Matthew Oakley

Deep Learning w/ Tensorflow

Dr. Rivas-Perea

12/10/17

Abstract

This paper discusses the work I have done regarding facial recognition and how I implemented it. It will also go into detail about the methodology I used in finding the best data and how I further researched it.

Introduction

My motivation for this work came from seeing when shows talk about facial recognition in cctv cameras and they utilize it in regards to tracking people down. I wanted to see for myself how easy it is to do this on static images and get an idea how close these shows and movies are to reality. This paper will discuss the background and related work that deals with similar problems. It also goes over the approach I have taken to tackle this problem. The experiments section will discuss the experiments I have done and how well they performed and the bugs I encountered. I will also go over the results and discuss what significance they have. The paper will also go over the overall findings and any references and I used.

Background

Before starting the project I asked competent people in the field if they knew a good basis. One recommendation to recognize faces is Viola and Jonas face detection algorithm to detect faces in images. I looked into universally known facial recognition algorithms and

determined how to improve them to build a more successful program. In the end I decided to just use Viola and Jonas because I did not have the time or skills to do my own face detection.

Methodology

The approach I took was one that would make it easier to acquire images for testing and training purposes. I decided the face I was gonna try and detect was President Donald Trump's. The reason for this is there are many images of him that are free for reuse and can easily be found via Google Images. The focus of my images is around politicians because many images of them are free for reuse as well. I have collected 100 Training and 100 Testing images which all do not contain images of Trump. The name of the images determined if they contained Trump or not. The format is "x-num" where x would be 1 or 0 depending if Trump is in the image or not. The num would be be leading zeros before a number that would be a counter, both images have the counter start at 0. The next method would be to run the Viola and Jonas algorithm over this data to box out where the faces would be. The faces that are returned will then be tested to see if they are Donald Trump's face or not. The faces with Trump are labeled with a 1 while not Trump is labeled with a 0. The dataset for faces is about ~13% of Trump's face while the rest is other faces or ties and random pictures. The next approach was to use an autoencoder to learn Donald Trump's face compared to other pictures. The autoencoder will have many of its parameters modified like number of layers, learning rate, and its epoch are set to a standard of 50 pretraining and 25 fine tuning.

Experiments

The first experiment I did was testing the error rate for only one layer at different sizes and learning rates. This is seen in (1) where the one layer has its learning modified from .0001 -

.1. In the chart where the testing error rate was .1364 this was where the autoencoder experienced explosive gradient. This happened in multiple experiments where the learning rate was set to .1. The next set of tests were with two layers and varying learning rates. The results from these tests can be seen in (2), (3), (4), (5). The best out of all of these results was .0036 error rate and the parameters for this were used in the experiments for (6) and (7). In these final sets of experiments the only parameters that was modified was the epoch for pre training. In the last set of tests the best error rate was .0018 for both 10 and 20 epochs. The different parameters for these tests were simply picked randomly.

Analysis

A common part of the results from the tests were that a learning rate of .1 caused gradient explosion. The tests with one layer showed which learning rates would produce bad results. The two layer tests helped show the impact that a learning rate has on the error rate. With .01 many of the experiments had gradient explosion. When testing the two layers with a learning rate of .1 it became unstable during some of the tests and the program could not finished during layer 2. The experiments showed that using a high learning rate of .1 is bad for the autoencoder. The tests in (6) showed that epochs did not have a significant factor on error rate and in some cases caused it to increase. The reason for this is that the network could be overtrained or how the autoencoder separates data for the final test resulted in a lower error rate.

Conclusion

Overall the tests showed that having a higher learning rate is bad. Another result from the tests is that having two layers lowers the error rate. One way to improve the tests is to include more images of President Trump's face so the autoencoder can have more testing data. Another

way to improve the practicality of this is the test in data the autoencoder has never seen before to make sure it is generalized.

Figures and Results

	0.0001	0.001	0.01	0.1	Avg
512	0.0145	0.0073	0.0273	0.1364	0.0464
256	0.0145	0.0236	0.0327	0.1364	0.0518
128	0.1164	0.0309	0.0745	0.1364	0.0896
64	0.1036	0.1273	0.1291	0.1364	0.1241
32	0.1364	0.1364	0.1364	0.1364	0.1364
Avg	0.0771	0.0651	0.0800	0.1364	

Figure 1. This is the autoencoder with one layer at 512, 256, 128, 64, 32 and a learning rate of .0001, .001, .01, .1.

