**Conv2D**

It’s the 2D convolution layer. The primary purpose of this layer is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. These small squares of input data are called as filters. We can have any number of filters to be used on the image dataset. Each filter creates a different feature map than the previous filter.

keras.layers.convolutional.Conv2D(filters, kernel\_size, strides=(1, 1), padding='valid', data\_format=None, dilation\_rate=(1, 1), activation=None, use\_bias=True, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=None, bias\_regularizer=None, activity\_regularizer=None, kernel\_constraint=None, bias\_constraint=None)

**So this is the syntax of the Conv2D from the keras documentation. When we compare this to our code we can see that**

model.add(Conv2D(32, (2, 2), padding='valid',input\_shape=x\_train.shape[1:], )) #32 filters of size 3\*3

model.add(Activation('relu'))

model.add(Conv2D(32, (2, 2)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(64, (2, 2), padding='valid')) #64 filters of size 3\*3

model.add(Activation('relu'))

model.add(Conv2D(64, (2, 2)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(num\_classes))

model.add(Activation('softmax'))

the number of filters used in the first two layers of conv2d are 32 and in other two they are 64 all of size 3\*3. We slide our filters over the original image at every position and compute element wise multiplication to get a new matrix. This new matrix is called as the feature map.

The size of the Feature Map (Convolved Feature) is controlled by three parameters that we need to decide before the convolution step is performed:

1. Number of filters: The more number of filters we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images.
2. Stride**:**Stride isthe number of pixels by which we slide our filter matrix over the input matrix. If we don’t specify the number of strides, it is by default 1. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.
3. **Padding:**Sometimes, 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 padding is that it allows us to control the size of the feature maps.

We have two options in keras. Padding = same or valid. ‘Same’ means that its zero padding and ‘valid’ means that no padding is applied to the matrix.

The other parameters is the activation function that can be used in the convolution step. Here instead of using in the conv2D function, we have written it separately in the activation function. The default activation function we use is the ‘relu’ rectified linear unit for the inputs and ‘softmax’ for the output. ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.

Other non linear functions such as **tanh**or **sigmoid** can also be used instead of ReLU, but ReLU has been found to perform better in most situations. The sigmoid function can be applied easily, the ReLUs will not vanish the effect during your training process. However, when you want to deal with classification problems, they cannot help much. The softmax function squashes the outputs of each unit to be between 0 and 1, just like a sigmoid function. But it also divides each output such that the total sum of the outputs is equal to 1

Initializers: Initializations define the way to set the initial random weights of Keras layers. They are of two types Kernal initializers and bias initializers . And they can be set as follows

kernel\_initializer='random\_uniform',

bias\_initializer='zeros' in the conv2D function.

**RESULTS:**

After interchanging all these parameters, the highest accuracy that I got was 64% which is not a good model and it is underfitting. This result was achieved by having two 32 and two 64 conv2d filters of size (2,2) stride of 1 and no padding involved. According to me the accuracy of this model can be increased by increasing the epoch size and using RMSProp instead of Stocastic Gradient Descent