FIT Report

Report 1

Handwriting detection and Note Conversion using OCR

Piętka Ł, Barot H, Krzysztosiak M, Markowicz J, Milewski M, Sobczak M

March 12 2025

Contents

1 Introduction

The aim of this project is to create a tool for digitizing handwritten notes written in the English language into readable text in ASCII format. The tools utilized in this project are existing OCR models along with various preprocessing techniques applied to training data in order to minimize computational time and discard unnecessary information.

We feel the applicability of such a tool is rather high as it would allow for a streamlined process of digital note making and sharing for those who prefer a traditional method of note keeping. In addition, such a tool could be utilized to increase the efficiency of tutors in correcting the test papers of their students.

1.1 Background Knowledge

This project focuses primarily on the use of Optical Character Recognition (OCR) models to convert handwritten text into digital text. OCR is a technology that is used to convert different types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera, into text files that can be easily edited, searched, and stored on a computer.

1.2 Problem Statement

The main problem that this project aims to solve is the inefficiency of converting handwritten notes to digital text. This is a common problem for students, teachers, and professionals who prefer to take notes by hand. The process of converting handwritten notes into digital text is time-consuming and error-prone. This project aims to develop a tool that can automate this process and make it more efficient and accurate.

2 Methodology and Procedure

There are various steps involved in the process of converting handwritten notes into digital text using OCR technology, and it is best to break down the problem into smaller problems. The following steps are involved in the process:

1. Data Collection
2. Data Preprocessing
3. Model Training
4. Model Evaluation
5. Model Deployment

2.1 Data Collection

The first problem we faced was deciding what kind of data we should collect to train the OCR model. The choice of the model should reflect the purpose of our project and, as such, we browsed for information and prepared a comparison between some notable datasets and their characteristics chosen based on our research. (Table 1)

We decided to use the freely available TrOCR model as it comes pre-trained for simple single-word recognition problems, however, it struggles with handwriting and full documents and thus needed to be fine-tuned.

Data found in the real world would differ in many aspects as the equipment used to take the images, the lighting used, and the handwriting of each person may vary, as such, the data should cover different lighting conditions, various handwriting styles, and varied image qualities (within reason).

For the training data, we used the GNHK dataset [1] which consists of real world examples of handwriting suitable for training and testing of the model.

The rationale behind choosing this dataset was simple as the data provided aligns with our goal, allowing us to fine-tune the model for the specific purpose of recognizing handwritten notes.

For the testing data, we plan to utilize both the GNHK dataset examples as well as our own handwritten notes.

Table 1: Comparison of Handwritten Text Datasets

Dataset	**Characteristics**	**Training Size**	**Testing Size**
CEDAR [2]	Scans of handwritten letters at 300 DPI, developed in 2002.	5321	N/A
MNIST (Archived) [3]	Grayscale images with normalized sizes, reducing preprocessing time.	60,000	10,000
GNHK [1]	Camera-captured handwritten notes with 1080p to 4k resolution, lighting, and backgrounds.	515	172

2.2 Data Preprocessing

The data collected for training and testing may come from different sources and as such different equipment (in this case a camera or a scanner) may have been used to obtain it, or it may differ in quality standards. Because of that, certain variables need to be controlled through normalization to provide any significant information. The preprocessing of data aims to solve the problem of controlling the quality of data used to fine-tune the model for our purpose, reduce the computational time, and narrow down the information we want to extract from real data.
After reading research papers on the subject [4], we decided on a list of variables that need to be controlled and the corresponding preprocessing approaches:

- Brightness levels and lighting
- Image size
- Image resolution
- Coloured image should be converted to grayscale
- Geometric distortions caused by the angle of the camera
- Compression and artifacts.

