# Assignment 2: Image Classification with Tiny imageNet using Convolutional Neural Network

G. Manasvi and S. Sai Avinash

October 5th, 2018

## 1 Introduction: Convolution Neural Network

The convolution neural network(CNN) is the workhorse of deep learning applications for image processing. It substantially reduces the number of parameters required to train an image when compared to a multi layer perceptron network. It, in fact, makes the process of learning and classifying images computationally tractable.

Originally Yann LeCun published the initial conceptualisation of convolution neural network. The work became famous due to its usage in the architecture called Alexnet, named after chief architect Alex Krizhevsky. It achieved a groundbreaking error of 15.8% classifying millions of images on a historically difficult dataset.
Current state of the art architectures surpass human level performance.

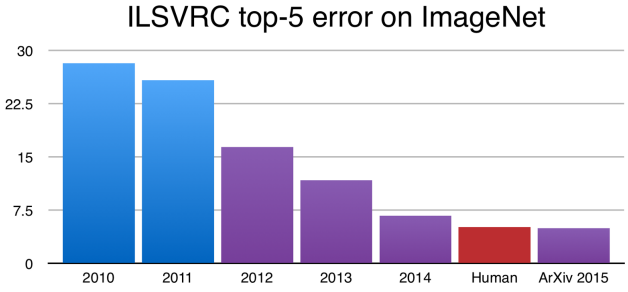


Figure 1

## 2 Architecture: How CNNs learn

Fundamentally regular neural network doesn't scale well for images. In a fully connected multilayer perceptron network, for training a $32 \times 32 \times 3$ (CIFAR -10) sized image, we would need 3072 weights only in the first layer itself.

CNNs try to isolate the important aspects of an image using the principle of convolution filters. Convolution layer is used compute the dot product (specifically, convulution) of the image with the filter. We can use multiple filters each of which can be used

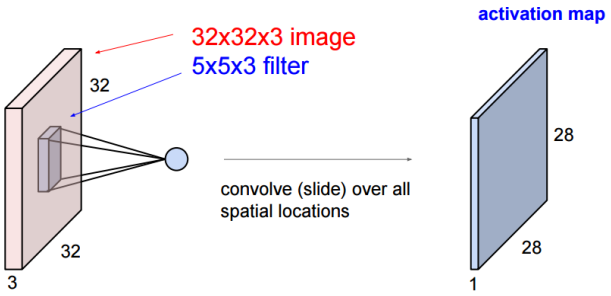


Figure 2: Convolution layer illustration.

to detect different features in an image. We can also have multiple convolution layers in a row to double down particular characteristics and make the computational process more tractable. Refer Figures 2 and 3.

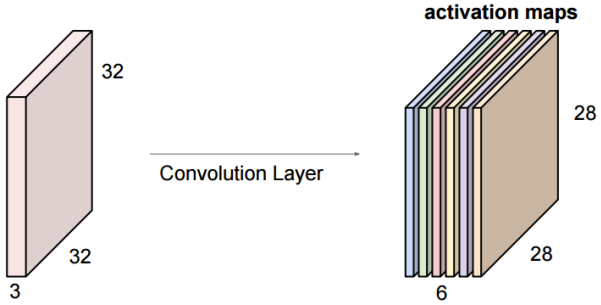


Figure 3: The result when we have multiple filters.

ReLU layer does an elementwise operation such as max(0,x). This doesn't change the dimensions of the image. There are several advantages for using the ReLU layer. One, the gradient computation is very simple (either 0 or 1 depending on the sign of x). Two, the computational step of a ReLU is easy: any negative elements are set to zero. Other activation functions such as tanh or sigmoid are more complicated and require considerable computational power over larger datasets. Refer Figure 4.

