Optical Character Recognition: Analysis of SVM and KNN Machine Learning Algorithms

Vasily Shcherbinin – BSc Computer Science, University of Surrey

Abstract—This report aims to provide insight into understanding and developing a useful computer vision system, extracting appropriate features for object recognition/classification, developing classification techniques and evaluating their performance. The proposed solution utilizes the SVM (Support Vector Machine) and KNN (K-Nearest Neighbors) machine-learning algorithms, and this report focuses on the analysis of the effectiveness of each approach. As is discussed further on, each of these approaches has their own advantages and disadvantages when applied to different kind of data – this report includes analysis of the appropriateness of applying aforementioned algorithms to convert images of typed and handwritten content into machine-encoded text, as well as recognize signatures and handwriting script author.

1 Introduction

OPTICAL character recognition (OCR) is an actively researched and still-developing area of applied computer science that finds implementations in all areas of every day life, specifically in anything that requires a conversion of an image containing text into digital format – examples include scanning passport information at the airport, tickets at a concert, converting text-to-speech.

Despite the widespread usage, OCR is far from a solved problem. Recognition of typewritten Latin-script text still does not yield 100% accuracy, with commercial OCR software accuracy varying in the range of 81% to 99% [1]. On the other hand, recognition of hand printed text and cursive handwriting are still areas where active research is performed to this day.

The aim of this report is to contribute to this research, specifically with the idea of using SVM (Support Vector Machine) and KNN (K-Nearest Neighbors) algorithms in order to train the machine for both printed and handwritten notes. The report also looks into experimenting with the classifiers parameters in order to see the effect of such adjustments on the recognition accuracy.

2 Literature Review

Utilizing SVM and KNN algorithms in order to improve character recognition has been attempted before by numerous researchers. Researchers at "Gheorghe Asachi" Technical University of Iaşi were able to record a 90% precision rate when training with sets corresponding to small or capital letters, but only 75% when training the SVM with sets of both small and capital letters [2]. Another researcher, this time one of the developers of OpenCV (Open Source Computer Vision) library, achieved an accuracy of 93.2% on printed text using KNN, and an accuracy of 94% using SVM [3].

Today, there is a consensus between researchers in terms of whether KNN or SVM is better suitable for the task of classifying image data – the prevailing consensus is that SVM generally performs better than KNN, and is therefore more commonly used in text classification problems. This notion is researched further in the work of researchers at the University of Michigan, who have concluded that SVM classifier outperformed KNN by 4%-12% [4].

Additionally, worthy of mention is the conclusion made by Rose Holley, manager of the Australian Newspaper Digitisation Program in 2009 – whilst discussing methods of improving OCR accuracy, the approach of manual intervention in the OCR process for each file to improve results has been claimed to be "not viable for cost-effective mass scale digitization" [1]. Further, in this report, evidence is presented that challenges this conclusion.

2.1 Literature Review - SVM

A Support Vector Machine (SVM) is a supervised machine-learning algorithm used mostly for classification purposes. SVM's are based on the idea of finding a hyperplane that best divides a dataset into two classes. The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly [5]. Pros and cons in terms of application to current work listed further in ***(Appendix 1).***

2.2 Literature Review - KNN

KNN is a fast classifier that falls under "supervised learning" category of algorithms. Being also non-parametric and instance-based, KNN avoids mismodeling the underlying distribution of data, as well as avoids learning any model at all, memorizing training instances instead [6].
In order to measure distances between a given sample and dataset, the algorithm utilizes the Euclidean distance, although other distance metrics can be used depending on the task [7]. The output depends on the k parameter, which represents the "k"-closest neighbor. For text recognition, k is recommended to be greater than 1 (one) (otherwise overfitting occurs), and smaller than 5 (five). Pros and cons in terms of application to current work listed further in ***(Appendix 1).***

3 Computer-Typed Text Recognition

3.1 Problem Description

Similarly to how a human eye functions in order to recognize a scene or object, so is the approach for a machine to learn to recognize objects – in our case, letters of a text. The goal is to produce a system capable of recognizing typed fonts with a high degree of certainty.

This process can be separated into four stages:

1. Feature Detection
2. Feature Extraction & Representation
3. Histogram Computation
4. Classification

There are various ways in which the above steps can be implemented, each suitable for its own purpose and with a common aim of improving accuracy of character recognition and overcoming the technical issues that might arise during the recognition process.

3.2 Technical Issues and Challenges

Despite numerous techniques developed for detecting/extracting features and classifying represented objects, object recognition remains a challenging task for numerous reasons. Challenges include forming different object categories (even if languages belong to the same writing system, alterations to characters in alphabets may occur – e.g. French characters É, â, æ, and others), ensuring accuracy of recognition regardless of the lighting conditions of the document (e.g. if analyzing a printed out script on paper), as well as excluding background clutter in order to retrieve text from a page with an image as background.

The suggested approach for feature detection and extraction utilized a concept of a bounding box, which would be drawn around a detected letter. Since punctuation was not included when the machine was trained and tested (by means of ignoring bounding boxes of a size smaller than particular area), problems occurred as letter *i* and *j* were not represented fully – the letter superscript dot would not be included together with the letters glyph. Additionally, discriminating between capital and small letters can be an issue (letters like c/C, s/S, v/V, w/W), as in some fonts the letter shapes are very similar, if not same – if a living person can at times make a mistake when filling in CAPTCHAS (Completely Automated Public Turing test to tell Computer and Humans Apart), it is understandable if the machine makes similar mistakes itself (depending on the way it was trained). Same can be said about other letters depending on font – at times, the letter for 1 (one) and l (small L) can look very similar, as well as some other examples of similar type.

Finally, recognizing spaces and newlines was a challenging task on a personal level.
Striving to minimize the amount of challenges and technical issues that would need to be solved in order to acquire results for further analysis, a decision was made to utilize PNG (Portable Network Graphics) format images with text written in a clear font on a white background as training data. The language of the text was chosen to be English ***(Appendix 2)***.

3.3 Approach to Development

1. Read in .PNG file containing training data.
2. Apply Otsu threshold in order to make the image binary and separate foreground pixels from background pixels.
3. Label image regions.
4. Draw bounding boxes, skipping bounding boxes smaller than of particular size. This is done in order to avoid punctuation (if it is present in the training data).
5. Sort detected characters, iterating through each bounding box one by one.
6. Extract characters by using numpy-slicing capabilities in order to extract specific area of document containing the letter. In order to extract the *i* and *j* superscript dot, the bounding box boundaries were extended 10 pixels upwards, 5 pixels downwards and 2 pixels to each side. For this approach to work, the training and testing image should be resized to a predefined standard size (around 500x600 pixels). Admitedly, this is not the neatest solution, as a properly done implementation would evaluate whether the previous bounding boxes and the current bounding boxes have overlapping upper and lower boundaries and, if that was the case, then merge them into one big box. The utilized approach unnecessarily creates padding around letters that do not have a superscript dot, and although that does not have a crucial role on the result, this aspect of the implementation can definetly be improved and adjusted. In addition, capital letters are extracted separate from lower-case letters.
7. Resize resulting image to 20x20 size in order to standardize the data.
8. Steps 1-7 are then repeated with testing image that contains data not present in the training data. Additionally, rather crudely, spaces and newlines in the text are recorded – if the distance between last bounding box right edge and the next bounding box left edge is larger than given number of pixels, consider this as a newline, which is "represented" as an image object itself. Similar approach is undertaken with spaces – if the distance between two letters is in a given range of pixels, but smaller than of newline – record a space by slicing a 1 by 1 extract from original image, which after resizing would look like a black square of size 20x20. Refer to ***Appendix 3*** to view the result after extraction is completed from a test text.

9. Save images to local storage into correct folders for easy visualization.
10. Because the machine is not trained to recognize "(" and ")", manual deletion of these character after extraction from the testing file is recommended, as they are considered as letters otherwise and lower the system accuracy. This step is especially crucial if during the evaluation step at the end one is to compare the resulting text with the original text – keeping the brackets in resulting text, but removing them from the original text (together with the punctuation) can cause the system to give unexpected results due to character shift.
11. Next step is to create the data features and assign them the correct labels. In this work, HOG (Histogram of Orientations) features were used, with the HOG parameters being altered in order to produce best possible result (to be analyzed). The data is then trained using either a Linear SVC classifier, a KNN classifier or the VotingClassifier (soft Voting/Majority Rule classifier for unfitted estimators) that aims to combine the two aforementioned algorithms together.
12. The images of letters of the testing image are then loaded and the classifier predicts what each of them is. This is then printed out in the console and added to a temporary list for later comparison with the documents original text in order to see the percentage that was detected correct. Result can be seen in ***Appendix 4***. [1]

3.4 Result Evaluation

During the success evaluation phase, several trends have become apparent. The following variables have been adjusted in order to visualize the effect of such adjustments on the accuracy of the system's retrieval: orientations and number of pixels per cell in the hog features for SVM and KNN algorithms, the number of neighbors in the KNN algorithm, as well as the image size - a hypothesis was established that with an increase in size of the stored image the accuracy of the system would also increase.

