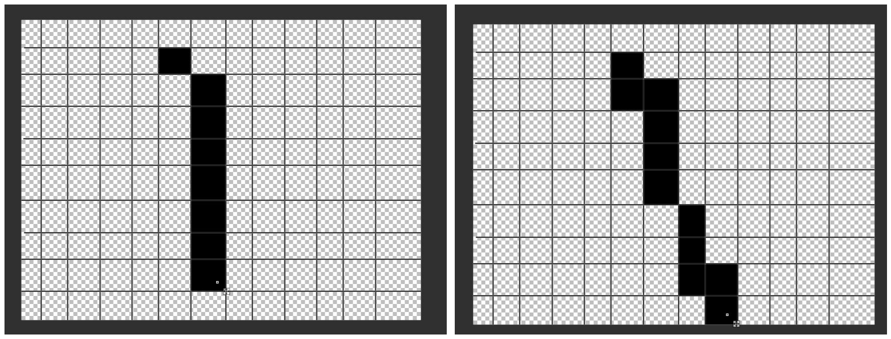
Cody Morgan
CS398

Recognition of Handwritten Numeric Digits

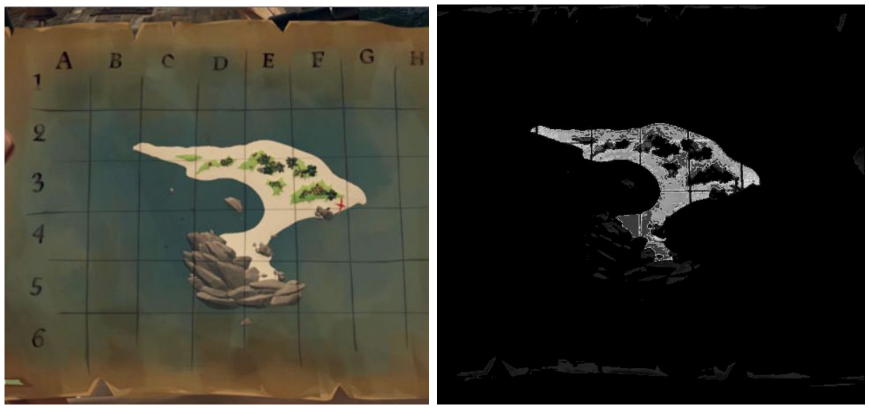
21 Nov 2020

Abstract

This project is intended to recognise handwritten numeric digits given a sample set of such digits. This is a difficult thing to do because humans are inconsistent, even with their own handwriting. For example the same person could write both of these ones:



I worked on an island recognition tool for the game Sea of Thieves this summer. It works based on color and tries to determine the shape of an island given a treasure map. This project was interesting and dealing with images, I feel, is a little more interesting than just data.



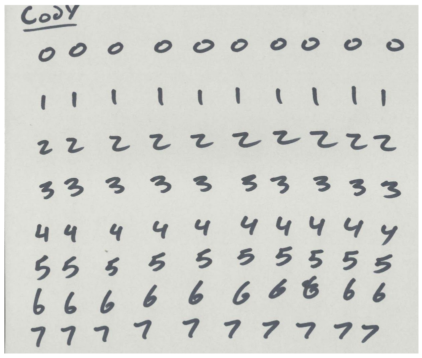
(unprocessed and preprocessed images)

I expect that this project will be similar in preprocessing methods and comparison - though I am hoping for better results by using KNN. First I will load the 5000 count training set (named digits.png). This image is then converted to black and white and cut into individual digits. This ends the preprocessing step. This is pretty common for image processing and in fact matches my steps for the island map matcher program. Pushing the image to black and white can help to eliminate distracting information and focus only on the crux of the issue - in this case the digits. After training, I use my own handwriting to find the best k value (tested 1-100). I also tested the KNN system on the MNIST larger dataset.

