CS440/ECE448 Spring2018
Assignment3: Pattern Recognition

Team Members: Haowen Jiang, Xiang Xiang, Haojia Yu
NetID: Haowenj2, xiangx2, hyu59

Part 1: Naive Bayes Classifiers on Digit classification

The purpose for this part is to recognize and classify digit numbers from the input based on Naïve Bayes Classifiers. For the extra part, we implement a function to recognition the input is human face or not.

Part 1.1 Single pixels as features (for everyone; 14 points)

For the digit classification, we implement the class into five sections. Firstly, we should deal with the input from file into python list or dictionary for future usage. This section should distinguish the digit input and ground truth label and save them into different containers. Secondly, training the input data and get the likelihood and priors through the function given in the assignment requirement. By the way, we test different value of k to get a better accuracy for the final result. In addition, we do the maximum a posterior according to the Naïve Bayes Model. Since the input data is composed by 0 and 1, the posterior probability should be calculated under current situation. Moreover, we get the evaluation part to calculate confusion matrix and so on then get odds ratio and plot the image.

For the training part, as I mentioned above, we use different value of k to get a better accuracy.

```
k value is  0.5 overall_accuracy is  0.9009009009009009
k value is  1.0 overall_accuracy is  0.8941441441441441
k value is  1.5 overall_accuracy is  0.8873873873873874
k value is  2.0 overall_accuracy is  0.8873873873873874
k value is  2.5 overall_accuracy is  0.8873873873873874
k value is  3.0 overall_accuracy is  0.8851351351351351
k value is  3.5 overall_accuracy is  0.8851351351351351
k value is  4.0 overall_accuracy is  0.8873873873873874
k value is  4.5 overall_accuracy is  0.8873873873873874
k value is  5.0 overall_accuracy is  0.8873873873873874
k value is  5.5 overall_accuracy is  0.8873873873873874
k value is  6.0 overall_accuracy is  0.8873873873873874
k value is  6.5 overall_accuracy is  0.8873873873873874
k value is  7.0 overall_accuracy is  0.8851351351351351
k value is  7.5 overall_accuracy is  0.8828828828828829
k value is  8.0 overall_accuracy is  0.8806306306306306
k value is  8.5 overall_accuracy is  0.8806306306306306
k value is  9.0 overall_accuracy is  0.8806306306306306
k value is  9.5 overall_accuracy is  0.8806306306306306
k value is  10.0 overall_accuracy is  0.8806306306306306
[Finished in 52.2s]
```

From the value I test above, it is easy to find the accuracy decreases when the value of k increases. Thus, I test the value from 0.1 to 1 and find when the k value is 0.1, we get the highest accuracy.

```
k value is  0.1 overall_accuracy is  0.9031531531531531
k value is  0.2 overall_accuracy is  0.9031531531531531
k value is  0.30000000000000004 overall_accuracy is  0.9031531531531531
k value is  0.4 overall_accuracy is  0.9009009009009009
k value is  0.5 overall_accuracy is  0.9009009009009009
k value is  0.6 overall_accuracy is  0.8986486486486487
k value is  0.7 overall_accuracy is  0.8963963963963963
k value is  0.7999999999999999 overall_accuracy is  0.8963963963963963
k value is  0.8999999999999999 overall_accuracy is  0.8963963963963963
k value is  0.9999999999999999 overall_accuracy is  0.8941441441441441
k value is  1.0999999999999999 overall_accuracy is  0.8941441441441441
```

The classification accuracy for each digit is:

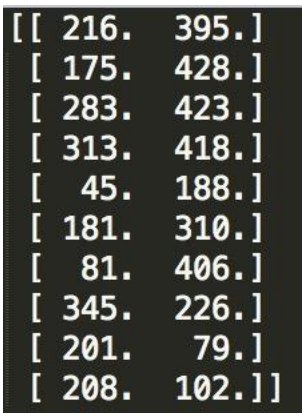
```
0.972222222222
0.955555555556
0.80487804878
0.878787878788
0.898305084746
0.741379310345
0.953488372093
0.957446808511
0.975
0.952380952381
```

The value of confusion matrix shows below (Attention: all the value smaller than 1)

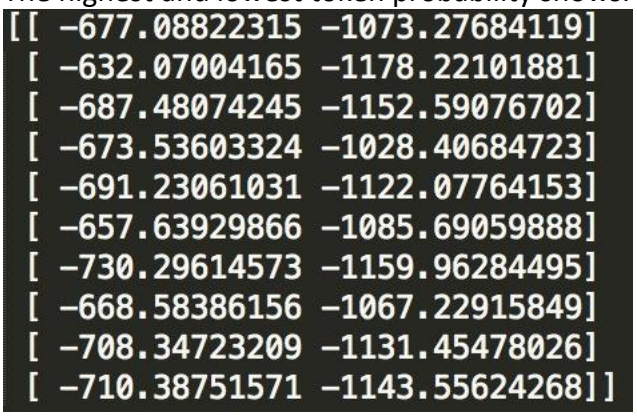
```
9.722e-01 2.778e-02 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00 9.556e-01 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 2.222e-02 0.000e+00 2.222e-02
0.000e+00 4.878e-02 8.049e-01 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 1.220e-01 2.439e-02
0.000e+00 0.000e+00 0.000e+00 8.788e-01 0.000e+00 0.000e+00 0.000e+00 3.030e-02 3.030e-02 6.061e-02
0.000e+00 0.000e+00 0.000e+00 0.000e+00 8.983e-01 0.000e+00 0.000e+00 5.085e-02 5.085e-02 0.000e+00
0.000e+00 5.172e-02 0.000e+00 0.000e+00 0.000e+00 7.414e-01 0.000e+00 0.000e+00 0.000e+00 2.069e-01
0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 9.535e-01 0.000e+00 4.651e-02 0.000e+00
0.000e+00 4.255e-02 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 9.574e-01 0.000e+00 0.000e+00
0.000e+00 2.500e-02 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 9.750e-01 0.000e+00
0.000e+00 0.000e+00 0.000e+00 2.381e-02 0.000e+00 0.000e+00 0.000e+00 2.381e-02 0.000e+00 9.524e-01
```

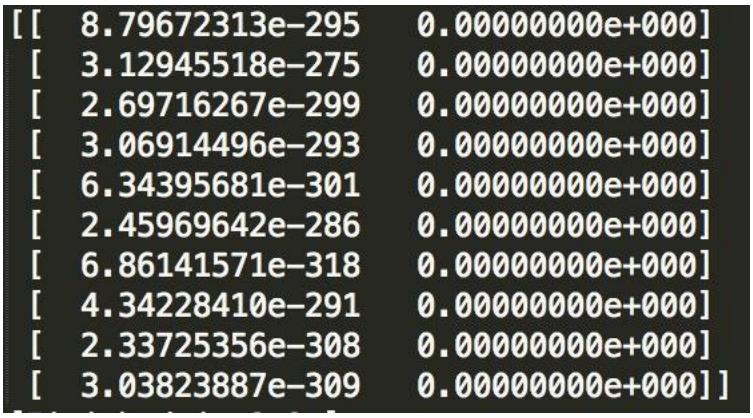
The row represents the ground truth label from 0 to 9 and column represents that the evaluate label. We can find that the right evaluation percentage is always higher than the rest.

The highest test token and lowest test token shows below. The row represents that ground truth label (digit number) from 0 to 9. The first column represents the highest token and second is the lowest token.



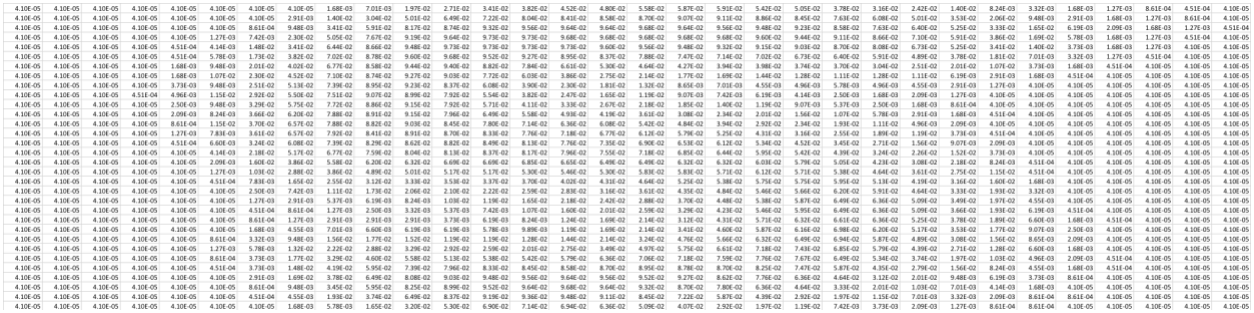
The highest and lowest token probability shows. The first picture is the log value.

