university of groningen / faculty of science and engineering

Convolutional Neural Network for off-line writer identification based on line segments of handwritten documents

Bachelor's Project Thesis

Eniko Puspoki, s3961656, e.puspoki@student.rug.nl,
Supervisor: Maruf A. Dhali, MSc

Abstract: Identifying individuals based on their handwriting can be challenging, but it is an essential step in document verification tasks. This is because everyone has their unique style of writing, which means that there are no two people who have exactly the same handwriting. However, when analyzing handwritten documents, the differences between writers can be hard to notice, and the higher the number of writers, the more difficult this task is. For this reason, utilizing computer technology in this field can be beneficial. In this thesis work, a low-cost convolutional neural network is presented, which is trained on two different datasets containing images of handwritten documents to identify writers with the classification of previously unseen data. The two datasets used are the IAM English Handwriting Database and the Firemaker image collection. The input to the model consists of patches of 113 x 113 pixels, cropped from the text's line segments. The model achieves an accuracy of 88.9% on the IAM images, and the highest accuracy obtained on the Firemaker images is 58%. In the Firemaker image collection, four different writing conditions were implemented when gathering the data, and the results suggest that within-writer variability might play a role in writer identification.

1 Introduction

Handwriting is based on the behavior of a person, it is a behavioral characteristic. It is mainly dependent on the writing method taught at the early stages of school, but also on personal preference, the copying of other styles from the people around you, and on the amount of writing experience (Schomaker, 2022). There are no two people who would be able to exhibit exactly the same handwriting, and no one can write down a word exactly the same way as they did the first time (Rehman et al., 2019). Because of this, handwriting is a very strong indicator of people's identity, since everyone has their own unique style that does not belong to anyone else. It can be used for proving someone's authenticity in forensic document verification tasks, for example to detect frauds or verify legal documents.

However, trying to analyse these documents only with a human eye without the aid of a computerized system can be very tiresome, therefore developing these systems and this way incorporating technology in handwriting analysis can significantly help this field of science.

Computers and machine learning have become more and more utilized for handwriting analysis. The term used for this field is called writer recognition, and it has two main components. One of them is writer verification, where the task is to compare a new handwriting sample with pre-stored samples for authentication, and is a binary classification problem. The other is writer identification - which is the one that is attempted in this project -, where the task is to find the correct writer from a list of other registered writers based on the resemblance between their handwriting, and is a multinomial classification problem (Rehman et al., 2019). A visual representation for these processes is visible on Figure 1.

This project took on the challenge of identifying writers based on line segments of handwritten documents independently of text. The data used in this

study were off-line - scanned images of the handwritten texts - as opposed to on-line which also includes information such as pressure, pen-angle and where the writing is usually performed on a tablet. The proposed model came from an already existing convolutional neural network called Deepwriter that managed to reach an accuracy as high as 99.01% on the IAM English Handwriting Database on 301 writers (Xing & Qiao, 2016). Moreover, it has also been tested with other datasets with the number of writers ranging from 300 to 657, and the accuracy was always above 93%. Deepwriter has five convolutional layers and around five and a half million parameters, which means that its usage is quite computationally expensive. The reason why in this project the model programmed is a simplified version of Deepwriter is that we wanted to find out if it is possible to achieve a similar performance but with a more efficient, less complex and therefore less computationally expensive model.

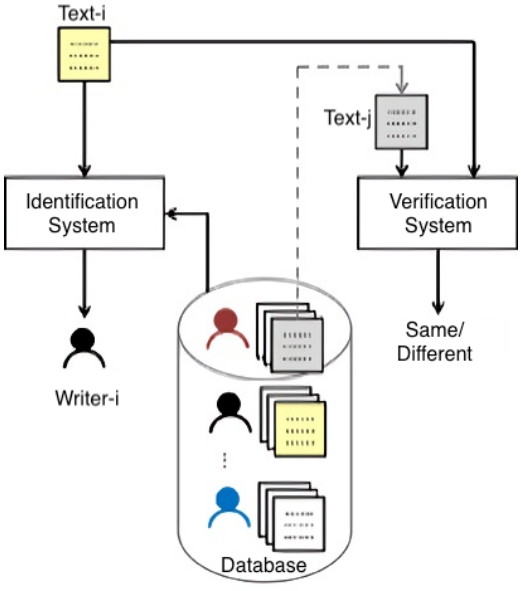

Figure 1: Writer identification and verification systems (Adak et al., 2019)