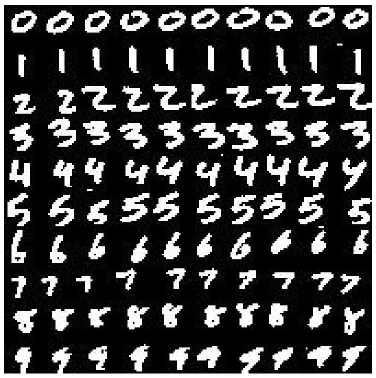
Cody Morgan
CS398
Recognition of Handwritten Numeric Digits
21 Nov 2020

Introduction

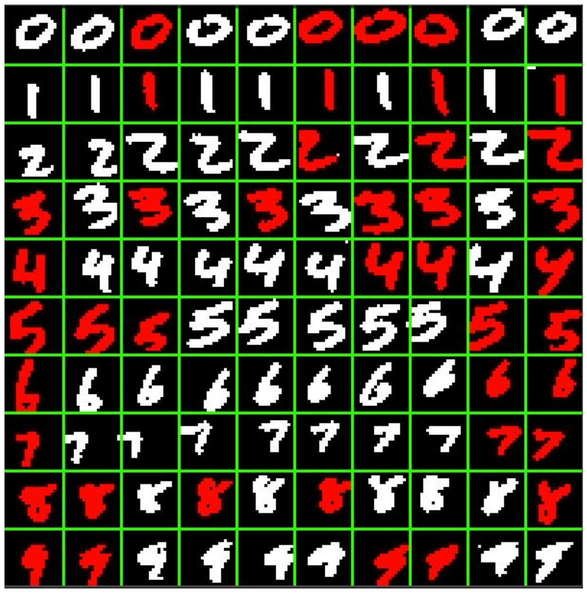
I was curious if I could get this to recognise my own handwriting so I created my own digits sheet



Of course, I had to preprocess this so that each digit matched the MNIST image size (20x20 pix) and push it to black and white. I'm not sure if there is any advantage in having the colors inverted - but I did to match the MNIST data. It shouldn't matter as long as all the data is processed in the same way.



I found that k=14 gave the highest accuracy of 55%. I wanted to see which numbers the system had a difficult time identifying so I marked them in red



It's obvious that some of these digits are badly formed, such as the first red 2 or the first two red 9's. I also noticed that several of the digits were slightly offset by just a few pixels - but enough that I believed it was interfering with the digit recognition. I worked to clean up my dataset a little - though I wanted to keep some of the malformed digits (humans are inconsistent after all). This raised my accuracy up to 88%. I also tested the KNN system on the larger MNIST dataset and found it had an accuracy of 81%.

Literature Review

Author:	Huseyin Kusetogullari
Article title:	ARDIS: a Swedish historical handwritten digit dataset
URL:	https://link.springer.com/article/10.1007/s00521-019-04163-3

Brief:

Huseyin used a different dataset than me but also used KNN. He was able to get 96.94% accuracy. He also discusses other methods such as the random forest technique. This paper is more of a comparison between methods - so it wasn't especially helpful but it is interesting to see what other methods are out there.

Author:	Hussein Moghnieh
Article title:	Scanned Numbers Recognition using k-Nearest Neighbor (k-NN)

URL:
https://towardsdatascience.com/scanned-digits-recognition-using-k-nearest-neighbor-k-nn-d1a1528f0dea
Brief:

He talks about preprocessing the images (which the previous source did not) as well as libraries that can be used to do this. He is taking his data straight from the user (like my handwriting sample). In fact, his method of separating the digits is exactly what I did for my island recognition program. This is a great source that goes into a lot of detail about exactly how to do this using KNN - this was very helpful in gaining an understanding of how KNN works on this kind of data.

Author:	Norhidayu binti Abdul Hamid
Website title:	Arxiv.org
URL:	https://arxiv.org/ftp/arxiv/papers/1702/1702.00723.pdf
Brief:	

This source used the same libraries as I did - these seem to be the de facto choice for this problem. They use k=1, though I'm not sure why, I would think that this would lead to inaccurate data. Similar to other sources, they compare KNN with other methods - such as neural networks. Interestingly, they found that the neural net had more issues with the number 9 than KNN. They perform their KNN a little differently than mine, but it was very informative. I think it's interesting that they use gaussian blur and grayscale instead of black and white. I would think that this would lead to more inaccuracies - but they claim an accuracy of over 99%.

