This is to provide a summary of performances among different feature transformation methods.

Method	Brief description	Error Rate	AUROC	Rank (in error rate)
Original	Just fit the model. No change made. Serve as benchmark.	0.0425	0.993241	3
Count overall numbers of white or black pixels	For each row of the image, I count the number of black and white pixels. Then I add 2 columns to the current data sets. That means the data set has 2 more features.	0.0330	0.995933	1
Histograms of parts of the data	For each row of the image, I count the number of pixel value occurences, dividing into 10 bins from 0 to 1. In total, there are 10 more columns a.k.a. features added to the data set.	0.0345	0.995529	2
Flip the image horizontally	For each row of the image, I flip the pixel values by reversing the list values. The result is that the images are flipped horizontally. I add the flipped data to the original data set, which doubles the data set.	0.0555	0.983933	4

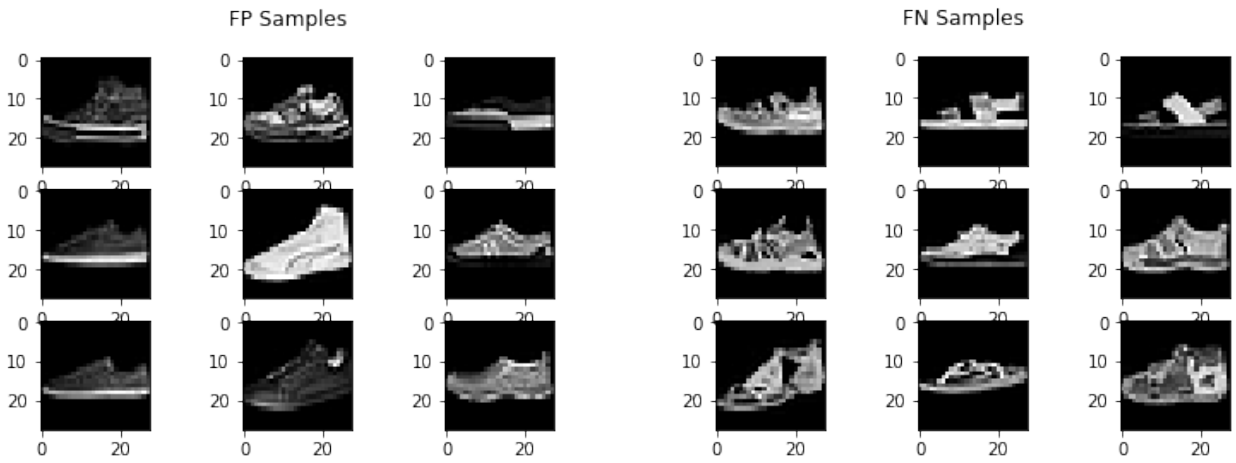
In order to make confusion matrices and to see false positive/ false negative examples, I divided the training set further to a sub-training set and a validation set with the ratio 70 – 30 (E.g.: 30% of training data are validation set).

1. **The original features a.k.a. Benchmark**

This model is purely fitted by the sub-training set and tested on the validation set. Then I created a confusion matrix using threshold 0.5 to see model performance. The True Positive Rate = 95%, which is already very good.

