Case Study IV

Krushang Shah

Prof. Richard Khan

Senior Design I

ID 23821856

Convolutional Neural Network

The term Convolutional Neural Network says that it is a Neural Network, that uses Convolution. A Neural Network is a multiple layer of neurons that are connected through different functions. The convolution operation is a linear operation that can be used instead of matrix multiplication in mathematics. A Convolutional Neural Network (CNN/ ConvNet) consists of an input layer, an output layer and several hidden layers that consist of different layers that do different types of operations.

The main advantage of a Convolutional Neural Network is that the pre-processing required for a Convolutional Neural Network is way lesser than that for general classification algorithms. In older and traditional algorithms, filters were designed by humans, however a Convolutional Neural Network has ability of improving the accuracy overtime by learning and improving the filters and characteristics by itself. The different types of hidden layers that are a part of a Convolutional Neural Network are

Convolutional Layer: This layer uses convolution to convert the matrix representation of an input image to a feature map, by multiplying each entry of the matrix with a feature detector, which are also known as filter and filter biases.

Local Connectivity: In a neural network with large amounts of neurons, it is almost impossible to connect each neuron to all the other neurons. Hence, this layer uses local spatial connectivity to establish a pattern between neurons and adjacent layers.

Spatial Arrangement: This layer controls the size of the output volume using three parameters depth, stride and zero-padding. The depth controls the number of neurons, the stride controls the dimensions like height and width, and the zero-padding adds the border of zeros on the input layer.

Parameter Sharing: This layer determines the number of free and shared parameters in a convolutional layer. This layer determines the depth slices in the input, which are slices of neurons in each depth, that have to use common biases and weights.

Pooling Layer: This is another important layer in a Convolutional Neural Network. It applies non-linear function like max pooling to take only the maximum cell from each block of elements. While this does not really make much difference to the result accuracy, it drastically decreases the time and memory footprint.

ReLU Layer: This layer is called Rectified Linear Unit, which applies non-saturating activation function max on the input, which takes maximum between the input value and zero, essentially changing all the negative values to zero.

Fully Connected Layer: This layer is used after multiple layers of convolution and pooling. This layer is responsible for the high-level reasoning of a neural network. It is a fully connected network, where each neuron has connections to all the activation functions in the previous layer.

Loss Layer: This layer determines the loss that occurred throughout the training of the network, and penalizes the network based on the predicted and true labels by comparing them.

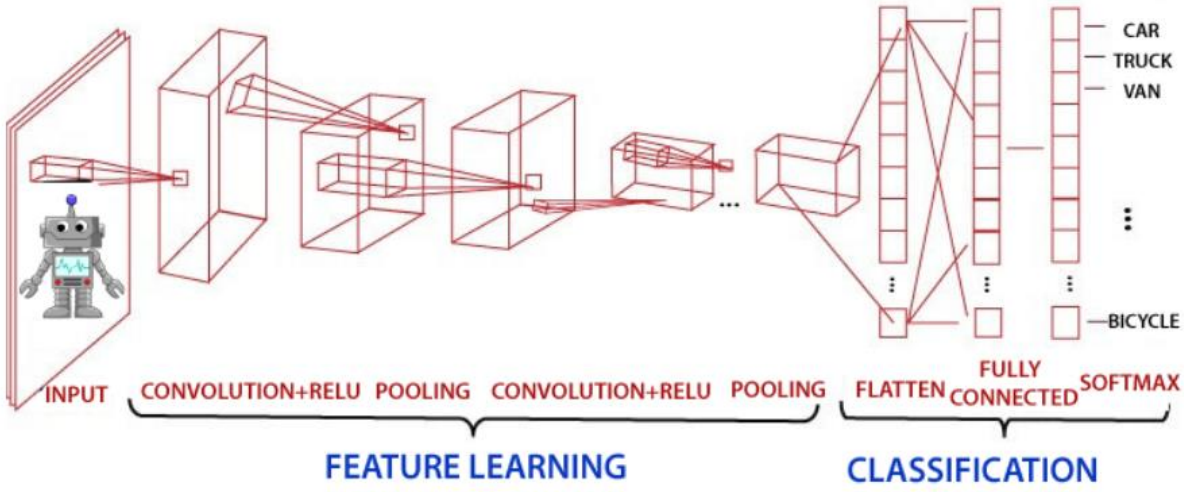

Figure 1: General example of a Convolutional Neural Network

The above shown diagram shows an example of a Convolutional Neural Network. Generally, a neural network can be separated into two main parts. One is Feature Learning where there are different types of mathematical and activation functions happening. And the second part is the classification, which combines the results from the previous layers, flattens the input into a tensor from a matrix and then classifies them based on different class descriptions. Here for example, an image of robot is used as an input, which then processes through Convolutional layer and ReLU layer, the convolutional layer convolutes the input image to make a feature map, using filters and filter biases. The ReLU layer removes all the negative values and only works with positive values, which improves the time footprint. Then it passes through pooling layer, which decreases the size of input by applying max pooling or other types of pooling. The neural network uses multiple instances of Convolutional layer and pooling layers to improve the accuracy of the model,

