Case Study II

Krushang Shah

Prof. Richard Khan

Senior Design I

ID 23821856

Convolutional Neural Network

The name "Convolutional Neural Network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in the place of general matrix multiplication in at least one of their layers.

A Convolutional Neural Network consists of an input layer, lots of hidden layers and an output layer. The hidden layers consist of multiple layers that each work in a different way, to train the model on training data and make the model as accurate as possible. The different types of hidden layers that are present in a Convolutional Neural Network are

Convolutional Layer: This layer is used for convolution of the input, along the width and height of input volume, which gets better overtime, with each pass.

Local Connectivity: This layer relies on the correlation of the input, where it exploits the spatial local correlation, and each neuron of this layer is connected with a small region of input volume.

Spatial Arrangement: This layer depends on various factors like strides, depth and zero-padding, to determine the size of the output volume, which tells us the number of neurons in the output layer.

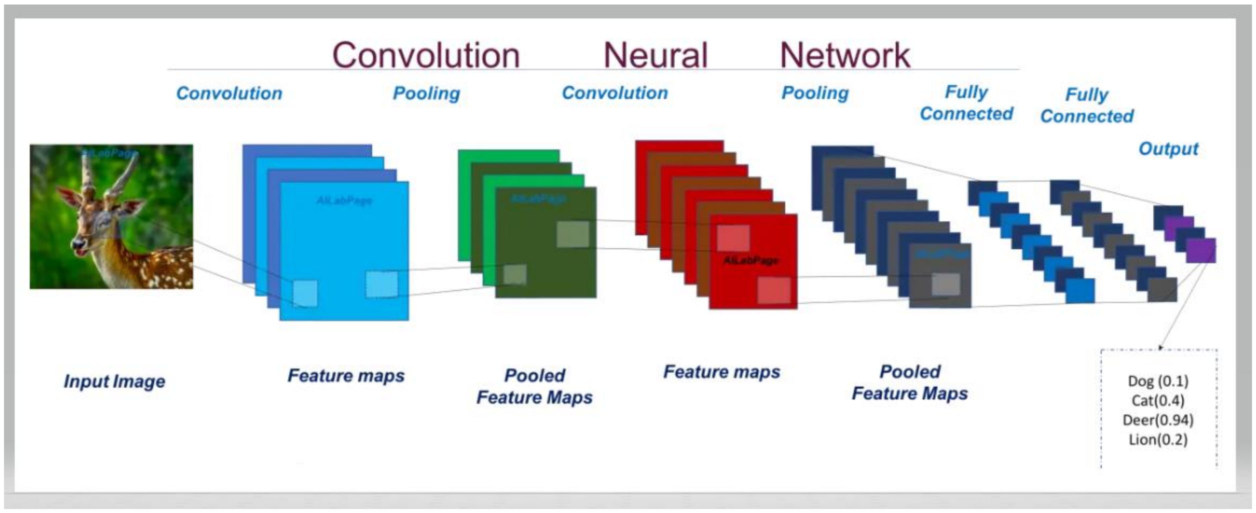
Parameter Sharing: This layer determines the number of free parameters in a neural network. This breaks the input into depth slices, where each slice uses same weights and biases, hence making the process faster and cheaper.

Pooling Layer: This layer takes the input and separates them in blocks as defined and then apples pooling on each block, to reduce the entire block into one cell. The pooling types that are usually used include max pooling, which takes the maximum element of the selected block for the next layer, and another type of pooling is average pooling, which takes the average of the entire block for the value of the cell for next layer.

ReLU Layer: ReLU stands for Rectified linear unit, which applies an activation function on the input, to take the maximum value, and eliminate all the negative values by setting them to zero.

Fully Connected Layer: This layer is connected to all activations of the previous layers. It is the layer after several convolutional and pooling layers, which does affine transformations and matrix multiplications, followed by a bias offset.