For **brightness levels** and **colours**, the images should be converted to grayscale for increased accuracy. Then a transformation called histogram equalization can be performed to normalize the brightness levels. Histogram equalization improves the contrast of an image by spreading out the most frequent intensity values, making the histogram of the output image approximately uniform, for that purpose we utilized the relevant function in the OpenCV library. [5]
A thresholding algorithm should also be used to separate the text and the background, including phenomena such as shadows caused by lighting that obscure the text, for this purpose, we utilize Otsu thresholding [4] via the relevant OpenCV function [6].
For **image size** and **resolution**, it is important to ensure that all images fed into the model are of a consistent size and resolution. This helps in maintaining uniformity and reduces the complexity of the model. Images that are too small may not contain enough detail for accurate recognition, while images that are too large may introduce unnecessary computational overhead. Therefore, resizing the images to a standard resolution is a crucial preprocessing step. We used OpenCV to upscale all images to a consistent resolution.
For **geometric distortions**, images may be taken at various angles, leading to distortions that can affect the accuracy of the OCR model. To correct these distortions, we applied geometric transformations such as rotation and perspective correction. These transformations help to align the text properly, making it easier for the OCR model to recognize the characters accurately. OpenCV provides functions for these transformations, which we utilized to preprocess the images.
For **compression and artifacts**, images can suffer from compression artifacts, especially if they are saved in lossy formats like JPEG. These artifacts can introduce noise and reduce the quality of the image. To mitigate this, we saved all images in a lossless format like PNG during preprocessing to avoid creating more artifacts. Additionally, we applied denoising techniques using OpenCV to remove any existing artifacts and enhance the quality of the images.

2.3 Limitations

In addition to all the tools and techniques utilized, we feel it is important to note the limitations of the assumptions and data collection.
The project assumes that images will be taken with the quality that is within reason, as such images should be taken with the intent of being legible, and thus input data with significant resolution problems, or pitch dark lighting ought to be ignored as they are not representative of the problem we aim to solve. The dataset we used presents a vast amount of images differing in quality and handwriting styles, however, we understand that more sources could be utilized.

Report 2

3 Re-evaluating: Report 1

In report 1, the aim of the project was stated to be creation of a program that utilizes OCR technology, which allows for recognition and digitization of handwritten notes into easily readable and storable text documents on a computer. However, the project was not clearly defined and the problem statement was not well articulated. The target population was not specified, and the aim of the project was not clearly defined. The project lacked a clear focus and direction, which made it difficult to understand the purpose and goals of the project.

Due to the target population being vague, multiple areas of this project were hard to define, and ambiguous as to what direction the project is headin in. This was a major roadblock because when we discussed questions like "Do the samples include math equations?" or "Do the samples include letters?" "Are the letters allowed to be geometrically malformed or the the samples have to be clean and perfectly written?". Our group memmbers had different approaches to these questions, and this led to a lot of confusion and miscommunication.

Hence, this report would like to revisit and address the flawed initial approach. Define the who the project is for, which will make it simpler to find source of raw data, and implement a model that will best suit the use case defined.

3.1 Problem Statement

Certain two members of our group particularly enjoyed writting english literature. Wether it was analitical essays, or fantasy stories. They both agreed that writting on a physical paper felt the most comfortable and natural. However, they couldn't organize their raw notes and digitizing them was really time consuming. Hence, we decided to create a program that would help them with this problem. The specific aim of this project is to create a program for the the two members of our group, that could assist in digitization of their english literature notes.

4 Data Collection Procedure:

In the previous report, we decided to use the GNHK dataset [1], as it best suited the freely available OCR model that we chose. However, this was a large dataset, that contained a variety of different samples. The sample types included math equations, plain english text, distorted handwritting and much more. Additionally these datasets conainted a the different types of english (british, american, asian) used all over the world.

This essentially ment, that this dataset ranged over a wider domain than what was needed for our project. This is displayed using a venn diagram in figure 1. Set A (red circle) is the collectioon of what GNHK has to offer, meanwhile Set B (blue circle) is what is required for our project. The intersection of the two sets is what we need.