Author:	RAHUL R. TIWARI et al.
Website title:	Iraj.in
URL:	http://www.iraj.in/journal/journal_file/journal_pdf/1-5-139024255920-25.pdf
Brief:	

This paper uses a k range of 3, which seems small to me. Though they do make the point that "increasing values of k caused the accuracy to drop. This may be because when digits are similar to each other, such as 8sand0s, finding additional nearest neighbours may cause the classifier to include other similar-looking but different digits and thus cause performance to drop. It thus seems that the additional computation necessary to use a k-nearest neighbour classifier is not worth the trouble with this problem, at least without additional pre-processing." This paper has shown me that lower K in this case may be better and I endeavoured to find out if that was also true in my case.

Author	Zhongheng Zhang
Article title:	Introduction to machine learning: k-nearest neighbors
URL:	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4916348/
Brief:	

This source uses R instead of python - so the libraries and methods are a little different but he explains the basics of KNN very well and shows it in a non python world. He also mentions that k ties should be broken at random (I chose to prefer lower K values based on the previous source's documentation). This source is more similar to our own implementations as he is doing everything by hand.

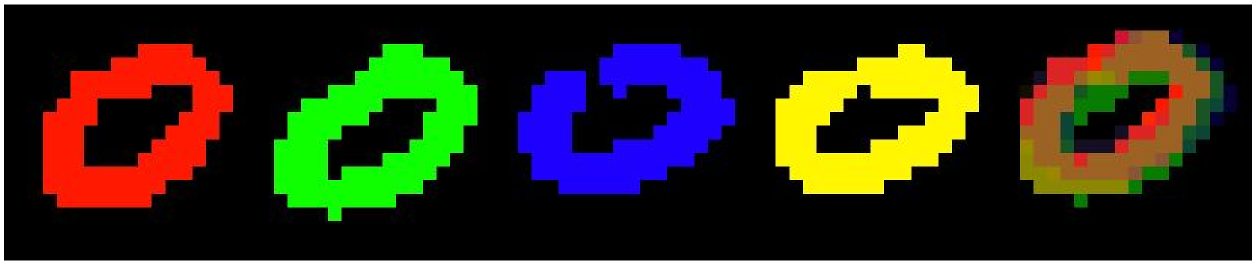
Cody Morgan
CS398
Recognition of Handwritten Numeric Digits
21 Nov 2020

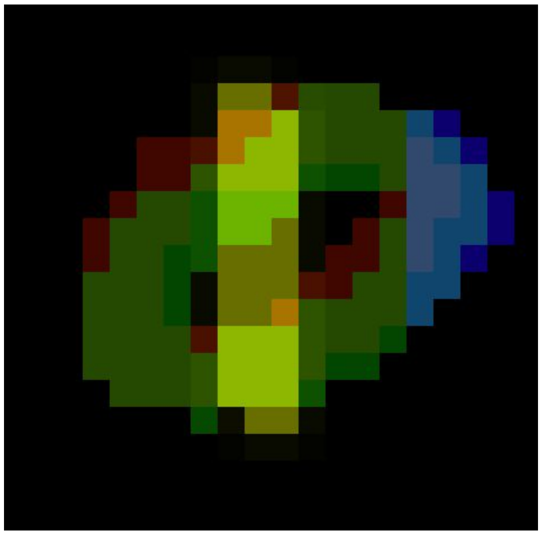
Datasets

I used the NMIST dataset which has already been preprocessed to be a consistent size and color (white on black). I used this dataset for testing, and a smaller subsection of the MNIST for training. I also used my own handwriting for testing.