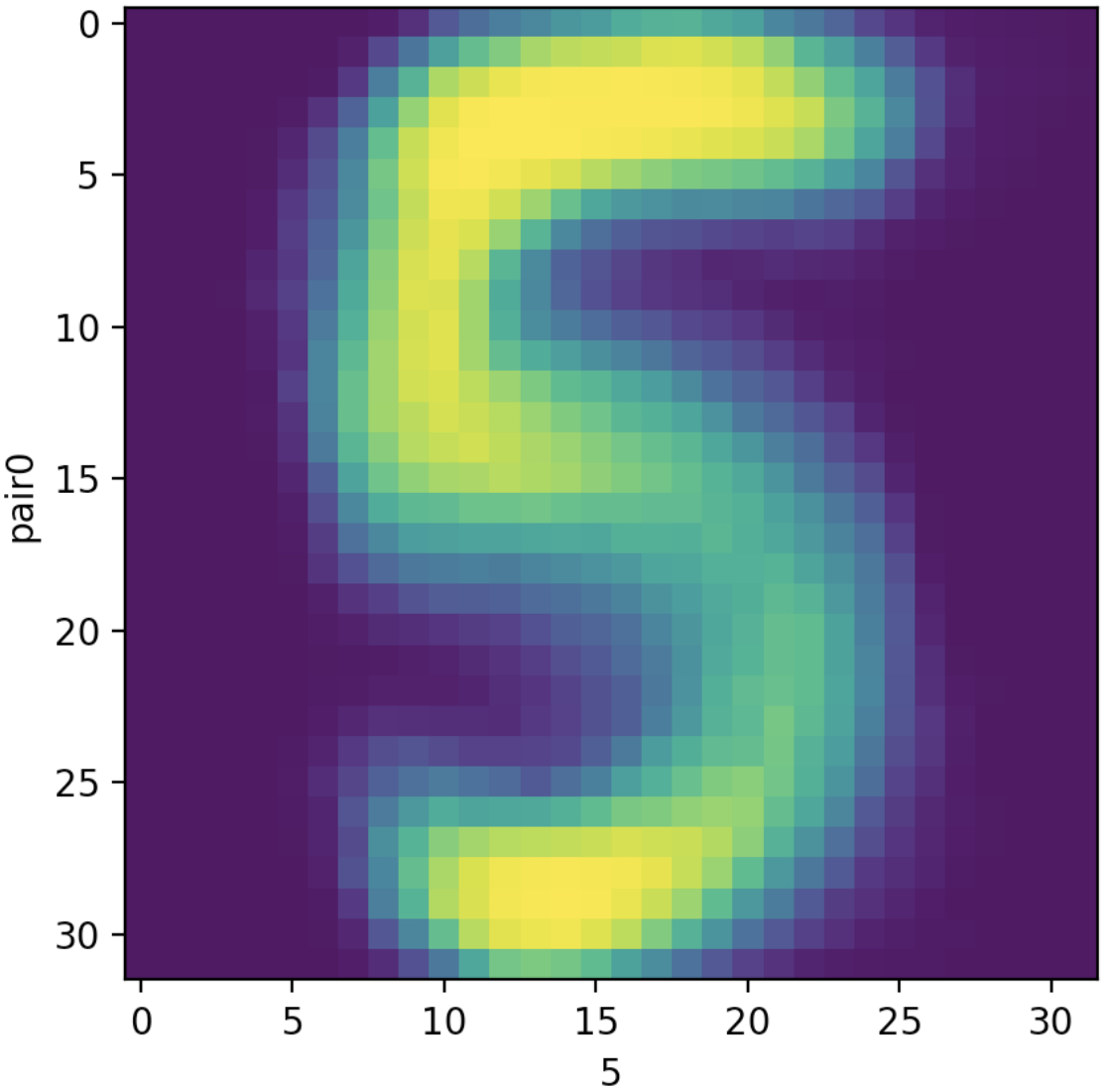


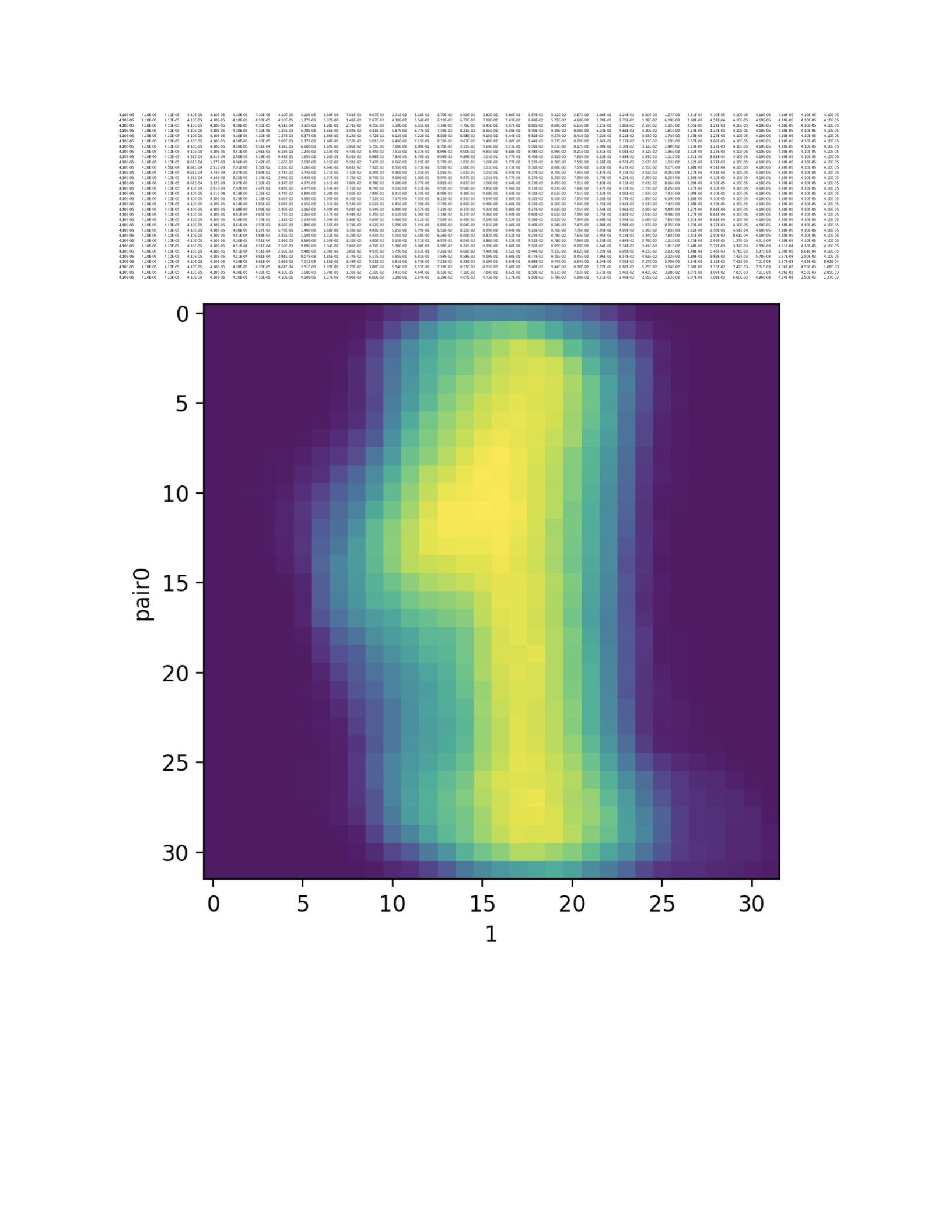


The four pairs of digits types have the highest confusion are (5, 1), (3, 9), (5, 9), (2, 8).

