Image Classification for Handwritten Digits Recognition.

Authored by: Neel Kanwal

we are going to design a fully connected network for training and validation of dataset then test the performance of the network on classifying the hand-written letters. The main goal is to learn how to set the optimal parameters and to compare the importance of those parameters on accuracy and cost function.

Evaluate the accuracy of the trained network

After we have designed the required network with three hidden fully-connected layers, we should train that providing as input the training dataset. After training it, we test the network by providing new images from validation dataset as input. In this way, we can see the performance of the network and if the network is able to recognize new characters that have never seen before. Below we plot the accuracy and the entropy cost function for validation and training dataset.

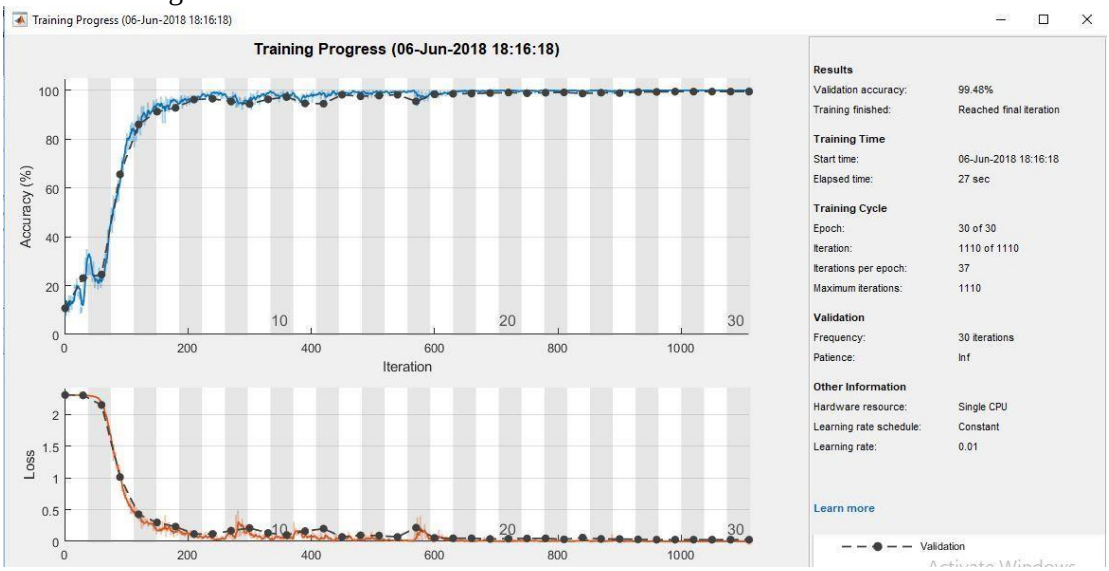

Figure 1: Training & Validation of Fully Connected Neural Network

We see that the Validation accuracy is 99.48% means that the network has provided a correct result for most of the new images and this is a very good performance. We did the calculations of accuracy also using the formula: Val_accuracy = sum(YTest == TTest)/numel(TTest) and we got the same result.

Check the classification of the image.

As we know from the theory, in the first few epochs the cost is higher, and accuracy is very low because the network does not have the optimal weights and bias. In the beginning, the values of weight and bias are chosen randomly and as possible of correction of the output is low but as we go on with other epochs we see that the accuracy increases faster and then remains almost steady.

We selected random different images from validation set for classification of hand written digit and the result gave from the network was the same with the number shown in the image.

Perform the analysis when changing: *Figure2: Classification* • Reducing the number of fully-connected layers will lead to a decrease of validation accuracy by 75.68% and the training time is increased because of backpropagation algorithm and insufficient weights and bias for activation.

• Increasing the number of fully-connected layers to 4 will increase the validation accuracy by 97.72% but also the training time. It will take more time to train the network because it has much more parameters. Another thing that we notice is that accuracy, in this case, takes much more time to start increasing significantly, let's say that at epoch number is 10, it starts to increase. This is due to the phenomena of "Vanishing gradient". Cascading layers will cause a decrease in the learning rate of the layers that are close to the input.

• If we decrease the size of minibatch to 150 the validation accuracy becomes better up to 99.48%, but as we are taking smaller mini batches means that network needs to do more iterations to pass all the image set before updating the weight and bias. So, it takes technically much more time to train it, around 1min. Whereas if we increase the size of minibatch to 220 the performance of network will be a little bit lower 99.32% but it will take less time to finish. Also, it reached the highest value of accuracy for more epochs.

