FIT: Report 1

Handwriting detection and Note Conversion using OCR

, Pietka Ł, Barot H, Krzyszosiak 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 tool utilized in this project are existing OCR models 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 and our own handwritten notes.

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 [**Lee2021s**]	Camera-captured handwritten notes with 1080p to 4k resolution, lighting, and backgrounds.	515	172

Table 1: Comparison of Handwritten Text Datasets

2.2 Data Preprocessing

The data obtained for training and usage may come from different sources due to equipment used (in this case a camera) or the quality standards, and thus 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 which obscure 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 images to a standard resolution is a crucial preprocessing step. We used OpenCV to resize all images to a standard resolution of 300 DPI, which is a common resolution for document scanning and provides a good balance between detail and computational efficiency.
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 in aligning 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 may 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. Additionally, we applied denoising techniques using OpenCV to remove any 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.

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`.