Hong Kong Metropolitan University

School of Science and Technology

BSc. (Hons) in Data Science and Artificial Intelligence
COMP S461F Data Science Interim Report
2024-2025

Deep Learning-Based System for Handwritten Character Recognition in Exam Papers

by

Chu Ying Ying,
Tam Oi Laam,
Tsao Sai Chak,
Wong Hok Man,
Cheung Yau Cheuk,
Lung Kwan Chak

Supervisor: Dr. Jimmy Kang

Date: 24 April 2025

Abstract

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Handwritten character recognition (HCR) is a pivotal research area in computer vision and machine learning, with significant applications in automating tasks such as exam grading, form processing, and digitizing historical documents. Deep learning has revolutionized HCR systems, offering improved accuracy and adaptability to diverse handwriting styles and complex document layouts.

In educational environments, manually processing exam papers is labor-intensive and susceptible to errors, especially when extracting details like course codes, exam numbers, student numbers, and handwritten answers.

A deep learning-based system can enhance efficiency and accuracy by recognizing handwritten characters and exporting structured data, thereby reducing the administrative burden on educators.

Recent progress in convolutional neural networks (CNNs) and recurrent neural networks (RNNs) has demonstrated high potential for accurate HCR. However, challenges persist, including handwriting variability, noise in scanned documents, and the need for intuitive interfaces to ensure practical adoption.

This project tackles these issues by developing a system that integrates deep learning models, manual region cropping capabilities, and a graphical user interface (GUI) to process exam papers effectively. The system aims to deliver high recognition accuracy, support manual extraction of specific document regions, and provide structured data outputs, meeting the demands of educational institutions.

1.2 Objectives

The primary aim of this project is to design and implement a deep learning-based system for handwritten character recognition in exam papers, with the following specific objectives:

1. **High Model Accuracy**: Achieve a recognition accuracy of at least 95% for all deep learning models to ensure reliable character recognition across varied handwriting styles.
2. **Manual Region Cropping**: Enable users to manually crop and extract specific regions from exam papers, such as course codes, exam numbers, and student numbers, to support precise data extraction.
3. **Character Set Recognition**: Develop the system to recognize English letters (A-Z) and digits (0-9), encompassing the primary characters used in exam paper annotations.
4. **User-Friendly GUI Application**: Create a graphical user interface (GUI) application that simplifies the process of uploading and processing PDF exam papers, ensuring accessibility for educators and administrators.
5. **Structured Data Export**: Implement functionality to export recognized data into Excel spreadsheets, facilitating integration with existing grading and record-keeping systems.

These objectives drive the development of a practical, user-centric system tailored to the needs of handwritten character recognition in educational settings.

1.3 Significance

1.4 Values

Chapter 2

Literature Review

2.1

Chapter 3

Method

The methodology for this handwritten character recognition (HCR) system involves a multi-stage pipeline: input handling, preprocessing, contour detection, classification, and structured data export. The system processes scanned exam papers in PDF format, extracts handwritten characters (digits 0-9 and letters A-Z), and outputs the recognized data into Excel spreadsheets. Two convolutional neural network (CNN) models, DigitNet and LetterNet, are employed for classification, supported by a binary classifier (BinaryNet) to distinguish between digits and letters.

3.1 Input Handling

The system begins by accepting PDF exam papers as input. Users can manually crop specific regions of interest (course codes, exam numbers, student numbers) via a graphical user interface (GUI). This manual region selection ensures precise extraction of relevant handwritten data, addressing the objective of supporting targeted data extraction.

3.2 Preprocessing

Preprocessing is a critical step to enhance the quality of input images and standardize them for model inference. The following steps are applied:

1. **PDF to Grayscale Conversion**: PDFs are converted to enhanced grayscale images using Contrast Limited Adaptive Histogram Equalization (CLAHE) to improve contrast and visibility of handwritten characters.
2. **Adaptive Thresholding**: Images undergo adaptive thresholding to create binary images, separating foreground (handwritten characters) from the background.
3. **Line Removal**: Vertical and horizontal lines (table grids in exam papers) are removed to reduce noise and isolate characters.
4. **Noise Cleanup**: Additional noise is cleaned using morphological operations to ensure clarity of character contours.

