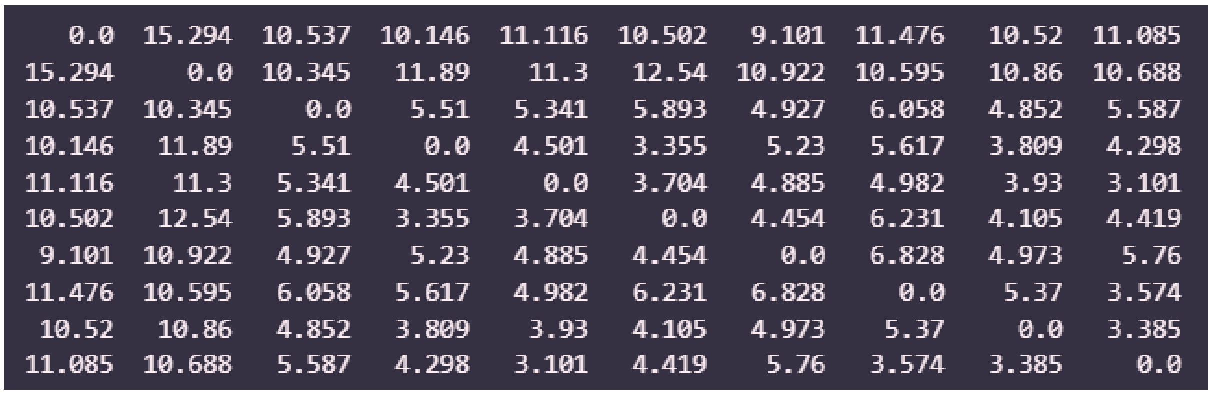
Assignment 0

Introduction
The goal of this assignment was to investigate the accuracy of several algorithms in classifying written numbers to their corresponding digits. The dataset used for this was a simplified version of the MNIST dataset which consists of 2707 handwritten digits represented by 16x16 vectors. For this assignment the dataset was split into a training set of 1707 images and a test set of 1000. Two additional data sets were created which store the class labels of the images, respectively for the train and test set.

Task 1
The aim of this task was to gain insight in the cohesion of the clouds of points in the dataset. To visualize this in the dataset we utilized different dimensionality reduction techniques. Furthermore we developed a relatively simple algorithm to classify the handwritten digits so we could compare it to other classifiers.

1. For this exercise we created a simple algorithm to classify the handwritten digits. We treated every digit d (d - 0, 1, …, 9) as a cloud of points in the 16x16 dimensional space created by all the training images of d. We then calculated the center, which is a 256-dimensional vector of the means of all the vectors of d. With these centers we could classify the test images by calculating the distance between its vector and the centers of the 10 clouds ($dist_{ij}$ - $dist(c_i, c_j)$).

We calculated the distances between the clouds to be the following.