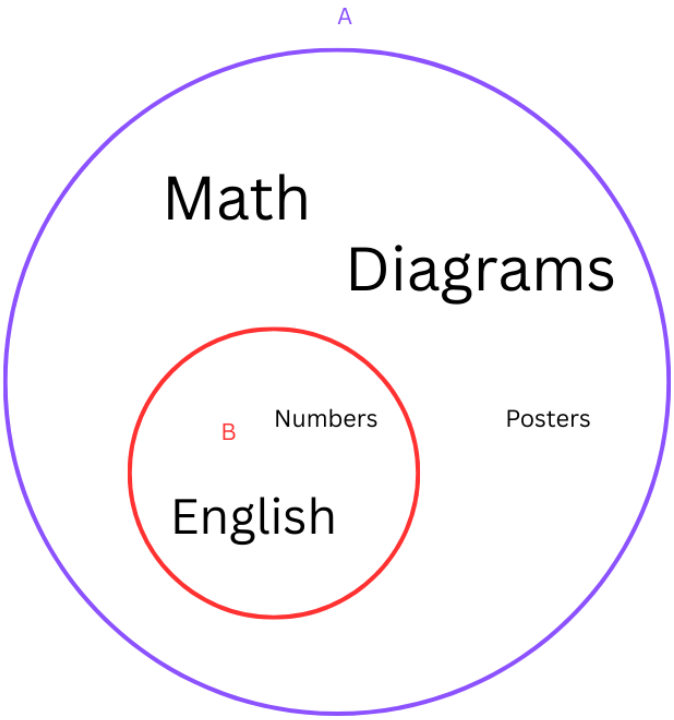


Figure 1: Venn Diagram of Dataset Characteristics vs. Project Needs

To collect the data we had two choices:

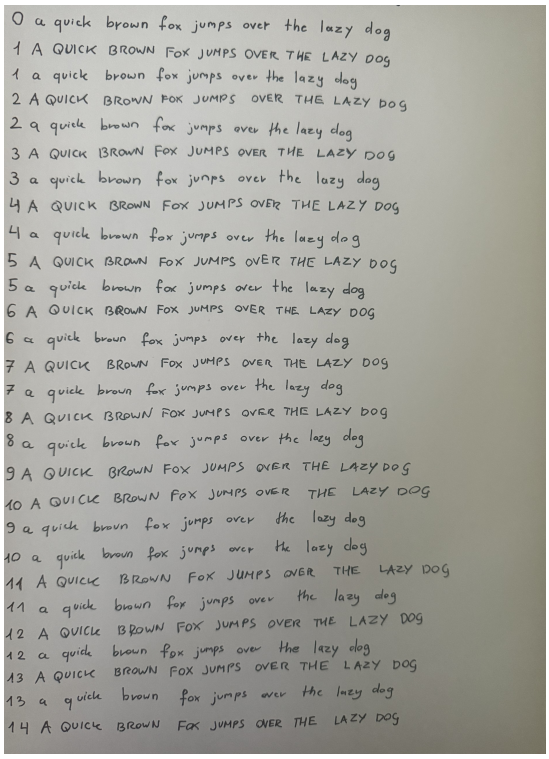
1. We could have used the GNHK dataset, and filtered out the samples that we did not need. However, this would be a time consuming process, and we would have to manually go through each sample.

