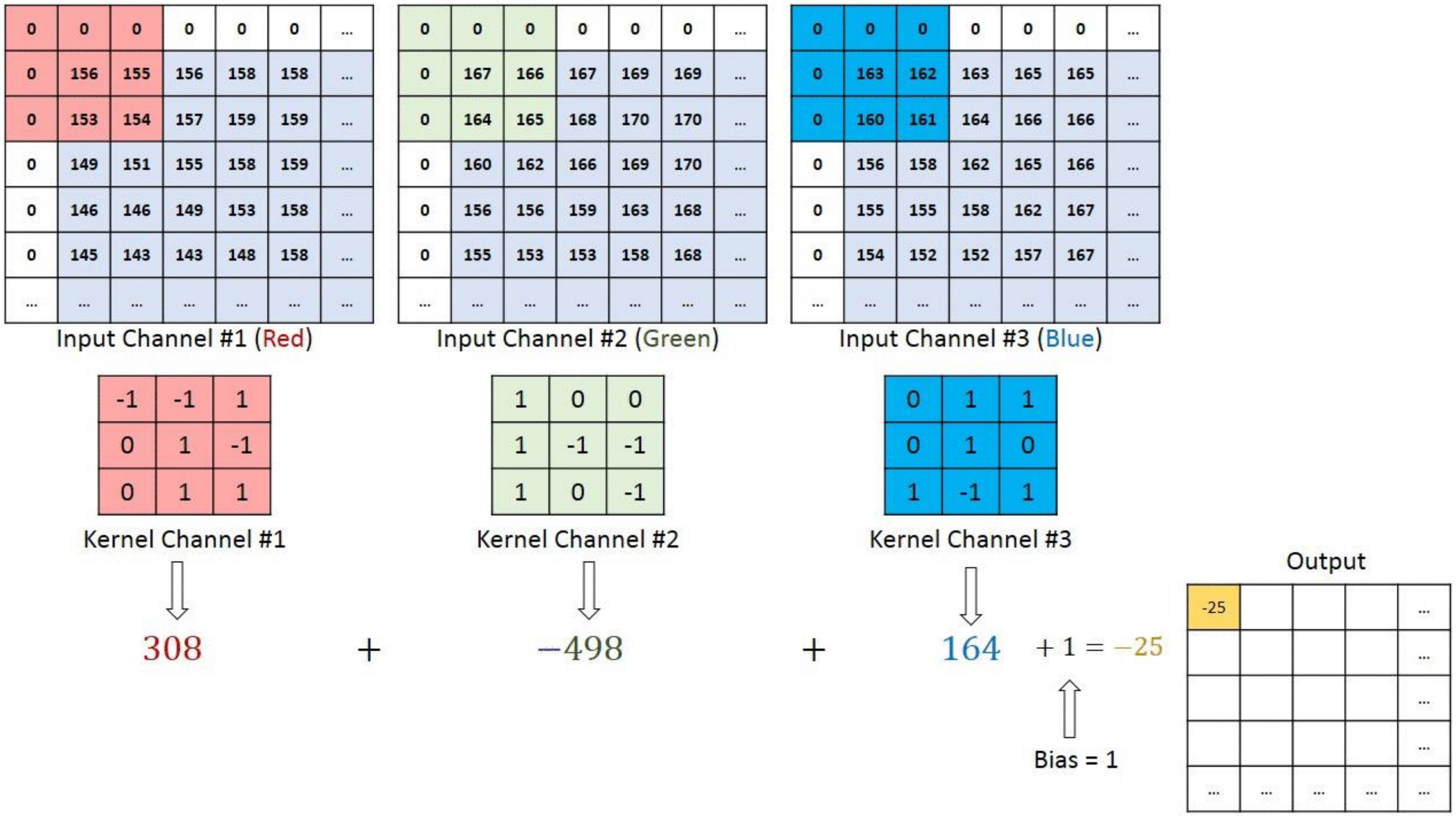
Striding

- No of steps taken for convolution.
- Suppose we choose a stride of 2. So, while convoluting through the image, we will take two steps – both in the horizontal and vertical directions separately.
- Stride helps to reduce the size of the image, a particularly useful feature.