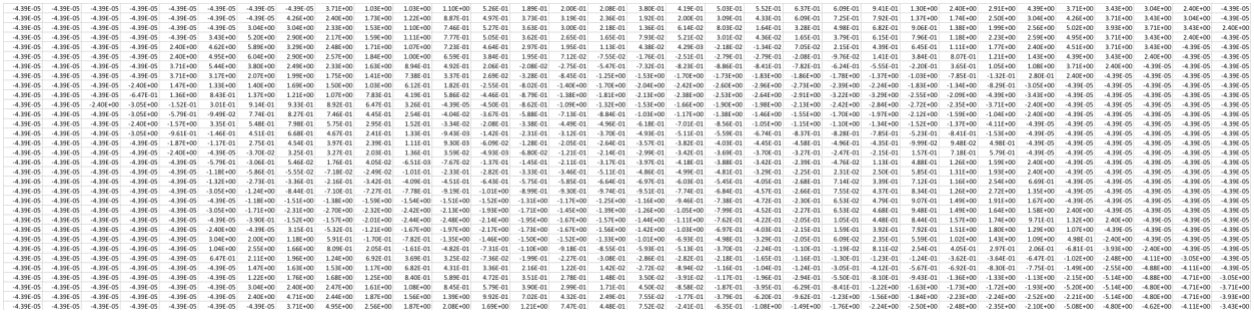
Label 5 and label 1

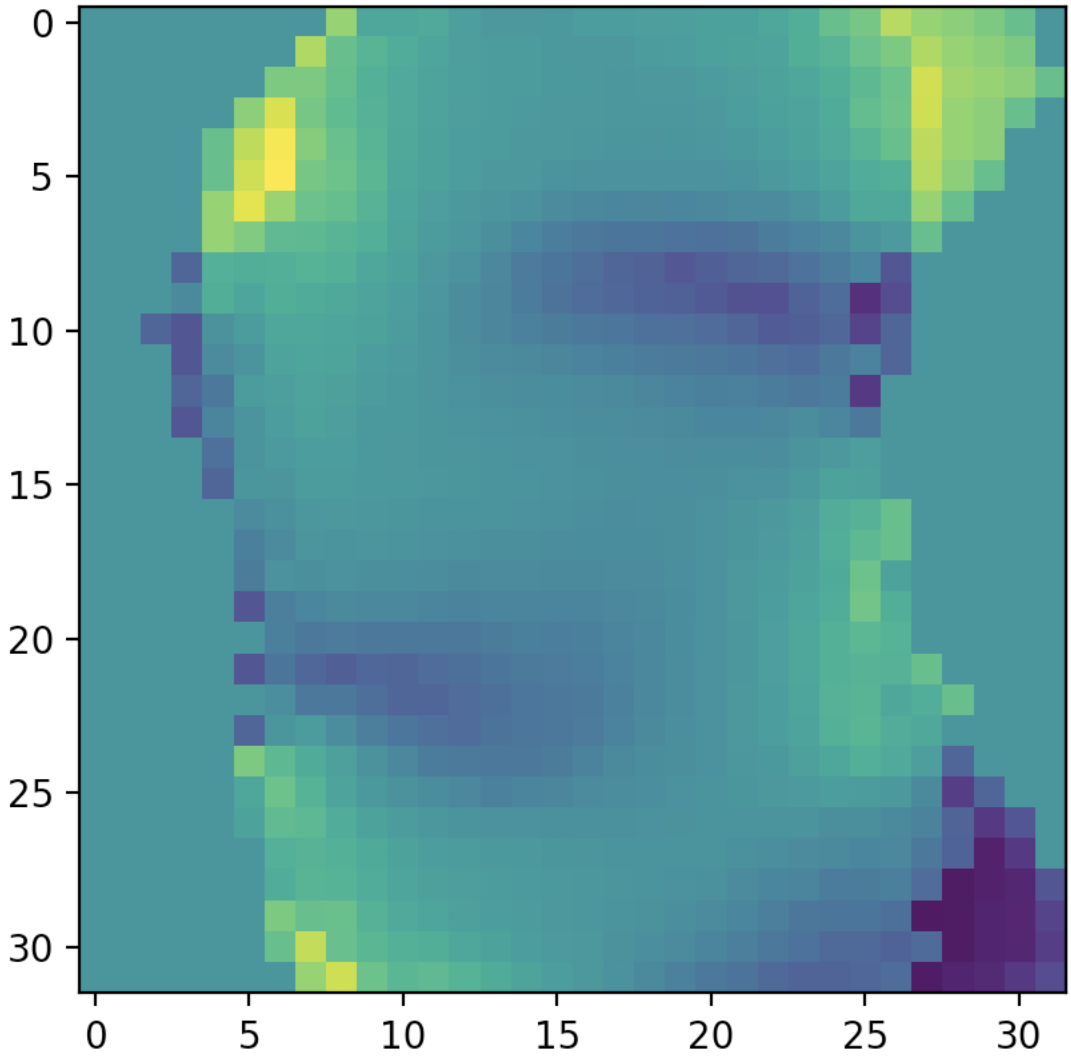




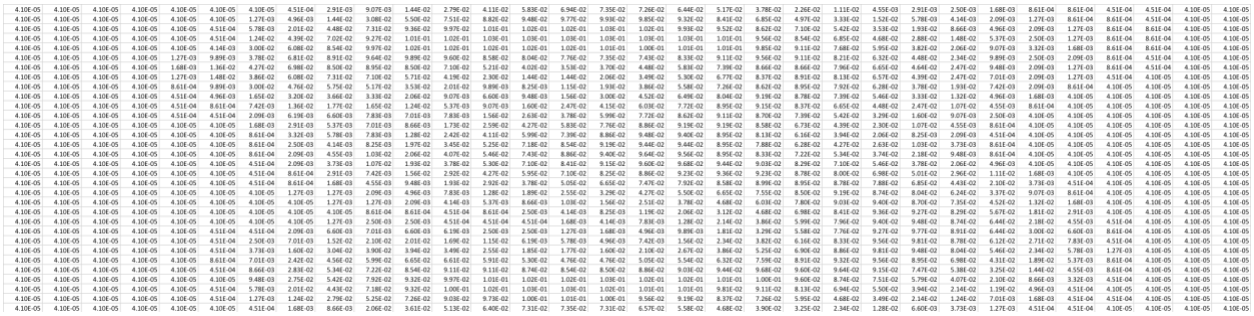

pair0
1
0
5
10
15
20
25
30

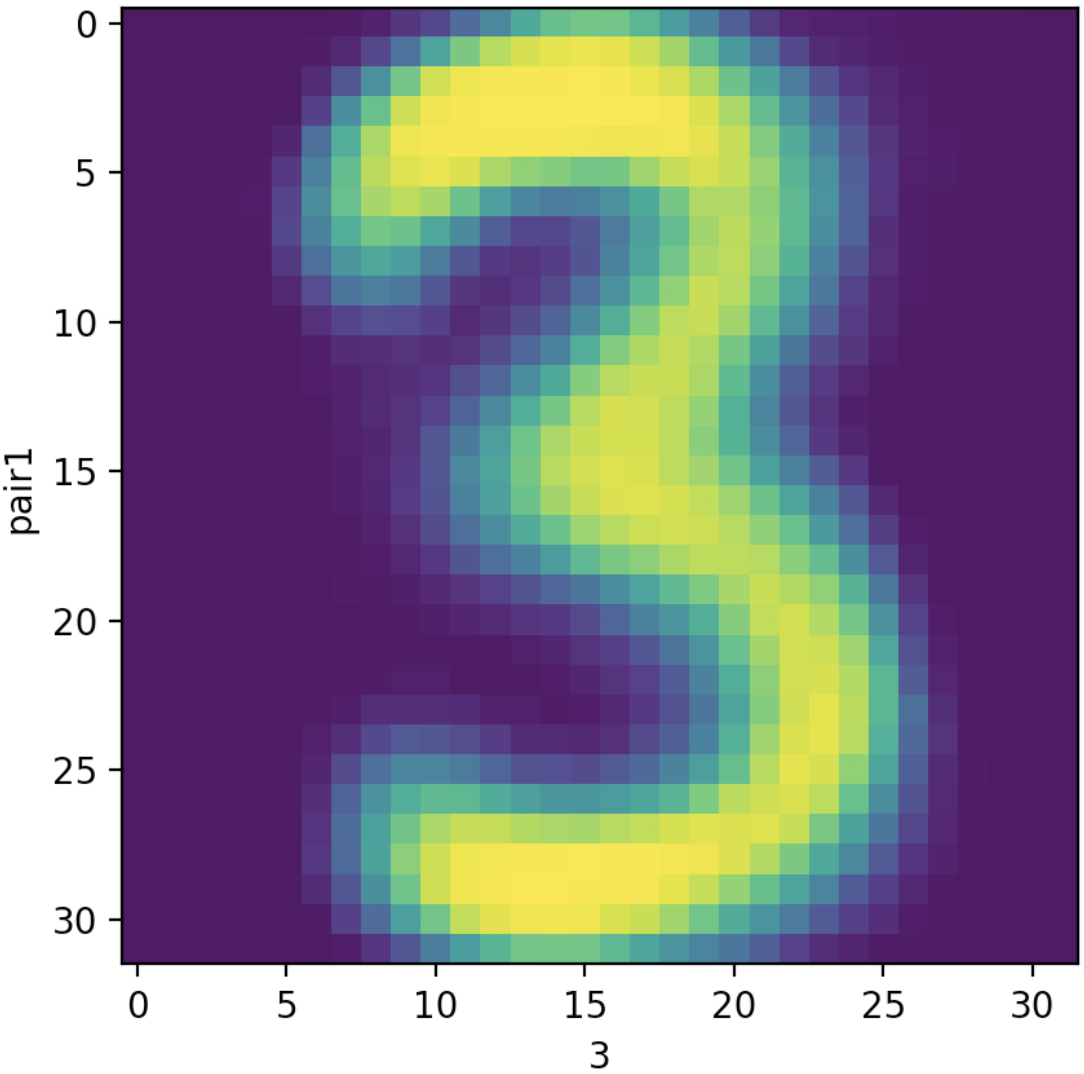
Odds Ratios

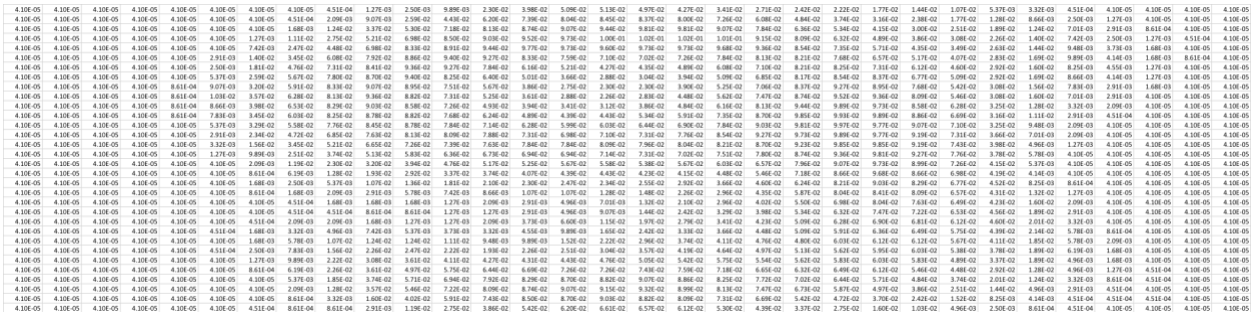


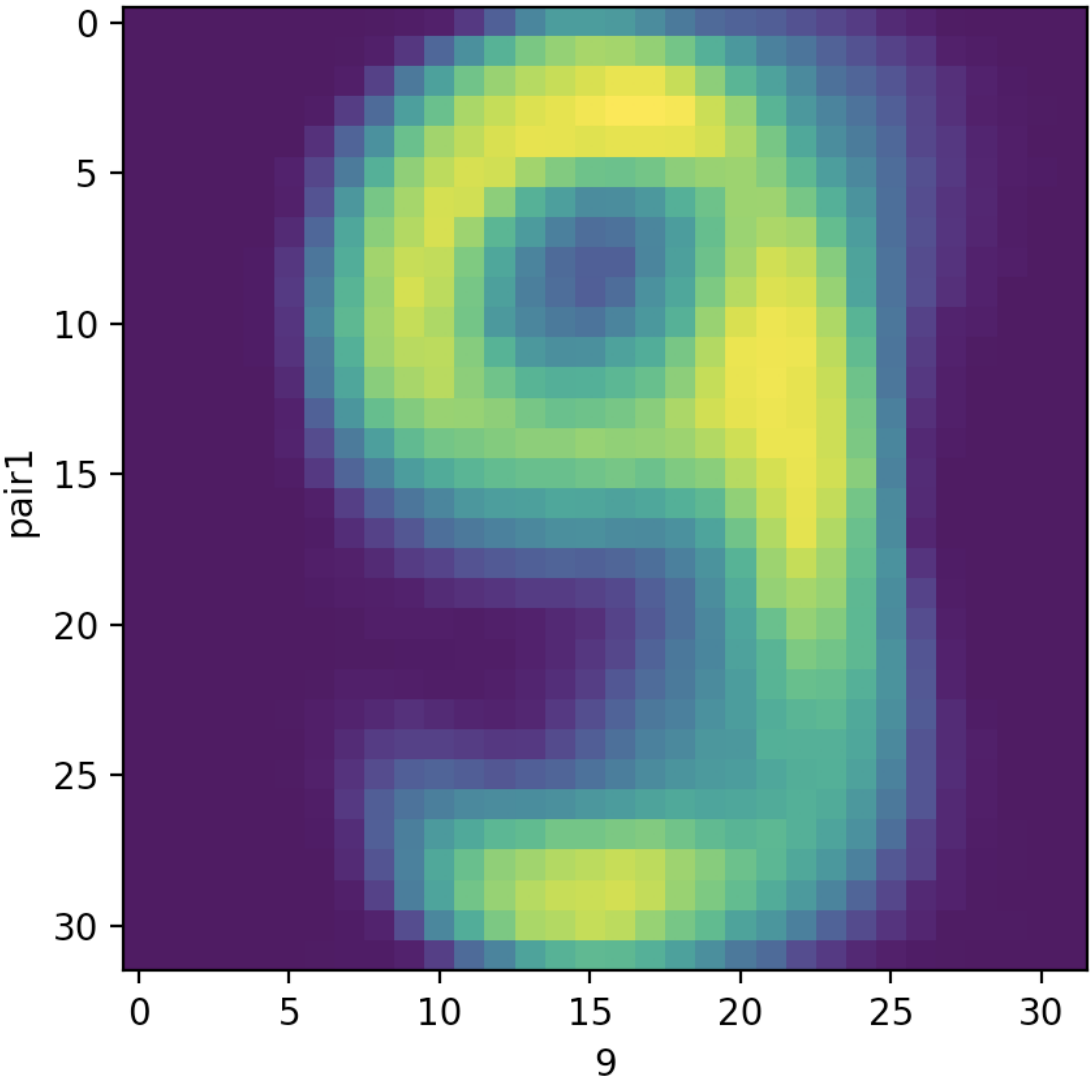


Label 3 and Label 9