2. The target population (two writters) are asked to send samples of their handwritting, of this exact same sentence.

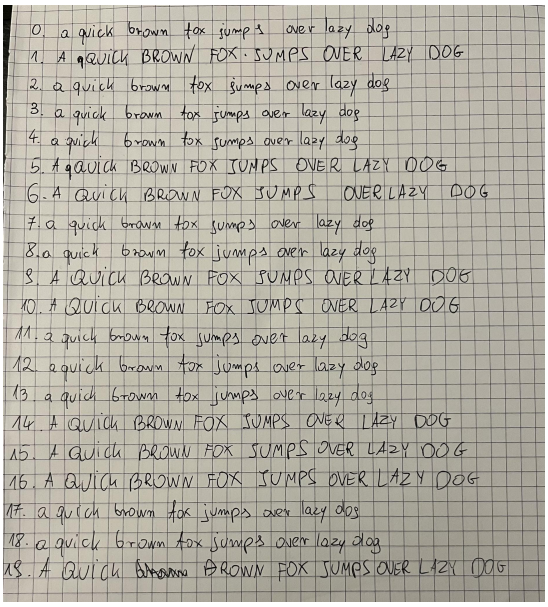
We decided to go with the second option, as it would be much easier to collect and it would be more represntative of the target population's needs. The sentence chosen was "The quick brown fox jumps over the lazy dog". This sentence was chosen because it contains all the letters of the english alphabet, and it is a common pangram used in handwriting samples. Additionally, the users were asked to write this sentence in all lowercase or uppercase, alternating between them. They were also asked to number each line, to get alphanumeric data.

5 Raw Data (samples and syntactic characteristics):

As the method collection procedure was described in the previous section, the raw data was collected from the end target users. Here is a sample image from the data from both users:



(a) User 1 -submitted handwriting



(b) User 2 -submitted handwriting

Figure 2: Side-by-side comparison of dataset and user handwriting samples

The comparison of the two samples as shown in figure 2 shows that, there are some similarities between the two users writting styles such as the letter “q” and “Q”, are close to identical between the two user. Despite the similarities, there are also some differences between the two users, such as the letter “f” , which are different between the two users.
This is important to note, as a decision must be made on to train two separate models for each user, or to train one model that recognises both handwritting styles. To understand which choice is more beneficial, we need to understand how we aim to make our model work. First and foremost, a sample must be normalized. This is done to get rid of geometric deformations, shadows, background. The normalization aims to create a binary image, where 0 means the background, and 1 means the text. This is done using a combination of histogram equalization and thresholding.

After the normalization is done, the image is split into individual letters. This helps creating a dataset of letters, with 61 classes (26 lowercase, 26 uppercase, 10 digits).

After the processing is done, we are left with a very normalized data. This leads to the question weather it is efficient to train different models for each user, or not. We believe training two models is feasible, as once the final product runs, a user can "log into" his profile, and have a personalized model. This will increase the accuracy of both models.

However, for the sake of this project, we will train one model, on the name of user 1, and if that will be a seamless process, we will retrain it on mixed data.

6 Normalization of Raw Data:

Lets choose User 1 as an example of Normalization. In report 1, number of image processing techniques were mentioned, such as histogram equalization, thresholding, resizing, and denoising. But firstly we need to start with smoothening of the image. There are three smoothening filers.

- **Gaussian filter**: Greate for general noise reduction
- **Median filter**: Attempts to preseves edges, while removing noise
- **Bilateral filter**: Provides a balance between smoothening and edge preservation.