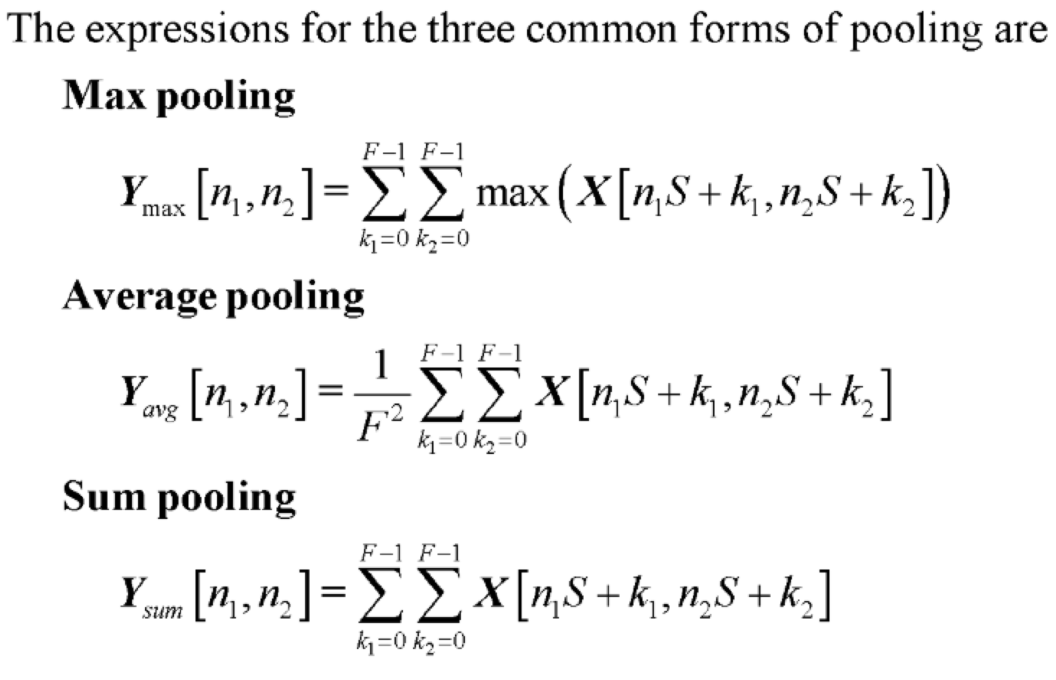
Loss Layer: This layer is used to check the loss that occurred during training, by checking the predicted labels and comparing them with true labels, to find the errors and the loss.



The above shown diagram is a general example of how a neural network would look like. There are multiple hidden layers that are associated with the neural network. Each layer performs its specific task, and the final goal is to make the neural network as accurate as possible. The above neural network has an image of a deer as the input, which then passes through convolutional layer, to develop feature maps, who's each one is convoluted using different filters and filter biases. The next step shown in the process is pooling, which reduces the width and height of the feature maps by using max pooling, to get pooled feature maps which are smaller in width and height compared to feature maps. The next to steps are doing the convolution and pooling, to get a better and more accurate model. Then the next layer is fully connected layer, which is connected to the activation functions of all the previous layers to develop a neural network. And the final step is where we get classified output, for example in the model shown above, it predicts that the image is of a deer, with 0.94 probability and the other classes also have some smaller probabilities. We finally apply SoftMax function, to set the value of the class that has maximum probability to one and all other classes to zero.

Pooling

The Pooling layer is done in a neural network for non-linear down sampling of the input data. There are several types of functions, to implement non-linear pooling, among which Max Pooling is the most common. In max pooling, the function partitions the input image into a set of non-overlapping rectangles and for each subregion, it outputs the maximum. In a neural network, a rough location is relatively more important than an exact location of a feature, and hence we use pooling, to get a rough idea of the image, and improve the cost of the function relative to time and memory both. It serves to reduce the number of parameters, memory footprint and amount of computation needed in the network, which in turn also controls overfitting. Hence, it is commonly used between two convolutional layers.