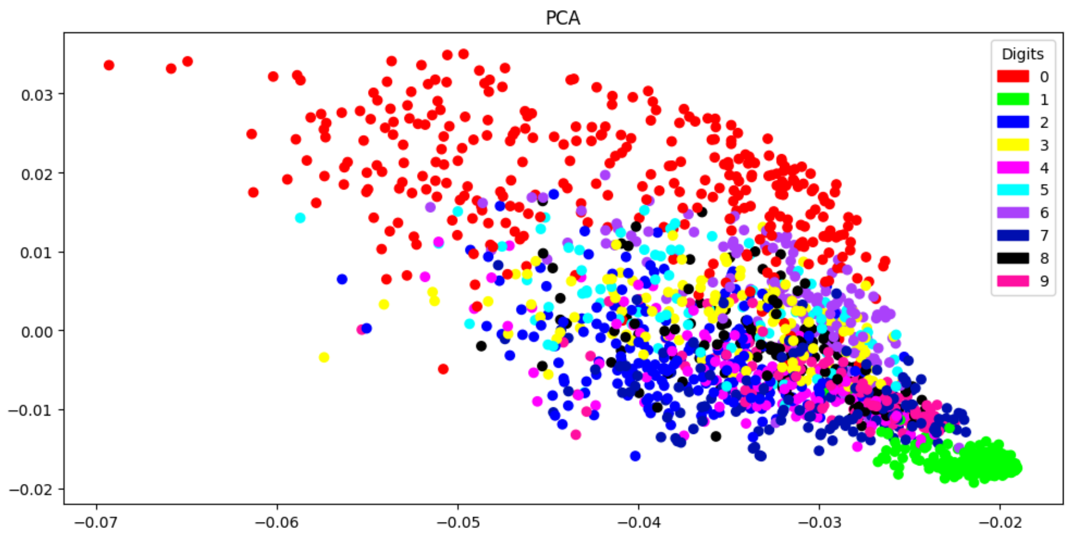
```
   0.0  15.294  10.537  10.146  11.116  10.502   9.101  11.476   10.52  11.085
15.294     0.0  10.345   11.89    11.3   12.54  10.922  10.595   10.86  10.688
10.537  10.345     0.0    5.51   5.341   5.893   4.927   6.058   4.852   5.587
10.146   11.89    5.51     0.0   4.501   3.355    5.23   5.617   3.809   4.298
11.116    11.3   5.341   4.501     0.0   3.704   4.885   4.982    3.93   3.101
10.502   12.54   5.893   3.355   3.704     0.0   4.454   6.231   4.105   4.419
 9.101  10.922   4.927    5.23   4.885   4.454     0.0   6.828   4.973    5.76
11.476  10.595   6.058   5.617   4.982   6.231   6.828     0.0    5.37   3.574
 10.52   10.86   4.852   3.809    3.93   4.105   4.973    5.37     0.0   3.385
11.085  10.688   5.587   4.298   3.101   4.419    5.76   3.574   3.385     0.0
```

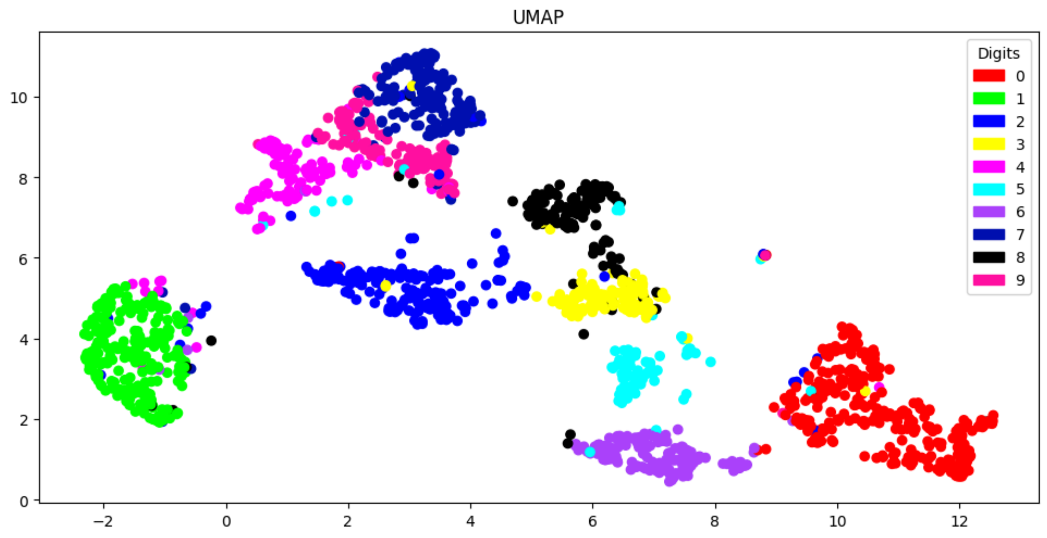
The x- and y-axis both represent the digits 0 to 9.

