CNN Model Intro with Fashion MNIST Implementation

Brief Introduction

This notebook will cover the following two major topics :

Understand the basic concepts of CNN model

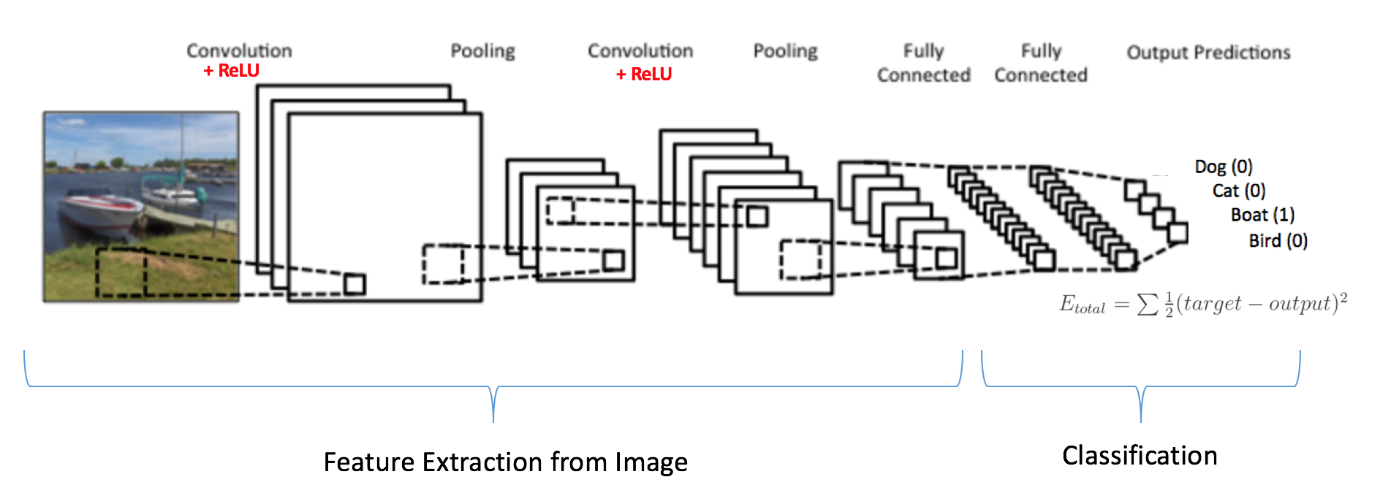
Implement CNN model in realtime using Fashion MNIST dataset

Understand the basic concepts of CNN model :

Mankind is an awesome natural machine and is capable of looking at multiple images every second and process them without realizing how the processing is done. But same is not with machines.

The first step in image processing is to understand, how to represent an image so that the machine can read it?

Every image is an cumulative arrangement of dots (a pixel) arranged in a special order. If you change the order or color of a pixel, the image would change as well.



Three basic components to define a basic convolutional neural network.

### The Convolutional Layer

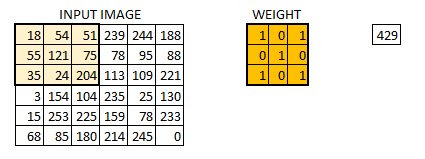
### The Pooling layer

### The Output layer

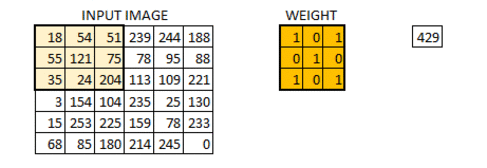
Let’s see each of them in detail

### The Convolutional Layer :

In this layer if we have an image of size 66. We define a weight matrix which extracts certain features from the images

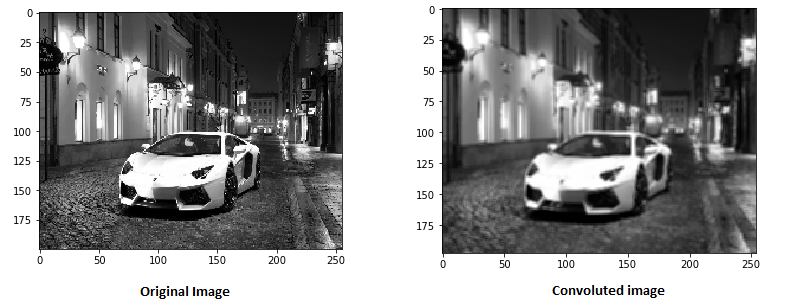


We have initialized the weight as a 33 matrix. This weight shall now run across the image such that all the pixels are covered at least once, to give a convolved output. The value 429 above, is obtained by the adding the values obtained by element wise multiplication of the weight matrix and the highlighted 33 part of the input image.



The 66 image is now converted into a 44 image. Think of weight matrix like a paint brush painting a wall. The brush first paints the wall horizontally and then comes down and paints the next row horizontally. Pixel values are used again when the weight matrix moves along the image. This basically enables parameter sharing in a convolutional neural network.

Let’s see how this looks like in a real image.