A convolutional layer with an input volume that has width 5, height 5, depth 3, and zero padding 1. There are 2 filters, with spatial extent 3 and applied with a stride of 2. It results in an output volume with width 3, height 3, and depth 2.



Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel



0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

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

Kernel Channel #1

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

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$$308 + -498 + 164 + 1 = -25$$

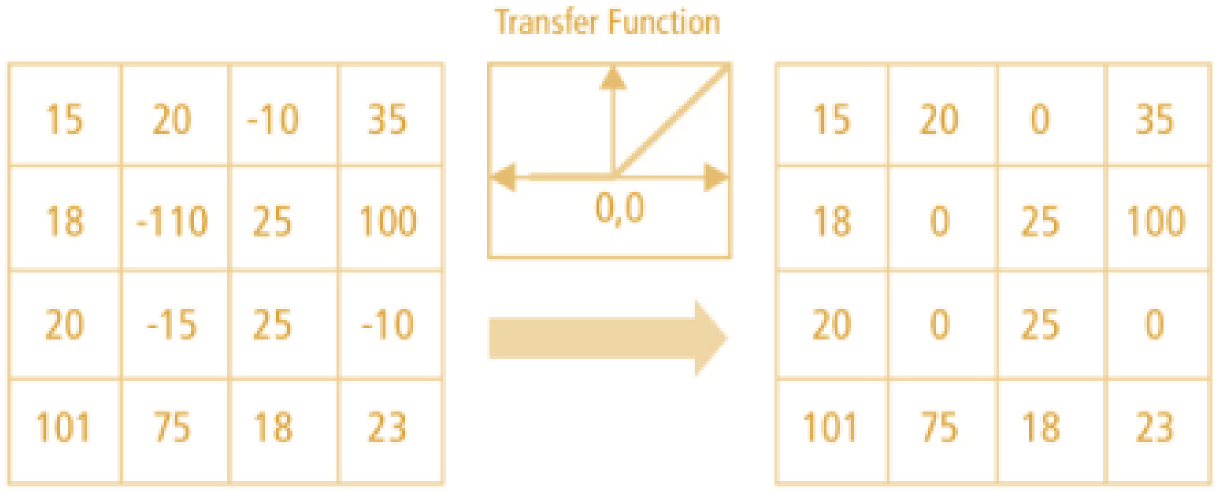
Bias = 1

Output

-25				...
				...
				...
				...
...	...	...	...	...

Non Linearity (ReLU)

- ReLU stands for Rectified Linear Unit for a non-linear operation. The output is *f(x) = max(0,x).*
- Why ReLU is important : ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.
- There are other non linear functions such as tanh or sigmoid that can also be used instead of ReLU. Most of the data
  bett



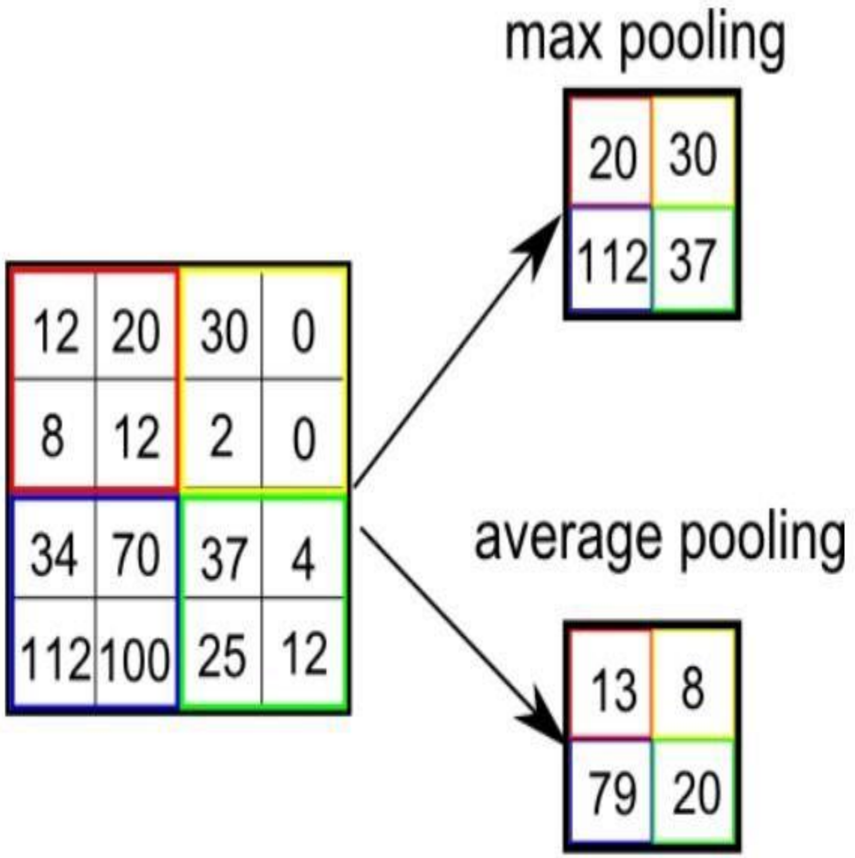
15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23

