Eye Blink Model

All the raw program output can be found in the *Resut-EyeBlink.txt* file.

From the previous assignment, we already know that the *Neural Networks* will have the best performance when it has 1 layer with 20 nodes. For this assignment, all the *Neural Networks Model* will use the same structure. And the cross validation will use 5-fold.

1 Feature Engineering

When doing the feature engineering, we used pixel intensity (normalize to range 0.0 – 1.0) to train the *Neural Networks*. The different between each cross validation is, we tried different scale of the image and check whether add or not add the momentum to the Back-Propagation algorithm will affect the result.

Here is the accuracy after running cross validation on the **Training Set**:

	Not Scaled	**Scaled Down by 2**	**Scaled Down by 4**
No Momentum	89.1580%	87.6720%	79.9395%
Add Momentum	89.7633%	87.4243%	80.1321%

From the accuracy shows in the table we can see that

- More features (no scale down of the image) will performance better
- Add momentum will performance better

2 Parameter Sweeping

2.1 Find the Best Parameters

After we checked our Feature Engineering result, we will use the full image's pixel intensity and add momentum to the Back-Propagation algorithm in our training.

To find the best *Neural Networks Model*, we can use, we need another round of parameter sweeping for it. Here is the sweeping range:

- Iteration to training the *Neural Networks Model*
  - Range 150 to 250, step 50
- Stochastic Gradient Descent Batch Size
  - [10, 30, 50, 75, 100, 125, 150, 200]
- Learning Rate
  - [0.01, 0.05, 0.1, 0.5, 1, 10]
- Momentum Beta
  - [0.25, 0.33, 0.5]

We can get 91.4695% cross validation accuracy with error bound [90.5612%, 92.3777%] on **Training Set**, when using the following parameters:

- Training Iteration 150
- Stochastic Gradient Descent Batch Size 10
- Learning Rate 1

- Momentum Beta 0.25

2.2 Process Analysis

From the program output of the parameter sweeping process, we can find that:

2.2.1 Training Iteration vs Learning Rate

From the comparation of Training Iteration and Learning Rate (Stochastic Gradient Descent Batch Size as 10, Momentum Beta = 0.25):

	0.01	0.05	0.1	0.5	1	10
150 Iteration	85.4981%	88.5250%	90.6714%	90.2587%	91.4695%	49.1469%
200 Iteration	86.8189%	89.6258%	90.8090%	90.1486%	91.3319%	56.6043%
250 Iteration	87.3693%	90.7815%	90.5889%	90.0660%	90.2036%	56.4942%

We can find that:

- when the learning rate is low, more iteration of the training will get better result, because lower learning rate will let the *Neural Networks Model* don't have enough time to go to the local or global optimal (underfitting) with smaller size of the iteration.
- With higher learning rate, the overall performance will better than lower learning rate because the *Neural Networks Model* now can find the local or global optimal. But with more iteration, the model might start overfitting the **Training Set**, so more iteration will have poorer performance.
- When the huge learning rate, the *Neural Networks Model* just simply cannot find the local or global optimal.

2.2.2 Stochastic Gradient Descent Batch Size

Since the implementation of the *Neural Networks Model* used the Stochastic Gradient Descent, we can update the weights of the model by certain small batch of the **Training Set** (150 Iteration, Momentum Beta = 0.25).

	0.01	0.05	0.1	0.5	1
*Batch Size **10***	85.4981%	88.5250%	90.6714%	90.2587%	91.4695%
*Batch Size **30***	75.6467%	87.4243%	88.3049%	91.1943%	90.7265%
*Batch Size **50***	74.4084%	85.0853%	85.9934%	89.8459%	90.7540%
*Batch Size **75***	72.2895%	82.5261%	86.6263%	88.0848%	90.0385%
*Batch Size **100***	70.0330%	80.6274%	84.2047%	86.2411%	88.2223%
*Batch Size **125***	66.9235%	78.8112%	84.5074%	86.9290%	87.2042%
*Batch Size **150***	67.1712%	77.4629%	82.5261%	85.6632%	87.5344%
*Batch Size **200***	70.2532%	75.3165%	81.0402%	86.8465%	86.6538%

From the output, we can see:

- With the increase of the batch size, since the weights of the *Neural Networks Mode* will update less time, the model will performance weak than before.

2.2.3 Momentum

Since we already know that momentum can help the *Neural Networks Mode* to avoid local optimal and increase the chance to find the global optimal.

We can also find that (150 Iteration, Stochastic Gradient Descent Batch Size as 10):

	0.01	**0.05**	**0.1**	**0.5**	**1**
*Beta **0.10***	84.6175%	88.8002%	90.6439%	90.2862%	90.6714%
*Beta **0.15***	85.3054%	88.9378%	90.4788%	90.0110%	90.6164%
*Beta **0.20***	85.8833%	88.9378%	90.8090%	90.2587%	90.0385%
*Beta **0.25***	85.4981%	88.5250%	90.6714%	90.2587%	91.4695%
*Beta **0.33***	86.1860%	89.1029%	90.4513%	90.3137%	90.1761%
*Beta **0.50***	86.4337%	89.5707%	91.0567%	90.4788%	90.2587%

- The momentum can help the *Neural Networks Model* to find the global optimal more quickly.
  - Large momentum can help power through local optimal.
  - Large momentum can help more to find the optimal.
- But some case the large momentum can let the model move further away from the optimal hence performance poorly.