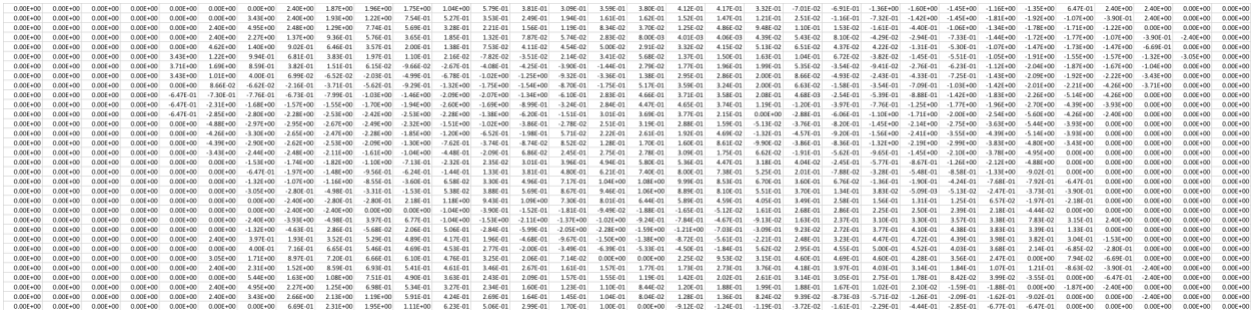


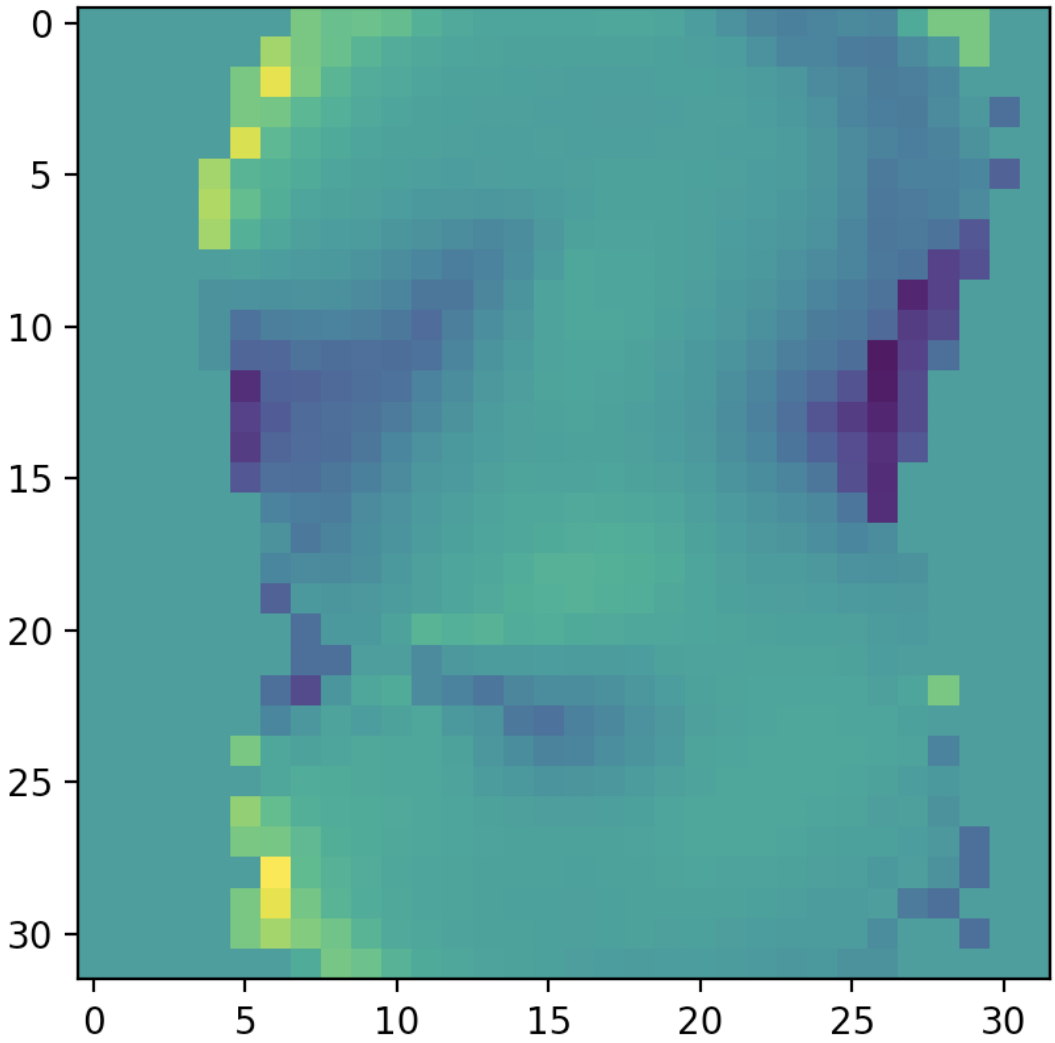

pair1
3
0
5
10
15
20
25
30



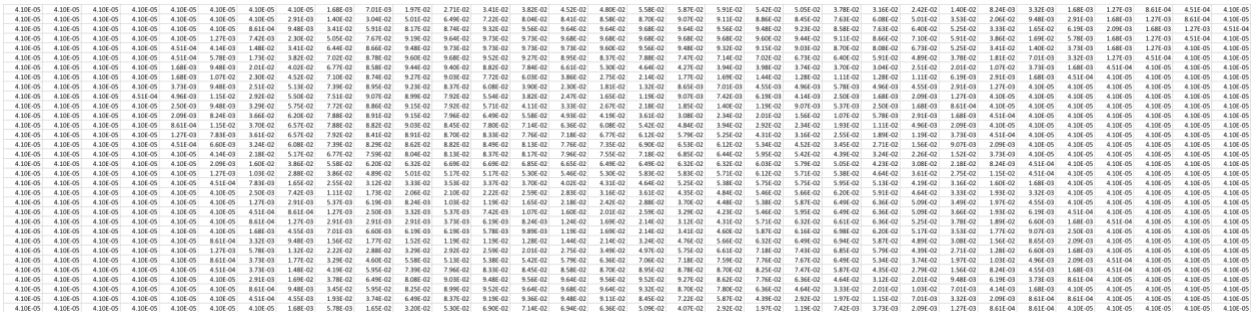

0
5
10
15
20
25
30
pair1
0
5
10
15
20
25
30
9

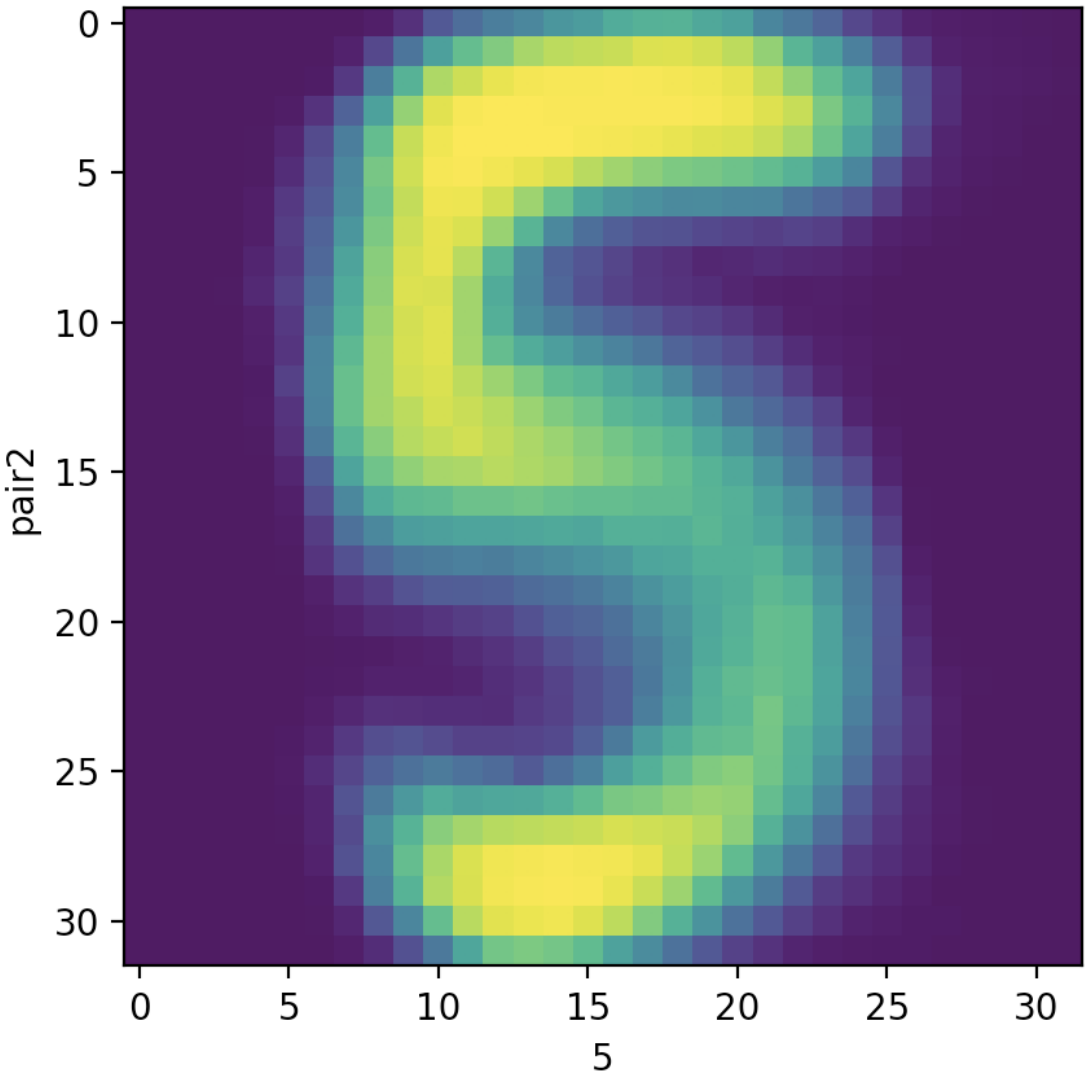
Odds Ratios



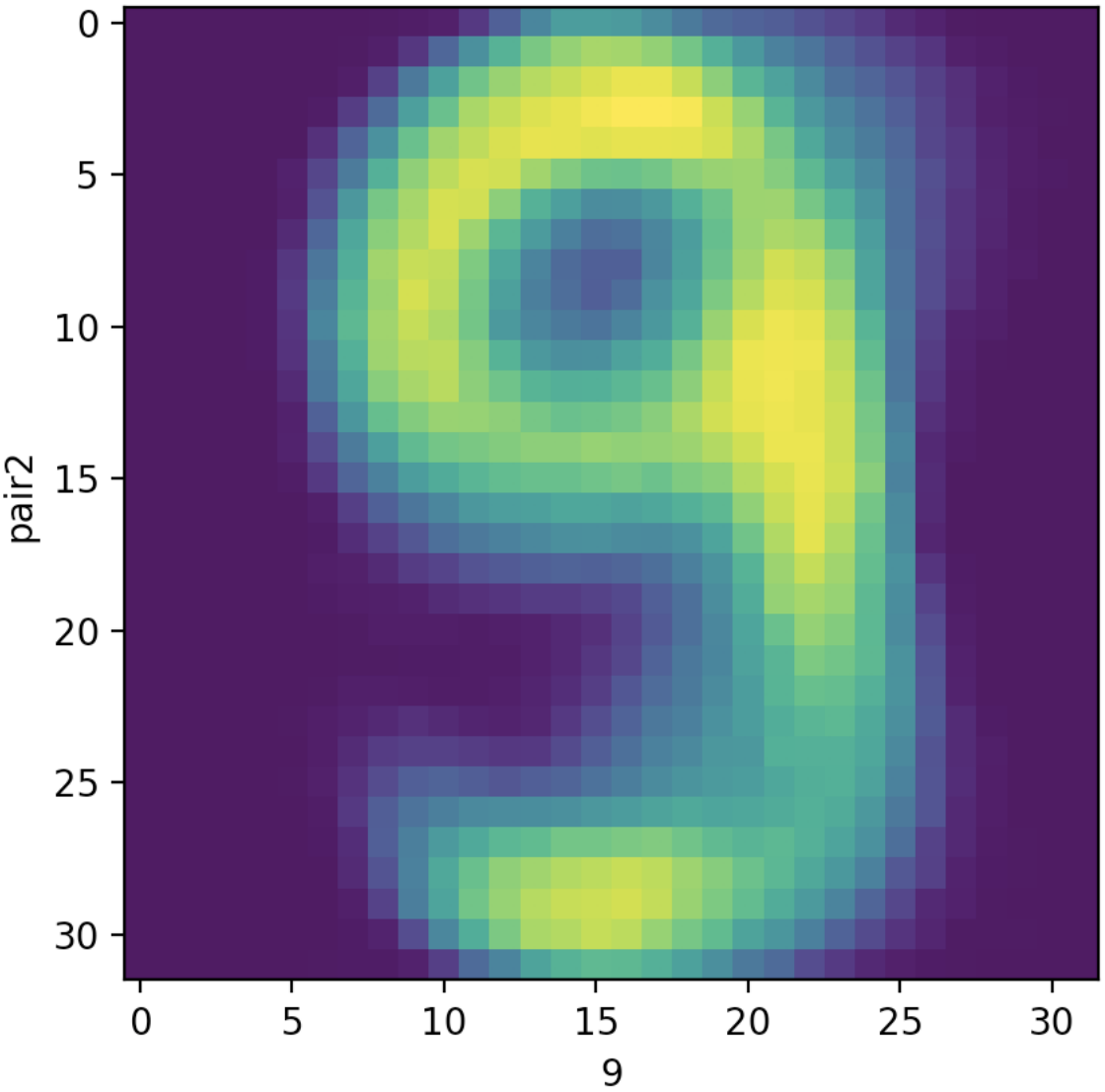


Label 5 and Label 9:

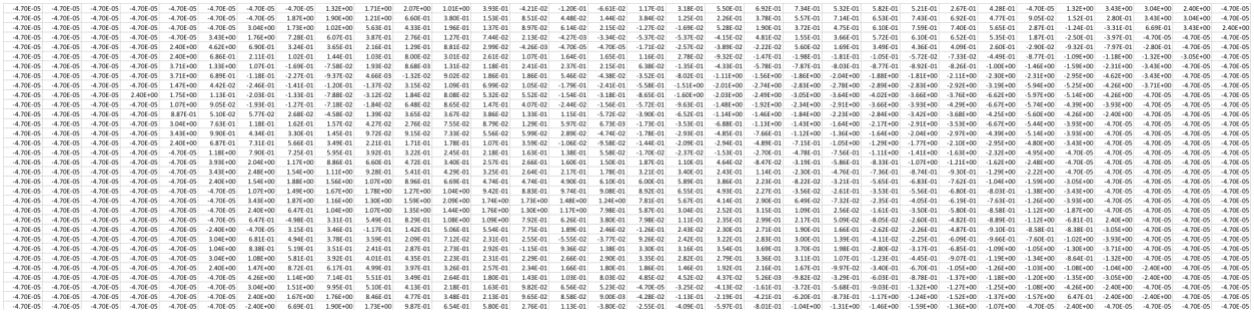


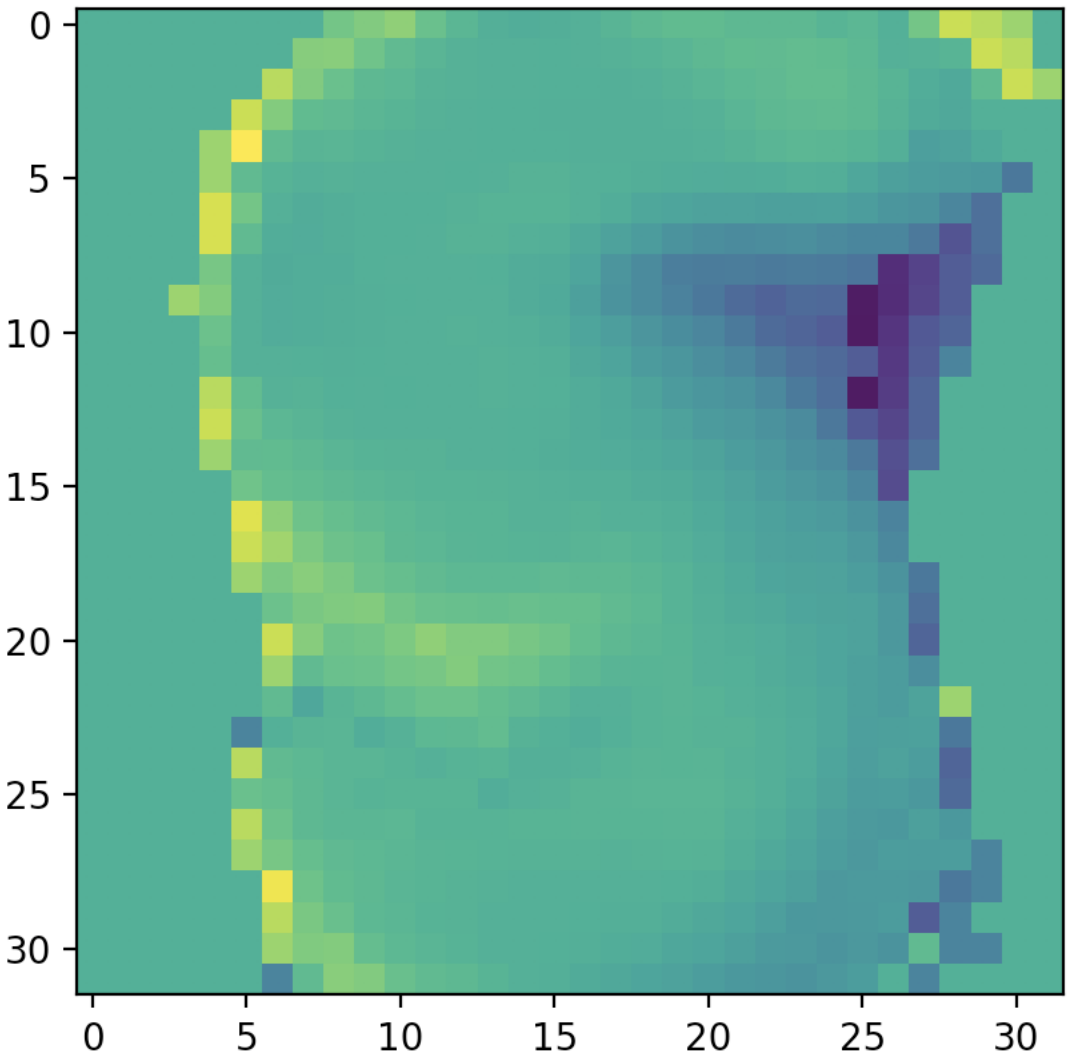

0
5
10
15
20
25
30
pair2
0
5
10
15
20
25
30
5

4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.51E-04	1.27E-03	2.50E-03	9.89E-03	2.30E-02	3.98E-02	5.09E-02	5.13E-02	4.97E-02	4.27E-02	3.41E-02	2.71E-02	2.42E-02	2.22E-02	1.77E-02	1.44E-02	1.07E-02	5.37E-03	3.32E-03	4.51E-04	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.51E-04	2.09E-03	9.07E-03	2.59E-02	6.43E-02	6.20E-02	7.39E-02	8.04E-02	8.45E-02	8.37E-02	8.00E-02	7.26E-02	6.08E-02	4.84E-02	3.74E-02	3.16E-02	2.38E-02	1.77E-02	1.28E-02	8.66E-03	2.50E-03	1.27E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	1.68E-03	1.24E-02	3.37E-02	5.30E-02	7.18E-02	8.13E-02	8.74E-02	9.07E-02	9.44E-02	9.81E-02	9.81E-02	9.07E-02	7.84E-02	6.36E-02	5.34E-02	4.15E-02	3.00E-02	2.51E-02	1.89E-02	1.24E-02	7.01E-03	2.91E-03	8.61E-04	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	1.27E-03	1.11E-02	2.75E-02	5.21E-02	6.98E-02	8.50E-02	9.03E-02	9.52E-02	9.73E-02	1.00E-01	1.02E-01	1.02E-01	1.01E-01	9.15E-02	8.09E-02	6.32E-02	4.89E-02	3.86E-02	3.08E-02	2.26E-02	1.40E-02	7.42E-03	2.50E-03	1.27E-03	4.51E-04	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	7.42E-03	2.47E-02	4.48E-02	6.98E-02	8.33E-02	8.91E-02	9.44E-02	9.77E-02	9.73E-02	9.60E-02	9.73E-02	9.73E-02	9.68E-02	9.36E-02	8.54E-02	7.35E-02	5.71E-02	4.35E-02	3.49E-02	2.63E-02	1.44E-02	9.48E-03	3.73E-03	1.68E-03	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	2.91E-03	1.40E-02	3.45E-02	6.08E-02	7.92E-02	8.86E-02	9.40E-02	9.27E-02	8.33E-02	7.59E-02	7.10E-02	7.02E-02	7.26E-02	7.84E-02	8.13E-02	8.21E-02	7.68E-02	6.57E-02	5.17E-02	4.07E-02	2.83E-02	1.69E-02	9.89E-03	4.14E-03	1.68E-03	8.61E-04	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	2.50E-03	1.81E-02	4.76E-02	7.31E-02	8.41E-02	9.36E-02	9.27E-02	7.84E-02	6.16E-02	5.21E-02	4.27E-02	4.35E-02	4.89E-02	6.08E-02	7.10E-02	8.21E-02	8.25E-02	7.31E-02	6.12E-02	4.60E-02	2.92E-02	1.60E-02	8.25E-03	4.55E-03	1.27E-03	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	5.37E-03	2.59E-02	5.67E-02	7.80E-02	8.70E-02	9.40E-02	8.25E-02	6.40E-02	5.01E-02	3.66E-02	2.88E-02	2.88E-02	3.04E-02	3.94E-02	5.09E-02	6.85E-02	8.17E-02	8.54E-02	8.37E-02	6.77E-02	5.09E-02	2.92E-02	1.69E-02	8.66E-03	4.14E-03	1.27E-03	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	8.61E-04	9.07E-03	3.20E-02	5.91E-02	8.33E-02	9.07E-02	8.95E-02	7.51E-02	5.67E-02	3.86E-02	2.75E-02	2.30E-02	2.30E-02	3.90E-02	5.25E-02	7.06E-02	8.37E-02	9.27E-02	8.95E-02	7.68E-02	5.42E-02	3.08E-02	1.56E-02	7.83E-03	2.91E-03	1.68E-03	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	8.61E-04	1.03E-02	3.57E-02	6.28E-02	8.13E-02	9.36E-02	8.82E-02	7.31E-02	5.25E-02	3.61E-02	2.88E-02	2.26E-02	2.83E-02	4.48E-02	5.62E-02	7.47E-02	8.74E-02	9.52E-02	9.36E-02	8.09E-02	5.46E-02	3.08E-02	1.60E-02	7.01E-03	2.91E-03	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	8.61E-04	8.66E-03	3.98E-02	6.53E-02	8.29E-02	9.03E-02	8.58E-02	7.26E-02	4.93E-02	3.94E-02	3.41E-02	3.12E-02	3.86E-02	4.84E-02	6.16E-02	8.13E-02	9.44E-02	9.89E-02	9.73E-02	8.58E-02	6.28E-02	3.25E-02	1.28E-02	3.32E-03	2.09E-03	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	8.61E-04	7.83E-03	3.45E-02	6.03E-02	8.25E-02	8.78E-02	8.82E-02	7.68E-02	6.24E-02	4.89E-02	4.39E-02	4.43E-02	5.34E-02	5.91E-02	7.35E-02	8.70E-02	9.85E-02	9.93E-02	9.89E-02	8.86E-02	6.69E-02	3.16E-02	1.11E-02	2.91E-03	4.51E-04	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	5.37E-03	3.29E-02	5.58E-02	7.76E-02	8.45E-02	8.78E-02	7.84E-02	7.14E-02	6.28E-02	5.99E-02	6.03E-02	6.44E-02	6.90E-02	7.84E-02	9.03E-02	9.81E-02	9.97E-02	9.77E-02	9.07E-02	7.10E-02	3.25E-02	9.48E-03	2.09E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	2.91E-03	2.34E-02	4.72E-02	6.85E-02	7.63E-02	8.13E-02	8.09E-02	7.88E-02	7.31E-02	6.98E-02	7.10E-02	7.31E-02	7.76E-02	8.54E-02	9.27E-02	9.73E-02	9.89E-02	9.77E-02	9.19E-02	7.31E-02	3.66E-02	7.01E-03	2.09E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	3.32E-03	1.56E-02	3.45E-02	5.21E-02	6.65E-02	7.26E-02	7.39E-02	7.63E-02	7.84E-02	7.84E-02	8.09E-02	7.96E-02	8.04E-02	8.21E-02	8.70E-02	9.23E-02	9.85E-02	9.85E-02	9.19E-02	7.43E-02	3.98E-02	4.96E-03	1.27E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	1.27E-03	9.89E-03	2.51E-02	3.74E-02	5.13E-02	5.83E-02	6.36E-02	6.73E-02	6.94E-02	6.94E-02	7.14E-02	7.31E-02	7.02E-02	7.53E-02	7.80E-02	8.74E-02	9.36E-02	9.81E-02	9.27E-02	7.76E-02	3.78E-02	5.78E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	2.09E-03	1.19E-02	2.30E-02	3.20E-02	3.94E-02	4.76E-02	5.17E-02	5.25E-02	5.67E-02	5.58E-02	5.38E-02	5.67E-02	6.03E-02	6.57E-02	7.96E-02	9.07E-02	9.73E-02	8.99E-02	7.26E-02	4.15E-02	5.37E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	8.61E-04	6.19E-03	1.28E-02	1.93E-02	2.92E-02	3.37E-02	3.74E-02	4.07E-02	4.39E-02	4.43E-02	4.23E-02	4.15E-02	4.48E-02	5.46E-02	7.18E-02	8.66E-02	9.68E-02	8.66E-02	6.98E-02	4.19E-02	4.14E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	1.68E-03	2.50E-03	5.37E-03	1.07E-02	1.36E-02	1.81E-02	2.10E-02	2.30E-02	2.47E-02	2.34E-02	2.55E-02	2.92E-02	3.66E-02	4.60E-02	6.24E-02	8.21E-02	9.03E-02	8.29E-02	6.77E-02	4.52E-02	8.25E-03	8.61E-04	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	8.61E-04	1.68E-03	2.09E-03	2.91E-03	5.78E-03	7.42E-03	8.66E-03	1.07E-02	1.07E-02	1.28E-02	1.48E-02	2.26E-02	2.96E-02	4.35E-02	5.87E-02	8.04E-02	8.41E-02	8.09E-02	6.57E-02	4.31E-02	1.32E-02	1.27E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.51E-04	1.68E-03	1.68E-03	1.68E-03	1.27E-03	2.09E-03	2.91E-03	4.96E-03	7.01E-03	1.32E-02	2.10E-02	2.96E-02	4.02E-02	5.50E-02	6.98E-02	8.04E-02	7.63E-02	6.49E-02	4.23E-02	1.60E-02	2.09E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.51E-04	4.51E-04	8.61E-04	8.61E-04	1.27E-03	1.27E-03	2.91E-03	4.96E-03	9.07E-03	1.44E-02	2.42E-02	3.29E-02	3.98E-02	5.34E-02	6.32E-02	7.47E-02	7.22E-02	6.53E-02	4.56E-02	1.89E-02	2.91E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.51E-04	2.09E-03	2.09E-03	1.68E-03	1.27E-03	1.27E-03	2.09E-03	3.73E-03	6.60E-03	1.15E-02	1.97E-02	2.79E-02	3.41E-02	4.23E-02	5.09E-02	6.28E-02	6.90E-02	6.81E-02	6.12E-02	4.60E-02	2.01E-02	3.32E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.51E-04	1.68E-03	3.32E-03	4.96E-03	7.42E-03	5.37E-03	3.73E-03	3.32E-03	4.55E-03	9.89E-03	1.65E-02	2.42E-02	3.33E-02	3.66E-02	4.48E-02	5.09E-02	5.91E-02	6.36E-02	6.49E-02	5.75E-02	4.39E-02	2.14E-02	5.78E-03	8.61E-04	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	1.68E-03	5.78E-03	1.07E-02	1.24E-02	1.24E-02	1.11E-02	9.48E-03	9.89E-03	1.52E-02	2.22E-02	2.96E-02	3.74E-02	4.11E-02	4.76E-02	4.80E-02	6.03E-02	6.12E-02	6.12E-02	5.67E-02	4.11E-02	1.85E-02	5.78E-03	2.09E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.51E-04	2.50E-03	7.83E-03	1.56E-02	2.26E-02	2.47E-02	2.22E-02	1.93E-02	2.26E-02	2.51E-02	3.04E-02	3.57E-02	4.19E-02	4.64E-02	4.97E-02	5.13E-02	5.62E-02	5.95E-02	6.03E-02	5.38E-02	3.78E-02	1.89E-02	6.19E-03	1.68E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	1.27E-03	9.89E-03	2.22E-02	3.08E-02	3.61E-02	4.11E-02	4.27E-02	4.31E-02	4.43E-02	4.76E-02	5.05E-02	5.42E-02	5.75E-02	5.54E-02	5.62E-02	5.83E-02	6.03E-02	5.83E-02	4.89E-02	3.37E-02	1.89E-02	4.96E-03	1.68E-03	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	8.61E-04	6.19E-03	2.26E-02	3.61E-02	4.97E-02	5.75E-02	6.44E-02	6.69E-02	7.26E-02	7.26E-02	7.43E-02	7.59E-02	7.18E-02	6.55E-02	6.32E-02	6.49E-02	6.12E-02	5.46E-02	4.48E-02	2.92E-02	1.28E-02	4.96E-03	1.27E-03	4.51E-04	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	5.37E-03	1.85E-02	3.74E-02	5.71E-02	6.94E-02	7.92E-02	8.29E-02	8.70E-02	8.82E-02	9.07E-02	8.86E-02	8.25E-02	7.72E-02	7.02E-02	6.44E-02	5.71E-02	4.84E-02	3.74E-02	2.01E-02	1.24E-02	3.32E-03	8.61E-04	4.51E-04	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	2.09E-03	1.28E-02	3.57E-02	5.46E-02	7.22E-02	8.09E-02	8.74E-02	9.07E-02	9.15E-02	9.32E-02	8.99E-02	8.13E-02	7.47E-02	6.73E-02	5.87E-02	4.97E-02	3.86E-02	2.51E-02	1.44E-02	4.96E-03	2.91E-03	4.51E-04	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	8.61E-04	3.32E-03	1.60E-02	4.02E-02	5.91E-02	7.43E-02	8.50E-02	8.70E-02	9.03E-02	8.82E-02	8.09E-02	7.31E-02	6.69E-02	5.42E-02	4.72E-02	3.70E-02	2.42E-02	1.52E-02	8.25E-03	4.14E-03	4.51E-04	4.51E-04	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05
4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.51E-04	8.61E-04	8.61E-04	2.91E-03	1.19E-02	2.75E-02	3.86E-02	5.42E-02	6.20E-02	6.61E-02	6.57E-02	6.12E-02	5.30E-02	4.39E-02	3.37E-02	2.75E-02	1.60E-02	1.03E-02	4.96E-03	2.50E-03	8.61E-04	4.51E-04	4.10E-05	4.10E-05	4.10E-05	4.10E-05	4.10E-05