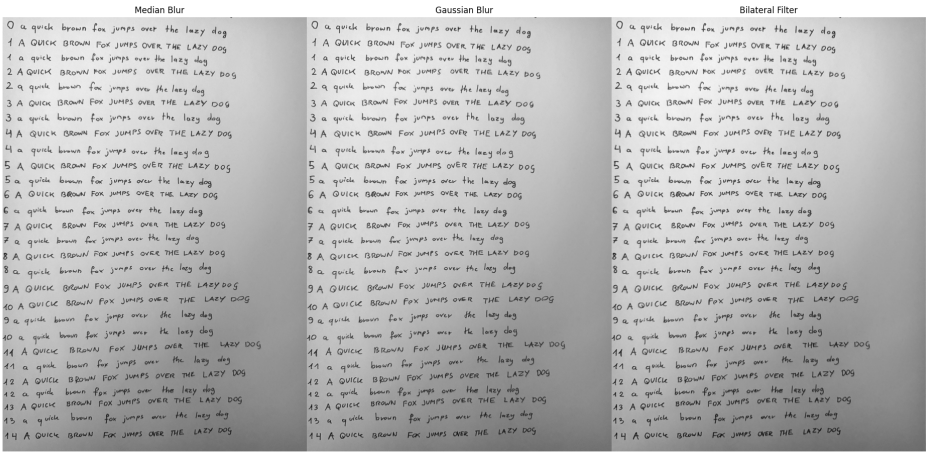


Figure 3: Comparision of three Smoothening filters applied to the same image

The figure 3 shows the comparison of the three filters. Although all of them look near identical when applied to the image, a subjective view on the comparison is that median blur makes the image darker, but lighting more balanced, than the other 2. Due to this reason, we decided to use the median filter for our project.

Moving onto thresholding, again we consider three different methods:

- **Otsu's method**: Minimizes intra-class variance. Good for images where there is clear distinction between foreground and background.
- **Adaptive Mean thresholding**: Averages neighbouring pixels and subtracts a constant value. Great for varying lighting conditions.
- **Gaussion thresholding**: Similar to adaptive mean, but uses a Gaussian weighted average. Great for images with not too dynamic lighting.

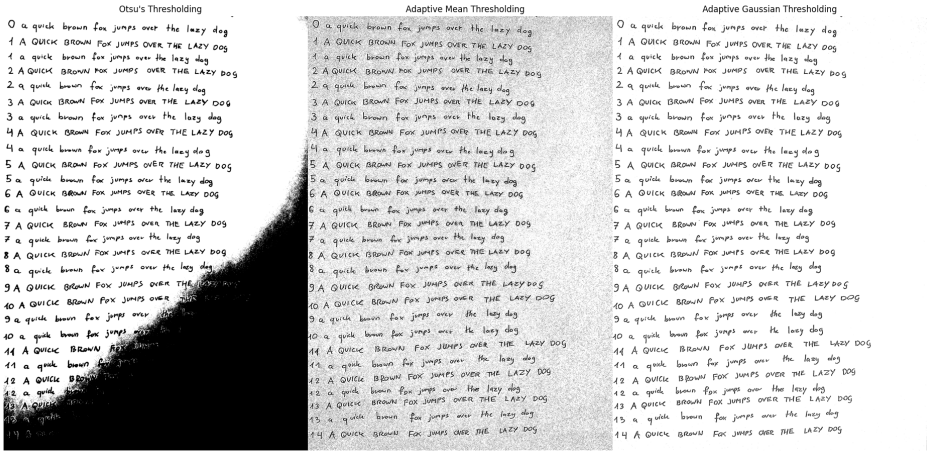


Figure 4: Comparision of three Thresholding methods applied to the same image

7 Extraction of words/letters:

8 Closing Thoughs:

References

[1] Alex W. C. Lee, Jonathan Chung, and Marco Lee. *GNHK: A Dataset for English Handwriting in the Wild*. 2021.

[2] University at Buffalo. Accessed: March 16, 2025. URL: https://cedar.buffalo.edu/Databases/index.html.

[3] Christopher J.C. Burges Yann LeCun Corinna Cortes. Accessed: March 16, 2025. URL: https://web.archive.org/web/20240119012243/http://yann.lecun.com/exdb/mnist/.

[4] Chris Tensmeyer and Tony Martinez. "Document Image Binarization using Recurrent Neural Networks". In: *arXiv preprint* arXiv:1509.03456 (2015). arXiv: 1509.03456 [cs.CV]. URL: https://arxiv.org/pdf/1509.03456.

[5] OpenCV Documentation Team. *Histogram Equalization in OpenCV*. OpenCV Documentation. Accessed: 12 March 2025. 2024. URL: https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html.

[6] OpenCV Team. *Image Thresholding - OpenCV Documentation*. Accessed: March 12, 2025. 2024. URL: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html.