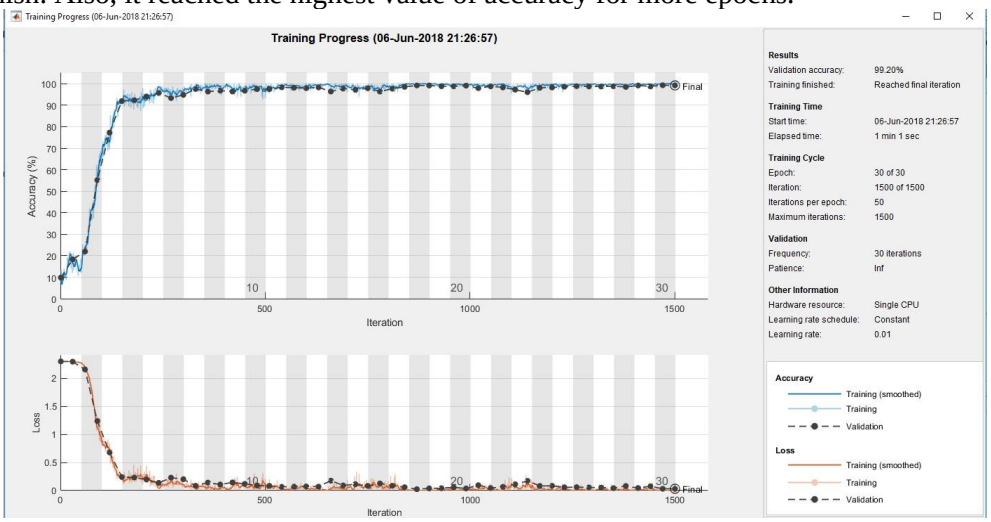

Figure 3: Variation of number of Epochs

• If we increase the learning rate to 0.1, we see that the validation accuracy is 10% but it took less time to finish the training. When we increase the learning rate (it is included in the formula of updating the weights and biases) we take bigger steps on changing the weights and we skip the lower point of our cost functions. The cost is very high, and the network is not performing good because of stochastic gradient.

Decreasing the rate will slow down a lot the training of the network, but we are more confident in the direction that we move on a cost function. We take small steps and training will be more complex. The advantage of this method is that we will never get stuck on a local minimum, we can find a low value on one of mini batches and move out.

• If we want to change the size of the fully-connected layers by putting it 30 the accuracy is 97.12% and it will take much more epoch times to reach the highest value. Whereas if we put the size 200 we see that it will peak the max accuracy at the third epoch time and reaches the validation accuracy 99.68% that is much better. Having more outputs increases the complexity of the network but also the performance because each neuron of next layer has more incoming edges and more info.

Overfitting of Neural Network

In order to make the network overfit we must provide a different input dataset for training or to increase the number of neurons, the parameters with respect to the number of the input dataset. In this way, the network will be able to recognize all the datasets that we provide to it, will learn with higher order the peculiarity of this data but will not be able to learn other types of data. So, when we come for validating our classification points start moving out of region. This is converse problem to early stopping and need to be covered with parallel validation testing during experiments.