The total number of training data amounted to 61*80+28+27 = 4935 letter images, with the testing data amounting to 1555 images.

Effect from variable adjustment can be seen in ***Appendix 7***. As can be seen, for this particular document (the document used for testing was the NVIDIA document provided), the SVC performed well on all settings, although it can be clearly seen that the result is best when utilizing 6-8 orientations with 6 pixels per cell – on the other hand, using 12 orientations yielded worse results on nearly all settings. The best result with using SVC without any post-adjustments to the document yielded an accuracy of 98% - a very respectable result even by today's standards.

On the other hand, utilizing KNN for classification gave a different result (***Appendix 8***). As can be seen from the graph, KNN failed to perform well on 12 orientations, but did very well otherwise, giving the best accuracy for 6-8 orientations at 6 pixels per cell – 98.2%. This is a good result not far from the one achieved by using SVM, but if one looks at the average accuracy obtained it becomes clear that SVM would be the preferable choice between the two. This result is backed up by other researchers who came to similar conclusions, as is discussed in the Literature Review section.

A surprising result arised whilst analyzing the effect of the number of K-Neighbors on the accuracy of retrieval using the KNN algorithm. As can be seen from ***Appendix 9***, with the HOG feature parameters being constant, KNN yields a very surprising result of 99% retrieval accuracy when K=2. Otherwise, at K=3, K=4 the accuracy of retrieval is around 98.2%, which is on-par with the result achieved by SVM. Although this fits with the common agreement that for text extraction K<=5 is the best approach, it was nonetheless surprising to see such high accuracy.

Finally, a test was conducted in order to determine whether the size of the extracted letter images had an effect on the retrieval accuracy. With the HOG feature parameters remaining constant, value for KNN remaining constant and only the image dimension being the dependent variable, it is possible to see from ***Appendix 10*** that the different image dimensions have an effect on the accuracy – the accuracy seems to follow a sinusoid pattern, first deterioting (20x20 to 40x40), improving (40x40 to 50x50), and deteriorating once more (50x50 to 60x60). This behavior repeats periodically, although the period interval is not constant. The best accuracy was yielded for image dimensions of size 50x50, yielding 99.2%, 98.6% and 98.6% for SVM, KNN and VotingClassifier respectively.

3.5 Post-processing as a mean of improving OCR accuracy

One of the ways to improve final accuracy of retrieval that was looked into is to first convert the resulting text into lower case letters, and only then compare it with the original text (also converted to lower letters before hand). This solved the issue with recognizing the difference between capital and small letters, and yielded a 99.9% similarity with lower-case version of original text. In situations where the letter-case is not important and rather the content retrieval is crucial, this could be a good post-processing method in order to improve the accuracy of retrieval – scanning car license plates, ticket id's and even passport details could all be technically done in lowercase – even if the machine would make a mistake in determining whether the letter is a capital or not, this would not turn out to be crucial and would not have any serious consequences (***Appendix 5***).

[1] *Please refer to the code supplied in .ZIP format together with this report for a better understanding of the approach to development.*

4 Handwriting Text Recognition

4.1 Problem Description

The goal is to be able to train the system to recognize handwritten text with a degree of confidence that would allow a human reader to understand the extracted digitalized text.
The approach to solving this problem would be similar as the one for solving the problem of recognizing printed pages, with minor differences that occur from the challenges present uniquely in this problem.

4.2 Technical Issues and Challenges

Handwriting is not standardized – the letter slant, the letter loops as well as other aspects of writing are not constant and vary from document to document, even if those texts were written by the same person. In handwritten texts, the machine had problems in perceiving "F" and "*F*" as the same letter due to the slant, returning "3F" as output result. Attempting to tackle this problem, the machine was trained using as much varied data as possible – this included letters with different slant and loops, allowing for a variance between each letter and allowing the system to be more accurate in its prediction. Additionally, cursive inter-connected letters are very hard to separate, and therefore the training and testing set handwritten letters were written with gaps between them.

4.3 Approach to Development

The approach to development resembled closely the approach undertaken for solving the problem of extracting computer-typed letters. Coincidentally, JPEG images were used instead of PNG, although this should not have had any effect on the accuracy of the system. As explained earlier, one of the difference in the development approach was the attempt to provide the classifier with more varied training data in order to improve recognition accuracy. Training data can be viewed in ***Appendix 6***.

The training dataset contained 61*16+5+5=986 training letters.

4.4 Result Evaluation

The retrieval accuracy for handwritten notes was tested using two short texts (document titled "Man" was written quite loosely and messy, whilst document titled "Work" was written out carefully and neatly), a page containing written out alphabet ***(Appendix 6)*** and two classifiers – aforementioned SVM and KNN. The results have made it apparent that KNN is not suitable for the task of recognizing handwritten notes, whilst SVM showed a respectable result, albeit not perfect.

Testing on the alphabet data yielded an adequate result. For HOG parameters at 6 orientations and 5 pixels per cell, KNN gave a result of 88.5% at K=5 and 83.7 for K=2, whilst SVM returned an accuracy result of 96.7%.

Problems started arising with the accuracy analysis of the extractions of the texts – as can be seen from the parameter input analysis of the crudely written text ***Appendix 13 (2nd graph)***, SVC for various variations gave an accuracy range from 75% down to 8.6%, although maintaining the trend of having a local maximum for all the orientations at 4 pixels per cell, and a local minimum at 7 pixels per cell, increasing with the increase of pixels per cell afterwards.
Similar can be seen for the same experiment but this time for the neatly written text (***Appendix 13 (1st graph)***) – this time local maximum peaked at 5 pixels per cell, giving best accuracy of retrieval at 87.5%, and falling to a minimum of 20% at pixels per cell value of 7.

The machine produced rather unsurprising results regarding the use of KNN in order to distinguish handwritten text – although yielding acceptable accuracy percentages (70%-87.5%) for the neatly written "Work" document ***(Appendix 14 (1st graph))*** for low value of orientation and low value for pixel per cell, it is straight away visible that SVM performed better on average. Even poorer was the accuracy of recognition for the "Man" document, with peak value at mere 55.6% and the majority of results being below the 30% mark for all three orientations ***(Appendix 14 (2nd graph))***. Interestingly, for both KNN and SVC the pixel per cell value of 7 (seven) dramatically decreases the accuracy of results to around 20%-30%.

Lastly, a test was conducted as to see the relationship between KNN retrieval accuracy and the value of K. For both documents, the resulting graph is very similar – with the increase of K, the percentage of correctly retrieved letters deteriorates, as would be expected. For the "Work" document, best KNN value equaled 87.5% at K = 5, whilst for the "Man" document –55.6% at K = 3, 4 and 5 ***(Appendix 15)***.

One possible method to enhance the machines accuracy is to use Neural Networks, which are effective in recognizing patterns and letters due to their unique approach to training. Usually, the output from the network is fed into a post-processing stage that uses knowledge about the neural network and the presented problem in order to generate very good recognition accuracy. Alternatively, optimizing image processing to keep the images original features as much as possible and looking into better segmentation techniques carries potential for the systems improvement.

5 Signature Recognition

5.1 Problem Description

Signature recognition is currently more widely implemented in the industry than the aforementioned handwriting recognition – signature recognizing software and hardware are present in banks, mail offices, as well as in many other locations of public importance.

Signature recognition can be seen as a simplified version of handwriting recognition, as with the signature the task is to recognize a single character and connect it with its owner. In the case of the implemented system, the machine must be able to recognize and distinguish between two different signatures, naming the owner when presented one during the test phase.

5.2 Technical Issues and Challenges

The challenges for signature recognition are similar to those encountered in the problem of handwritten character recognition – signatures are by no means stable (despite the idea of them being constant, as with any handwritten characters variance is expected to occur), and therefore the machine must be trained with a wide range of data in order to be able to produce a correct result. Additionally, it is possible that problems would occur with the recognition of signatures that are long or have many detailed elements like underlining, dashes, complex loops etc. Nonetheless, this issue could be neutralized by increasing the size of the extracted image (in order not to cluster many small details into small 20x20 image, making recognition difficult).

5.3 Approach to Development

In order to identify the signature within the image, we follow the same process as before – apply threshold in order to discriminate between background and foreground pixels, specify the borders of the bouding box and compute each signature as one separate box.

The training process is also the same as mentioned before, with the exception of sorting the extracted signatures not by letter labels, but by the signature owner. The training data was provided to be as varied as possible in order to accommodate for the human error when writing the signature – training process included training with both neat and messy signatures, as well as different slants.

All together, training set contained 22 signatures.

5.4 Result Evaluation

The initial hypothesis was that the result would reflect similar ideas as those seen in handwriting recognition. This was seen to be the case – SVM was more accurate in its recognition, often returning a 100% accuracy result. Using SVM the machine had absolutely no problem in recognizing and distinguishing between two absolutely separate and uncommon signatures. As mentioned before, by pure coincidence both used signatures were not very long and complex (***Appendix 11***) – possible additional tests of the machine with more complex signatures could reveal flaws within the system. A possible improvement to the system would be to test the accuracy of retrieval with Neural Networks. Otherwise, one can conclude that in making a choice between utilizing SVM or KNN for the task of recognizing handwritten notes, SVM would be the clear favourite.