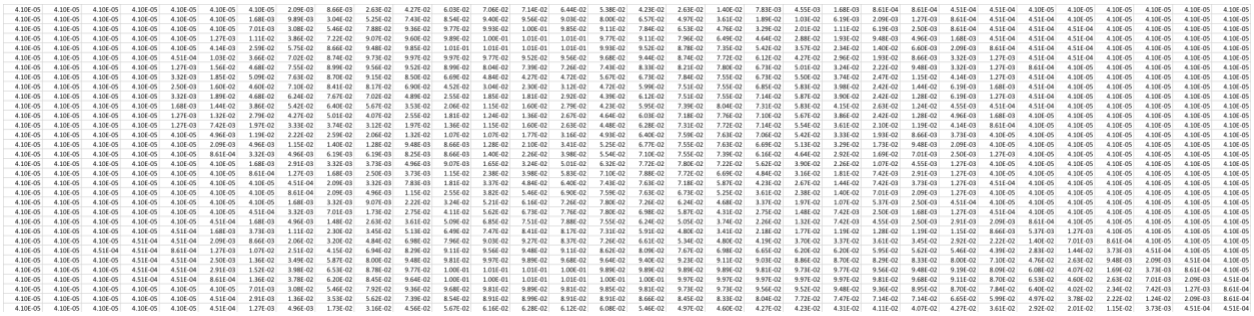


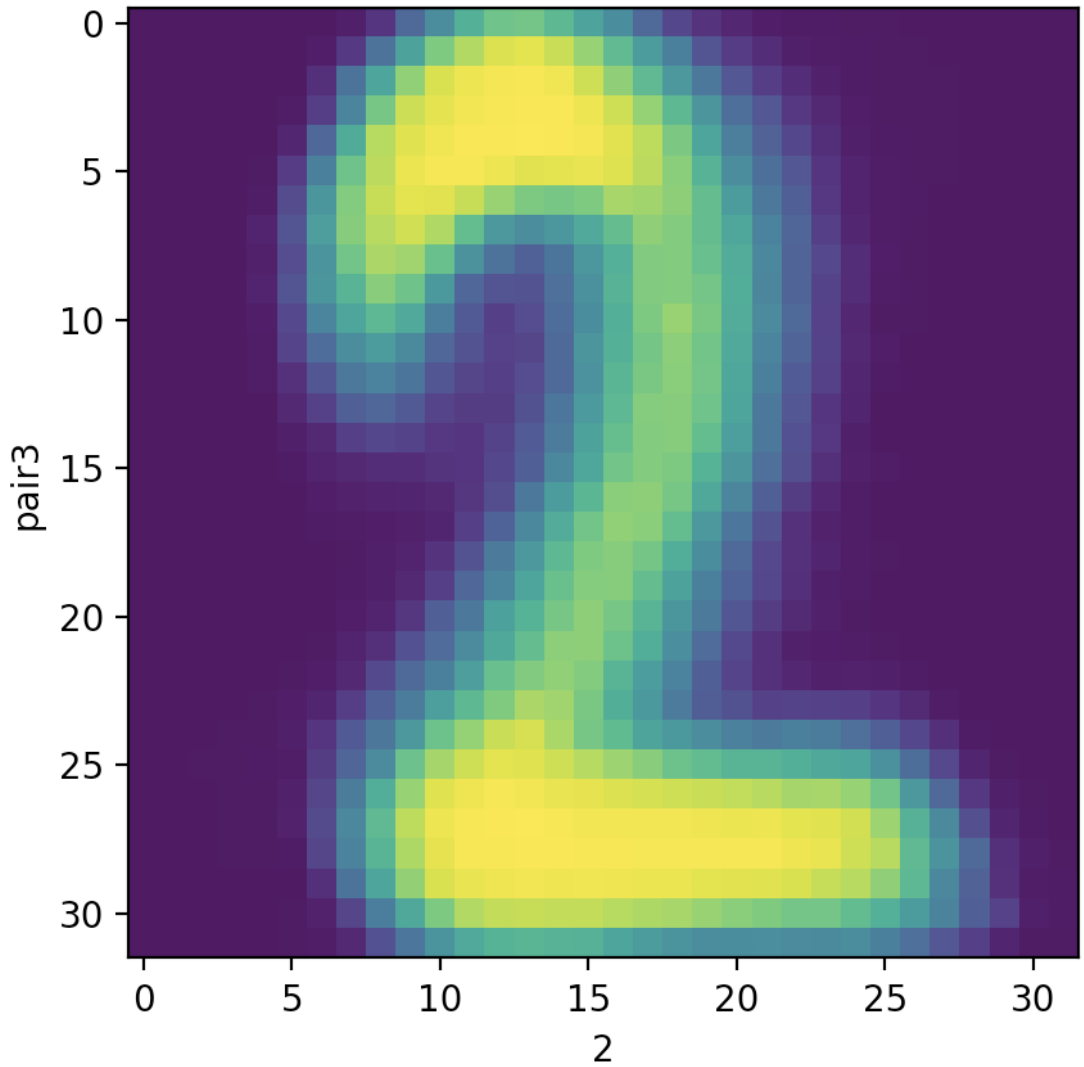
Odds Ratios

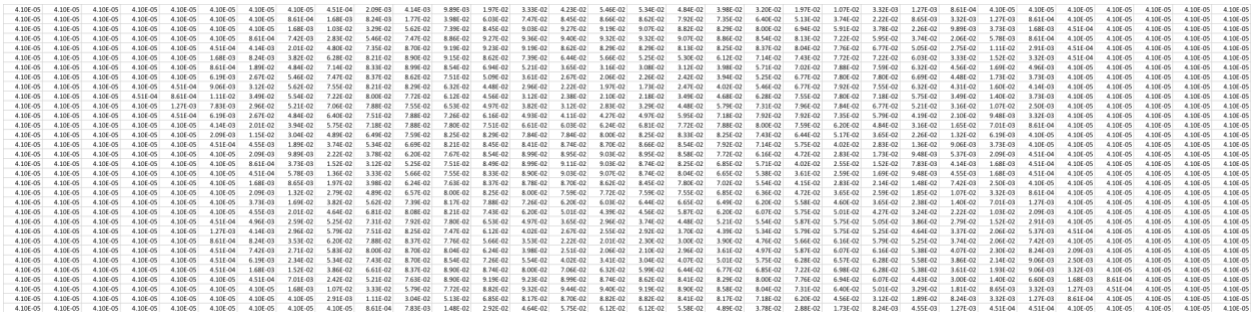


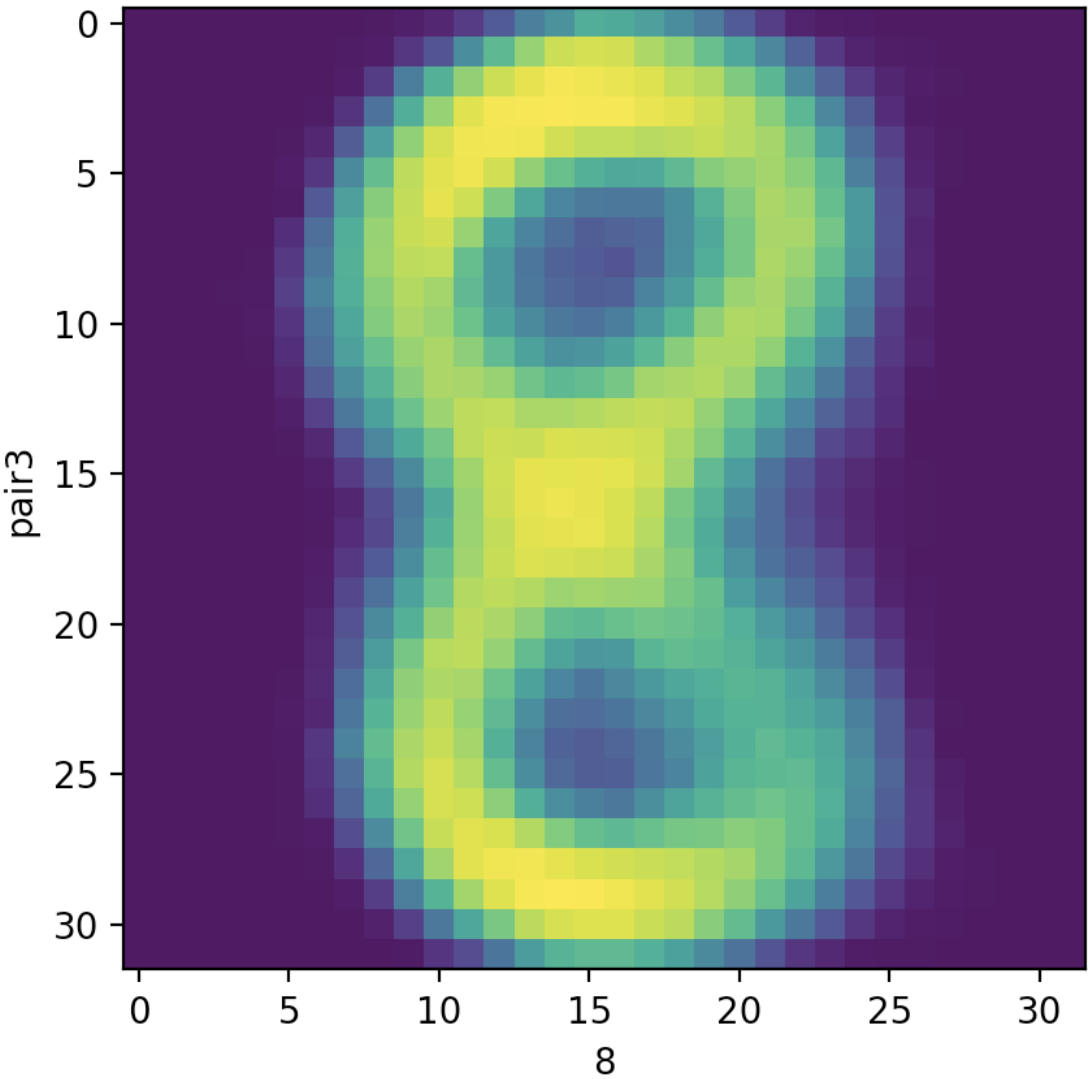


Label 2 and Label 8:

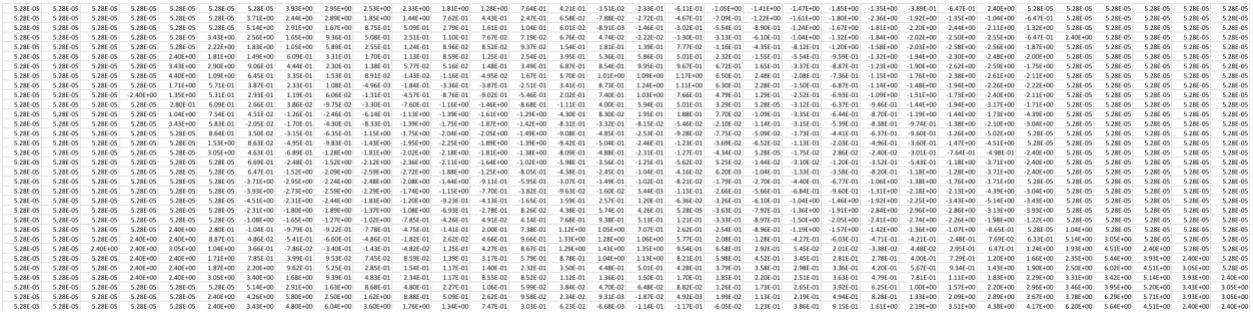


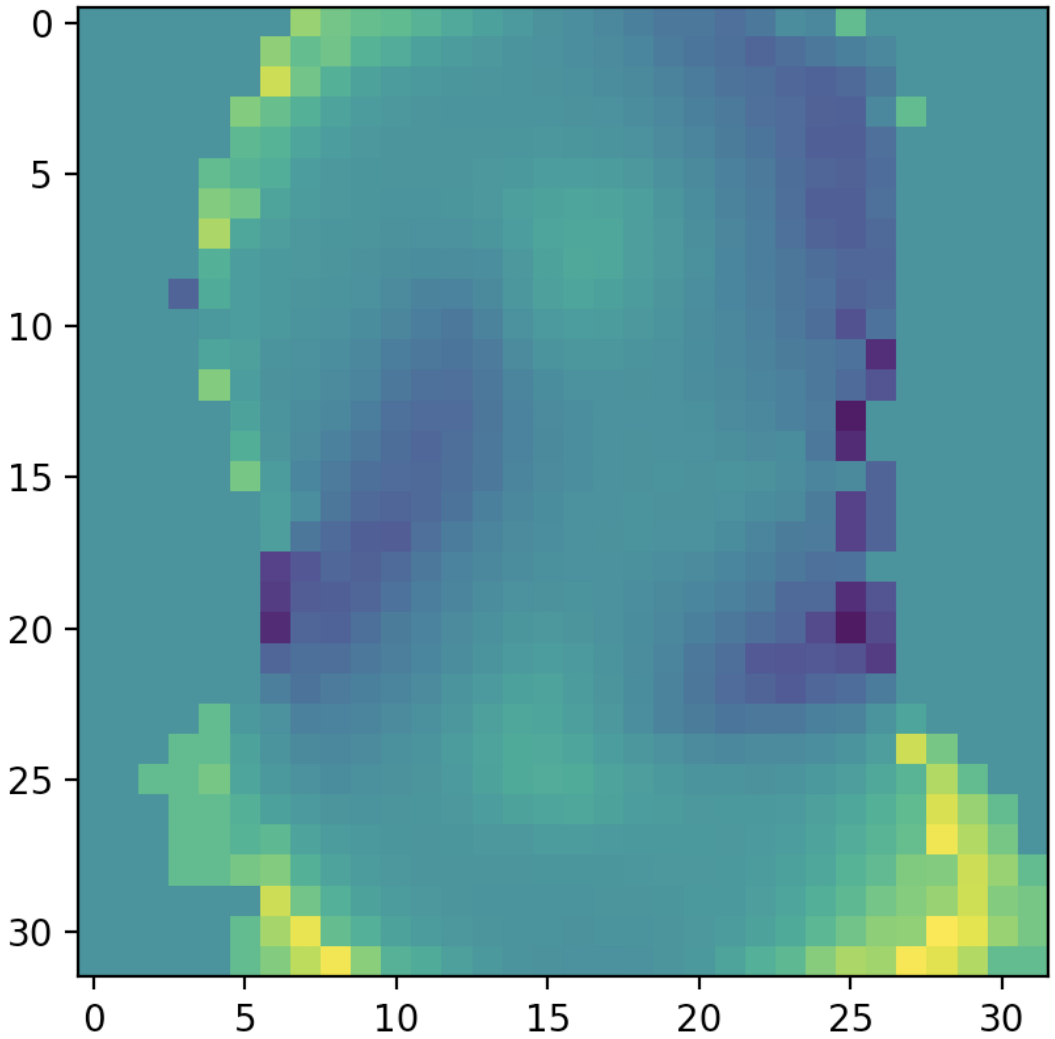

pair3
2
0
5
10
15
20
25
30




pair3
0
5
10
15
20
25
30
0
5
10
15
20
25
30
8

Odds Ratios





Part 1 Extra Credit

The face classification is similar to digit classification. Since the data of the input file is a little bit different, I replaced the ‘ ’ to 0 and ‘#’ to 1. In that case, we can treat the face classification similar to the digit. However, we found that the fetch of k value is different. The accuracy will increase

with the value of k. Thus, we use k = 10 in this section. The accuracy and confusion matrix show below.

```
overall_accuracy is:  0.7666666666666667
confusionmatrix is :
 [[ 0.97402597  0.02597403]
 [ 0.45205479  0.54794521]]
```

Part 2. Digit Classification via Machin Learning Methods

Part 2.1 Digit Classification with Perceptrons

For our implementation, we assign each class a weight vector, name it w_i , $i = 0,1, \dots, 9$, and each of it is of dimension of (1, 1024) when no bias term included or (1, 1025) when bias term included. We combine all the weight vectors together to form a (10, 1024) or (10, 1025) two dimensional array. For each of the input image, we turn it into an array with dimension of (1024, 1), or (1025, 1), according to the case of existence of bias term. Then we multiply these two arrays, and finally get a 2D result array of (10, 1). Each row of the result array stands for the score of the corresponding class weight vector give for the image, then, we just pick up the class which give out the highest score according to the decision rule stated in the lecture. If the classifier mismatches the image, then the related weight vectors will be updated according to the update rule stated in lecture. After tuning the optimal parameter, we test our trained model on the test set and check the overall accuracy.

1. Optimal Parameter Setting

For this part, we tried miscellaneous parameter settings, and below is the optimal setting:

Learning rate decay function: 0.00001 (tried from 1 to 10e-8);

Bias or no bias: bias (averagely, bias could bring around 5% higher accuracy than no bias);

Initialization of weights: zeros (tried both zeros and random, when the epoch is pretty large, random could also bring good accuracy, but zeros could bring relatively high accuracy with much less epoch number);

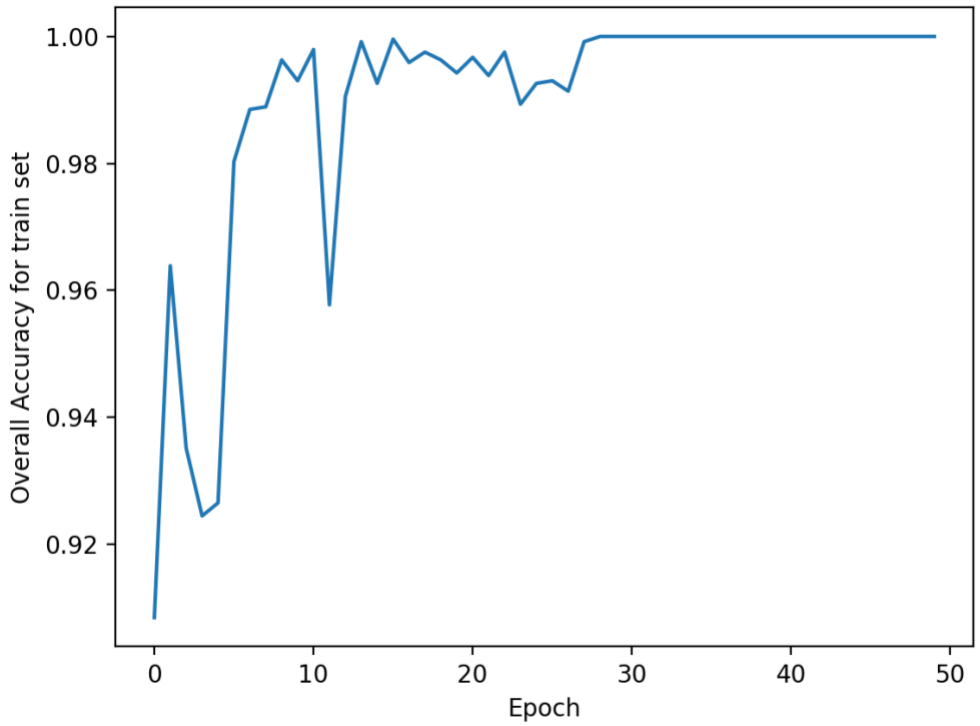
Ordering of training examples: fixed (tried both fixed and random, though random sometimes could bring higher accuracy than fixed, but usually the accuracy is lower than fixed, and the performance is quite unstable);

Number of epochs: 11 (tried from 1 to 50, though when epoch reaches 29, the training set accuracy get 100%, and the over-fitting occurs then, and the test set accuracy is lower than the case with epoch of 11, which brings the highest accuracy on test set in our implementation).