Predicted	0	1
True		
0	1897	74
1	97	1928

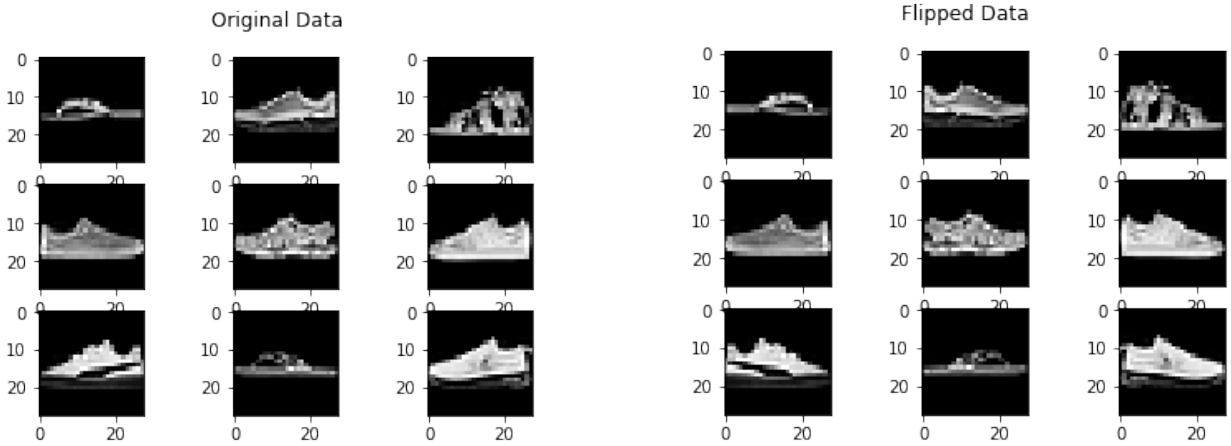
Looking at FP examples, which are predicted sandals but actually sneakers, we can see it might be the shapes of the sneakers that look like sandals, and vice versa for the FN samples.



2. Flip the image horizontally

I wrote a function flipping an image horizontally. For each line (variable), I reshaped into a 28 x 28 matrix. I reversed the values each line of the 28 x 28 image, then returned to the original array shape. I combined the flipped data set with the original data set. Doing this way helps me double the training data size without having to collect more features.

An example of flipped data is as below:

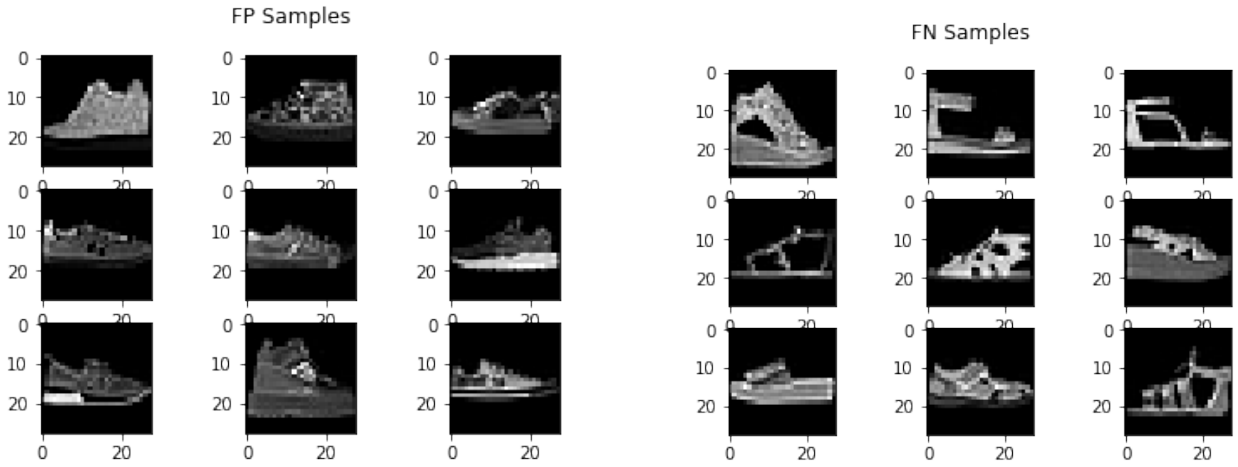


The expectation is that the merged data set will boost accuracy score, as we provide more training data points. However, the result is opposite. The error rate is pretty high when uploading to the leaderboard, which is 0.055499999999999994. AUROC is 0.9839330000000001.

When fitting using the sub-training set and test on the validation set (threshold = 0.5), the TPR actually reduced to 93.55%. This suggests that I may need to change the classifier or combine the flipped data set with data sets using other feature transformation methods to see if I can get better results.

