**Neural Networks and Deep Learning**

Gadfrey Balacy

CSC-180

Fall 2017

Project 3 – Neural Network and Deep Learning

**Introduction 2**

**Part 1 Average and Median 3-4**

**Part 2 Median 5-6**

**Part 2 Average 7-8**

**Part 3 Image Classification 9-13**

**Appendix A (How to Run the Program) 14**

Introduction

A Neural Network is an artificial interconnected group of nodes, similar to a vast network of neurons in the brain. Each node represents an artificial neuron. In which neuron fires when a certain threshold is met. Like biological neurons, artificial neurons change the limit of the neurons to fire and the resistance of the connection between neurons. Also, the way artificial neuron networks learn is classified in three ways supervised learning, unsupervised learning and reinforcement learning. The one we are going to use for our project is supervised learning.

Supervised learning is learning in which the input and output are given to the neural network. In which the error difference is calculated between the targeted output and the actual output. The error is use to change the weights of the neural network. The form of supervise learning compose of three parts. A training set in which example of inputs, with known correct outputs, are used for training. The training pair one input/output example in the training set. The testing set with examples of inputs, with known correct outputs and is used for testing. This method of learning corresponds well with our objectives for all parts of the project.

With part 1 and part 2 of the project consist of the neural network learning the average and median of three inputs. In which we provide a floating point number between 1 and 50 in which a training set consisting of 20 and 50 sets of training pairs. Also, a testing set in which we see how well the trained network generalizes the problem. The supervised learning algorithm that we are using for our project for part 1 and part 2 is either a Perceptron Learning or an extension of that program called Backpropagation depending on the settings of our neural network. With part 3 of the project being considerably more complex of image classification, we have to use new tools to train and create a neural network.

Similar to part 1 and part 2 we are using supervised learning method but with more complex tools. Part 3 of the project uses a methodology called deep learning that refers to convolutional neural networks (CNN). Convolutional neural networks utilize a significant amount of neurons and a large number of layers and use sophisticated or specialized organization of layers. The specific tools we are using for image classification for this part of the project is TensorFlow and TFLearn a higher level API of TensorFlow. The architecture we are using for this project is AlexNet. The programming language we are using is python.

**Part 1 Average Median**

**Settings Average Median**

****

The neural network base on this settings learns the training set for the average of the three inputs but not the median of the three data. You can see by the amount of error between the output and actual value in the median error values. The learning rate for training set was .5400 but a significant portion of the values meeting that criteria are the average outputs. With only 3 of output from the neural network for the average not matching the requirements of 1.0 for the training set. It took 4 seconds to complete the training. I did have a configuration with a more substantial amount of iteration did not increase the learning rate percentage.

Similar to the training set the generalization mostly happen the average neural network output. It contributed the highest for meeting the testing criteria of 1.5. The learning rate for the testing case was .5600. The neural network generalizes will for the average but not for the median.

The neural network learned the three problems well for the average but not for the median. You can tell by the amount of error in the median outputs compared to the actual. You large errors in both the training set and testing set. Also, it did not learn the median well because similar inputs that past in the training set did generalize.

There were other settings in which there was a higher learning percentage in the training set and testing set. But the only reason it had higher learning percentage and testing because some of the values for the output for the median was similar to the average output. That setting was using linear activation for both outputs.

**Part 2 Median**

**Settings Median**



The result from the setting of the neural network surprises me. Because of the configuration of this neural network having the lowest iteration that I tried. Also, with this arrangement of the neural network, I try to increase the iteration but the higher the iteration value it decreased the learning percentage of training data and lowered the generalization of the neural network. With this setting, I had a learning percentage of .5143 and generalization of .40. For my training data and testing data. The precision of the criteria for the training set was .5 for the training set and .75 for the testing set. The training for this program is fast since it has low settings of iteration and has a decent amount of momentum. It took about 3 seconds to complete the application.

The program did not generalize as well as it learns the training set. There is a higher error rate between the output and the actual value of testing data. In the training set, the most significant error amount was 1.6 between the output and actual. While the testing data was 3.3622 between output and actual. The highest error rate for both the training data and testing data occur to similar problems. That problem was with inputs values not being not having a significant amount of difference between each input. From the training data, it seems that my neural network had problems finding the median when the input values did not have a sizeable significant difference in the amount of the different inputs.

The neural network did not learn the problems well in this configuration. Considering there was a significant amount error inside the testing set in which the similar input values were present inside the training set.

**Part 2 Average**

**Settings Average**

****

The neural network learns the training set to a precision of .0001. With a hundred percent learning rate. It learns the training set well because the average is a linearly separable. That’s reason why I choose to use linear as activation function for this neural network. I did try with other activation function, but those other functions did give better results than with linear activation. The training takes less than a second to complete. The neural network linearly separates the outputs quickly.