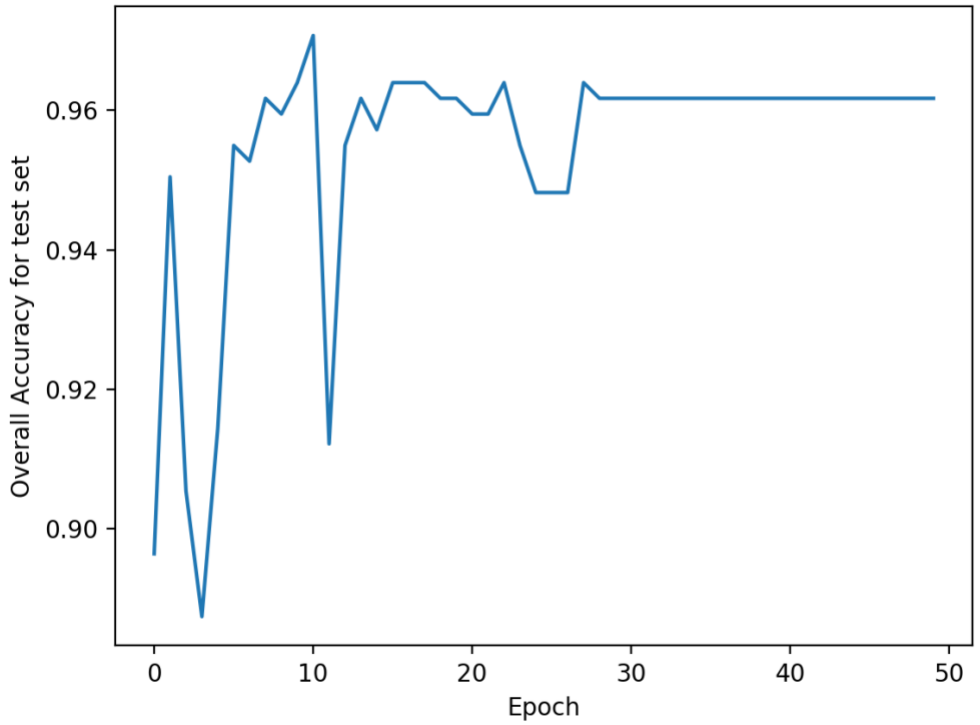
2. Output Results

(1) Training Curve

The training curve for training set is shown below:



Like mentioned above, though the training set accuracy could reach 100% when epoch get the number of 29 or higher, the over-fitting situation occurs, and the performance of the model on test set later on is not good, while when epoch picks up the value of 11, the test set accuracy get maximum in our implementation (details are shown in the figure of test set accuracy below).



(2) Overall Test Set Accuracy

The overall test set accuracy is shown below:

```
The accuracy for test set is:  0.9707207207207207
```

This result is acquired under the optimal parameter settings above.

(3) Confusion Matrix

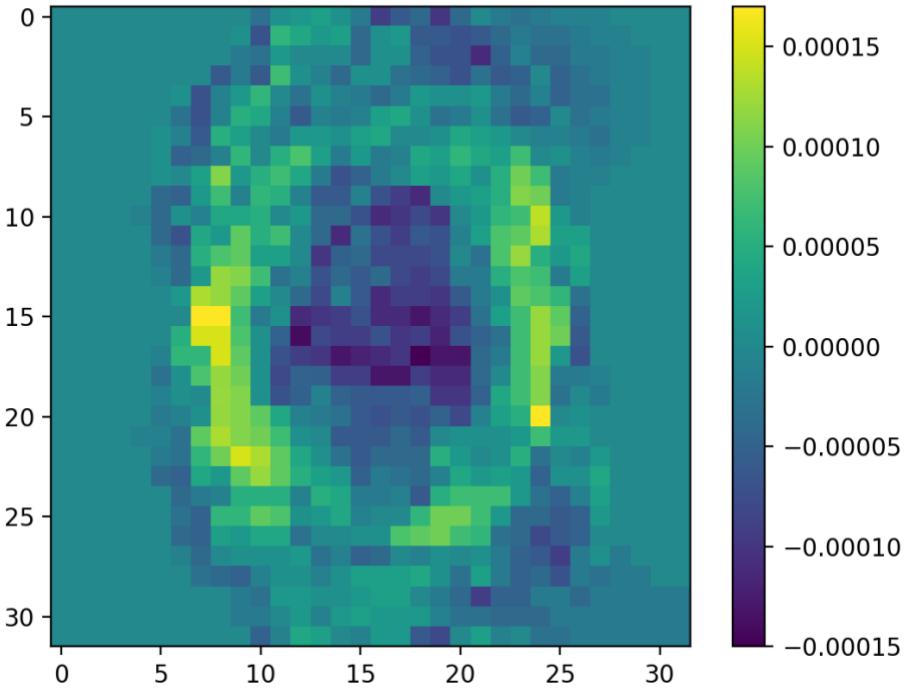
```
1.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00 9.556e-01 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 4.444e-02 0.000e+00 0.000e+00
0.000e+00 0.000e+00 9.512e-01 0.000e+00 0.000e+00 2.439e-02 0.000e+00 0.000e+00 2.439e-02 0.000e+00
0.000e+00 0.000e+00 0.000e+00 1.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00 1.695e-02 0.000e+00 0.000e+00 9.492e-01 0.000e+00 0.000e+00 0.000e+00 1.695e-02 1.695e-02
0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 1.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 0.000e+00 2.326e-02 0.000e+00 9.767e-01 0.000e+00 0.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 1.000e+00 0.000e+00 0.000e+00
0.000e+00 2.500e-02 0.000e+00 0.000e+00 0.000e+00 2.500e-02 0.000e+00 0.000e+00 9.500e-01 0.000e+00
0.000e+00 0.000e+00 0.000e+00 2.381e-02 0.000e+00 2.381e-02 0.000e+00 0.000e+00 2.381e-02 9.286e-01
```

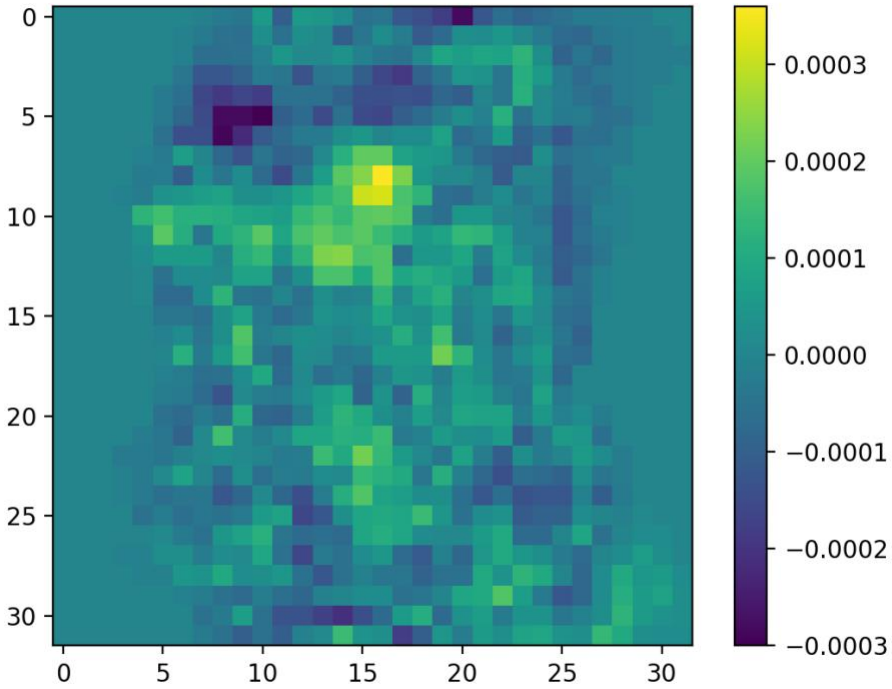
Class pair (0, 0) is at up left of the matrix, (9, 9) is at bottom right of the matrix.

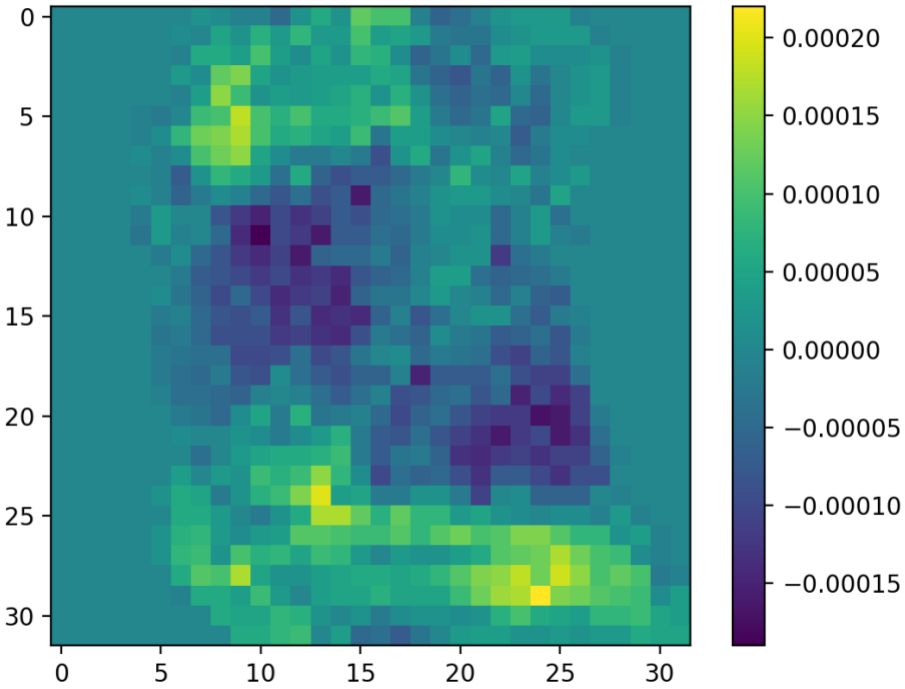
Part 2 Extra Credit:

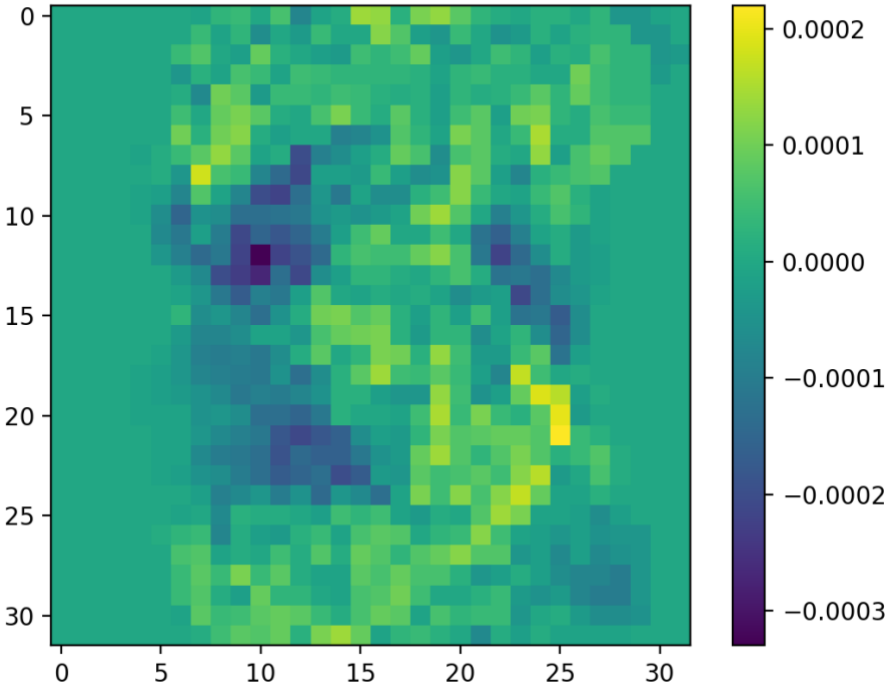
1. Weight Visualization

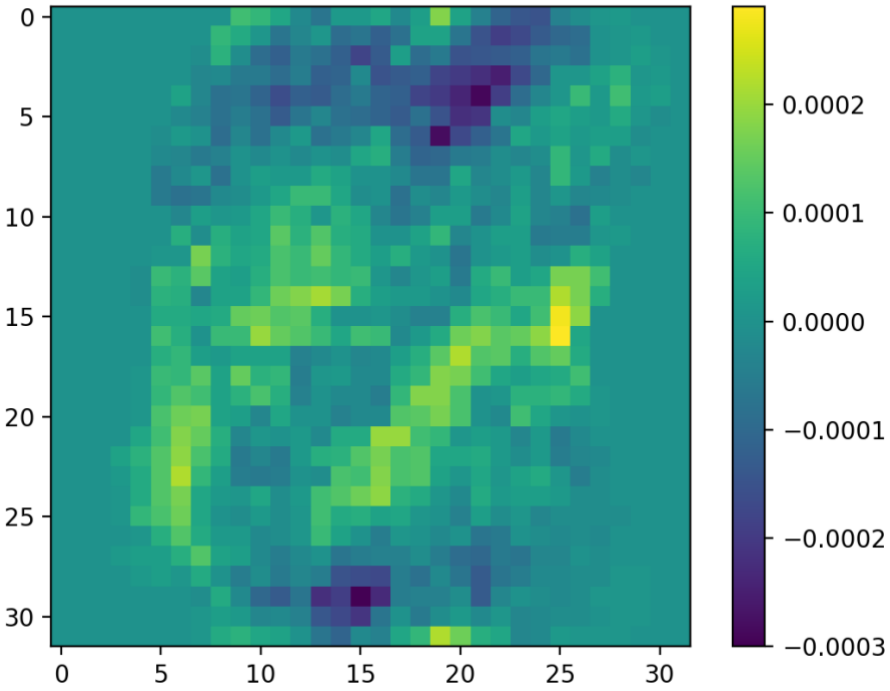
Below shows the 10 class weights visualization images, in the order of 0 to 9:

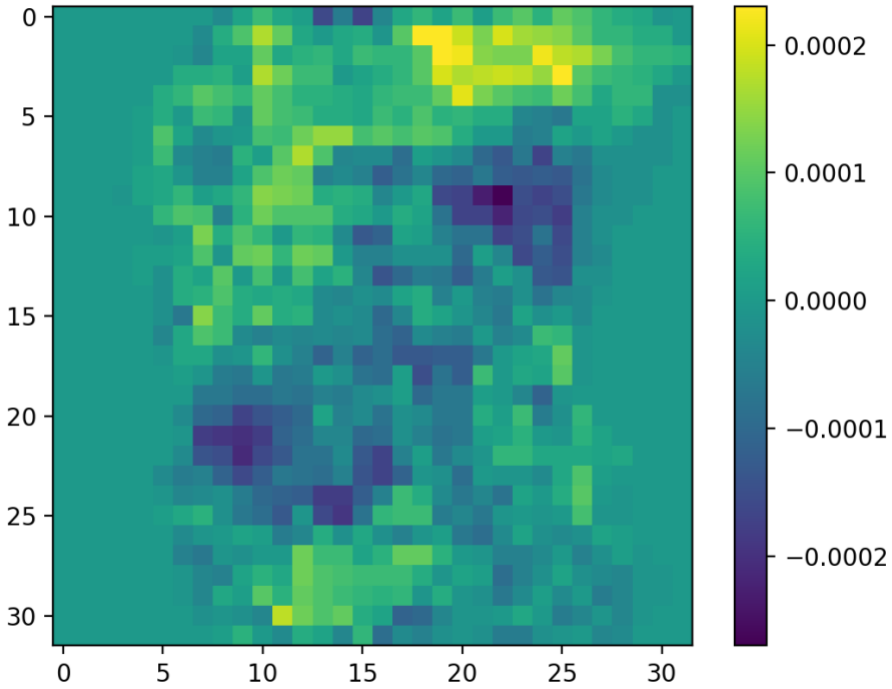


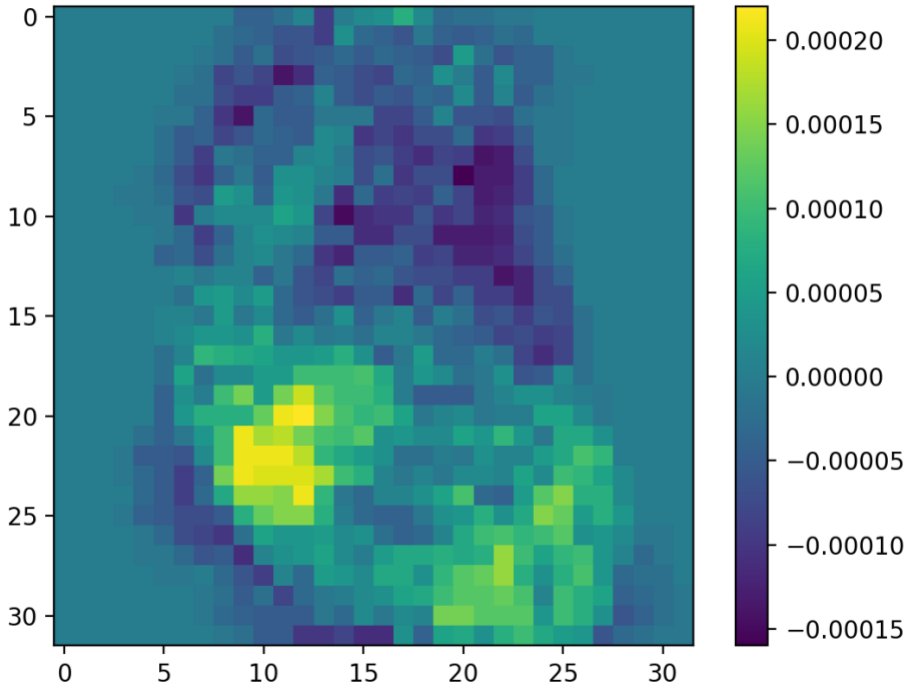

0
5
10
15
20
25
30
0
5
10
15
20
25
30
0.0003
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003

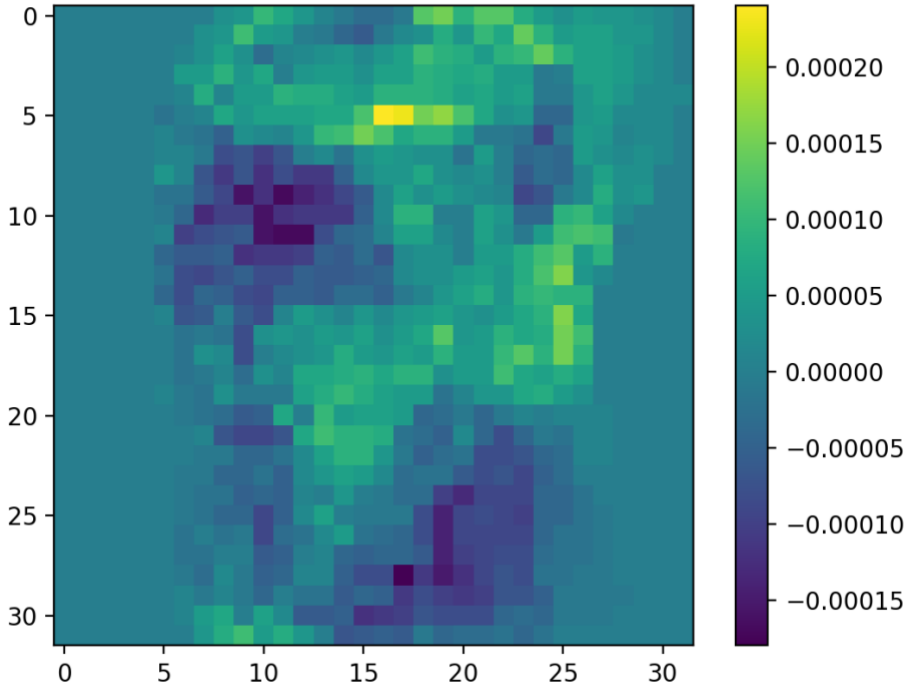

0
5
10
15
20
25
30
0.00020
0.00015
0.00010
0.00005
0.00000
−0.00005
−0.00010
−0.00015

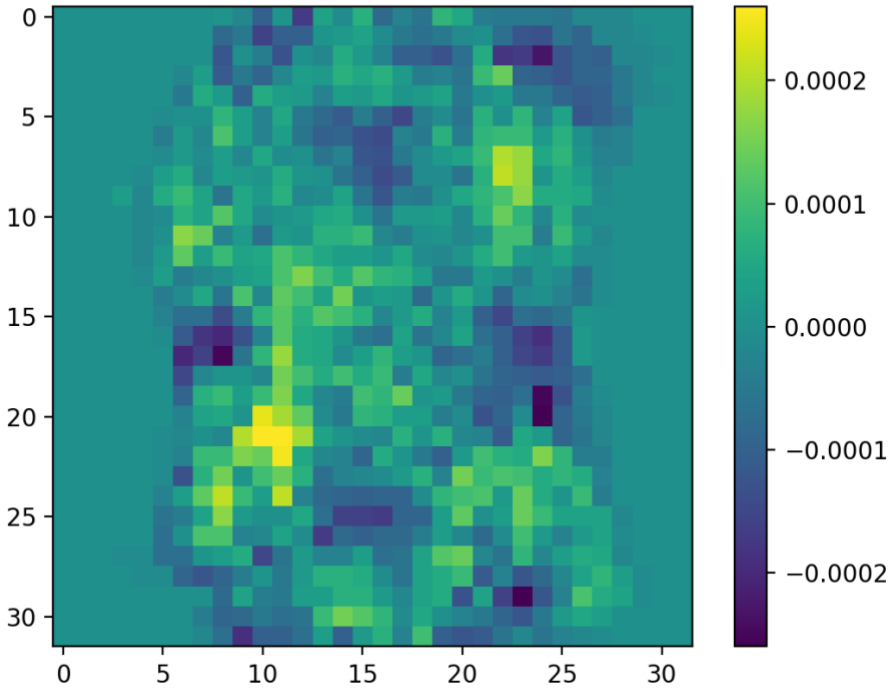

0
5
10
15
20
25
30
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003

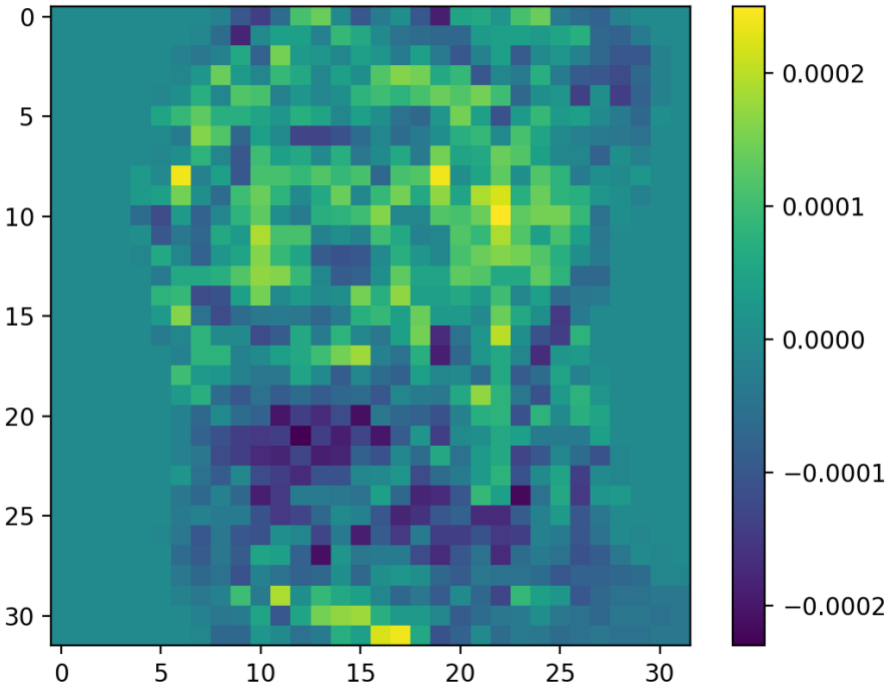

0
5
10
15
20
25
30
0
5
10
15
20
25
30
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003


0
5
10
15
20
25
30
0
5
10
15
20
25
30
0.0002
0.0001
0.0000
−0.0001
−0.0002


0
5
10
15
20
25
30
0.00020
0.00015
0.00010
0.00005
0.00000
−0.00005
−0.00010
−0.00015


0
5
10
15
20
25
30
0.00020
0.00015
0.00010
0.00005
0.00000
−0.00005
−0.00010
−0.00015


0
5
10
15
20
25
30
0
5
10
15
20
25
30
0.0002
0.0001
0.0000
−0.0001
−0.0002



The images above show that the most important pixels for each digit class are located in which pixels inside the weight visualization image are brighter, i.e., the visualization value (shown by the color bar in the visualization images) is higher than the other pixels. This can be explained easily by observing the shape of these visualization images. We can find that for each digit class, the corresponding weight visualization also look just like that digit. In the original image, the pixels with value of 1 contributes to the difference of digits, so the weight at these locations should be high enough to specify and strengthen the unique pattern. On top of this, the signs of the weights mean that the locations with positive values are expected to have "1" for this digit class, while negative means those locations are expected to have "0" for this digit class.

2. Classifier from Other Source

For this extra part, we utilized Support Vector Machine methods, in detail, refers to the "multiclass.OneVsRestClassifier", "multiclass.OneVsOneClassifier" and "svm" methods from "sklearn" packages.

(source: scikit-learn.org)

We utilized svm.LinearSVC as the basic binary classifier, and above this, we implemented three kinds of multi-classification methods: OneVsRest, OneVsOne, and Crammer Singer.

And among these functions, we found the OneVsOne classifier gives the highest test set prediction accuracy. Below shows the results of prediction accuracy on test set of sklearn method:

```
The overall accuracy for test set via sklearn method is:  0.9797297297297297
```

And the confusion matrix (pair (0, 0) is located up left, (9, 9) located bottom right):

```
1.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00 9.778e-01 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 2.222e-02 0.000e+00 0.000e+00
0.000e+00 0.000e+00 9.268e-01 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 4.878e-02 2.439e-02
0.000e+00 0.000e+00 0.000e+00 1.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00 1.695e-02 0.000e+00 0.000e+00 9.661e-01 0.000e+00 0.000e+00 0.000e+00 1.695e-02 0.000e+00
0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 1.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 0.000e+00 2.326e-02 0.000e+00 9.767e-01 0.000e+00 0.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 0.000e+00 2.128e-02 0.000e+00 0.000e+00 9.787e-01 0.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 1.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 2.381e-02 9.762e-01
```