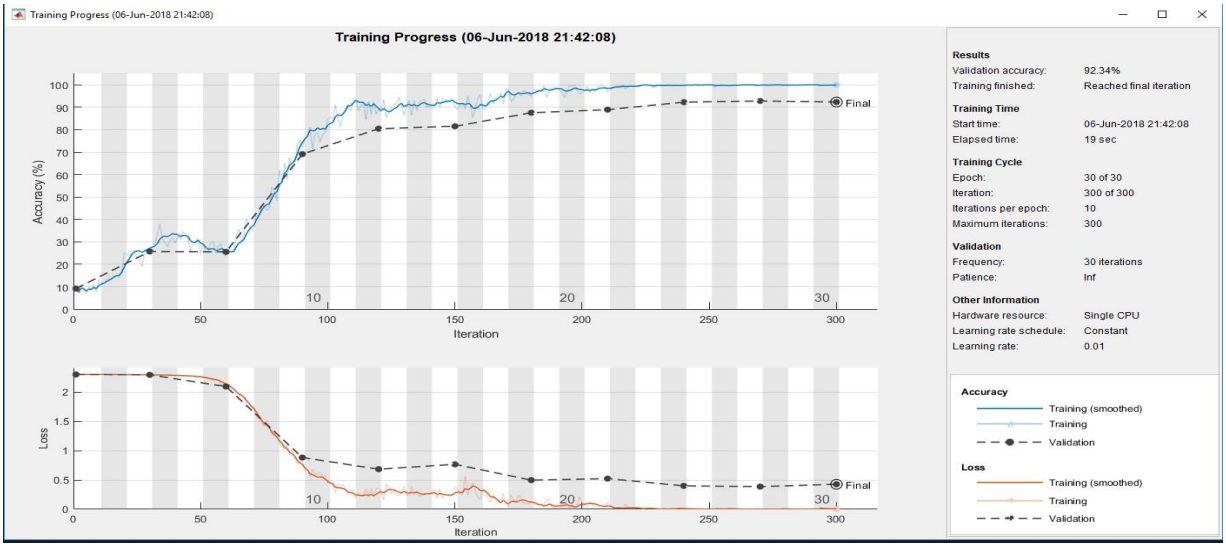

Figure 4: Network Overfitting

We decreased the number of training data to 200 and we noticed that the accuracy of training data is high, around 98% whereas the accuracy of validation data is 92.34%. This means that our network can learn well just the training data for which it is also trained, meaning that are chosen the optimal parameters. By having more layers, we will have more neurons and more parameters, the training will be more complex and will be very efficient for the training data but will not work well for new one. That's why it is better to have a less complex network that can work well for the unknown type of datasets.

Deep Learning Part 2 - Convolutional neural networks

In this part of lab, we must design a convolutional neural network that has to classify the handwritten letters. The procedure, in the beginning, is the same as in the previous lab, we should load the dataset, chop it in two parts for the testing and validation. The different lies in hidden layer which are based on convolution architecture now.

Test the performance of the network

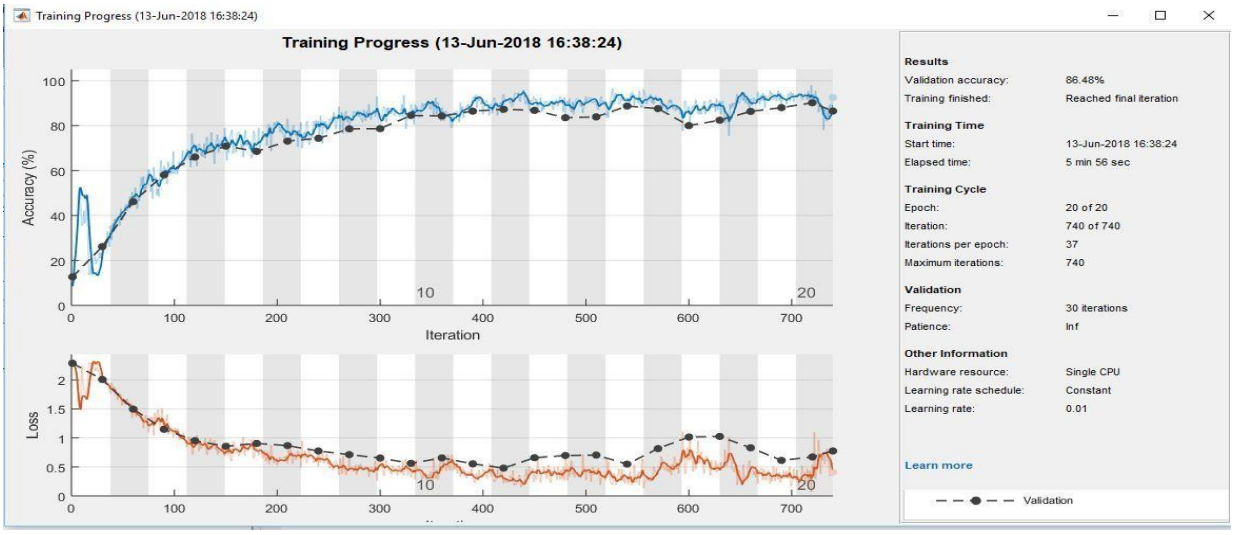

Figure 5: Convolution Neural Network

We design the network following the instructions and then we test the performance of the network by providing as input the validation dataset. The performance is shown in the figure below and it is conspicuous the validation

accuracy is around 86 %. Another important thing to be mentioned is that with this network it takes much more time to be trained because of shared weights and pooling. From the plot, we can say that the accuracy peaks its maximum value after 10 epochs and can be seen also fluctuations. So, the performance of our network varies a lot during initial epochs.

The learned filters

The learned filters for conv1 and conv2 are outputs of layers We followed the instructions and we got that for the same sample of the image the activation of the first convolutional layer is not so clear as the one of the second layer. In practice, the activation is a filter output and it can distinguish features of different images. From the theory, we know that the early layers learn slowly then latest layers. Below are provided the weights learned by both layers.