2 Data

2.1 IAM

In this project there were two datasets used. The first one is called the IAM English Handwriting Database, which contains scanned forms of copied handwritten texts from 657 writers (Marti & Bunke, 2002). The authors of the data have developed their own line segmentation method and made the line segments available online, so the images did not require many preprocessing steps. But since this experiment worked with the 50 most common writers, - and most common here means those writers who have the highest number of samples available - these 50 writers had to be separated. The total number of line images from these 50 writers was 4909. The images were split into train, validation and test sets, with ratios of 0.64, 0.16 and 0.2 respectively. Example line segments are visible on Figure 2.

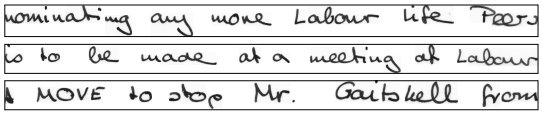

Figure 2: Example line segments from the IAM dataset

2.2 Firemaker

The other dataset is called Firemaker, which is a Dutch handwriting image collection (Schomaker & Vuurpijl, 2000). In this case the participants had to write in four different conditions during the data gathering process. "The conditions are: p1: copied, natural style, p2: copied, UPPER case, p3: copied and forged, i.e.,"try to write in a different style than your natural style", and 4 self generated, i.e., text produced to describe a given cartoon." (Schomaker & Vuurpijl, 2000). Example images from the different conditions can be seen on Figure 3. The image collection contains 1000 scanned documents from 250 writers, therefore four documents in different conditions per writer. Since the images contained the full forms, they had to be preprocessed and line segmentation had to be performed.

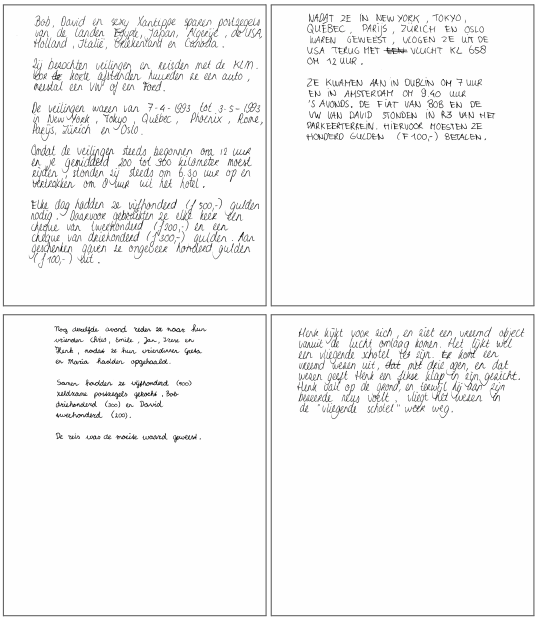

Figure 3: Example images from the Firemaker dataset (conditions one-four respectively)

2.2.1 Preprocessing

First, the top and bottom of the images were cropped to remove the labels containing the demographic data of the writers, leaving only the handwritten text on the images, see Figure 4.

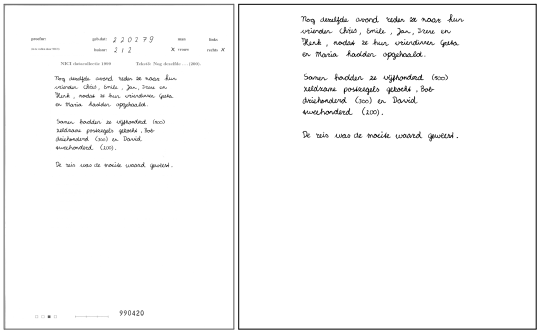

Figure 4: Removing the labels

The next step was to binarize the images with Otsu's automatic image thresholding method (Bradski, 2000), which means that based on a calculated threshold the pixels of the images were changed to either black or white (foreground and background).

Then the edges of the text were located in order to remove as much of the excess white pixels as possible, an example is visible on Figure 5. After this, the lines had to be separated from the text.