Methods

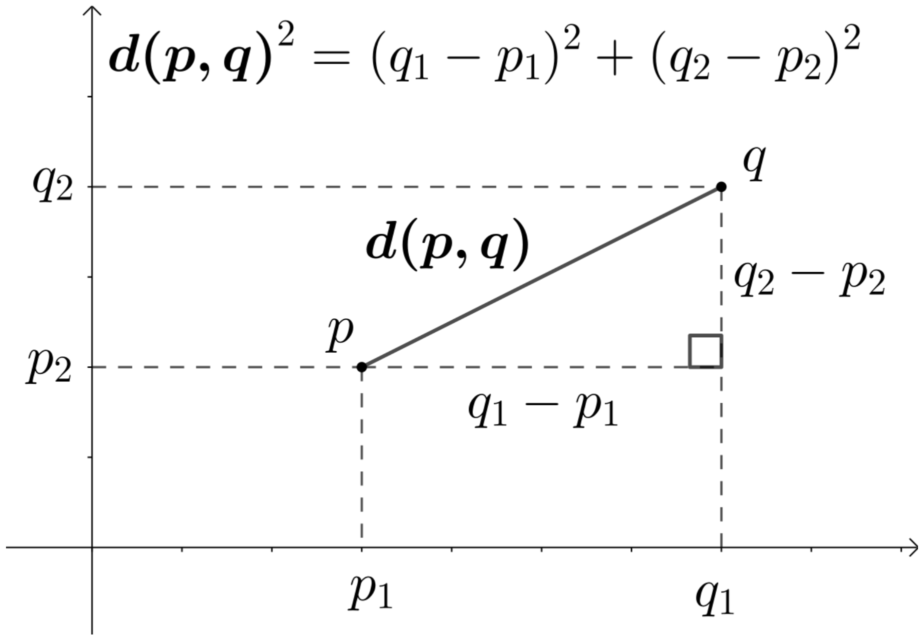
KNN model is trained on digits like those in red, green and blue - then new digits (in yellow) are compared with the training data. You can see that there is a lot of overlap with the training set when compared to other training digits they will have fewer overlapping pixels





(Attempting to match a one with training 0 data)

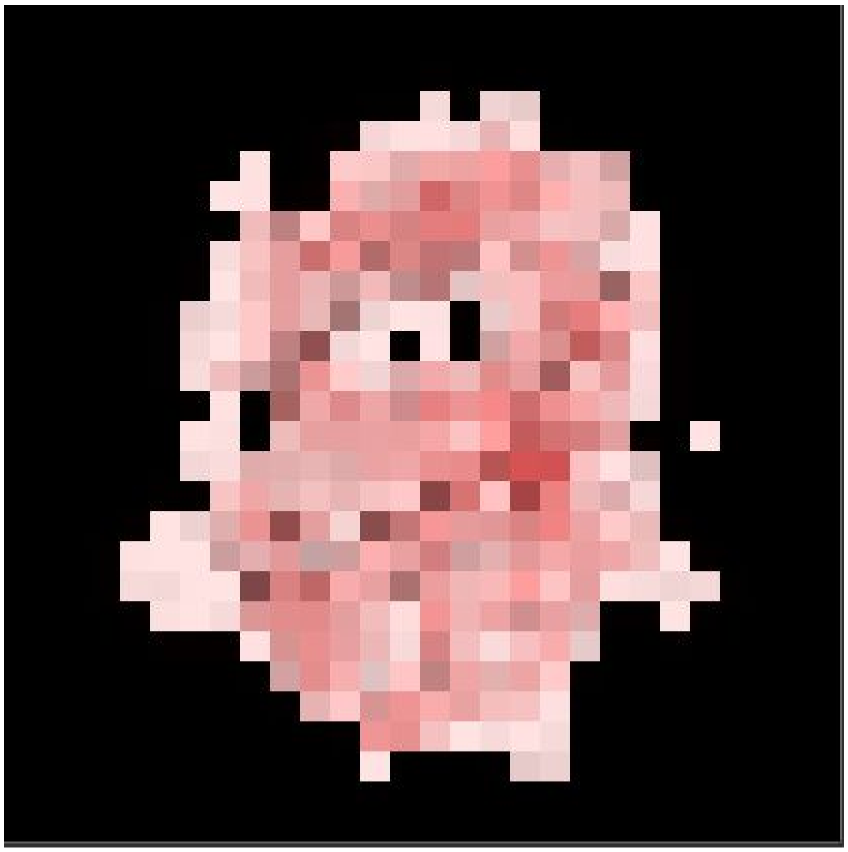
The distance is calculated using Euclidean distance - giving preference to the closest k neighbors.