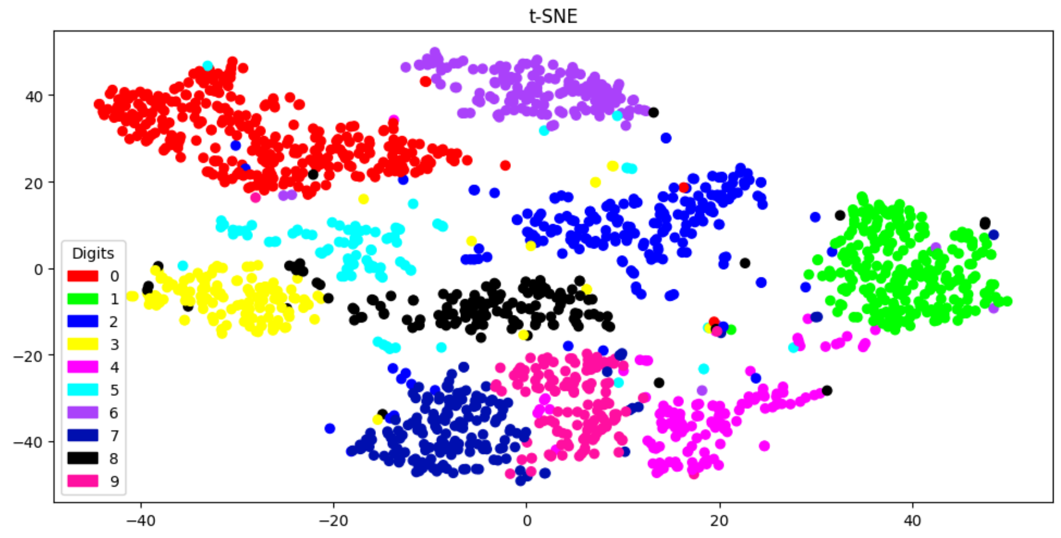
The distances between the centers of the clouds seem to be quite substantial, therefore we expect the accuracy of the classifier to be decent.
As we can see the lowest value in the table (apart from the intercepts) is 3.101. This number represents the distance between the centers of the clouds of the digits 4 and 9. Because of this we expect the digits 4 and 9 to be the most difficult to separate.

2. For this exercise we used three different dimensionality reduction algorithms to visualize the data, namely PCA, U-MAP and T-SNE.

For PCA and T-SNE we used two components as we wanted to visualize the data in 2D. For the PCA we also added a filter where we limited the value of the rows to be equal or smaller than 1 to eliminate the outliers. Below are the resulting visualizations.





