Computational Intelligence 2020 **(Due: 14/05/2020)**

Submission Assignment #4

Instructor: Jakub Tomczak *Name:* Tyron Landman, *Student Id:* 2674357

1 Introduction

In this report I will be using two neural networks to analyze and distinguish differences between them while undertaking the task of image classification. Image classification can be a difficult thing for computers as instead of seeing a cat as a cat or a dog as a dog, the computer sees binary. As a result,distinguishing between different classes of images can be difficult especially between similar classes.
Because of this, I will be looking at implementing two currently used Neural Networks which can be used for image classification to try and create a model that can more accurately distinguish between images.
The neural networks that I have chosen to implement and test are, Fully-connected Neural Network (FCN), and a Convolutional Neural Network.
An FCN can be thought of as having a series of layers whereby each layer has n number of nodes/neurons. In an FCN each neuron in the a layer, is directly connected to each neuron in the second layer and so on until the final layer.
As for a CNN, has both an input and output layer, with 1 or more hidden layers in-between. These hidden layers alter the information through activation functions, convolving, pooling, and can even include fully connected layers as well.

2 Problem statement

To test these two neural networks(NNs) I will be doing image classification on the data-set named "Imagenette". This data-set consists of 10,000 images belonging to 10 different classes. The goal of each algorithm will be to train on a subset of this data called the training-set. Once completed, we will be able to take the trained model and test it on another subset of data called the test set to determine the overall progress and quality of the algorithm.
To further enhance the quality of the model we will add in a third subset of data that we will call the validation set. this subset of data will be used as the model is trained. This is to assist in avoiding any over fitting issues that may result from training too many times on the training set, giving the illusion of an increased accuracy during training but then failing during the actual testing phase.
For these tests, we will use Cross Entropy Loss, as our objective function. As this decreases and gets closer to 0 we will be reaching our goal of a perfect model that can accurately differentiate between the different classes in out data-set.
As for the individual NNs, for the FCN, I will be using two linear layers, Rectified linear unit (ReLU) as the activation function, and softmax for the actual output.
For CNN, I will run it through a convolve and ReLU before pooling it, then do a linear and ReLU combination, before finally adding a linear and softmax combination for the output.

3 Methodology & Experiments

The first part of answering our problem statement was to import the Imagenette data-set. Once this was done, as mentioned above, I split this data-set into three distinct groups: Training Set, Validation Set, and Test set. After this, the code for both the FCN and CNN were implemented following the structure mentioned before as well.
From here, I created a for loop to iterate over the entire data set calling the range an Epoch. For example, for one iteration of the entire data it would be one epoch. From here, I create two more for loops, one to train the model on the training set, and once completed one to validate the updated model. Having this combination allowed us to iterate over the entire training and validation data allowing me to best avoid issues such as over fitting in the training set. This is especially useful as when the number of epochs increases, more iterations inside the training set for the model may lead to a higher chance of over fitting.
The for loop containing the training set, would take batches of the data within the training set and run it through the FCN model. From there it would be able to calculate the loss using the cross entropy objective function. After this accuracy would be tested by comparing correct and incorrect valuations done by the model at this point in time and appended to a list for plotting at a later point.
Once the training was complete, the same process would be run except this time using the validation set. What we would expect from this, given the validation set has fewer images, is to have a loss slightly higher, and an accuracy slightly lower than that received in the training set.
Lastly, once the predefined amount of epochs has been run, the model created will enter it's third and final phase. This phase will be to run the test set data through the latest model to determine the final accuracy of the model.
Since the FCN and CNN models themselves won't be altered at any point in time, the most important hyper-parameters to be tested to determine end results will be the number of epochs and the learning rate. These two variables will both be adjusted and tested creating overall a total of 18 separate runs (9 for FCN, and 9 for CNN). This will allow us to see a performance metric displaying how both hyper-parameters affect each NN as well the how each NN performance against the other in different contexts.
For values of these variables will be as below:

Epoch = [10,50,100]
Learning Rate = [0.1, 0.01, 0.001]

Using the different combination of the above variables we will be able to see how the overall accuracy can change over time given the amount of times the model gets to be trained (epochs) and how quickly it can learn (learning rate). Some issues to keep in mind, and the reason I have set the experiments up like this is that as the learning rate decreases, the model is able to become more accurate. However the issue is that is would take many more epochs to improve the performance. Alternatively, with a higher learning rate we run into the issues of the model potentially reach a point where it can't get better. This is because as the model gets closer to the optimum performance, the learning rate may cause the weights to jump back and forth for each of the epochs.
Finally, to simplify the graphs and to avoid unnecessary information, I have decided to keep only the first and last of both the loss and accuracy. This will help condense the data to a manageable amount that can be displayed within the report especially as epochs increase up to 100.
Further to this, we will be able to see just how far each of the NNs come over a given epoch much more clearly.