6 Writer Identification System

6.1 Problem Description

Creating a writer identification system combines within itself elements of handwriting and signature extraction and identification. Essentially, the goal is to teach the machine to identify the author of a handwritten text.

6.2 Technical Issues and Challenges

The challenges for author recognition from handwritten notes are similar to those encountered in the problem of handwritten character recognition. Another challenge – if two authors have a similar handwriting, distinguishing between them becomes times more difficult. Further on, distinguishing between authors of non-Latin texts, like Chinese, Hindu or Arabic texts poses an even greater challenge, as letters in those writing systems follow a strict structure and don't have room for much author creativity that could later be used as features to distinguish between the two authors.

6.3 Approach to Development

Similar as was done with signature, handwritten letters are divided and labeled by authors rather by the actual letter label. After this, SVM is used to extract each letter and predict to whom it belongs. In order to not rely on 100% accuracy for each letter, a majority approach was utilized – if the majority of letters belongs to a particular author, a conclusion would be made that that author wrote the letters ***(Appendix 12)***. SVM was chosen as previous tests have shown that handwriting characters are better analyzed with SVM rather than KNN.

6.4 Result Evaluation

Varying orientations and pixels per cell rendered peculiar results for pixels per cell > 5 – suspection falls that overfitting occurs and therefore the result always indicates just one of the two authors, regardless of the text being tested. For using majority to identify the author, it was found that using 6 orientations and 4 pixels per cell was the best choice of parameter in order to identify the author (***Appendix 16***).

7 Conclusion

By developing a useful computer vision system based on two different classifiers SVM and KNN, it has become evident that these algorithms both have their own applications and purposes in the world of OCR, but SVM being the clear favourite in recognizing handwritten notes and signatures, whilst for printed text the two classifiers both have achieved and shown to be reliable and viable algorithms to utilize in the development of a computer vision system.

8 REFERENCES

[1] Holley, R. (2009). How Good Can It Get?. *D-Lib Magazine*, [online] 15(3/4). Available at: http://www.dlib.org/dlib/march09/holley/03holley.html [Accessed 6 May 2017].

[2] TĂUTU, E. and LEON, F. (2012). OPTICAL CHARACTER RECOGNITION SYSTEM USING SUPPORT VECTOR MACHINES. *BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI*, [online] LVIII(2), p.42. Available at: http://www12.tuiasi.ro/users/103/F2_2012_2_Tautu.pdf [Accessed 6 May 2017].

[3] Docs.opencv.org. (n.d.). *OpenCV: OCR of Hand-written Data using kNN*. [online] Available at: http://docs.opencv.org/trunk/d8/d4b/tutorial_py_knn_opencv.html [Accessed 6 May 2017].

[4] KIM, J., KIM, B. and SAVARESE, S. (n.d.). Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines. [online] Available at: http://www.wseas.us/e-library/conferences/2012/CambridgeUSA/MATHCC/MATHCC-18.pdf [Accessed 6 May 2017].

[5] Bambrick, N. (2016). *Support Vector Machines: A Simple Explanation*. [online] Kdnuggets.com. Available at: http://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html [Accessed 7 May 2017].

[6] Zakka, K. (2016). *A Complete Guide to K-Nearest-Neighbors with Applications in Python and R*. [online] Kevinzakka.github.io. Available at: https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/ [Accessed 7 May 2017].

[7] Hu, L., Huang, M., Ke, S. and Tsai, C. (2016). The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus*, 5(1).

[8] Ray, M. (2012). *Nearest Neighbours: Pros and Cons*. [online] Www2.cs.man.ac.uk. Available at: http://www2.cs.man.ac.uk/~raym8/comp37212/main/node264.html [Accessed 7 May 2017].

9 APPENDIX

Appendix 1 – SVM and KNN Pros and Cons

SVM Pros and Cons [5]

Pros

- Provides a high level of accuracy
- Produces good results on small clean datasets.

Cons

- Not suited for large datasets due to training time.
- Uneffective on noisy datasets with overlapping classes.

KNN Pros and Cons [8]

Pros

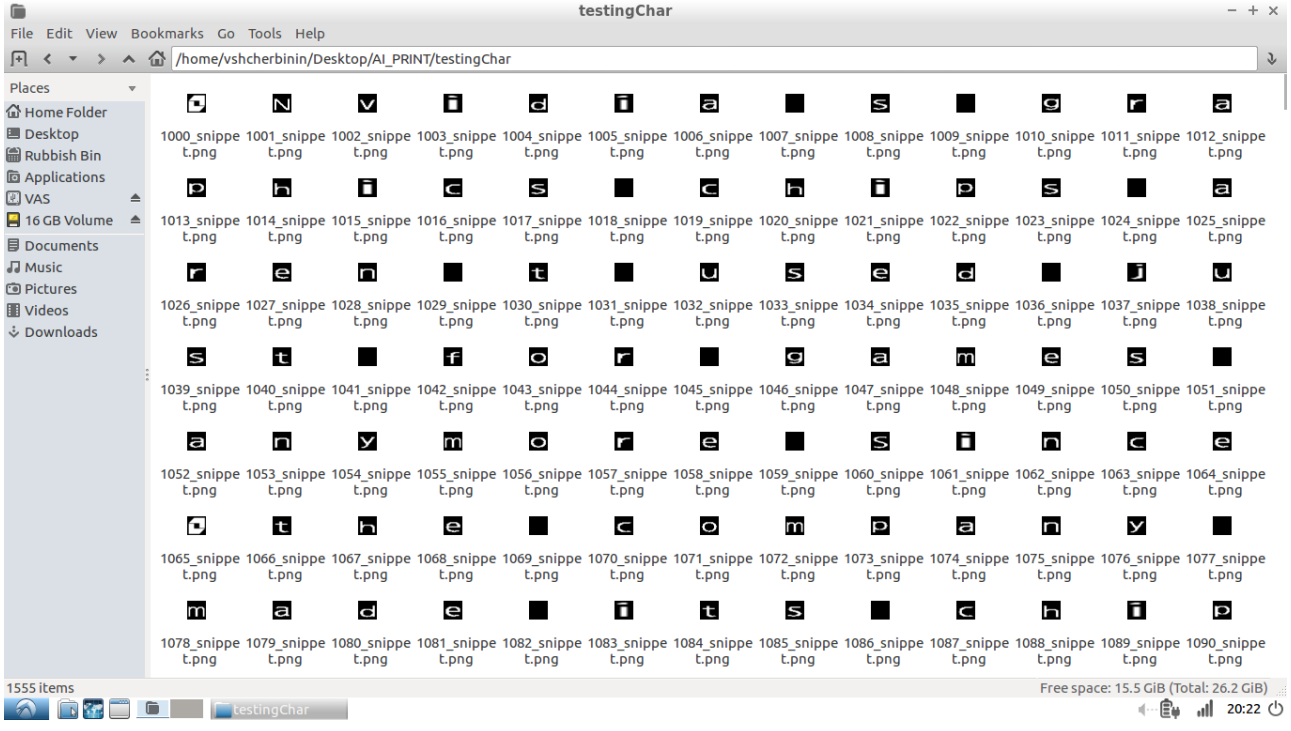
- Implementation is simple
- Flexible to feature / distance choices
- Multi-class cases are well handled
- Given sufficient representation data can do well in practice

