EE 6663 MNIST Homework Report, Task 7

## Team Structure

Jacob English was in charge of Task 1. David Masters worked on Tasks 2 and 4. Nick Leyder was assigned Task 3. Robin Kelby was responsible for Tasks 5 and 6. Everyone worked on the report for Task 7.

## Task Breakdown

Task 1: Jacob English - For the first task, a formatting function was used to merge the NumPy MNIST data set with the classification labels. The formatted data could then be passed to the NumPy functions for calculating the mean, median, and variance of the pixels for each digit. The two functions for this task can be found in ‘mmv.py’ within the GitHub repository.

Task 2: David Masters - I took the median weights for each pixel for each digit, and then normalized the median to get the weights. After testing Robin noticed the mean was more accurate so we replaced the median with the mean.

Task 3: Nick Leyder - For Task 3, I worked with David to write the function to apply the weights based off the input from Task 2. My work can be found in the applyWeights.py as the function applyWeights that takes the weights returned from task 2 and applies them to the samples passed in for testing .

Task 4: David Masters - There were two parts I did for this: generating the bias and classifying.
For each digit I generated the bias to be the offset that made that digit perfectly classified. To do that, for that digit, I set the bias for the corresponding digit to be the smallest bias that made all outputs classify that digit as >=1, and for the rest of the bias for that digit, I set it to be the largest number that made all outputs <= -1. To generate the overall I just averaged the bias from each digit.
To classify the output I just took the argmax of the output from task3 + the bias.

Task 5: Robin Kelby - For Task 5, I coordinated with the other students to incorporate Tasks 1-4 together in one script. My work for Task 5 is contained in the file main.py, which calls the functions written to implement Tasks 1-4. Because of the nature of these tasks and the pipeline for classification, I had to read and understand the code implementing Tasks 1-4 to ensure that the inputs were sent in the proper way.

Task 6: Robin Kelby - For Task 6, I scaled and output many different images to visually show the results. These images are saved to the images/ directory in the Github repo. I also calculate the numbers and percentages of misclassified examples to stdout to calculate the overall accuracy of the model.

Task 7: All - For Task 7, we choose to have everyone write about their individual tasks and have Nick write the results section.

## Development Process

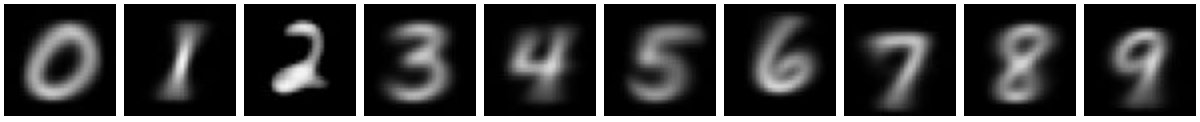
## Results

The model was trained with 60000 digits and tested against 10000 independent digits. Overall using the mean to classify the images had the most success with a classification rate of 82.16%, whereas the median and variance had a total rate of 75.58% and 72.84%. The hardest digit to detect was 5 as it never had a classification rate above 64.47%, an average of 57.32% and is the worst rate for both the mean and median. Digits 0 and 1 were the easiest to detect with the average rates of 91.22% and 89.2% respectively.

| Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mean (%) | 92.05 | 93.84 | 76.46 | 84.16 | 80.80 | 64.47 | 88.21 | 82.50 | 76.19 | 79.89 | 82.16 |
| Median (%) | 94.39 | 81.24 | 67.25 | 81.50 | 69.96 | 44.29 | 81.74 | 77.05 | 78.75 | 76.32 | 75.58 |
| Variance(%) | 81.23 | 92.52 | 72.10 | 72.48 | 50.51 | 63.20 | 83.93 | 82.69 | 48.46 | 77.01 | 72.84 |
| Average(%) | 91.22 | 89.20 | 71.93 | 79.38 | 67.09 | 57.32 | 84.62 | 80.74 | 67.80 | 77.74 | 76.86 |

Classification accuracy for the digits using the mean, median, variance, and then the average.