The expressions for the three common forms of pooling are

Max pooling

$$\boldsymbol{Y}_{\max}\left[n_1, n_2\right] = \sum_{k_1=0}^{F-1} \sum_{k_2=0}^{F-1} \max\left(\boldsymbol{X}\left[n_1 S + k_1, n_2 S + k_2\right]\right)$$

Average pooling

$$\boldsymbol{Y}_{avg}\left[n_1, n_2\right] = \frac{1}{F^2} \sum_{k_1=0}^{F-1} \sum_{k_2=0}^{F-1} \boldsymbol{X}\left[n_1 S + k_1, n_2 S + k_2\right]$$

Sum pooling

$$\boldsymbol{Y}_{sum}\left[n_1, n_2\right] = \sum_{k_1=0}^{F-1} \sum_{k_2=0}^{F-1} \boldsymbol{X}\left[n_1 S + k_1, n_2 S + k_2\right]$$

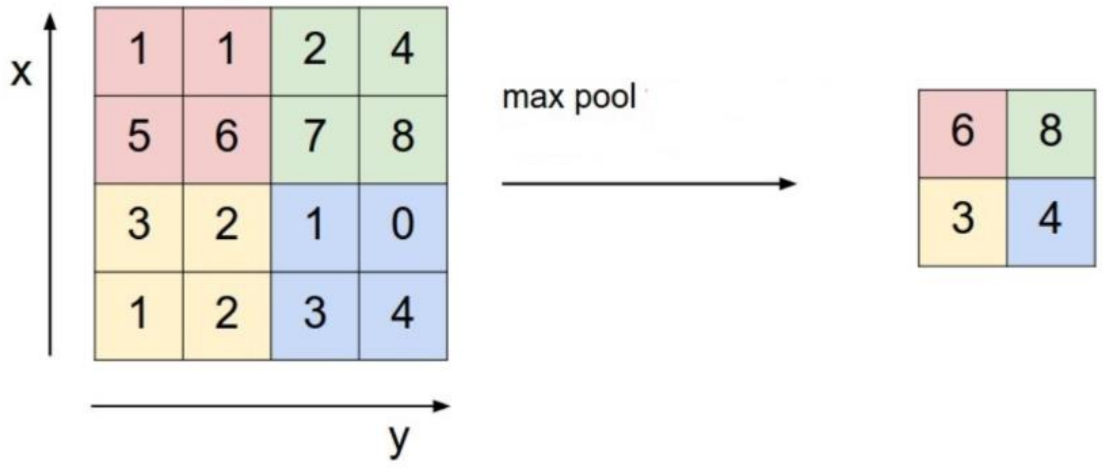
The above equations show the different types of pooling that are applied on a neural network.

Max Pooling: The max pooling layer takes the maximum along the defined width and height, k1 and k2 respectively. The function takes the maximum from each sub-region to output the maximum, which reduces the size that the neural network is working on drastically and it has almost the same results as using all the exact features of the image.

Average Pooling: The average pooling layer takes the sum of all the elements present in the selected region, and then divides it by the number of elements in the region, to obtain the average of the values present in the region. It reduces the whole image along its width and height, to a smaller region, that only has the averages of all the subregions.

Sum Pooling: The sum pooling layer takes the sum of the elements that are present inside the non-overlapping subregions. Each cell in the output layer, represents a part of the input layer, and whose value is sum of all the elements present in that part of the input layer.

Pooling



This is an example of how max pooling works. The input layer is a 4*4 matrix with 16 elements. The input layer is broken down into four non-overlapping subregions, each containing a 2*2 matrix of four elements each.

First Region (in Red): max(1, 1, 5, 6) -> 6

Second Region (in Green): max(2, 4, 7, 8) -> 8

Third Region (in Yellow): max(3, 2, 1, 2) -> 3

Fourth Region (in Blue): max(1, 0, 3, 4) -> 4

Each region of the input layer which was of four elements is reduced down to one element. Hence, reducing the input layer from 16 elements to 4 elements.