Cons

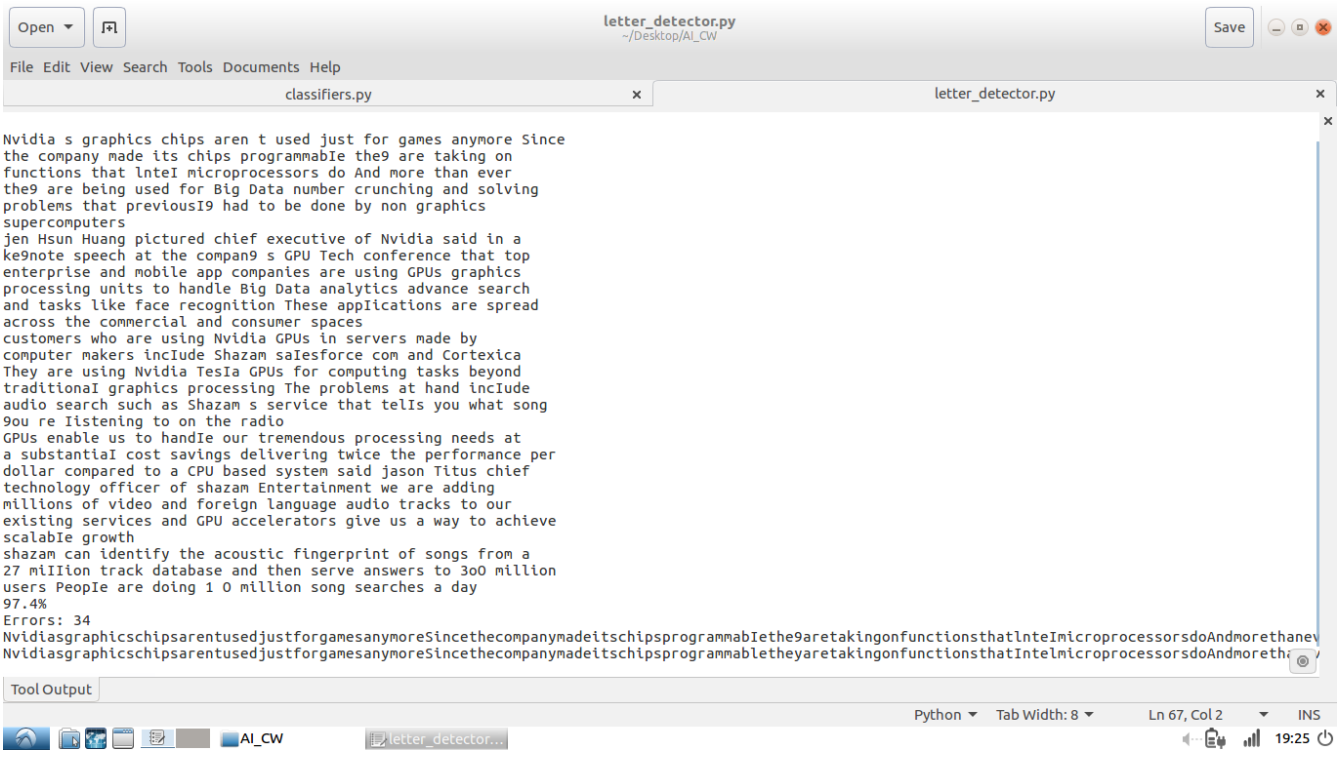
- Commends a large search problem to find nearest neighbor.
- Must have a meaningful distance function.

Appendix 2 – Computer-Typed Training Data

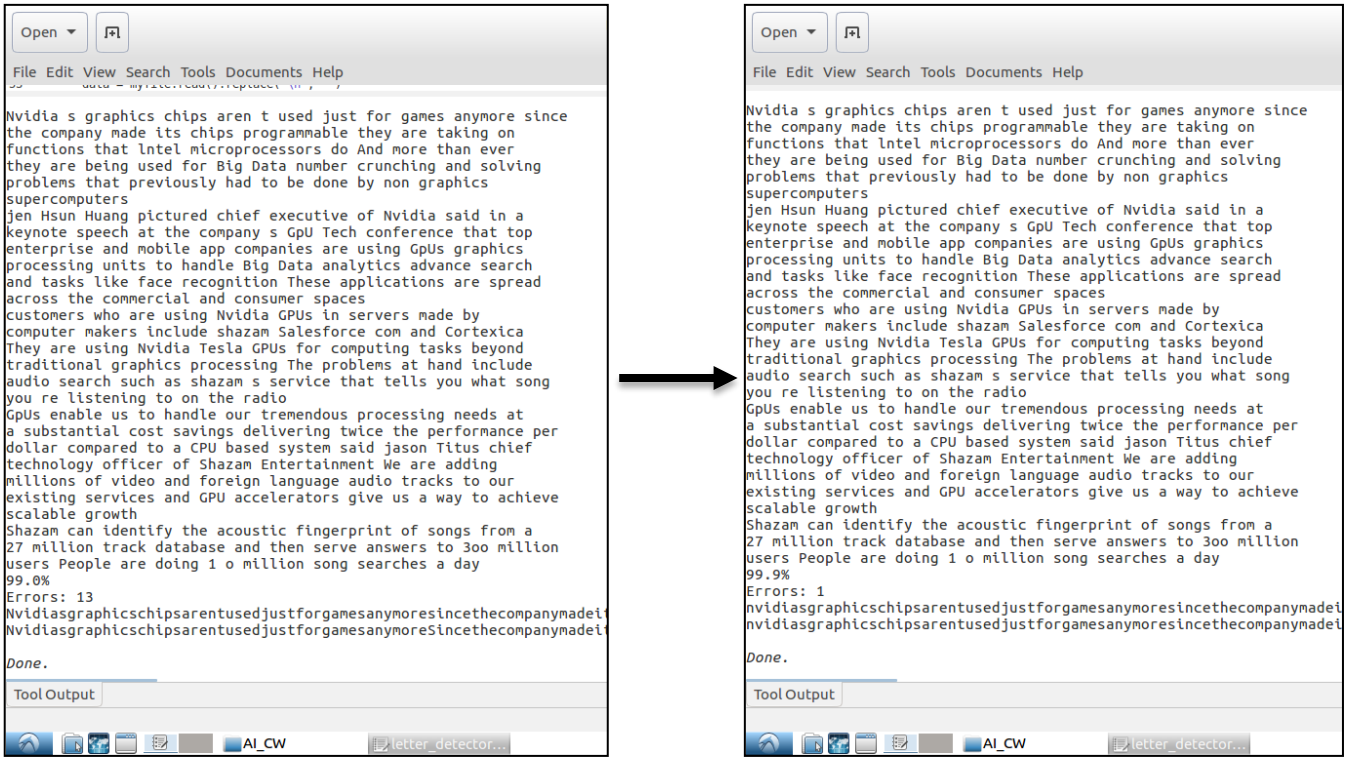
Appendix 3 – Test File Extraction Result



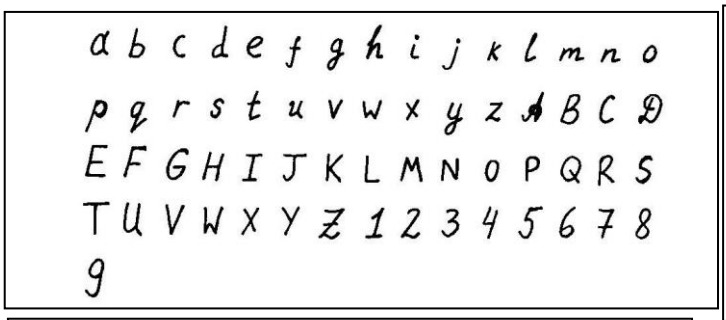
Appendix 4 – Typed-Letter Extraction Result (SVM)

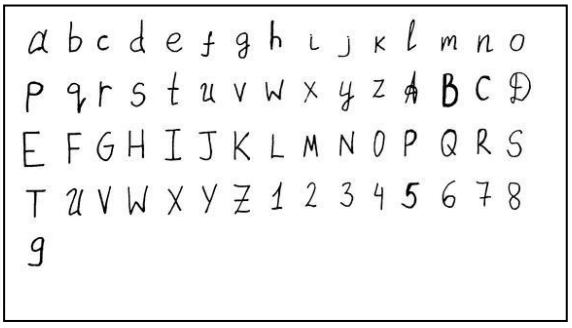


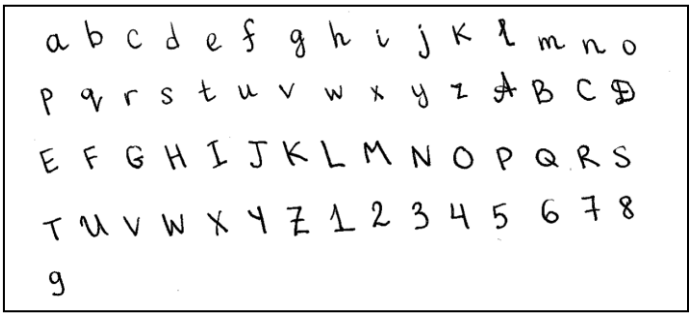
Appendix 5 – Post-processing improvement of result retrieved with KNN=2.