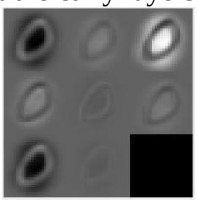

Figure 6:Activation from the first layer

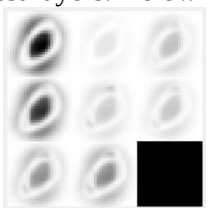

Figure 7: Activation from the second layer

Change the size of filters

If we change the size of the filter will for sure number of neurons for block of 28*28 and the entropy because of different segments made by layer for parametric variation. We shall try to elaborate this by following two different gradients of increase in the filter size and reduction in filter size.

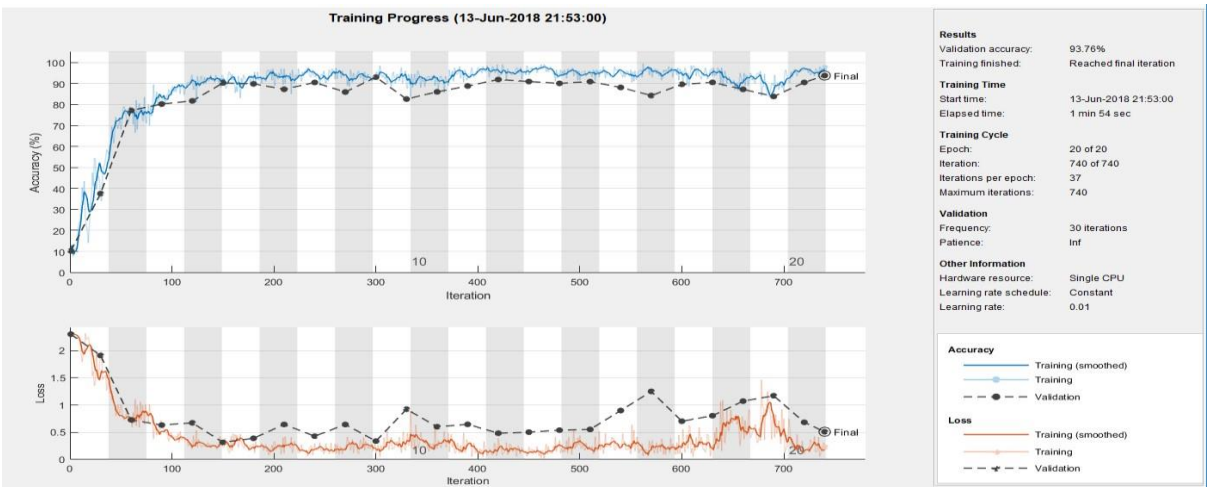

Figure 8: Reduction in size of Filter

The above simulation depicts when we reduced the filter size in both layers to [4,4] and [2,2] and the output/activation of the second layer changed a lot and becomes more difficult to distinguish because we have a smaller number of outputs by neuron and their convolution does not result similarly. Also, the accuracy decreased significantly to 93.78%.

A view of weights from first convolution and second convolution layer, in this case, is below.

Figure 9: First Convolution Layer

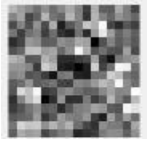

Figure 10:Second Convolution Layer

Size of the weight matrix depends on the parameters which we initialized in the network formulation. These matrices are reshaped to make them more feasible for visibility of montage. Hidden layers with more number of outputs own a larger range of weights and biases so they return matrices of higher dimensions. This training takes more time overall.

By increasing the size of the filter, we notice that training the network takes much more time because the number of neurons are changed. Whereas fragmentation into less part increases the number of iteration. This practice results in improvement of accuracy but the reduced performance by affecting the training time and cost.

