Assignment 2 - Neural Network Eye Detector

Bryon Kucharski

February 2019

I ran different activation functions by changing the model paramters in trainNN.py. I ran the minibatch version by commenting out $model = trainNN(X, Y, threshold)$ in run.py and replacing it with $model = trainNN\_batched(X, Y, threshold)$. I ran the Ensemble Averaging by running the file *ensemble_averaging.py*.

1 Model with Sigmoid Activation Functions and Gradient Descent with Momentum

classification error in the test dataset: 7.10 %
test threshold used: 0.98
learning rate: 0.5
momentum: 0.9
weight decay: 0.001

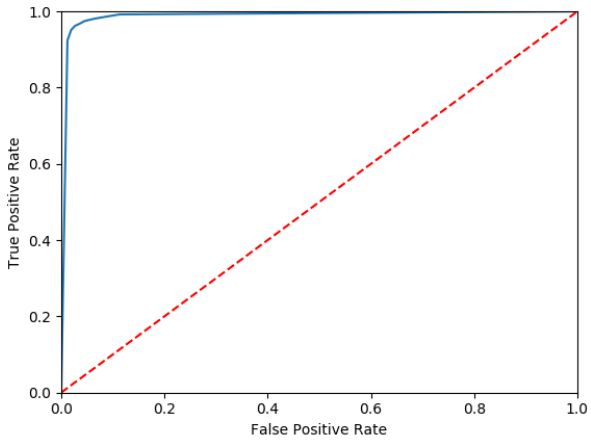

Figure 1: Sigmoid ROC curve

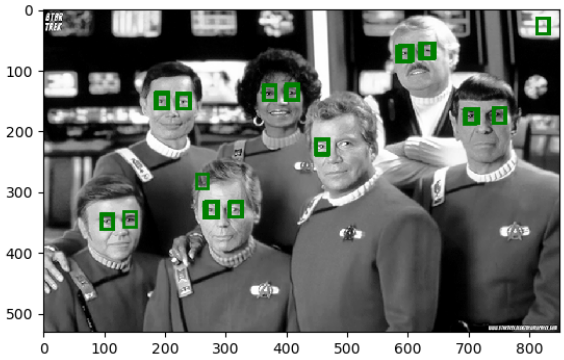

Figure 2: Sigmoid Startrek 1

Figure 3: Sigmoid Startrek 2

2 Why the threshold might need to be selected different from 0.5

It is intuitive to think that the decision threshold should be 0.5 when deciding between two classes. This is not always the case in reality, and it depends on the problem and how many of each label you have in the dataset. Looking at the ROC curve explains why this may be the case. Ideally, you want the area under the curve to be as close to 1 as you can get because this means youre model is

correctly predicting labels. From my ROC graphs, the area under the curve is maximized when the true positive rate is close to 1 and the false positive rate is close to 0. If i was to choose a threshold of 0.5, that would I would have more false positives, which in result lowers the area under the curve and prediction accuracy. Therefore, I want a threshold closer to 1.

3 Bonus: Model with ReLu Activation Functions and Gradient Descent with Momentum

classification error in the test dataset: 3.27 %
test threshold: 0.999
learning rate: 0.05
momentum: 0.9
weight decay: 0.001

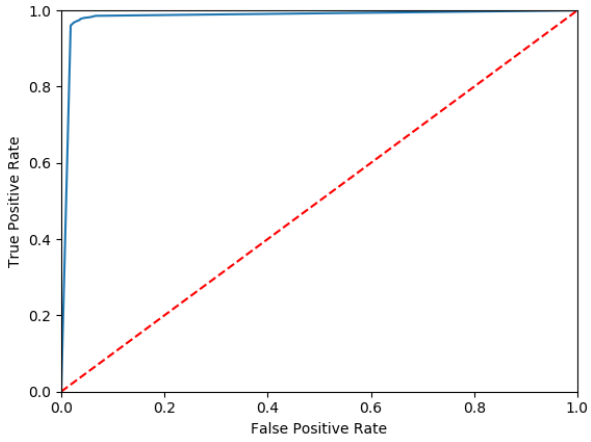

Figure 4: ReLu ROC curve

4 Bonus: Model with ReLu Activation Functions and Gradient Descent with Momentum and mini-batch training

classification error in the test dataset: 3.39%
test threshold: 0.9995
learning rate: 0.0009
momentum: 0.99

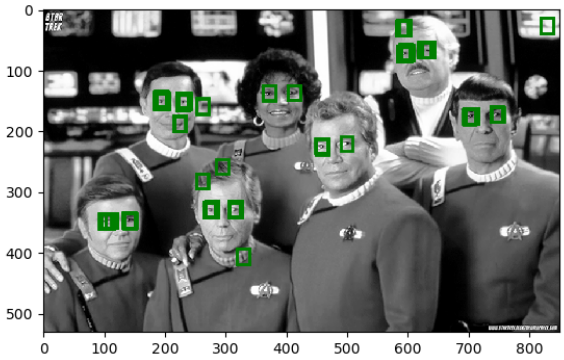

Figure 5: ReLu Startrek 1

Figure 6: ReLu Startrek 2

weight decay: 0.001

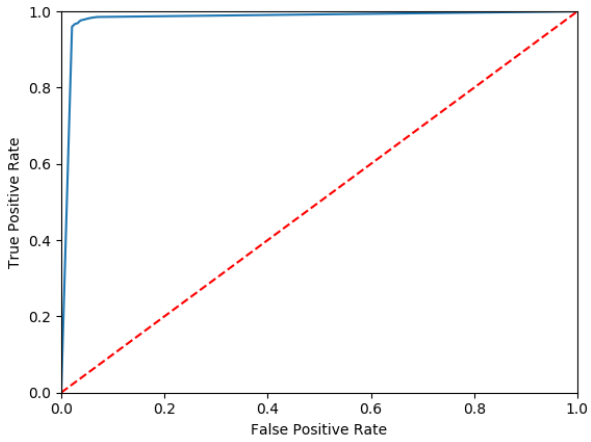

Figure 7: Mini batched ReLu ROC curve

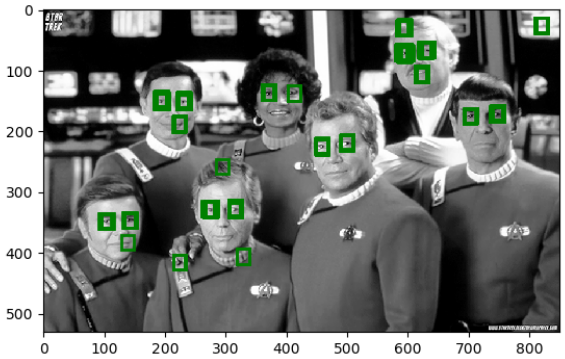

Figure 8: Mini batched ReLu Startrek 1

5 Bonus: Ensemble Averaging with ReLu Activation Functions and Gradient Descent with Momentum and mini-batch training

I trained three different networks using reLu activation functions and gradient descent with momentum. I used the same parameters from section 3.
classification error in the test dataset: 3.69 %

Figure 9: Mini batched ReLu Startrek 2