4 Results and discussion

Epochs	Learning Rate	Start Loss	End Loss	Start Accuracy	End Accuracy
10	0.1	2.29	2.1	14.39	38.24
10	0.01	2.318	2.25	12.25	21.2
10	0.001	2.32	2.318	8.05	11.43
50	0.1	2.29	1.97	14.54	51.05
50	0.01	2.32	2.13	10.93	36.19
50	0.001	2.32	2.283	11.64	16.24
100	0.1	2.29	1.89	13.96	59.86
100	0.01	2.31	2.07	10.87	41.94
100	0.001	2.32	2.25	10.92	19.85

Table 1: Training Data, FCN

From table 1, we have the performance metric of the FCN during the training portion of the experiment. As we can see the tests completed where the model had a higher learning rate of 0.1 had the highest end accuracy results of 38%, 51% and 60%. This would make it seem like a higher learning rate, even if just at these lower epoch amounts, would be the best route we can take. However we can still see a steady rise in performance in both the learning rates of 0.01 and 0.001, slower, but still increasing.

Epochs	Learning Rate	Start Loss	End Loss	Start Accuracy	End Accuracy	Final Accuracy
10	0.1	2.35	2.19	19.01	36.7	35.41
10	0.01	2.4	2.33	10.56	21.2	19.54
10	0.001	2.408	2.404	7.75	11.13	9.81
50	0.1	2.35	2.16	18.24	39.5	40.36
50	0.01	2.4	2.225	11.06	35.77	33.76
50	0.001	2.4	2.37	11.34	15.14	14.93
100	0.1	2.34	2.11	18.87	42.68	43.82
100	0.01	2.4	2.17	10.14	38.52	37.96
100	0.001	2.4	2.33	10.56	19.23	18.24

Table 2: Valid Data / Final Accuracy score, FCN

From the table above, we can see the validation and finally the final accuracy score of the FCN. As per above, the learning rate of 0.1 had the greatest influence on accuracy of all the learning rate. Further to this, what is interesting to note is that even though the accuracy was much higher during the training set for a 0.1 learning rate, the valid end accuracy is much closer to that of 0.01 and 0.001. We can see this especially in epoch 100 where the end accuracy for a 0.1 learning rate is 59.86% but only 42.68% in the valid one. As well as for the learning rate of 0.001 where the end accuracy during training was 19.85% and then 19.23% in the valid set, creating the more consistent results.
Finally, from the final accuracy count, we can see that the learning rate of 0.1 had the best results overall. This is expected as it learns quite a bit faster but what should be taken into consideration is as epochs rise the percentage and consistency of the lower learning rates do start to out perform the learning rate of 0.1. It would be no surprise that if the epoch was to be increased to 1,000 that the performance of the lower learning rates would outperform the higher learning rate.

Epochs	Learning Rate	Start Loss	End Loss	Start Accuracy	End Accuracy
10	0.1	2.3	2.078	12.65	40.03
10	0.01	2.3	2.28	9.85	16.13
10	0.001	2.231	2.319	8.26	10.66
50	0.1	2.31	1.83	11.93	65.34
50	0.01	2.3	2.1	10.9	38.84
50	0.001	2.321	2.304	10.31	15.28
100	0.1	2.308	1.679	11.98	79.66
100	0.01	2.32	2.039	10.95	44.5
100	0.001	2.321	2.298	9.04	12.44

Table 3: Training Data, CNN

From table 3 we see the result of the performance of the CNN in the training portion of the experiment. Instantly, we can see that the overall performance of the CNN is better than the FCN. However, upon closer inspection we can see the the learning rate of 0.01 and 0.001 perform more poorly than their predecessors. This could be because of what I previously discussed about over fitting. The 0.1 learning curve maybe be learning quicker because of the new model increasing the accuracy whereas the lower learning rates help avoid this. We can see further proof of this in the next table.

Epochs	Learning Rate	Start Loss	End Loss	Start Accuracy	End Accuracy	Final Accuracy
10	0.1	2.37	2.182	14.58	37.39	37.45
10	0.01	2.4	2.36	10.49	16.2	15.08
10	0.001	2.407	2.405	8.31	10	9.02
50	0.1	2.38	2	14.51	52.75	50.85
50	0.01	2.404	2.194	10.14	36.83	34.97
50	0.001	2.407	2.389	11.34	15.14	14.47
100	0.1	2.372	1.994	14.65	54.51	54.24
100	0.01	2.405	2.151	10.28	40.56	40.28
100	0.001	2.407	2.386	9.15	12.25	11.24

Table 4: Valid Data / Final Accuracy score, CNN

As mentioned in the discussion of table 3, we can see huge drop in overall accuracy for both the valid and final accuracy score. This shows that the initial training for the higher learning rate did create a false illusion of accuracy but was really just over fitting. However, it should be noted that the overall performance of the learning rates 0.1 and 0.001 both did better than their counterparts in FCN with the higher learning rate of 0.1 scoring almost 11% higher in accuracy. Interestingly enough, the lowest learning rate got worse. Not only this, but the starting and end loss at the highest epoch range for the learning rate of 0.001 was also higher. This may mean that because of the extra calculations completed during the model's hidden layers the actual learning rate is slower for the lower learning rate, or having too low of a learning rate can be counter productive and not a universal/ go to thought that should be used when working with machine learning algorithms.

5 Conclusion

In conclusion, it is clear that in most cases the CNN model would be the better to use of the two. The exception potentially being with the lower learning rate. Otherwise, we see that the CNN can achieve a higher accuracy faster, this pro however is balanced with the high potential of over fitting that we have seen.