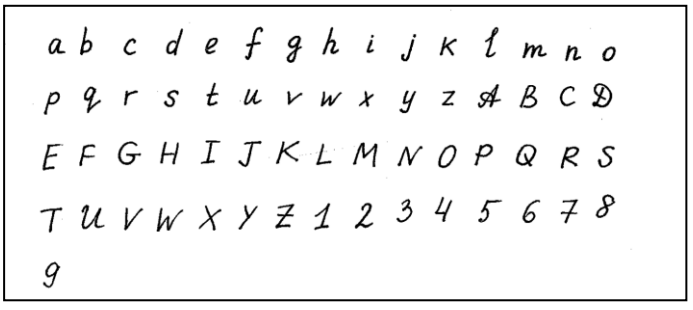


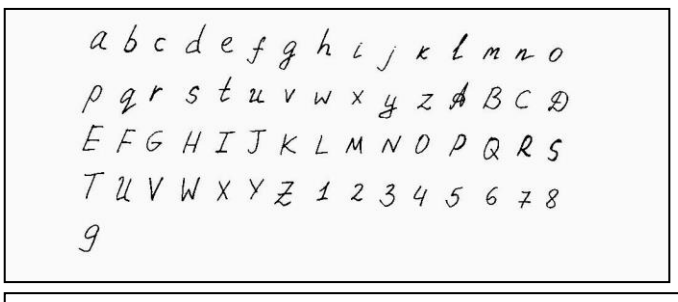
Appendix 6 – Handwriting Training and Testing Data

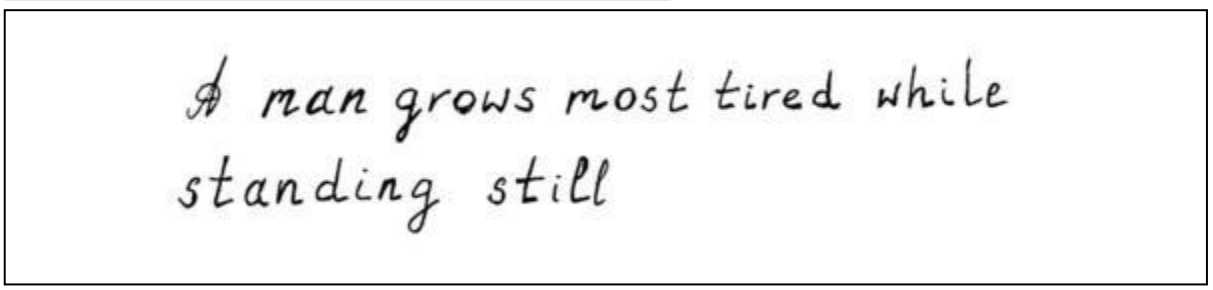








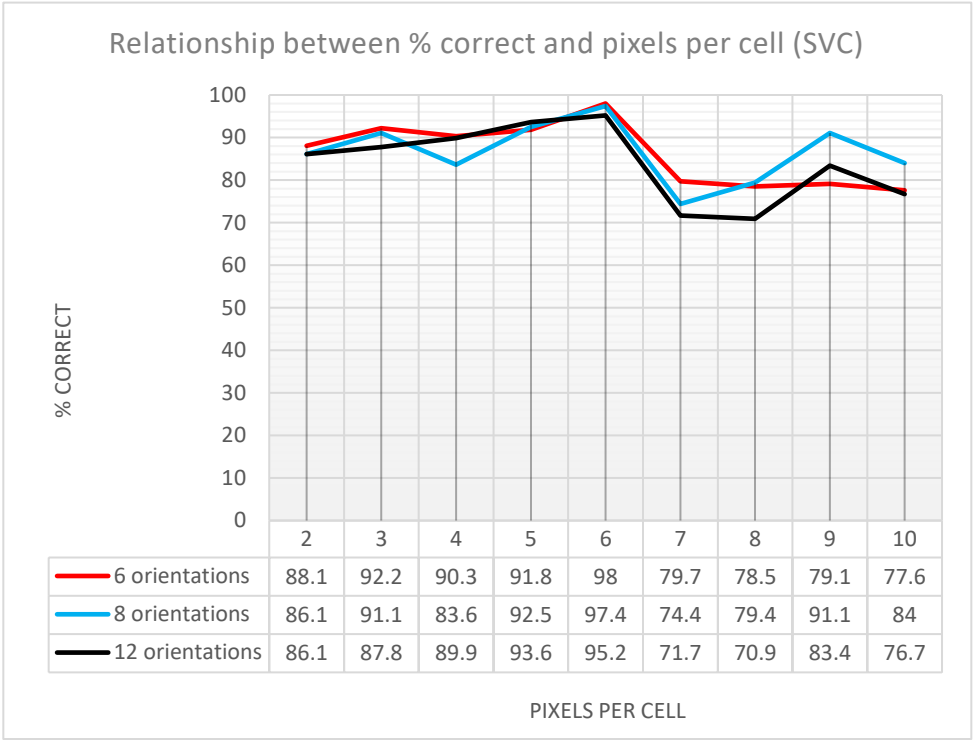




A man grows most tired while standing still

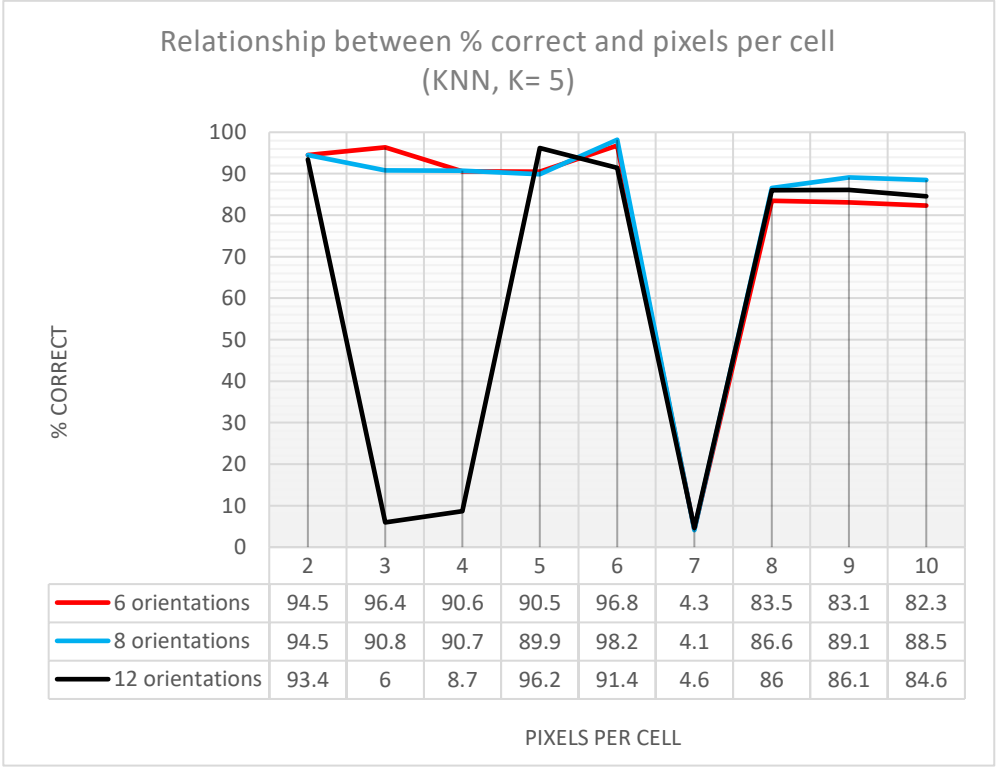
Work hard Dream big

Appendix 7 – Relationship between % correct and pixels per cell using SVC for shazam.txt document



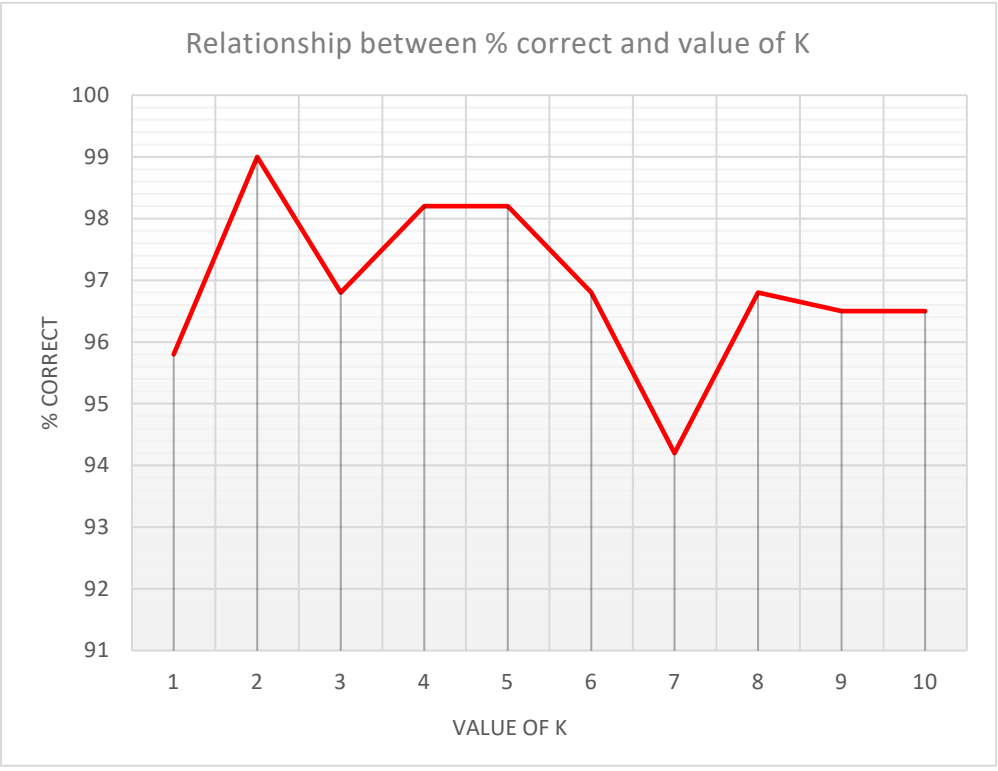
	2	3	4	5	6	7	8	9	10
6 orientations	88.1	92.2	90.3	91.8	98	79.7	78.5	79.1	77.6
8 orientations	86.1	91.1	83.6	92.5	97.4	74.4	79.4	91.1	84
12 orientations	86.1	87.8	89.9	93.6	95.2	71.7	70.9	83.4	76.7