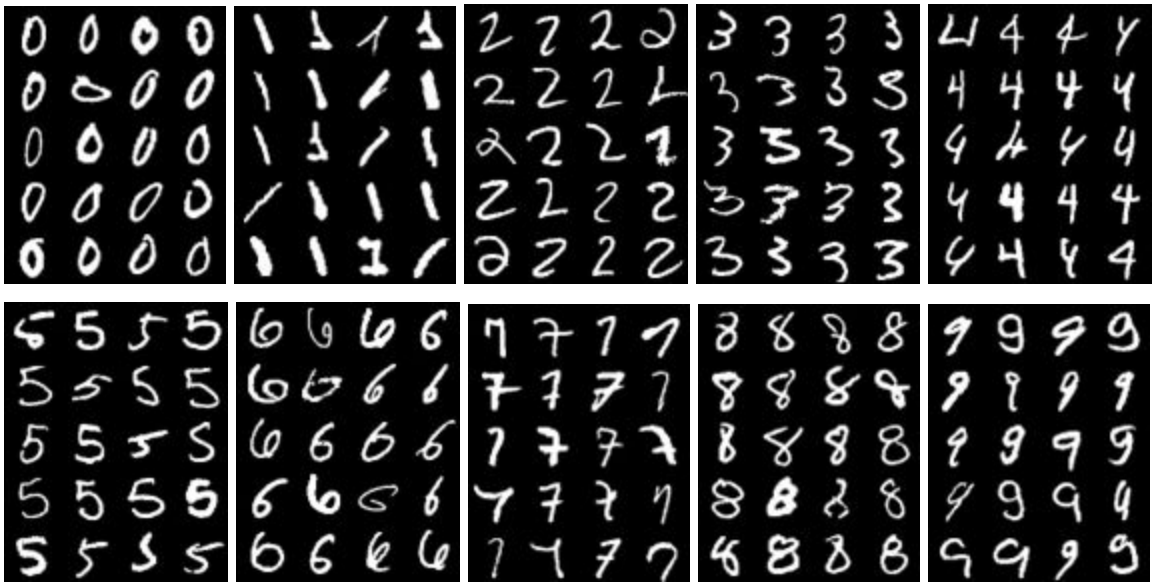
The means calculated from the training set.



The medians created from the training set.

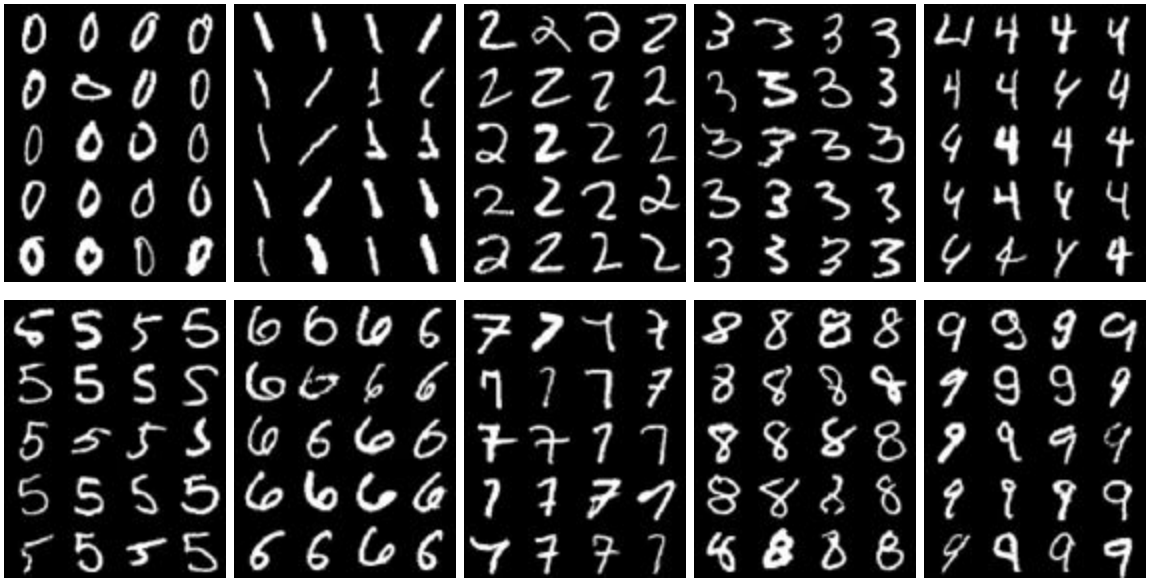
Below are 20 digits that were misclassified by the models. All three models shared some common issues when dealing with incomplete or extra markings on the digits.