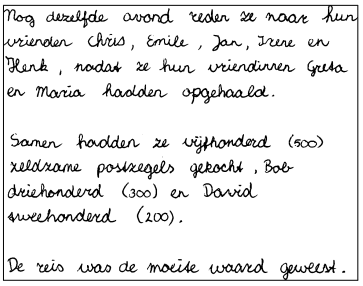

Figure 5: Edge detection on the binarized image

The line segmentation process started by counting the number of black pixels in each row of the images. The horizontal projection plane of the counted pixel density can be seen on Figure 6.

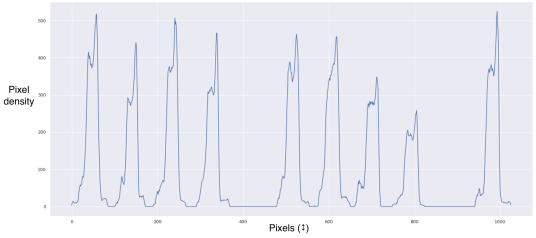

Figure 6: Horizontal projection plane of the pixel density

The number of curves on Figure 6 represents the number of lines, and the lower points in the graph are indicating the start and end points of the lines, since that is where the least amount of black pixels are. However, the graph in its current form contains too many local minimums which would represent the separating points, so the graph had to be smoothed out (Armstrong, 2019), see Figure 7.

Now we have the correct local minimums which are representing the breaking points, which are visible on Figure 8.

After segmenting the images based on the found points, another edge detection was done to remove the excess white parts from the shorter lines, this way the images were exactly the size of the lines, see Figure 9.

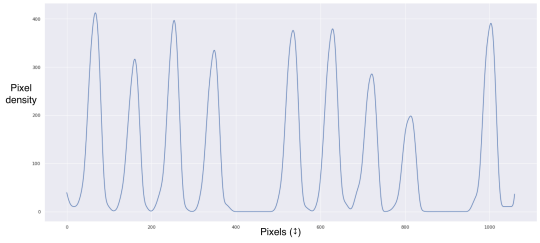

Figure 7: Smoothed horizontal projection plane of the pixel density

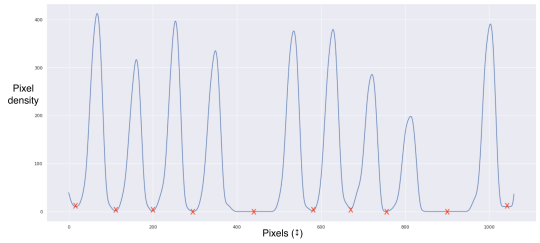

Figure 8: Local minimum points pinpointing the start and end of the line segments

However, in a few cases the segments were not containing the right, errorless information, because for example there was an extra local minimum point that produced a small segment of only white pixels or a small segment containing only a few black pixels. To handle these situations, images where the percentage of white pixels was more than 95% were omitted, and images with a height smaller than 40 (this value was found after experimenting and checking the average height of the line segments) were also omitted. This method seemed to produce good results, and the line segmentation process was concluded.

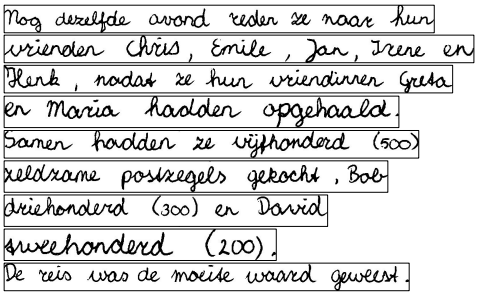

Figure 9: Produced line segments

Afterwards the next step was selecting the 50 most common writers to be used in this experiment, which was done the same way as with the IAM dataset, to end up with as much data as possible.
Then the line segments were divided into four groups based on the conditions (one-four), and split into train, validation and test sets. The ratios were 0.64, 0.16 and 0.2 respectively.
Since there was only four forms available per writer, as the last step of the preprocessing, data augmentation had to be done. The program that was used for this is called imagemorph (Bulacu et al., 2009), and it works by generating synthetic examples using random geometric distortions. Examples for such distortions are visible on Figure 10. 10 augmented samples were created for every image, this way the number of samples per condition ended up being around 3000-5000.
After all four of the train-validation-test groups were augmented the images were converted back to grayscale, and that concludes the preprocessing of the Firemaker database.