Appendix 8 - Relationship between % correct and pixels per cell using KNN for shazam.txt document

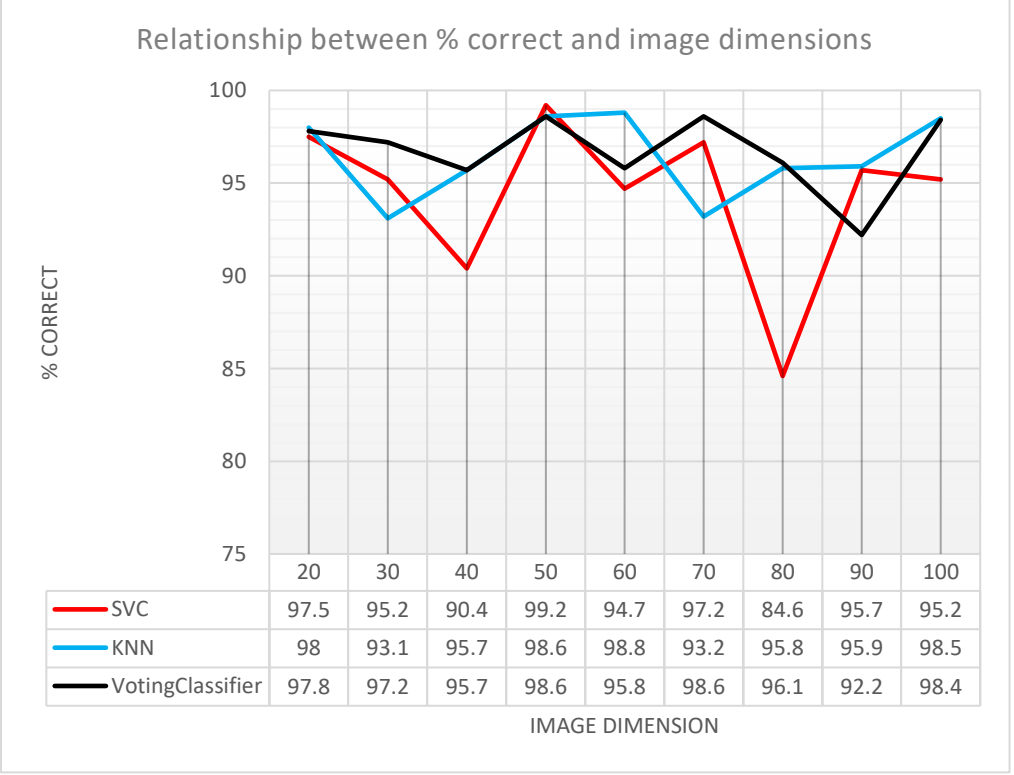


	2	3	4	5	6	7	8	9	10
6 orientations	94.5	96.4	90.6	90.5	96.8	4.3	83.5	83.1	82.3
8 orientations	94.5	90.8	90.7	89.9	98.2	4.1	86.6	89.1	88.5
12 orientations	93.4	6	8.7	96.2	91.4	4.6	86	86.1	84.6

Appendix 9 - Relationship between % correct and value of K (Computer-Typed)



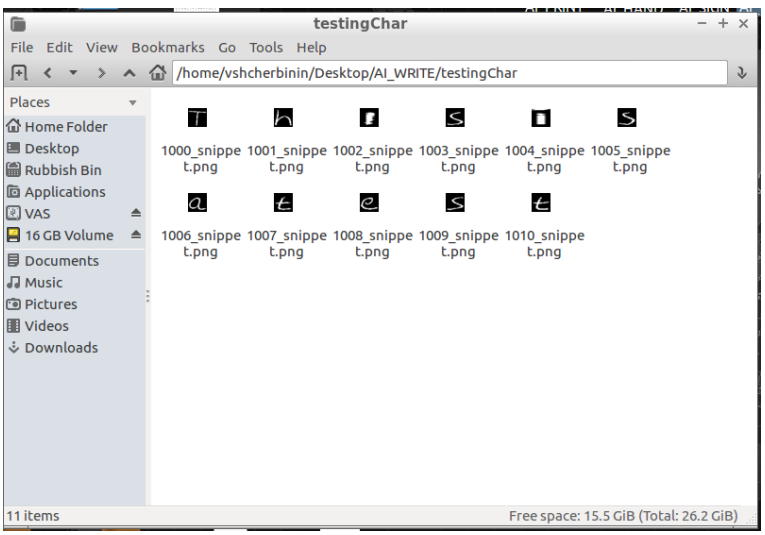
Appendix 10 - Relationship between % correct and image dimensions

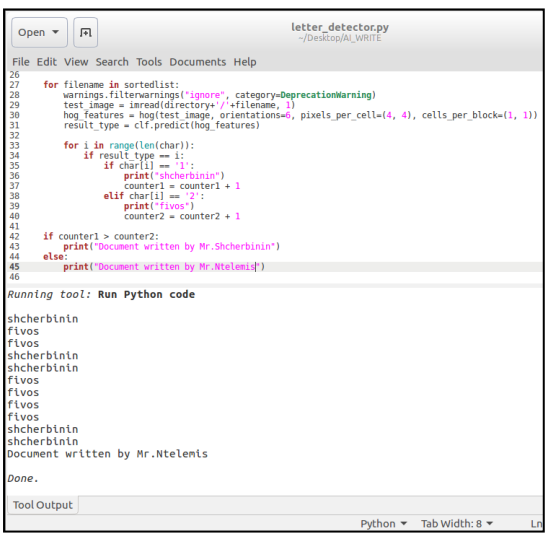


	20	30	40	50	60	70	80	90	100
SVC	97.5	95.2	90.4	99.2	94.7	97.2	84.6	95.7	95.2
KNN	98	93.1	95.7	98.6	98.8	93.2	95.8	95.9	98.5
VotingClassifier	97.8	97.2	95.7	98.6	95.8	98.6	96.1	92.2	98.4

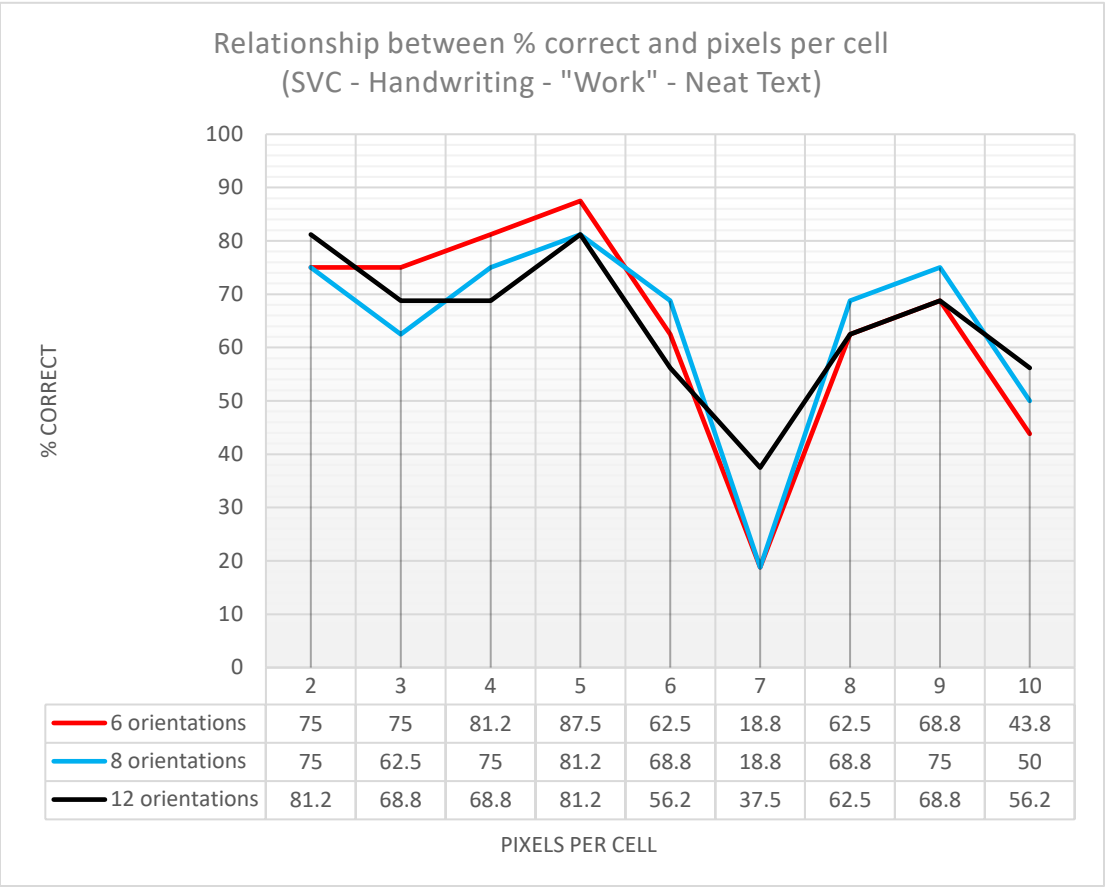
Appendix 11 – Signature Training and Test Data