15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

Pooling Layer

- Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature.
- This is to **decrease the computational power required to process the data** through dimensionality reduction and hence speed up the computation.
- There are two types of Pooling: Max Pooling and Average Pooling. **Max Pooling** returns the **maximum value** from the portion of the image covered by the Kernel. On the other hand, **Average Pooling** returns the **average of all the values** from the portion of the image covered by the Kernel.

Max Pooling also performs as a Noise Suppressant. Max Pooling performs a lot better than Average Pooling.

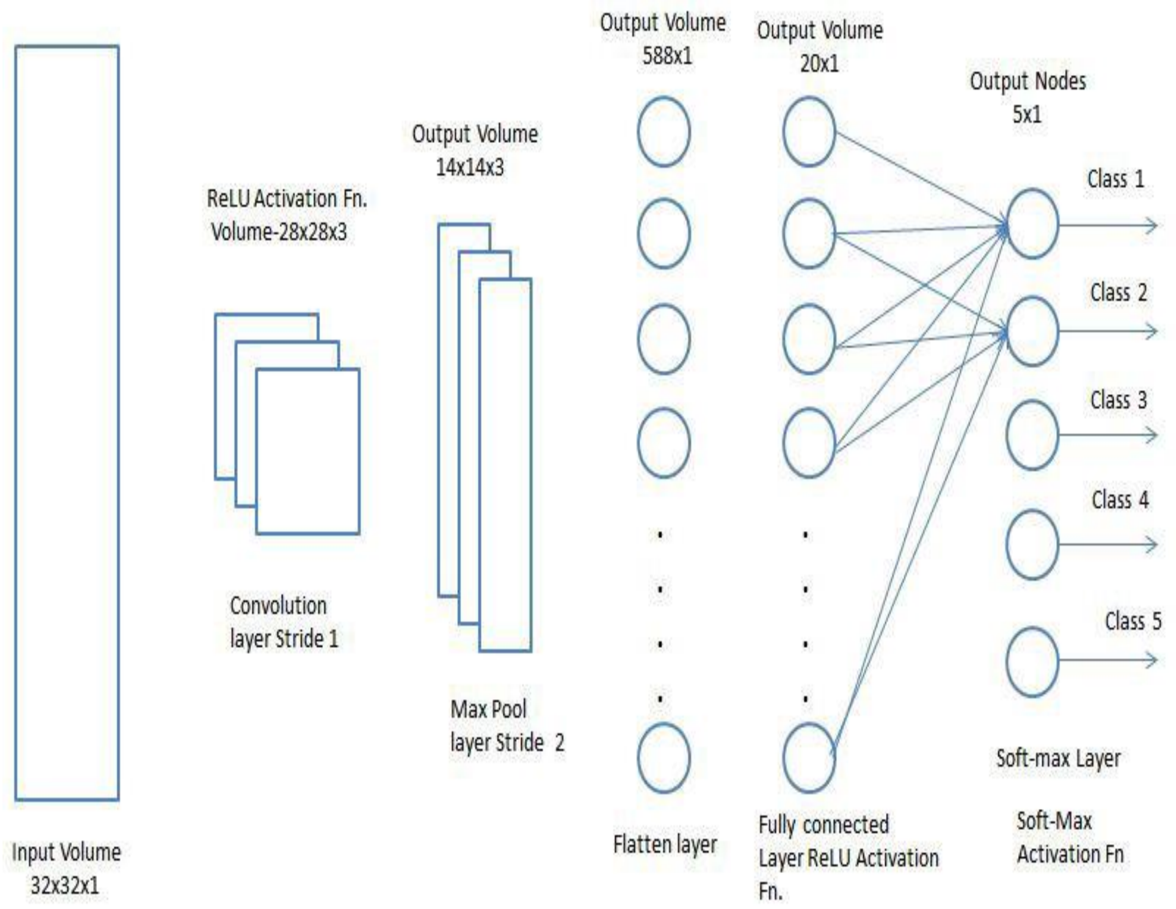


- The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network.
- Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.
- After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

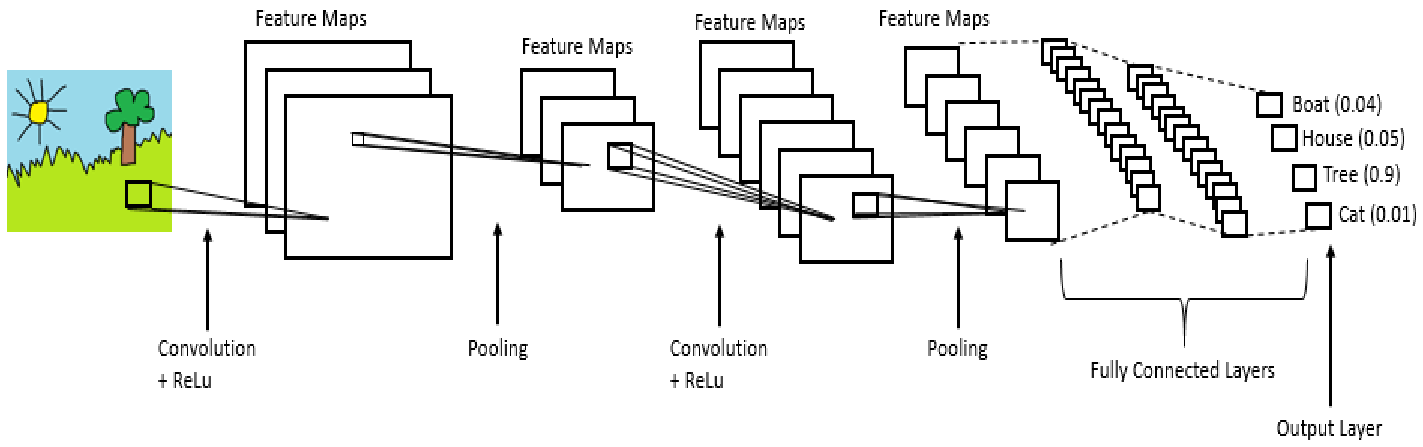
Fully Connected Layer

- After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.
- The layer we call as FC layer, we flattened our matrix into column vector and feed it into a fully connected layer.
- In this layer, the flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

FC Layer


Output Volume 588x1
Output Volume 20x1
Output Nodes 5x1
Output Volume 14x14x3
ReLU Activation Fn. Volume-28x28x3
Class 1
Class 2
Class 3
Class 4
Class 5
Convolution layer Stride 1
Max Pool layer Stride 2
Soft-max Layer
Input Volume 32x32x1
Flatten layer
Fully connected Layer ReLU Activation Fn.
Soft-Max Activation Fn

Complete CNN architecture



Summary

- Provide input image into convolution layer
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size
- Add as many convolutional layers until satisfied
- Flatten the output and feed into a fully connected layer (FC Layer)
- Output the class using an activation function (Logistic Regression with cost functions) and classifies images.

references

- https://pavisj.medium.com/convolutions-and-backpropagations-46026a8f5d2c (CNN back propagation)
- https://www.saama.com/different-kinds-convolutional-filters/
- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53------ -
- https://www.cs.toronto.edu/~lczhang/360/lec/w04/convnet.html#:~:text=There%20is%20one%20bias%20for,Nevertheless%2C%20the%20biases%20are%20there
- https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/
- https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-1-convolution-operation
- https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/
- https://cs231n.github.io/convolutional-networks/