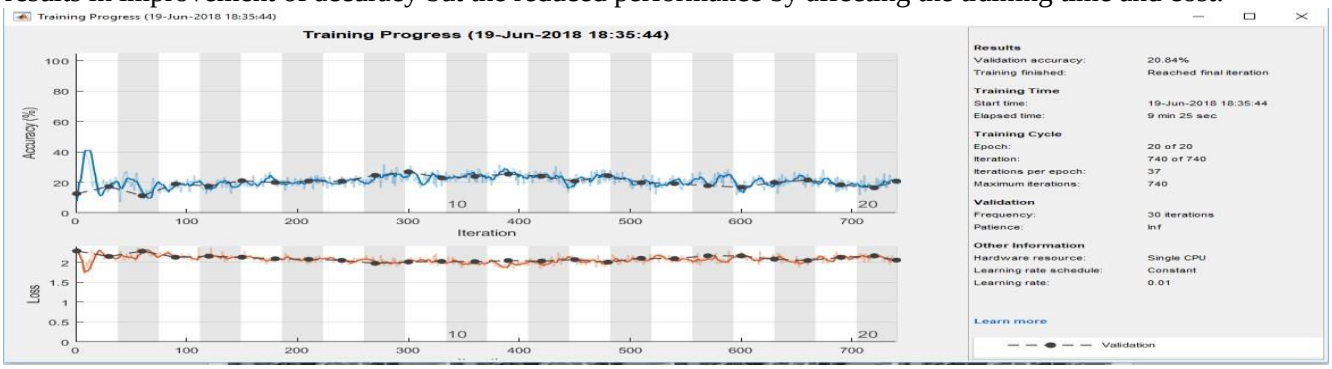

Figure 11 : Increment in size of Filter

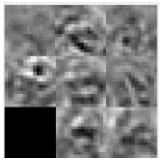

Figure 12: First Convolution Layer

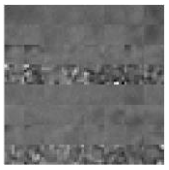

Figure 13:Second Convolution Layer

Inserting batch Normalization layer

The main goal of adding the batch Normalization layer is to normalize the weights and errors activation of successive layers, to be somehow equal and in this way to speed up the learning gradient and reduce the sensitivity. In this way, the performance of the network will be better. The plot will be much steadier, meaning that the value of the accuracy will be bounded and will not vary too much. So, it will be much easier for the network to reach the highest value, better performance. It is more noticeable when we are training the network with high learning rate, that having an unbounded variance of activations energy will lead to unstable accuracy plot. We increase the learning rate to 0.15 and below are shown the plots with this layer and without.

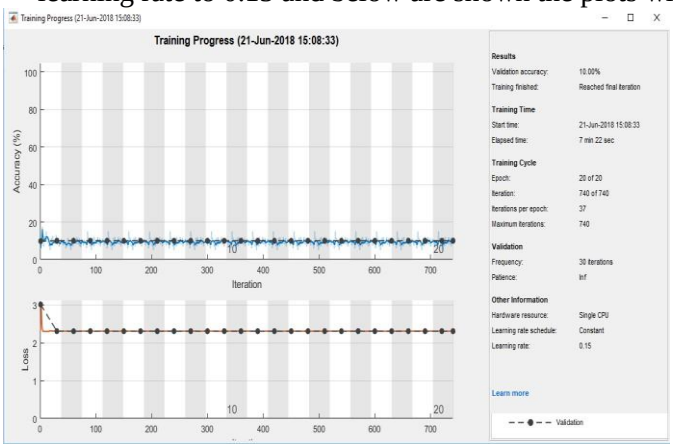

Figure 14: Without Batch Normalization Layer

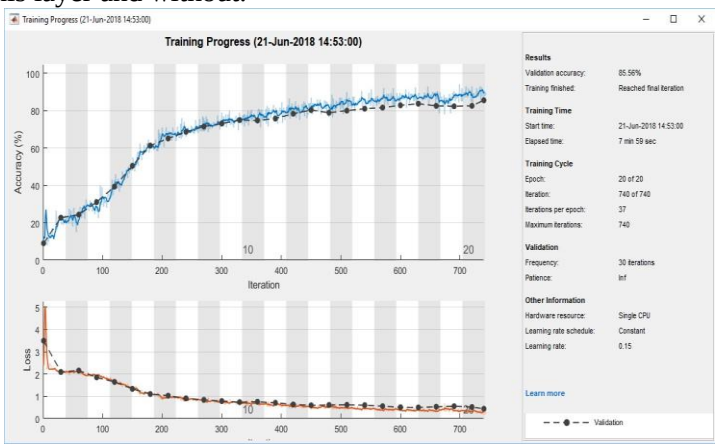

Figure 15: With Batch Normalization Layer

Fully-connected neural network vs Convolutional Neural Network (Conclusion)

In a the fully-connected neural network we have weights per each edge that connects two consecutive nodes. Consequently, we have much more parameters and training the network is more complex due to a high number of computations. Whereas in the case of a convolutional neural network, we share the weights per each layer and so we reduce a lot number of parameters. This network is less complex follows receptive fields to make convolution to provide the output. In our lab, whereas it is also observable that convolution neural network takes a lot more time as comparative to fully connected one because it has to perform sliding convolution of every out parameter to all other input parameters for successive hidden layer. There is possibility to use pooling layers to boost performance by down sampling from preceding layers to successive hidden layers.