The neural network generalizes to 1.0 or 100% of the test cases. Neural network learns the three problems well by having a 100% generalization of the test cases.

**Part 3 Image Classification**

**Description**

My image classification problem is two different classification with one classification having a significant difference appearance. That would be a car and tree. The other classification problem with similar to features a car and truck. I wanted to see how well the neural network would handle images that are contrasting and pictures that are similar.

Images:

Car and Tree

Car and Truck

**Develop Training and Testing Sets**

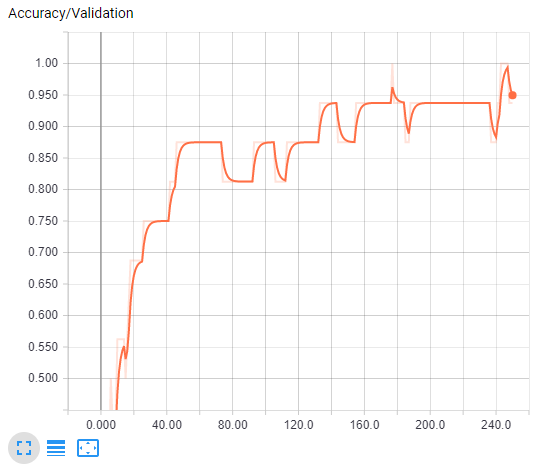
The way I got my training and testing set is by going to google images and typing the pictures that I wanted for my training and testing sets. Then I use the website resizemypicture.com to resize the pictures to a height of 100 and width of 100. Later on, I wanted to have more data for the training set and testing, so I decided to resize the truck and car originals to 300 \* 300-pixel image.

**Architecture**

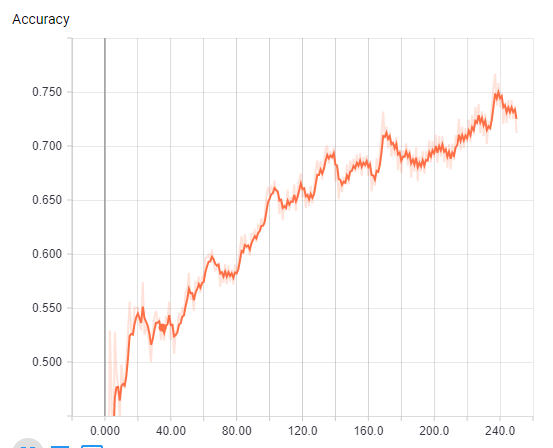
The architecture that I used was the AlexNet style architecture for my training data. The reason why I decided to use AlexNet style is because what our professor said about how it was an improvement from LeNet and it was one of the best architecture to use for relu activation function and also if you had the necessary graphics card you could use your GPU that would increase the rate in which it can process the training data. Also, it would decrease the learning time for the Neural Network. Another reason that I wanted to use this architecture is it allows us to use relu. The reason why I wanted to use relu as the activation function is that of its efficient computation, sparse activation and the reduction of the likelihood of the gradient to vanish. Which results in the neural network learning faster.

**Summary**

Car and Tree

The training for the image classification problem between the car and tree took 33 minutes and 38 seconds. The neural network for this classification generalizes very well as you see from this graph. 

As you can see the accuracy of the validation of testing case goes higher as the amount of epoch is occurring. The neural network at one point at 244 period hits a hundred percent validation accuracy. At the end of the program running a 95% percent accuracy. Once the application ended I use the predicted class to see how well this validation graph corresponds to the testing case. The result was the neural network having guess 15 of the 16 images in the test case correctly. That’s a 93.75 percent accuracy.The neural network learn the problem very well as you can tell by the how it well it generalize and also you tell by the accuracy overtime in the training set.