but not overfit on the training data. The next part of a Convolutional Neural Network is the Classification part. After processing the input data from multiple layers, it finally passes to the flattening layer, where the data gets flattened from a matrix or grid form to a tensor. The next layer is the fully connected layer. In the fully connected layer, all the neurons are connected with the activation functions from the previous layers. The next step is passing the values through softmax function to take the maximum of one object and make others zero. The final step is the classification of the input image into the appropriate category based on the value obtained in the probability.

ReLU Layer

ReLU stands for Rectified Linear Unit. The ReLU layer operates based on each element. It takes in the values of each element and compares them with zero and returns the maximum. Essentially, it keeps the positive values unchanged and changes the negative values of the input to zero. Convolution is a linear operation and to compliment that, we use ReLU layer, which is a non-linear function to introduce a non-linearity to out model, to make it operable on real life data and avoid overfitting.

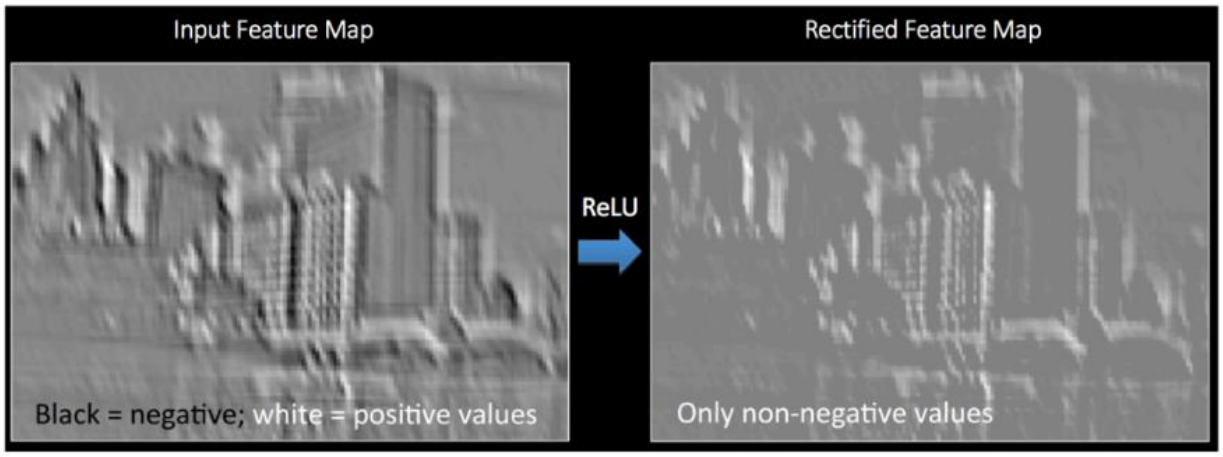

Figure 2: ReLU Operation

The above image shows a clear example of the ReLU layer. The negative values are the black parts of the first image and the positive values are the white part in the first image. In the second image we can see that after passing the image through the ReLU layer, we get an image that only consists of gray and white pixels. The white pixels represent positive values and the gray pixels represent values that are zero. The ReLU layer also converts the negative values to zero, hence the second image does not have any black pixels.

There are multiple non-linear activation functions that can be used in place of ReLU, based on the need for the model. The most common are

Sigmoid/Logistic: This can be used to prevent jumps in the output values. The output values of each neuron are bounded between 0 and 1. It is computationally expensive and follows the path of a sigmoid curve.

Hyperbolic Tangent/TanH: This is almost similar to sigmoid function, but it is zero centered which makes it easier to model all types of inputs.

Rectified Linear Unit (ReLU): This function converges quickly. It looks like a linear function, but it is a non-linear function. When the values tend to zero, or are negative, this function makes them zero, which does not allow back propagation and the model cannot learn.

Leaky ReLU: This function is almost similar to ReLU, but it has a small slope for negative values instead of making them zero, to enable the backpropagation that is missing in the ReLU layer.

Softmax: This function normalizes the output between 0 and 1 and divides by their sum**.** This is used before the classification step, to get the value of a specific class for the output.

Swish: This function is recently developed, which performs better than ReLU but has almost the same cost. It has a steep curve for positive values and very small values for negative input values.

Parametric ReLU: This makes the negative slope of a Leaky ReLU into a learnable curve, so that back propagation can be possible and the slope of the negative part can be send as a parameter back to the function.