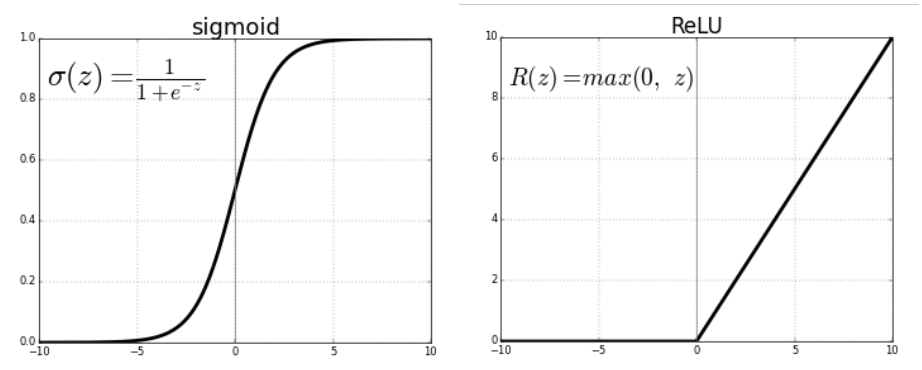


Figure 4: We illustrate the simplicity of the ReLU over a sigmoid.

Pooling layer is perhaps one of the most important layers in the CNN. without the right pooling functionality we would end up training the CNN for longer periods of time. It is common to insert pooling layers between successive convolution layers. A pooling layer generally are maxpool [Figure 5] (which takes the maximum value over the receptive field of the filter) or average pool (takes the average of all values over the receptive field). Fully connected layers are utilised after all the convolutions, ReLU, and pooling layers. The role is similar to that in a multi layer perceptron network.

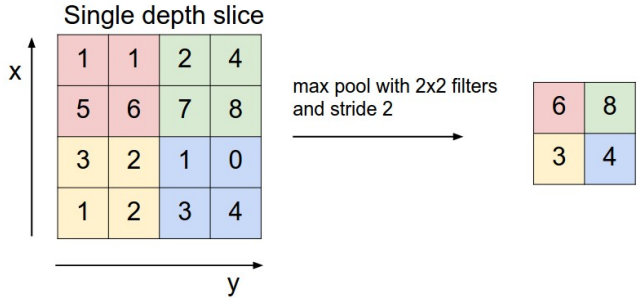


Figure 5: The image shows the operation of Maxpooling

## 2.1 Backpropagation in CNN

The concept used here used is the chain rule itself. But it gets a little trick as generally all the weights in a filter can contribute in the output of a convolution layer. The output of the convolution layer is what propagates the error.

Say we have an input X of size $3\times3$ and a filter of size $2\times2$ without any padding stride = 1. The simple example is chosen just to make it easier to explain our current understanding of the process. Refer Figure 6 and 7.

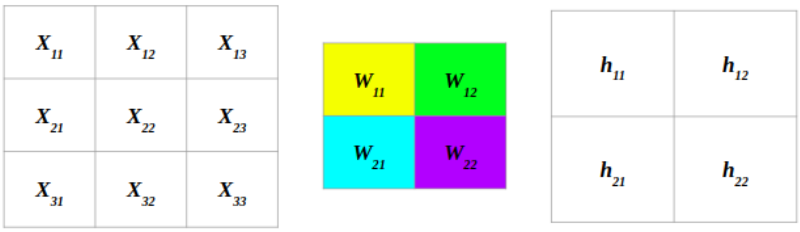


Figure 6

$$h_{11} = W_{11}X_{11} + W_{12}X_{12} + W_{21}X_{21} + W_{22}X_{22}$$

$$h_{12} = W_{11}X_{12} + W_{12}X_{13} + W_{21}X_{22} + W_{22}X_{23}$$

$$h_{21} = W_{11}X_{21} + W_{12}X_{22} + W_{21}X_{31} + W_{22}X_{32}$$

$$h_{22} = W_{11}X_{22} + W_{12}X_{23} + W_{21}X_{32} + W_{22}X_{33}$$

Figure 7

Now, for implementing the back propagation step for the current layer, we can assume that we get $\delta$h as input (from the backward pass of the next layer) and our aim is to calculate $\delta$w and $\delta$x. Thus if we utilise the equations showed in Figure 7 we get the equations used to update the weights in the filter [refer Figure 8] .