Appendix 12 – Writing Identification Result of Test

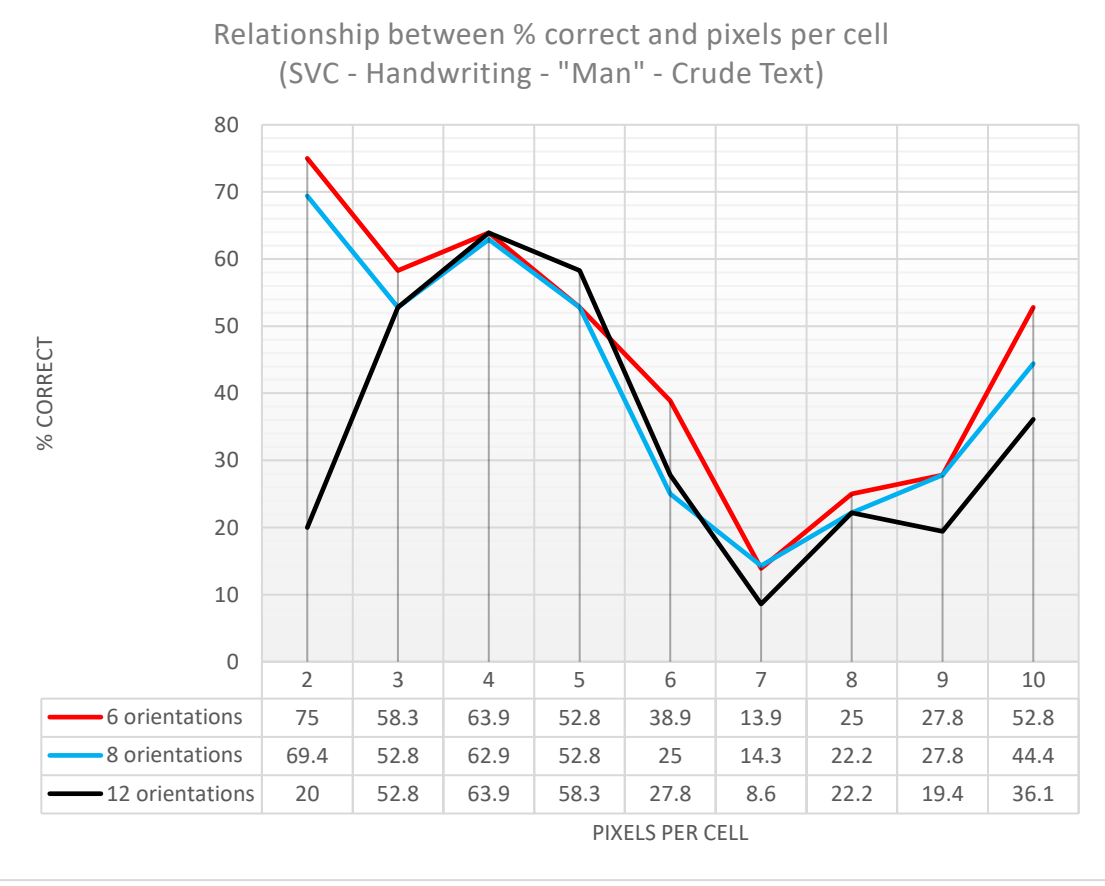




Appendix 13 - Relationship between % correct and pixels per cell using SVC (Handwriting)

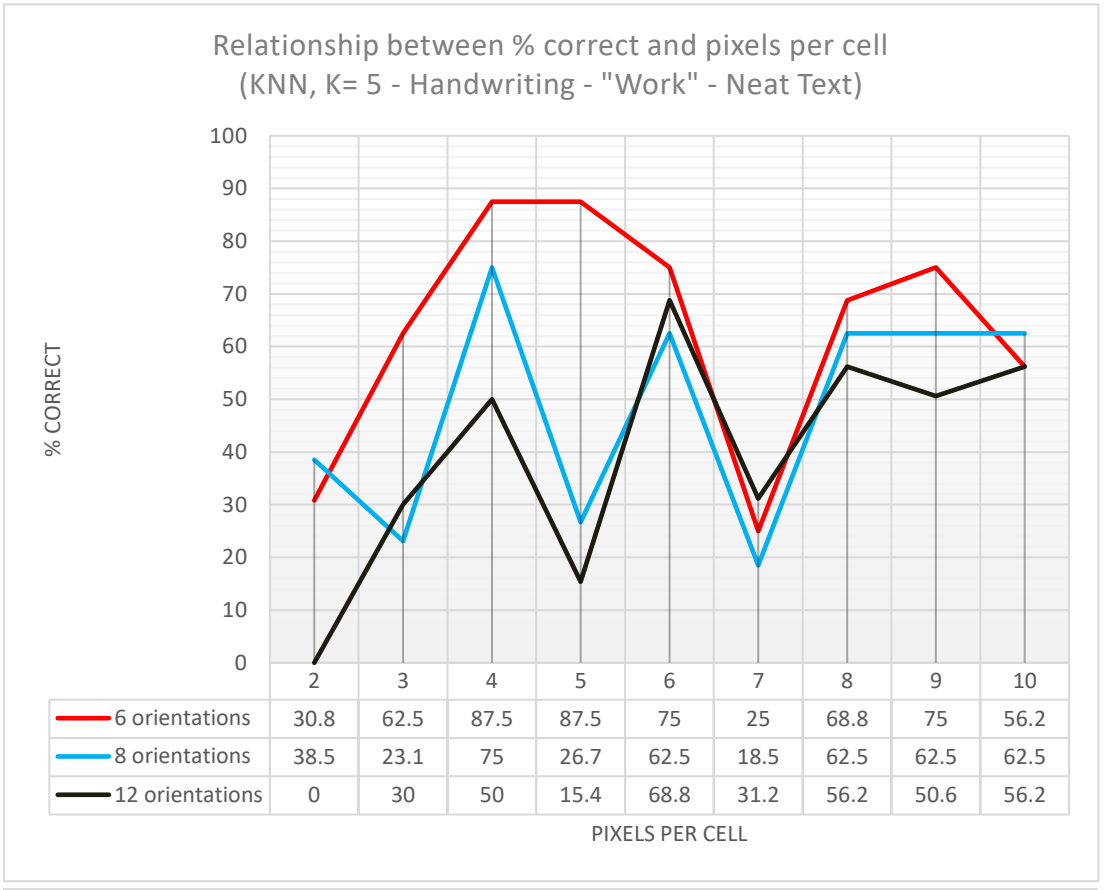


	2	3	4	5	6	7	8	9	10
6 orientations	75	75	81.2	87.5	62.5	18.8	62.5	68.8	43.8
8 orientations	75	62.5	75	81.2	68.8	18.8	68.8	75	50
12 orientations	81.2	68.8	68.8	81.2	56.2	37.5	62.5	68.8	56.2