3 Categorize Mistakes

After we have the parameters to build the current best *Neural Networks Model*, we can output the most **False Negative** (Closed eye, but predicted as opened eye) and most **False Positive** (Open eye, but predicted as closed eye) samples from the **Training Set**.

3.1 Most False Negative Sample

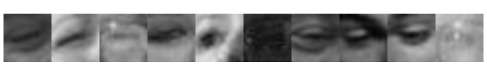
Here are the 10 most False Negative samples (ascending order by the predict raw value):

From the samples, we can know that:

- 3 of the 10 eye images are blur
- 2 of the 10 eye images have hair
- 3 of the 10 eye images have glasses or makeup
- 1 of the 10 eye images include noise and in different angle

3.2 Most False Positive Sample

Here are the 10 most False Positive samples (descending order by the predict raw value):



From the samples, we can know that:

- 6 of the 10 eye images have small eye
- 3 of the 10 eye images are blur
- 1 of the 10 eye images include noise and in different angle

3.3 Improvement

To improve out *Neural Network Model*, maybe we can add more features to handle different situation:

- Y-Gradient for reducing the effect of hair

- X-Gradient for reducing the effect of makeup and check for small eyes

For the gradient, we can use the same featurize method:

- Divide the image into a 3 x 3 grid and for each grid location include a feature for the min, max, average X-Gradient and for Y-Gradient among the locations in the grid
- Normalize the gradient value to 0.0 – 1.0 range for maintain same scale as the pixel intensity features, also can prevent the overflow when doing the calculation of Back Propagation.

3.4 Evaluation the Best Model with new Features

With these new features the best *Neural Network Model* can get 93.6434% cross validation accuracy with error bound [92.8501%, 94.4366%] on **Training Set**. And by using for predicting the **Test Set**, we can get the following evaluation statistic:

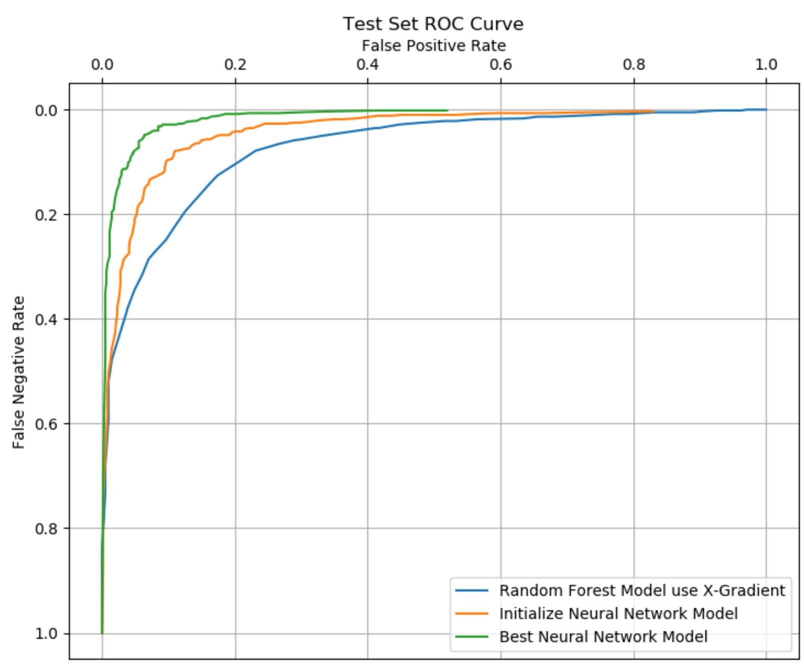
- Accuracy: 93.9769%
- Error Bound: [92.6375%, 95.3163%]
- Precision: 93.2455%
- Recall: 94.6488%
- False Positive Rate: 6.6775%
- False negative Rate: 5.3512%

And the Confusion Matrix as:

	PREDICT TRUE	PREDICT FALSE
ACTUALLY TRUE	566	32
ACTUALLY FALSE	41	573

4 Compare Models

4.1 ROC Curves



By running the 3 different models on the **Test Set**, and sweeping the threshold, we can produce the ROC curves as showing above. And from the graph we can find that:

- When the 3 models have same FPR, the FNR from best *Neural Networks Model* will be lesser than both the best *Random Forests Model* and the *Neural Networks Model* without any parameter tuning and feature engineering.
- When the 2 models have same FNR, the FPR from best *Neural Networks Model* will be lesser than both the best *Random Forests Model* and the *Neural Networks Model* without any parameter tuning and feature engineering.
- Overall, best *Neural Networks Model* will guaranty performance better than other 2 models.

4.2 Compare Other 2 Models

From previous assignments, we can know that:

- The best *Random Forests Model* is using X-Gradient, and it has:
  - 85.9736% accuracy on **Test Set** with error bound [84.0185%, 87.9287%].
- The *Neural Networks Model* without any parameter tuning and feature engineering can have:
  - 87.6720% cross validation accuracy on **Training Set** with error bound [86.6031%, 88.7409%].
  - 90.5941% accuracy on **Test Set** with error bound [88.9506% 92.2375%].

Since our best *Neural Networks Model* have the lower bound 92.6375% on **Test Set** and lower bound 92.8501% on **Training Set** cross validation, which are both higher than the other 2 models' upper bound. We can now say the best *Neural Networks Model* is guaranty better than those 2 models.

4.3 More Improvement?

To continue improve our *Neural Networks Model* we can take the following actions:

- More iteration of Feature Engineering
- More Training Data
- Add shuffle when do the Stochastic Gradient Descent Mini Batch update
- Use some more fancier Model such as Deep Learning?