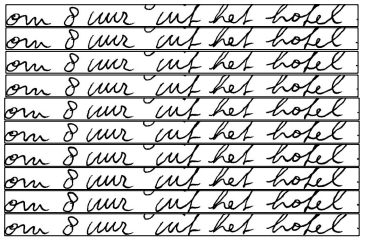

Figure 10: Data augmentation with imagemorph

3 The model

As mentioned before, the model used in this experiment was a simplified version of Deepwriter. The main difference is that the presented network is less complex, since it has only three convolutional layers as opposed to Deepwriter which has five. The ordering and the parameters of the layers were set similarly to Deepwriter. The structure of the network is visible on Table 1. The last fully connected layer was a vector of length 50, which is the number of writers. It was followed by the Softmax activation function to end up with the prediction probabilities of each writer's likeliness of being the cor-

Table 1: Network structure of the model

Conv1, filters=32, kernel=(5, 5), strides=(2, 2)
Activation, ReLU
MP, pool=(2, 2), strides=(2, 2)
Conv2, filters=64, kernel=(3, 3), strides=(1, 1)
Activation, ReLU
MP, pool=(2, 2), strides=(2, 2)
Conv3, filters=128, kernel=(3, 3), strides=(1, 1)
Activation, ReLU
MP, pool=(2, 2), strides=(2, 2)
Flatten
Dropout, value = 0.5
Dense, units = 512
Activation, ReLU
Dropout, value = 0.5
Dense, units = 256
Activation, ReLU
Dropout, value = 0.5
Dense (output layer), units = 50
Activation, Softmax

rect one. The model was compiled with Categorical crossentropy loss and the Adam optimizer.

As discussed earlier, the input images were line segments, but before feeding them to the network there was an additional step. The creators of Deepwriter decided to augment the data by resizing the shorter side of the images to 113 with original aspect ratio and then randomly cropping out 113 × 113 big image patches from the resized line segment images. Keeping the original aspect ratio turned out to be very important, as it contains information about the handwriting attributes of the writers, and the identification accuracy decreased significantly when the input images were distorted (Xing & Qiao, 2016). The same procedure was implemented with the line segment images used in this project, example inputs to the model can be seen on Figure 11.

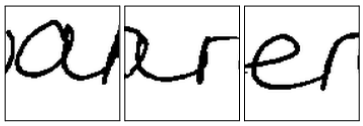

Figure 11: 113 × 113 image patches (the input images)

Deepwriter has a multi-stream structure, which means that they used two adjacent image patches as input, taking into account the spatial relationships between these image patches. By using only one image as input we do lose some information that could help determine the correct writer, but this way we can eliminate the expense of storing the spatial relationships.

Deepwriter used a batch size of 256, but since the proposed model is simpler it was set to 16, and other implementations of Deepwriter also reported to have used this value (Dwivedi, 2018; Reddy, 2018; Castillo, 2018). Since the creators of Deepwriter do not specify the number of epochs or iterations per epoch, it was calculated based on the number of training images and the batch size, this way the number of iterations per epoch was 3268. The size of the validation steps was calculated similarly, which ended up being 842. Concerning the number of epochs, at first the training was run for 15 epochs, but at some point the model did not improve anymore and that was always around the 10th epoch, therefore the experiments were run for 10 epochs. The differences between the accuracies and losses per epoch are visible on Table 3 and Table 4 in Appendix A.

4 Results

4.1 Results on IAM

A five-fold cross validation was performed on the IAM images, and the model achieved an accuracy of 88.9% on 50 writers, with a standard deviation of 0.0075. These results were very close to other simpler implementations of Deepwriter where the researchers have reported an accuracy of 81-94% with the IAM dataset (Dwivedi, 2018; Reddy, 2018; Castillo, 2018).