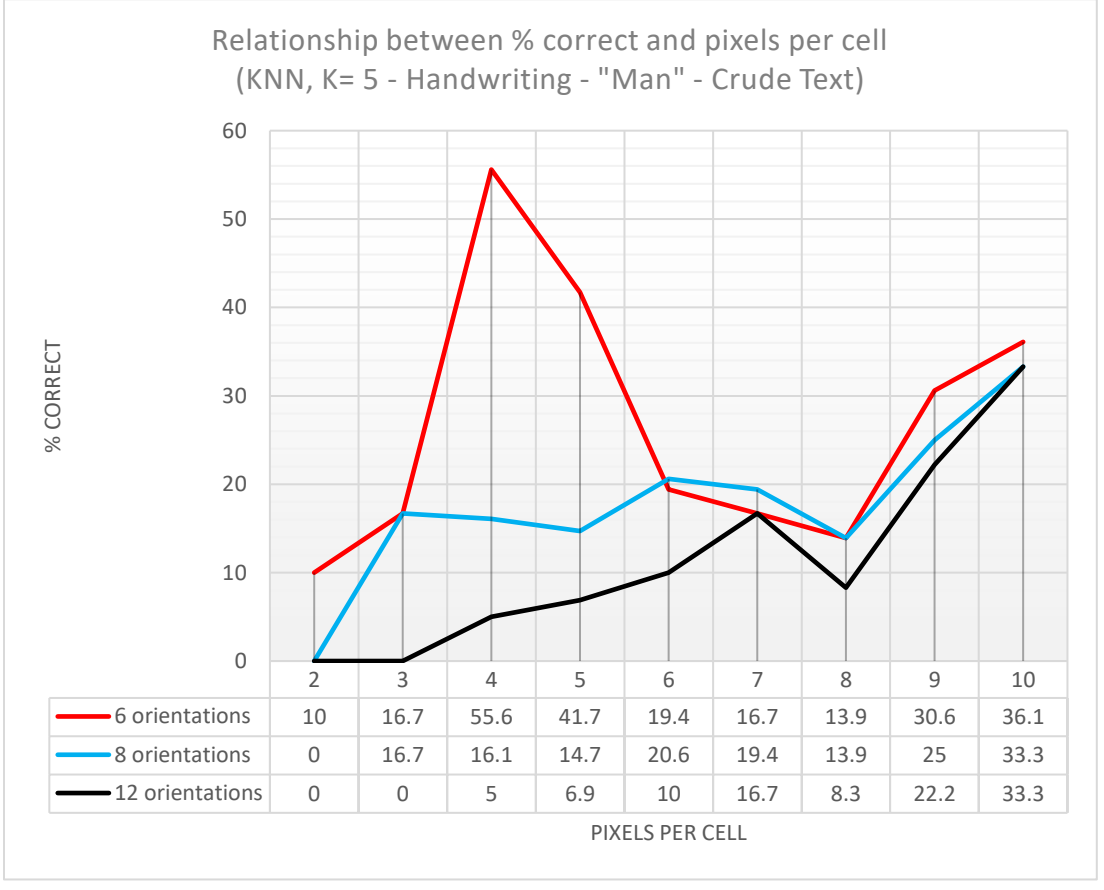


	2	3	4	5	6	7	8	9	10
6 orientations	75	58.3	63.9	52.8	38.9	13.9	25	27.8	52.8
8 orientations	69.4	52.8	62.9	52.8	25	14.3	22.2	27.8	44.4
12 orientations	20	52.8	63.9	58.3	27.8	8.6	22.2	19.4	36.1

Appendix 14 - Relationship between % correct and pixels per cell using KNN (Handwriting)

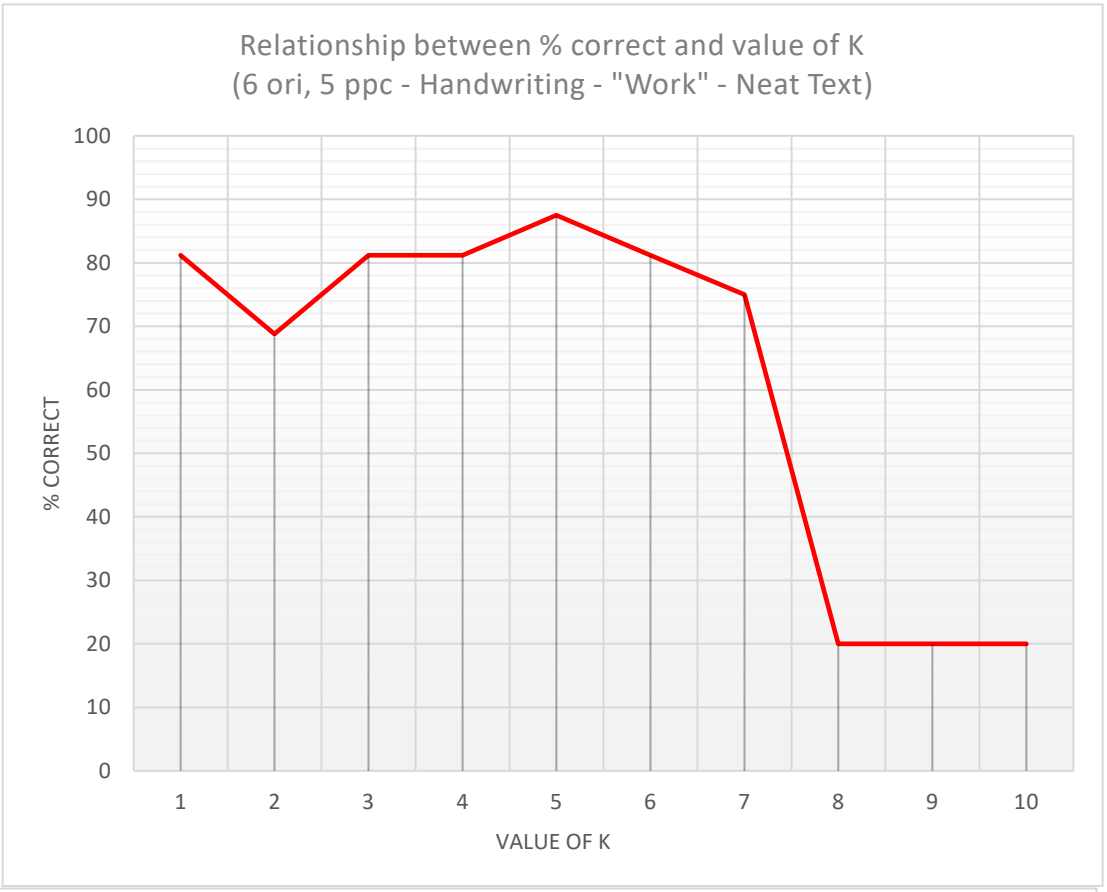


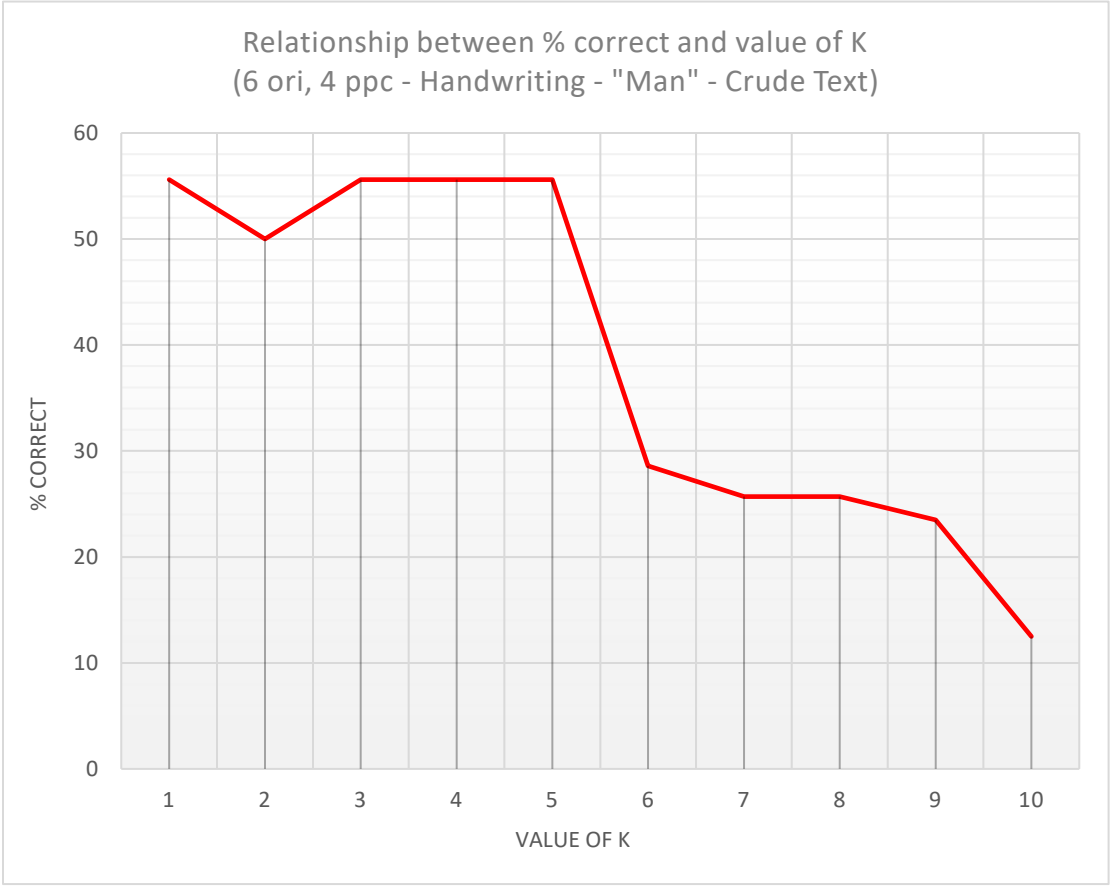
	2	3	4	5	6	7	8	9	10
6 orientations	30.8	62.5	87.5	87.5	75	25	68.8	75	56.2
8 orientations	38.5	23.1	75	26.7	62.5	18.5	62.5	62.5	62.5
12 orientations	0	30	50	15.4	68.8	31.2	56.2	50.6	56.2



	2	3	4	5	6	7	8	9	10
6 orientations	10	16.7	55.6	41.7	19.4	16.7	13.9	30.6	36.1
8 orientations	0	16.7	16.1	14.7	20.6	19.4	13.9	25	33.3
12 orientations	0	0	5	6.9	10	16.7	8.3	22.2	33.3

Appendix 15 - Relationship between % correct and value of K (Handwriting)





Appendix 16 - Relationship between % correct and pixels per cell (SVC – Writer Identification)