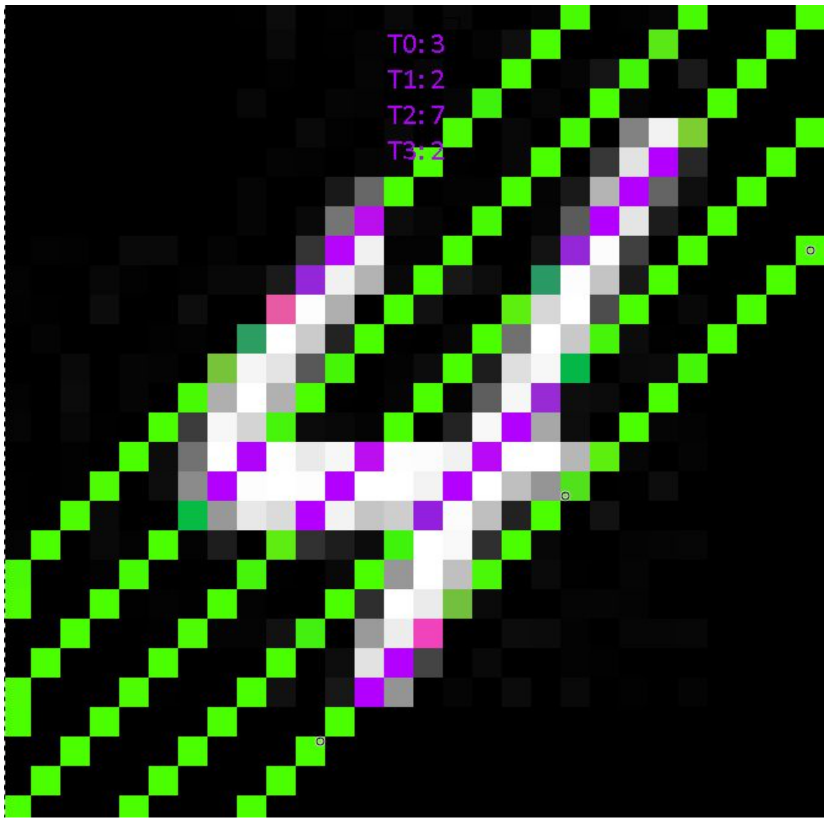
Experimental Result

Experimentally I found that k=14 gave the best accuracy - but after reading other papers I decided to use lower values of k to avoid overgeneralization between digits. I don't think there is overfitting - but it is possible that I could train the model better using more samples - or more diverse samples. Accuracy could also be increased by preprocessing the images more to keep them more centered

Cody Morgan
CS398
Recognition of Handwritten Numeric Digits
21 Nov 2020


(Heatmap of MNIST pixels)

You can see that some numbers are going to be easier than others to identify - 2 and 7 for example share many of the same pixels. This is also why the lower k value makes more sense - we want to go with the closest neighbor and not worry about outliers, because the second and third closest neighbors are probably very similar to the actual digit. One way to fight this would be to take diagonal sampling to help determine digit values.



Instead of comparing each pixel - this would compare diagonal sums. This would give less weight to individual pixels and more weight to the actual shape of the digits

Conclusion

It is possible to recognise handwritten digits using a multitude of methods - in this case we have found that KNN is one such well known solution. By comparing pixels of a training set we are able to identify handwritten digits correctly 88% of the time. There are always improvements to be made or other methods to investigate - but I believe that this would be suitable for non-vital applications such as games.

References

“K-Nearest Neighbours,” GeeksforGeeks, 29-Jul-2019. [Online]. Available: https://www.geeksforgeeks.org/k-nearest-neighbours/. [Accessed: 23-Nov-2020].