Mean



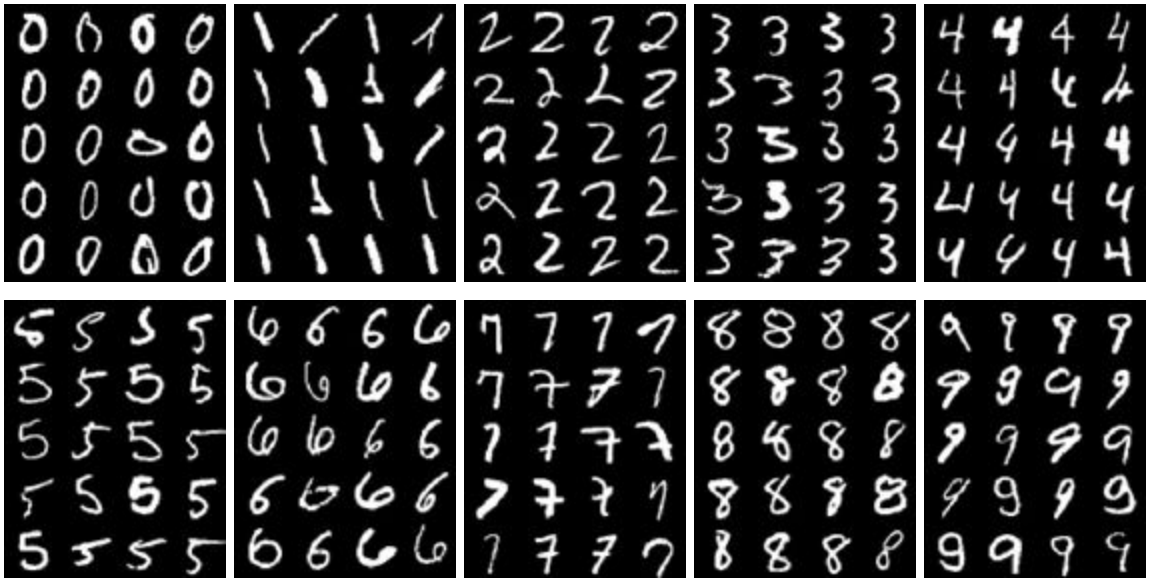
Misclassified digits.

Median



Misclassified digits.

Variance



Misclassified digits.

## Source code

The code can be found at: https://github.com/robinjanette/EE6663_MNIST

Input/Output Format

Preprocessing: MNIST loads as a 60,000x784 matrix for the data and a 60,000 length vector as the label. I think using the labels this should be separated into 10  Nx784 matrices, one Nx784 matrix for each digit. The weights of the MNIST matrix are range 0-255, but I think this should be normalized to range 0-1

Task1: A function that takes in a Nx784 matrix, representing a single digit, and return a 3 tuple of vectors, each with length 784, representing the mean, median, and variance of each pixel for the digit

Task2: A function that takes in a list of 3 tuples/ the list of outputs on all digits of task 1, and returns a 784x10 matrix, representing the weights for the network. Weights are in range -1 to 1

Task3: A function that takes a 784x10 matrix, W, from task2, and a Nx784 matrix, T, that is the entire test set, and returns TxW, which will be Nx10 matrix, where the row represents the test case and the column represents the value of the digit

Task4: A function that takes the output matrix of task3 and the variance matrices from task 1 and returns a vector with each element representing the classification of a test case