Predicted	0	1
True		
0	3769	225
1	258	3740

It seems like the FP and FN samples are not different from the original version FP and FN samples.



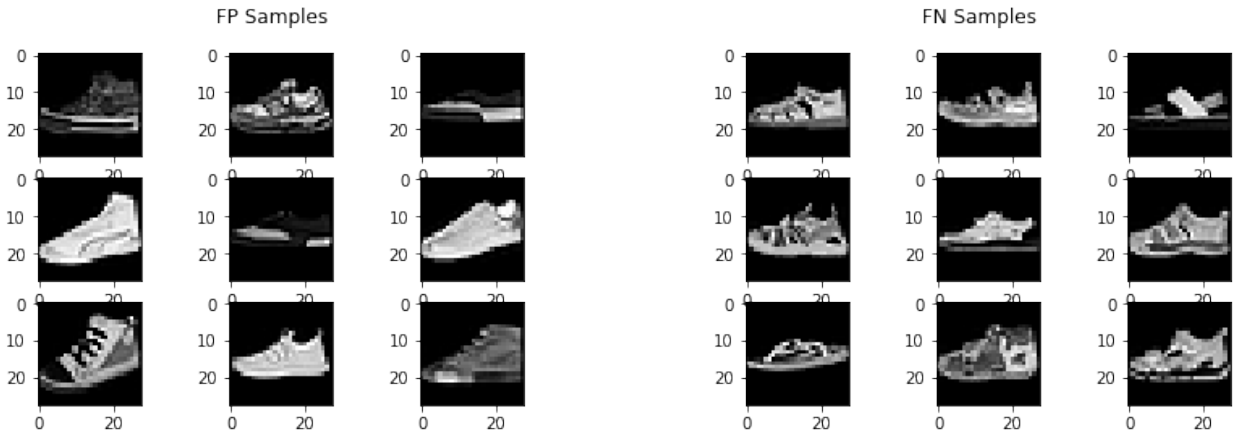
3. Count overall numbers of white or black pixels

I write a function adding 2 columns for each row of the image. The 2 columns are count of white pixels and count of black pixels. So basically 2 more features are added overall. This is a way of adding more variations in the independent variables without having to actually collect more features.

The expectation is that doing is will increase the accuracy and reduce the loss. When uploading to the leaderboard, the figures look pretty good. The error rate is 0.03300000000000003 and the AUROC is 0.995933.

When running on the validation set, the accuracy rate is good as well with True Positive Rate = 96%.

Predicted	0	1
True		
0	1922	49
1	81	1944



Looking at FP and FN examples, we can see that the classifier doesn't make simple mistakes. For example, many of the FP samples look quite slim, which are similar to a sandal look. Similarly, many of the FN samples look bulky, reflecting the general sneaker forms.

4. Histograms of parts of the data

The third feature transformation method that I use is to add histogram of parts of the data. Instead of counting the number of 0s and 1s, I count the values using intervals from 0 to 1. Then I add the bins (a.k.a. columns) into the data set. In total, there are 10 bins. The bins are:

- $0 < x < 0.1$
- $0.1 <= x < 0.2$
- $0.2 <= x < 0.3$
- $0.3 <= x < 0.4$
- $0.4 <= x < 0.5$
- $0.5 <= x < 0.6$
- $0.6 <= x < 0.7$
- $0.7 <= x < 0.8$
- $0.8 <= x < 0.9$
- $0.9 <= x$

Doing this helps me create 10 more features without having to collect more data. However, the downside of this method is much longer computation time, as more computations need to be done. The expectation is that this method of transformation would bring higher accuracy and less error than the previous method (only count 0 & 1).

Uploading the result to the leaderboard, the error rate is 0.034499999999999975, and AUROC is 0.9955290000000001. It is interesting to see that doing this has higher error rate than the last method.

Predicted	0	1
True		
0	1917	54
1	94	1931

On the validation set, it's also interesting to see that the True Positive Rate becomes only 95%. The conclusion for FP and FN samples is similar to the previous method.