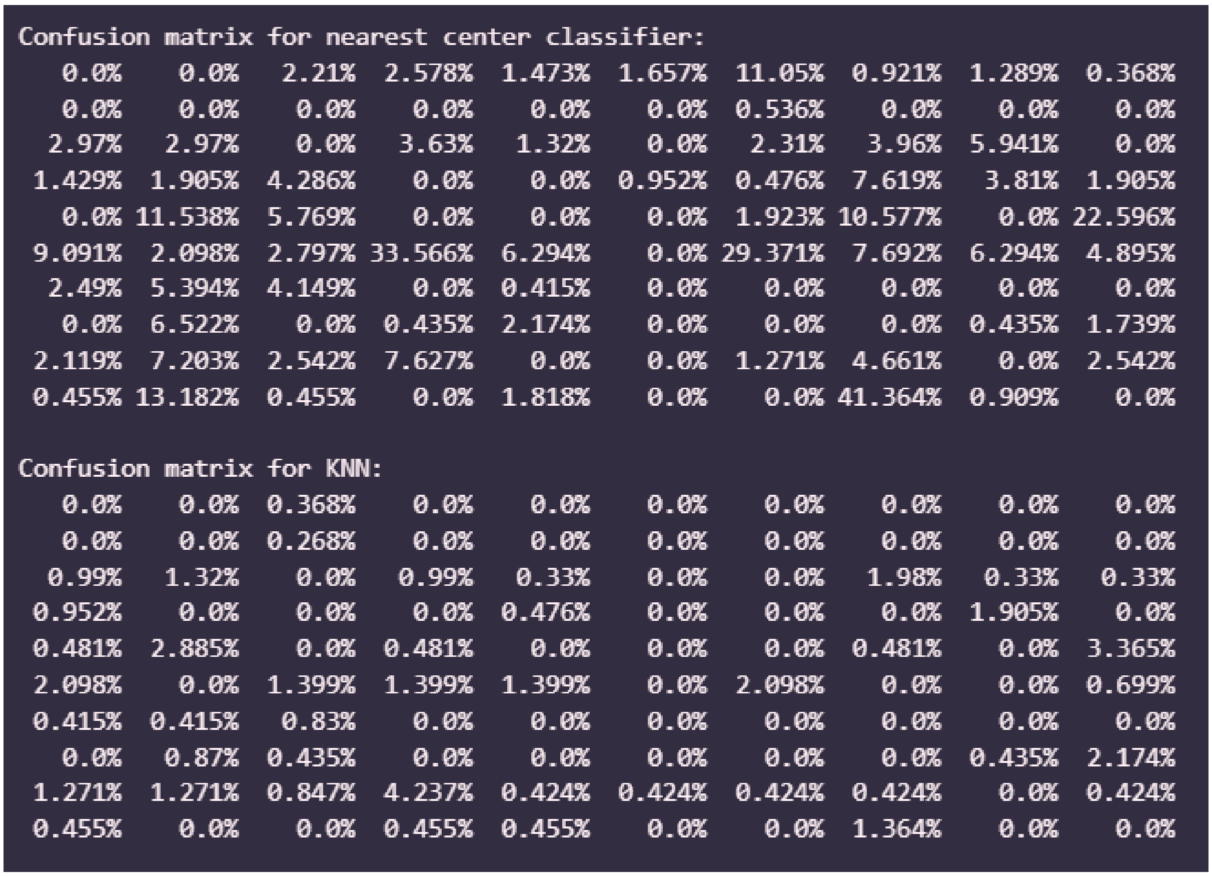


The visualizations agree mostly with what we saw in the between-class distance matrix. For example the largest distance in the matrix is between 0 and 1 which are on opposite sides in all images. The opposite goes for 4 and 9 (the smallest distance) as they, as expected, overlap in all figures.

3. For this task we again used the center of the cloud of each digit and utilized it to implement a *Nearest mean classifier*. We did this by taking the smallest distance from a vector from the set, to the centers of the digits calculated in 1. We then assigned the tested vector to the digit for which the center was the closest. The accuracy was then calculated by calculating the percentage of correctly classified vectors.

This classifier was first applied to all points from the training set. The percentage of correctly classified digits came down to 77.97%. We also tested the classifier on the test set. This accuracy came down to 72.8%.

4. Lastly, we also trained a K-Nearest-Neighbor classifier. The value of k was set to 5 as 3 seemed to be too little for a spread out dataset like this, while 7 might convolute it too much since a lot of the digits get misclassified often.
For the training set this classifier resulted in an accuracy of 94.1% and for the test set the accuracy was 90.7%. Below the confusion matrices from respectively the NMC and KNN classifier are plotted.



```
Confusion matrix for nearest center classifier:
   0.0%    0.0%   2.21%  2.578%  1.473%  1.657%  11.05%  0.921%  1.289%  0.368%
   0.0%    0.0%    0.0%    0.0%    0.0%    0.0%  0.536%    0.0%    0.0%    0.0%
  2.97%   2.97%    0.0%   3.63%   1.32%    0.0%   2.31%   3.96%  5.941%    0.0%
 1.429%  1.905%  4.286%    0.0%    0.0%  0.952%  0.476%  7.619%   3.81%  1.905%
   0.0% 11.538%  5.769%    0.0%    0.0%    0.0%  1.923% 10.577%    0.0% 22.596%
 9.091%  2.098%  2.797% 33.566%  6.294%    0.0% 29.371%  7.692%  6.294%  4.895%
  2.49%  5.394%  4.149%    0.0%  0.415%    0.0%    0.0%    0.0%    0.0%    0.0%
   0.0%  6.522%    0.0%  0.435%  2.174%    0.0%    0.0%    0.0%  0.435%  1.739%
 2.119%  7.203%  2.542%  7.627%    0.0%    0.0%  1.271%  4.661%    0.0%  2.542%
 0.455% 13.182%  0.455%    0.0%  1.818%    0.0%    0.0% 41.364%  0.909%    0.0%

Confusion matrix for KNN:
   0.0%    0.0%  0.368%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%
   0.0%    0.0%  0.268%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%
  0.99%   1.32%    0.0%   0.99%   0.33%    0.0%    0.0%   1.98%   0.33%   0.33%
 0.952%    0.0%    0.0%    0.0%  0.476%    0.0%    0.0%    0.0%  1.905%    0.0%
 0.481%  2.885%    0.0%  0.481%    0.0%    0.0%    0.0%  0.481%    0.0%  3.365%
 2.098%    0.0%  1.399%  1.399%  1.399%    0.0%  2.098%    0.0%    0.0%  0.699%
 0.415%  0.415%   0.83%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%    0.0%
   0.0%   0.87%  0.435%    0.0%    0.0%    0.0%    0.0%    0.0%  0.435%  2.174%
 1.271%  1.271%  0.847%  4.237%  0.424%  0.424%  0.424%  0.424%    0.0%  0.424%
 0.455%    0.0%    0.0%  0.455%  0.455%    0.0%    0.0%  1.364%    0.0%    0.0%
```

NMC: the highest percentage in the matrix is 41.364% which represents 9 getting wrongly classified as a 7.