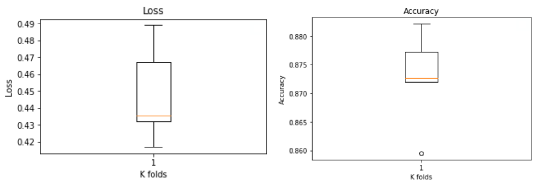

Figure 12: Loss and accuracy of the model on the IAM images

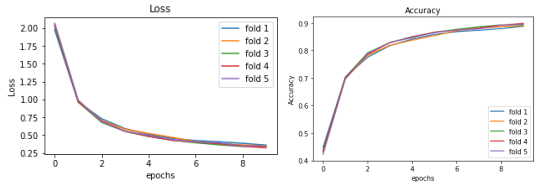

Figure 13: Learning curves from five-fold cross validation on the IAM images

Figure 13 displays the learning curves during the cross validation steps, the curves are following almost exactly the same path, therefore we can state that the model was actually learning the handwriting patterns of the writers.

4.2 Results on Firemaker

The results on the Firemaker images did not turn out to be as good as the results on the IAM images. The model was tested on the different conditions separately as well as the conditions together. The best accuracy that the model achieved was 58% both on the first condition, and on the first and fourth condition combined. Interestingly, in the second condition where people had to copy a text in uppercase, the model was not learning at all. The

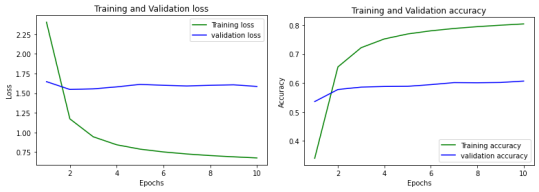

Figure 14: Curves from condition one

two graphs on Figure 14 are the curves from condition 1, and it is visible that the model did not learn that much between the first and last epoch compared to experiments with the IAM images, but it did slowly improve over time. The accuracy of the model was also checked before the data augmentation with imagemorph, which for the different conditions were between 26-28%, thus data augmentation significantly improved the performance of the model.

Table 2: Results on the Firemaker images

Condition	accuracy
1	58%
2	2%
3	45%
4	54%
all	2%
1,4	58%
1,3,4	50%

5 Conclusions

The differences in performance between the conditions suggest that within writer variability - noise in an individual's writing - played a role in the classification. In the second condition, where the participants had to write in uppercase, the model did not manage to learn the writing attributes of the writers. This means that the variability of the letter shapes introduced so much noise to the individuals' handwriting that the model could not acquire the true handwriting habits of the writers. And if we look at the results when trying different conditions together, condition one and four produced the best results. In both of these conditions the task was to write in natural style, therefore the noise in the handwriting was little, making it possible for the model to learn the unique handwriting styles.

To solve the mentioned within writer shape variability problem, an on-line recognition algorithm could be used (Schomaker, 2022). But that would require such samples which would have to be taken by a pen-based computer system in order to capture all relevant information about the writing habits of the writers, allowing the model to find handwriting features less sensitive to within writer variability and more sensitive to between writer variability.

Additionally, a cleaner line segmentation method may produce better results. Although the described method seemed to work fine, after manually going over the images, in some cases it did cut off parts of letters that were large enough to be in the lines above or under them. Examples for such occurrences are visible on Figure 15.

A possible way to improve this could be that after locating the start and end points of the segments from the horizontal projections of the pixel density, the A* path planning algorithm could be used to

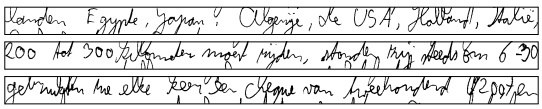

Figure 15: Examples of overlapping and stripped line segments

find a path through the lines that does not cut off or leave out parts of letters. An implementation of such process is visible on Figure 16. This could potentially prevent the top and bottom of the letters from being cut, this way improving the quality of the line segments.

