The program structure

Dataloader.py:

- Converts the mnist data into tables

Main.py:

- At the moment it just takes a number from the test data and tests it against the train data using **KNN.py**

KNN.py:

GetErrorPrecentage():

- Returns the percentage of numbers the algorithm fails to recognise

CreatePixelBoolCoordinateNumber()

- Converts an image table into a list of colored pixel coordinates

CreatePixelBoolCordinateTable():

- Uses **CreatePixelBoolCoordinateNumber()** toconvert multiple data entries into lists of colored pixel coordinates.

GetNeighbouringCoordinateRange():

- Returns a list of surrounding coordinates of a coordinate

GetClosestNeighbour():

- Returns the distance of the closest colored coordinate to a given coordinate. Either with the range of **GetNeighbouringCoordinateRange()** or the whole pixel coordinate image.

DistanceToNearestNeighbour():

- Controls the search radius of **GetClosestNeighbour()** to optimize runtime returns the distance to the nearest colored coordinate of a given coordinate

CompareNumberWithBoolCordinates():

- Checks the distance between colored pixels between a given image and images from the test data using **DistanceToNearestNeighbour()**. Then it returns a list containing the index of the training data and the distance, the list is sorted so the smallest distances comes first.

GetTheMajorityNeighbourNumber():

- Gets the closest numbers from the train data using a list form **CompareNumberWithBoolCordinates()**. Then it chooses the number that the majority of the closest numbers are.

Time Complexity

- Converting to bool tables takes O(n*784) time
- Number recognition worst case takes $O(n*784^2)$

The Programs Functionality

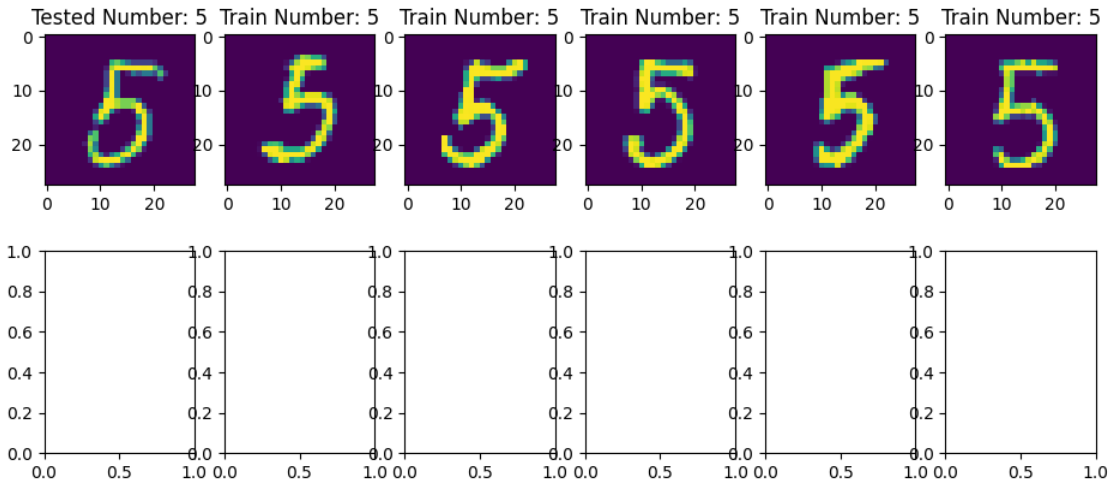
Right now the program can recognize handwritten digits from with an accuracy of (91% succeed,due to me not wanting to spend half a day waiting for the result, i tested only 10 numbers against a training set of 10000 entries wich makes the recognition weaker)

The program output looks like this:

```
Set train data size: 10000
Set search radius: 2
Set K: 5
1 for regular recognition. 2 for error precentage: 1
```

Regular recognition:

```
Generating boolcoordinte table
Booltable conversion time:  17.49900794029236
Testdata index: 45
BoolNumber conversion time:  0.002001047134399414
time elapsed on distance measurments: 293.8802661895752
The number that the algorith recognized 5
The digit tested was a 5
plotting...
```



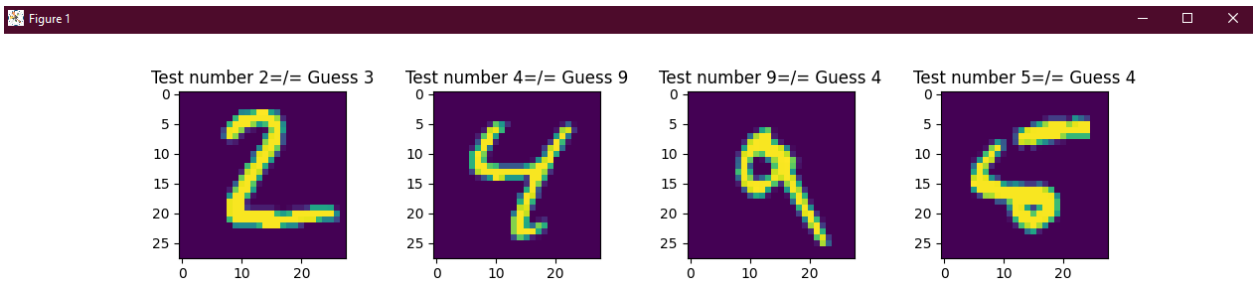
It plots the nearest neighbours and the test number

Error percentage test:

```
Set train data size: 1000
Set search radius: 3
Set K: 7
1 for regular recognition. 2 for error precentage: 2
Set testsample size: 20
Time spent on enrty 0 124.82067966461182
Time spent on enrty 1 200.11228942871094
Time spent on enrty 2 82.73014450073242
Time spent on enrty 3 124.26035523414612
Time spent on enrty 4 129.0278835296631
Time spent on enrty 5 94.9687569141388
Time spent on enrty 6 137.9752128124237
Time spent on enrty 7 145.24754428863525
Time spent on enrty 8 166.68641328811646
Time spent on enrty 9 127.11046361923218
Time spent on enrty 10 200.4140510559082
Time spent on enrty 11 150.889098405838
Time spent on enrty 12 102.49612498283386
Time spent on enrty 13 154.68954706192017
Time spent on enrty 14 103.91040992736816
Time spent on enrty 15 170.04250383377075
Time spent on enrty 16 116.93624496459961
Time spent on enrty 17 160.882173538208
Time spent on enrty 18 220.38625073432922
Time spent on enrty 19 100.85362195968628
0.0
Time elapsed: 2818.0958766937256
```

The 0.0 is the percentage of wrong guesses meaning in this case that it got all of them right.

An example of the plotted data if it guesses incorrectly is shown below.



Things missing

- Optimization can always be done

References:

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=576361

Advice from Hannu, which I'm very thankful for.