## CONVOLUTION NEURAL NETWORKS

There are four main operations in the ConvNet shown in Figure 3 above:

**1. Convolution**

**2. Non Linearity (ReLU)**

**3. Pooling or Sub Sampling**

**4. Classification (Fully Connected Layer)**

**CHANNEL:**

1. **In a CNN channel tells u depth, for color RED, BLUE, GREEN. A colored image will have channels equal to 3.In general three layer of 1-d matrix stacked over each other.**
2. **For a grayscale image, channel = 1 which a 2-D matrix.**
3. **The value in the matrix ranges from 0-255, 0 for black and 255 indicating white.**

**CONVOLUTION STEP**

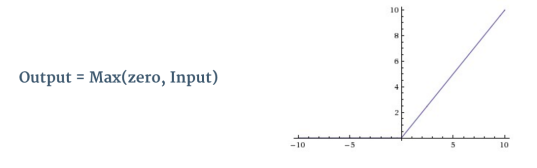
1. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.
2. For an actual 28x28 pixels image (grayscale) we choose a filter = 3X3 is called a ‘filter ‘or ‘kernel’ or ‘feature detector’.
3. Thus we slide on actual image pixel using this filter, we slide by steps decided by a parameter called **stride.**
4. If stride =1 we move the 3x3 matrix one pixel by one, if two two by two.
5. Larger stride gives lesser feature maps.
6. The matrix formed by moving the the filter is called convolved matrix/features or Activation Map.
7. Padding is another parameter .we add zero padding around border and edges 1: to adjust i/p and o/p size. 2: to detect around borders and edges.

Three parameters before applying a convolution layer:-

1. **DEPTH: -** Depth corresponds to the number of filters we use for the convolution operation. If we use 3 filter, thus producing three different feature maps. You can think of these three feature maps as stacked 2d matrices, so, the ‘depth’ of the feature map would be three.
2. **STRIDE:-** Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.
3. **Zero-Padding:-** it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A nice feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is also called wide convolution, and not using zero-padding would be a narrow convolution

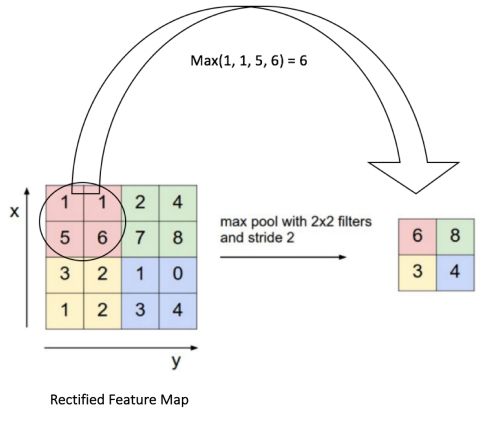
**RELU ACTIVATION:-**

1. ReLU stands for Rectified Linear Unit and is a non-linear operation
2. ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.
3. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear.
4. Because convolution is a linear process.



**The Pooling Step:-**

1. Reduces the dimensionality of feature map retaining the most important feature.
2. MAX, AVG, SUM 3 types of pooling is done.
3. Example: For a max-2d polling out of four matrix numbers [6 4 5 3] it will be out 6 to represent the feature map.
4. Max\_pooling seems to perform better since network invariant to small transformations, distortions and translations in the input image.
5. Reduces number of dimension, leads to less computation and also checks overfitting.



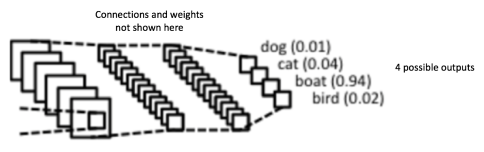
Full Connected Dense Network (MLP)

The full connected layer or dense layer is Multi-Layer perceptron.

Full means all neurons from i/p are connected all neurons on o/p.

Generally it is a Softmax regression (Multi class logistic classifiers) function applied. We can use other classifiers as SVM instead too.

The purpose of the Fully Connected layer is to use the features from previous layers (conv + pool) for classifying the input image into various classes based on the training dataset.



For example, in above image we are classifying the features derived from convolution and pooling layers into four classes’ dog, cat, boat, & bird with a probability score. The probability score should add up to 1.

FINAL STEPS OF A CONVOLUTION NN

**STEPS:-**

1. Initialize the weights, bias and other parameter for all layer.
2. The network takes the image input and extracts features using forward propagation(CONV + POOL)
3. The extracted features are used by a SOFTMAX CLASSIFIER (FULL CONNECTED NN) to assign probabilities to the classes defined.
4. Calculate the total error at o/p layer

**Total Error = ∑ ½ (target probability – output probability) ².**

1. The backpropagation algorithm is used using gradients to update the initially assigned random weights in order to minimize the TOTAL ERROR.
2. The values of the filter matrix and connection weights get updated

**STEPS 2-5 are repeated until error is minimized☺**

When a new (unseen) image is input into the ConvNet, the network would go through the forward propagation step and output a probability for each class (for a new image, the output probabilities are calculated using the weights which have been optimized to correctly classify all the previous training examples). If our training set is large enough, the network will (hopefully) generalize well to new images and classify them into correct categories.