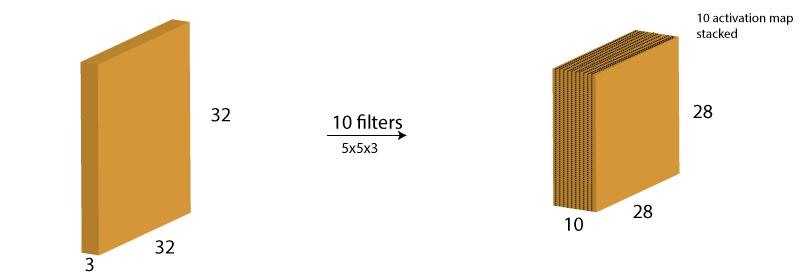


#### Having Multiple filters & the Activation Map

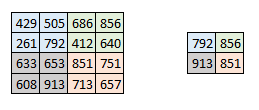
* The depth dimension of the weight would be same as the depth dimension of the input image.
* The weight extends to the entire depth of the input image.
* Convolution with a single weight matrix would result into a convolved output with a single depth dimension. In case of multiple filters all have same dimensions applied together.
* The output from the each filter is stacked together forming the depth dimension of the convolved image.

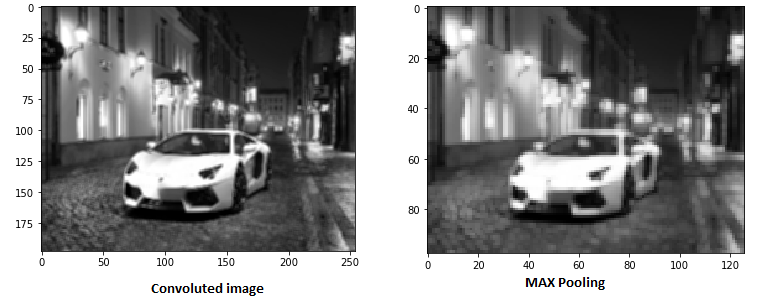
Suppose we have an input image of size 32323. And we apply 10 filters of size 553 with valid padding. The output would have the dimensions as 282810.

You can visualize it as –



### The Pooling Layer

If images are big in size, we would need to reduce the no.of trainable parameters.For this we need to use pooling layers between convolution layers. Pooling is used for reducing the spatial size of the image and is implemented independently on each depth dimension resulting in no change in image depth. Max pooling is the most popular form of pooling layer.Here we have taken stride as 2, while pooling size also as 2. The max operation is applied to each depth dimension of the convolved output. As you can see, the 44 convolved output has become 22 after the max pooling operation.

Let’s see how max pooling looks on a real image.In the above image we have taken a convoluted image and applied max pooling on it which resulted in still retaining the image information that is a car but if we closely observe the dimensions of the image is reduced to half which basically means we can reduce the parameters to a great number.

There are other forms of pooling like average pooling, L2 norm pooling.

#### Output dimensions

It is tricky at times to understand the input and output dimensions at the end of each convolution layer. For this we will use three hyperparameters that would control the size of output volume.

1. No of Filter: The depth of the output volume will be equal to the number of filter applied.The depth of the activation map will be equal to the number of filters.
2. Stride – When we have a stride of one we move across and down a single pixel. With higher stride values, we move large number of pixels at a time and hence produce smaller output volumes.
3. Zero padding – This helps us to preserve the size of the input image. If a single zero padding is added, a single stride filter movement would retain the size of the original image.

We can apply a simple formula to calculate the output dimensions.

The spatial size of the output image can be calculated as( [W-F+2P]/S)+1. where, W is the input volume size, F is the size of the filter, P is the number of padding applied S is the number of strides.

Let us take an example of an input image of size 64643, we apply 10 filters of size 333, with single stride and no zero padding.

Here W=64, F=3, P=0 and S=1. The output depth will be equal to the number of filters applied i.e. 10.

The size of the output volume will be ([64-3+0]/1)+1 = 62. Therefore the output volume will be 626210.

### The Output layer

* With no of layers of convolution and padding, we need the output in the form of a class.
* To generate the final output we need to apply a fully connected layer to generate an output equal to the number of classes we need.
* Convolution layers generate 3D activation maps while we just need the output as whether or not an image belongs to a particular class.
* The Output layer has a loss function like categorical cross-entropy, to compute the error in prediction. Once the forward pass is complete the backpropagation begins to update the weight and biases for error and loss reduction.

### Summary:

* Pass an input image to the first convolutional layer. The convoluted output is obtained as an activation map. The filters applied in the convolution layer extract relevant features from the input image to pass further.
* Each filter shall give a different feature to aid the correct class prediction. In case we need to retain the size of the image, we use same padding(zero padding), otherwise valid padding is used since it helps to reduce the number of features.
* Pooling layers are then added to further reduce the number of parameters
* Several convolution and pooling layers are added before the prediction is made. Convolutional layer help in extracting features. As we go deeper in the network more specific features are extracted as compared to a shallow network where the features extracted are more generic.
* The output layer in a CNN as mentioned previously is a fully connected layer, where the input from the other layers is flattened and sent so as the transform the output into the number of classes as desired by the network.
* The output is then generated through the output layer and is compared to the output layer for error generation. A loss function is defined in the fully connected output layer to compute the mean square loss. The gradient of error is then calculated.
* The error is then backpropagated to update the filter(weights) and bias values.
* One training cycle is completed in a single forward and backward pass.