$$\partial W_{11} = X_{11}\partial h_{11} + X_{12}\partial h_{12} + X_{21}\partial h_{21} + X_{22}\partial h_{22}$$

$$\partial W_{12} = X_{12}\partial h_{11} + X_{13}\partial h_{12} + X_{22}\partial h_{21} + X_{23}\partial h_{22}$$

$$\partial W_{21} = X_{21}\partial h_{11} + X_{22}\partial h_{12} + X_{31}\partial h_{21} + X_{32}\partial h_{22}$$

$$\partial W_{22} = X_{22}\partial h_{11} + X_{23}\partial h_{12} + X_{32}\partial h_{21} + X_{33}\partial h_{22}$$

Figure 8

# 3 Our Architecture

We have arrived at our architecture after a system of trial and error. Initially we used a regular architecture for CNN which consists of the following layers:

- Convolution (C),
- Max-pooling (MP),
- Convolution(C),
- Max-pooling (MP),
- Convolution (C),
- Fully Connected (FC)

. We arrived at an unimpressive accuracy (certainly less than 20%). Next, we increased the convolution layers without any change in the position of pooling layers. We did not notice a drastic improvement. Further, we increased the MP layers and we actually saw a decrease in accuracy. Finally, we increased the pooling layers and played around with average pooling to arrive at our current loss and figures.

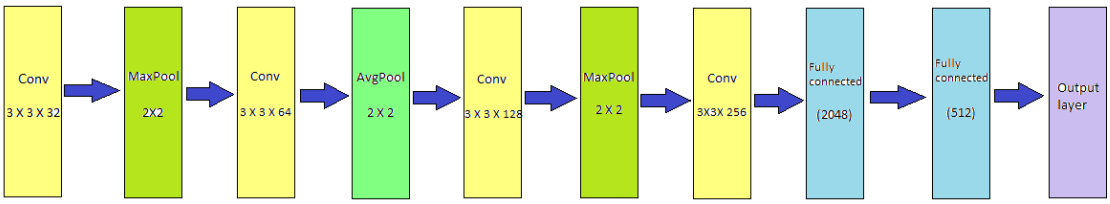


Figure 9: Our Architecture

# 4 Results

In this section we present the results of our CNN. It was run for 20 iterations to achieve this accuracy. We are confident it would lead to a higher accuracy over longer training periods as we can see an increasing trend.

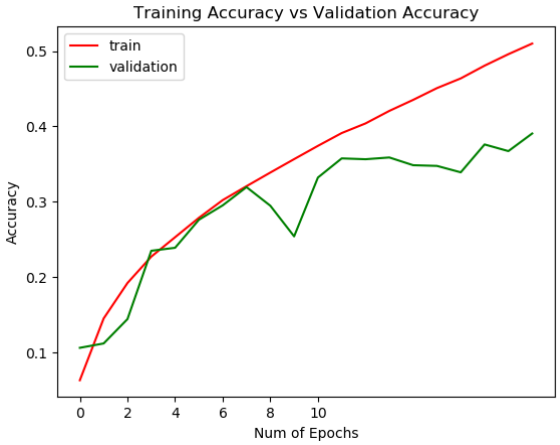


Figure 10: We can see an increasing trend line for accuracy over the 20 epochs. Our validation accuracy is approaching 40%

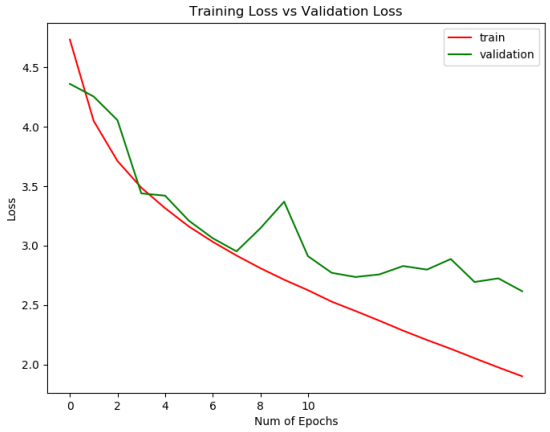


Figure 11: We can see the loss function, as expected, decreases. The decrease in loss for training data seems to be exponential

# 5 References

1. Class Lectures and Notes
2. CS231n: Convolutional Neural Networks for Visual Recognition.
3. Internet blogs for the images used.