0.0001	512	256	128	64	Avg
256	0.0255				0.0255
128	0.1364	0.1327			0.1346
64	0.0055	0.0091	0.0327		0.0158
32	0.1291	0.0164	0.0491	0.1364	0.0828
Avg	0.0741	0.0527	0.0409	0.1364	

Figure 2. This is the autoencoder with two layers and the learning rate with .0001. The output number is the testing error.

0.001	512	256	128	64	Avg
256	0.0036				0.0036
128	0.0073	0.1291			0.0682
64	0.0036	0.1364	0.0527		0.0642
32	0.0073	0.0091	0.1345	0.1345	0.0714
Avg	0.0055	0.0915	0.0936	0.1345	

Figure 3. This is the autoencoder with two layers and the learning rate with .001. The output number is the testing error.

0.01	512	256	128	64	Avg
256	0.0255				0.0255
128	0.1364	0.1364			0.1364
64	0.0109	0.1273	0.1327		0.0903
32	0.1309	0.1345	0.1364	0.1345	0.1341
Avg	0.0759	0.1327	0.1346	0.1345	

Figure 4. This is the autoencoder with two layers and the learning rate with .01. The output number is the testing.

0.1	512	256	128	64	Avg
256	0.1345				0.1345
128	0.1364	0.1364			0.1364
64	**error**	0.1364	0.1364		0.1364
32	**error**	0.1364	0.1345	0.1364	0.1358
Avg	0.1355	0.1364	0.1355	0.1364	

Figure 5. This is the autoencoder with two layers and the learning rate with .1. The output number is the testing.

0.001	50	40	30	20	10
512 / 64	0.0036	0.0109	0.0036	0.0018	0.0018

Figure 6. This is the autoencoder with two layers of 512/64 and a learning rate of .001. These tests were done based on the previous best results. The 50, 40, 30, 20 and 10 are the epochs that were changed.

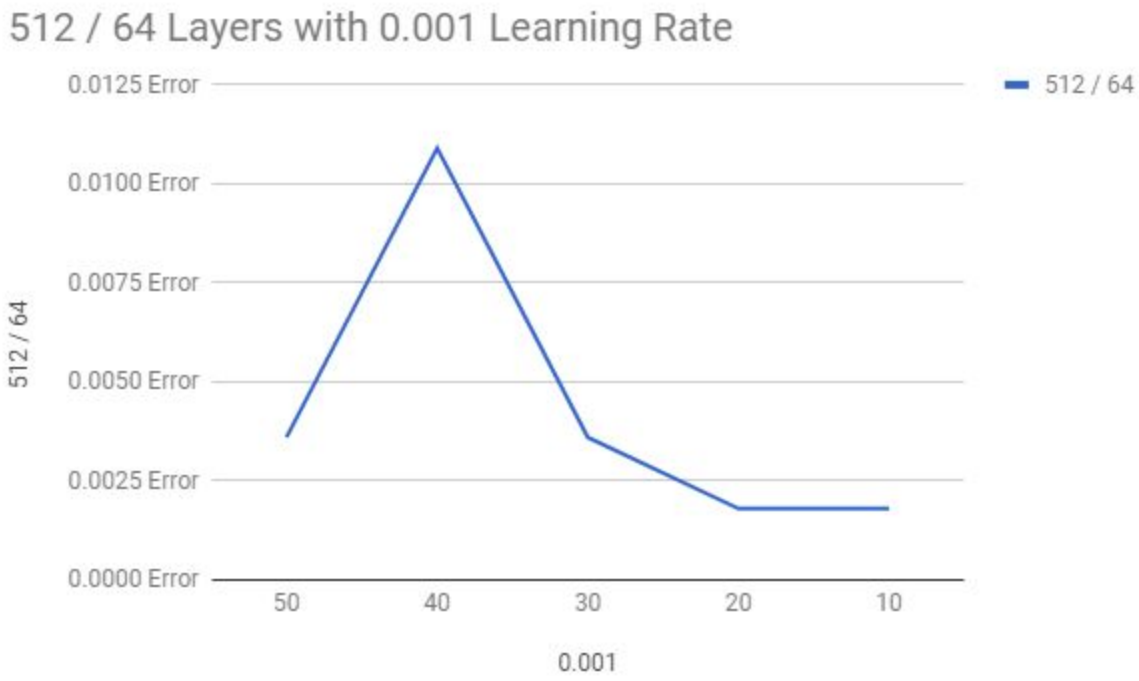

Figure 7. This is the accompanying graph for figure 6.