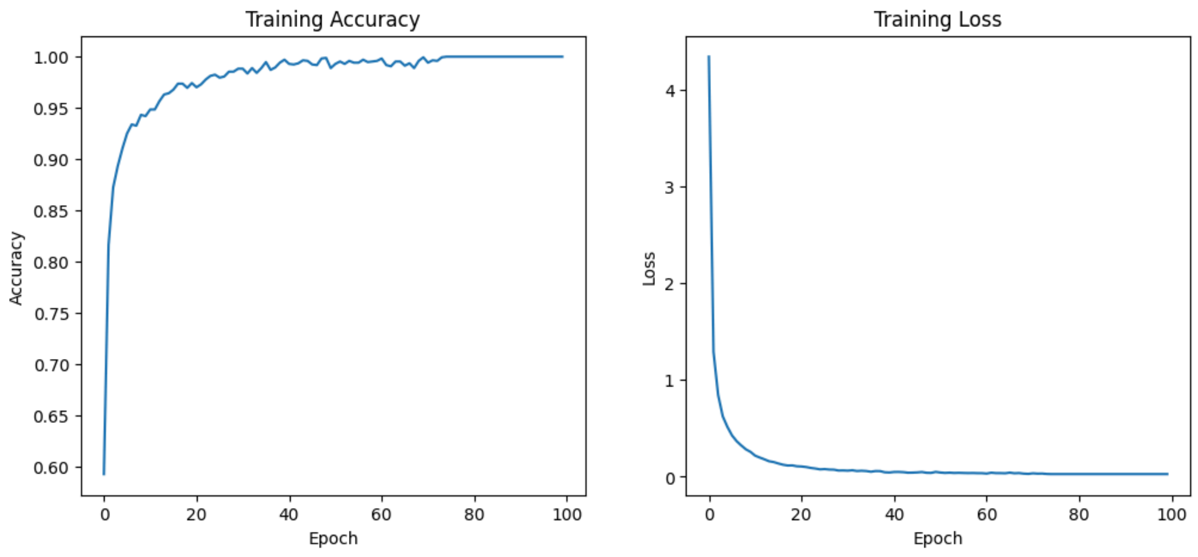
KNN: The highest percentage in the matrix is 4.237% which represents 8 getting wrongly classified as a 3.

Overall we found that the KNN-classifier appears to perform better on both the train and test set. We can also see that the digits 9 and 7, and 8 and 3, for respectively NMC and KNN, seem to be the most difficult to classify correctly. It is interesting to note that this is different than we had expected from the distance matrix in exercise 1.

Task 2

For this task we created a multi-class perceptron training algorithm. This contains 10 independent linear classifiers which each predict whether the digit is part of a certain class.

To do this we used the training set to train the network. Following the model's training cycle, the analysis focuses on the training accuracy, training loss, and final test accuracy. In order to evaluate the network's learning behavior, the model was trained over a predetermined number of epochs, and training accuracy and loss were monitored.



The model's training accuracy increased with each epoch after it had a somewhat low accuracy at first. The training loss also decreased in line with this, suggesting that the model was improving and making fewer mistakes over time. The model attained almost flawless training accuracy by the end of the training process.

After training was finished, a test dataset that wasn't used during training was used to assess the model. With a test accuracy of 0.83, or 83%, the model demonstrated a respectable level of generalization to previously unseen data. However, given the discrepancy between test and training accuracy, which implies the model may have retained some information from the training set, there might still be space for improvement, especially in avoiding overfitting.

When we compare this to the test accuracies in task 1 this is more than the nearest mean approach but less accurate than the KNN classifier. Therefore, the KNN classifier seems to overall be the most accurate method.

<u>Contributions</u>
Rajat: Task 1 code
Saghar: Task 2 code
Hannah: Report