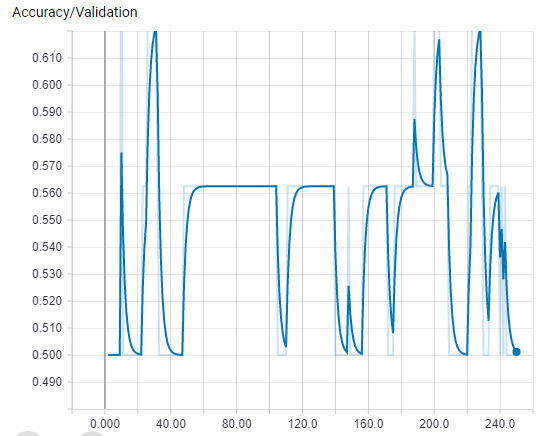


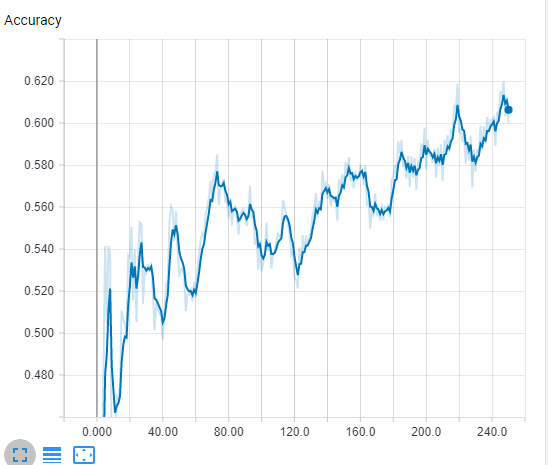
As you can tell by this graph the accuracy of neural network is getting higher as the amount of epoch are occurring.

Car and Truck

The training for the image classification of the Car and Truck run 1 with 100 by 100 took 32 minutes and 29 seconds.

The car and truck neural network did not generalize the problem well as you can tell with the validation accuracy over time.

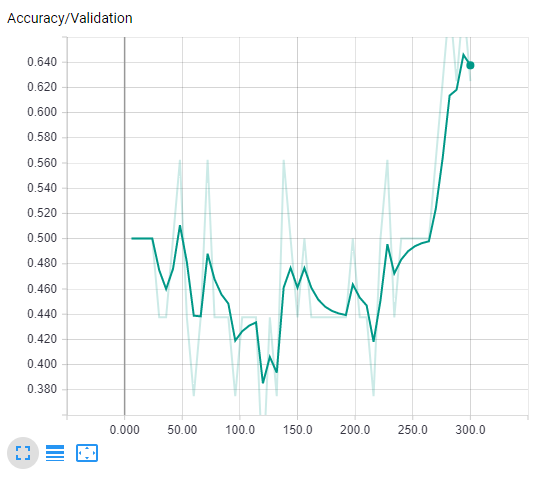


You can tell it did not generalize the problem well because you have this peaks and valleys of accuracy over time. Instead of a gradual increase accuracy similar to the Car and Tree problem. This lack generalization of this problem might not be because of the neural network it might be my test cases has images that are similar to a car or vice versa cars that look like trucks. The reason I say that because the neural network in training set accuracy in increasing over time. As you can see in this graph. 

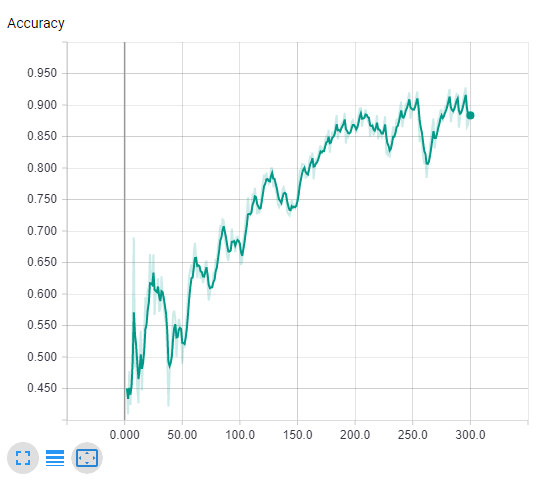
The neural network I believe is learning the problem very well as you can tell by the graph above. The increase of accuracy of the training set overtime tells that neural network is learning the problem very well.

With the problems that occur in the previous run of the program, I decided to trim some images in the test folder and training folder in both the car and truck. To remove pictures that the neural network might mistake as a car when it’s a truck and vice versa. Also, to increase accuracy and to give more image data for the neural network to process I decided to change the dimensions of the images to 300 by 300 in the training set and testing set. Also, I change some of the code to send the images in batches instead of the whole sale.

Car and Truck Run 2

This run took 1 hour and 19 minutes. The neural network in this instant improve dramatically in the accuracy of the validation compare to the previous run of the program. As you can see from this graph. 

You can see a trend in which the accuracy of the validation starts to pick up around the 250 epoch. With the validation accuracy improving dramatically. So when I saw this graph having such improvements of precision in the test case I believe that it generalizes the problem well, but I was mistaken when I checked the test case and resulted in only having 10 out of the 16 correctly. But that wasn’t the issue it was the high percentage of neural network thinking a truck was a car and also some of the picks of the neural network where just guesses. Instead of knowing if it was a car or truck.

Even with the validation accuracy with the test cases not being right. I do believe the neural network did learn the problem well by the high accuracy of the training set. As you can see in this graph. 

The accuracy in the training set is outstanding with points of the accuracy going above 90 percent.

Appendix A

Start Command Line

Go Destination of the Training, Predict and DeepNeuralNet Programs

Run Training Program:

python train.py

After training program is completed

Run Predict Program

python predict.py