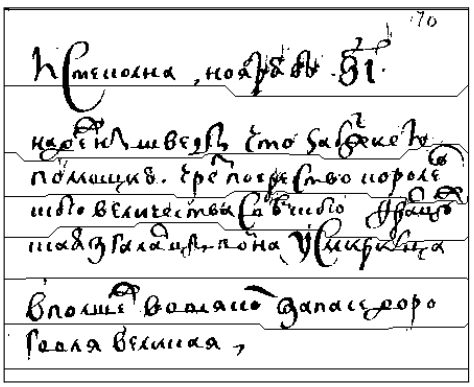

Figure 16: Example application of the A* path planning algorithm for line segmentation (Surinta et al., 2014)

Overall the objective of this project was to create a system that can identify writers based on off-line handwritten documents independently from text, moreover the goal was to achieve similar results to Deepwriter but in a less computationally expensive way. Although the model performed well on the IAM dataset, in its current state it is not a generalized model. Testing with more datasets and a cleaner line segmentation method could reveal more information about the usefulness of this model in automated handwriting analysis, and potentially answer the question whether it is possible to identify writers with a simple, low computational cost model.

References

Adak, C., Chaudhuri, B. B., & Blumenstein, M. (2019). An empirical study on writer identification and verification from intra-variable individual handwriting. *IEEE Access*, *7*, 24738-24758. doi: 10.1109/ACCESS.2019.2899908

Armstrong, I. (2019, 08). *Handwritten text recognition.* Retrieved from `https://github.com/IrinaArmstrong/HandwrittenTextRecognition`

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Bulacu, M., Brink, A., Zant, T., & Schomaker, L. (2009, 01). Recognition of handwritten numerical fields in a large single-writer historical collection. In (p. 808-812). 10th International Conference on Document Analysis and Recognition. doi: 10.1109/ICDAR.2009.8

Castillo, D. (2018, 11). *Iam writer recognition.* Retrieved from `https://github.com/diegocasmo/iam_writer_recognition`

Dwivedi, P. (2018, 01). *English deep writer.* Retrieved from `https://github.com/priya-dwivedi/Deep-Learning/tree/master/handwriting_recognition`

Marti, U.-V., & Bunke, H. (2002, 11). The iam-database: An english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, *5*, 39-46. doi: 10.1007/s100320200071

Reddy, T. (2018, 06). *Offline handwriting recognition using cnn.* Retrieved from `https://github.com/TejasReddy9/handwriting_cnn`

Rehman, A., Naz, S., & Razzak, M. (2019, 04). Writer identification using machine learning approaches: A comprehensive review. *Multimedia Tools and Applications*, *78*. doi: 10.1007/s11042-018-6577-1

Schomaker, L. (2022, 06). *Handling within-writer variability and between-writer variation in the recognition of on-line handwriting. *.*

Schomaker, L., & Vuurpijl, L. (2000, Jan). *Firemaker image collection for benchmarking forensic writer identification using image-based pattern recognition.* Zenodo. doi: 10.5281/zenodo.1194612

Surinta, O., Holtkamp, M., Karaaba, M., Oosten, J.-P., Schomaker, L., & Wiering, M. (2014, 05). A* path planning for line segmentation of handwritten documents. In (Vol. 2014). doi: 10.1109/ICFHR.2014.37

Xing, L., & Qiao, Y. (2016, 10). Deepwriter: A multi-stream deep cnn for text-independent writer identification. In (p. 584-589). doi: 10 .1109/ICFHR.2016.0112

A Appendix

Table 3: Accuracy of the model on the IAM dataset for 50 writers

Epochs	val_acc	val_acc diff
1	0.7507	-
2	0.8395	0.0888
3	0.8661	0.0266
4	0.8721	0.0060
5	0.8727	0.0006
6	0.8816	0.0089
7	0.8889	0.0073
8	0.8922	0.0033
9	0.8971	0.0049
10	0.8975	0.0004
11	0.8851	-0.0124
12	0.9011	0.0160
13	0.8948	-0.0063
14	0.8992	0.0044
15	0.9015	0.0023

Table 4: Loss of the model on the IAM dataset for 50 writers

Epochs	val_loss	val_loss diff
1	0.7955	-
2	0.5127	-0.2828
3	0.4387	-0.0740
4	0.4218	-0.0169
5	0.4258	0.0040
6	0.4012	-0.0246
7	0.3741	-0.0271
8	0.3669	-0.0072
9	0.3496	-0.0173
10	0.3588	0.0092
11	0.3998	0.0410
12	0.3468	-0.0530
13	0.3637	0.0169
14	0.3464	-0.0173
15	0.3377	-0.0087