5. **Transformers and Augmentation**: Images are resized to 28x28 pixels (MNIST format) and converted to PyTorch tensors. Normalization is applied with a mean of 0.1307 and a standard deviation of 0.3081. Data augmentation techniques, including random affine transformations (e.g., ±5° rotation, 10% translation), color jitter (brightness/contrast tweaks with probability 0.3), and salt-and-pepper noise (probability 0.2), are used to mimic real-world distortions and improve model robustness.

3.3 Contour Detection and Classification

The classification process involves three steps:

1. **Contour Detection**: Character contours are identified in the binary image, and nearby contours are merged to ensure each character is treated as a single entity.
2. **Region Extraction**: Each detected contour is extracted and resized to 28x28 pixels (MNIST format) for compatibility with the models.
3. **Two-Step Classification**:
    - A binary classifier (BinaryNet) first determines whether the character is a digit or a letter.
    - Based on the binary classification result, the character is passed to either DigitNet (for digits) or LetterNet (for letters) for specific classification.

3.4 Model Architecture

Two CNN models, DigitNet and LetterNet, are designed to recognize digits (0-9) and letters (A-Z), respectively. Both models follow a similar architecture but are trained on different datasets tailored to their respective character sets.

- **DigitNet Architecture**:
    - Input: 28x28x1 grayscale image
    - 3x3 Conv1 (64 filters) → 2x2 Pooling → 3x3 Conv2 (128 filters) → 2x2 Pooling → 3x3 Conv3 (256 filters)
    - Dropout (to prevent overfitting) → Flatten

    - FC1 (256 units) → FC2 (256 units) → FC3 (10 units, corresponding to digits 0-9)
    - Output: Logits for digits [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
    - Loss Function: Cross-entropy loss
    - Activation: Sigmoid
- **LetterNet Architecture**:
    - Input: 28x28x1 grayscale image
    - 3x3 Conv1 (64 filters) → 2x2 Pooling → 3x3 Conv2 (128 filters) → 2x2 Pooling → 3x3 Conv3 (256 filters)
    - Dropout → Flatten
    - FC1 (256 units) → FC2 (256 units) → FC3 (26 units, corresponding to letters A-Z)
    - Output: Logits for letters [A, B, C, ..., Z]
    - Loss Function: Binary logistic loss
    - Activation: Sigmoid

3.5 Structured Data Export

After classification, the recognized characters are organized into structured data (course codes, exam numbers, student numbers) and exported as Excel spreadsheets. The GUI facilitates this process by allowing users to map extracted characters to specific fields, ensuring seamless integration with existing grading systems.

3.6 Implementation Details

The system is implemented using PyTorch for model development and training. The GUI is built using a Python-based framework to ensure user-friendliness.

Chapter 4

Results

This section presents the performance evaluation of the three deep learning models—DigitNet, LetterNet, and BinaryNet—developed for handwritten character recognition in exam papers. Each model was trained on specific datasets (MNIST and EMNIST) and evaluated based on loss and accuracy metrics over multiple epochs. The results demonstrate the models' effectiveness in achieving the project objective of at least 95% accuracy, with low loss values, ensuring reliable performance for real-world exam paper processing.

4.1 DigitNet Model Performance

The DigitNet model is a classification model designed for handwritten digit recognition (0-9). It was trained on the MNIST dataset, a standard benchmark for digit recognition tasks. The training process spanned 20 epochs, and the model's performance is illustrated in Figure 4.1, which plots loss and accuracy against epochs for both training and test sets.

The loss plot shows a rapid decrease in both train and test loss during the early epochs, stabilizing around 0.0155 by the end of training. Similarly, the accuracy plot indicates a quick improvement, with train and test accuracy converging at 99% (0.99). The close alignment of train and test metrics suggests good generalization performance without overfitting. These results confirm that DigitNet meets the project's accuracy goal of 95% and demonstrates robust performance for digit recognition tasks.

Figure 4.1: DigitNet Performance on MNIST Dataset

4.2 LetterNet Model Performance

The LetterNet model is a classification model for handwritten uppercase letters (A-Z). It was trained on the EMNIST dataset, specifically using the subset containing only uppercase letters, as lowercase letter recognition was not required for this project. The training process spanned 15 epochs, and the model's performance is shown in Figure 5.2.

The loss plot reveals that both train and test loss decrease steadily, stabilizing at an impressively low value of 0.0015. The accuracy plot shows that test accuracy closely follows train accuracy, both reaching 98% (0.98) by the end of training. This close alignment indicates strong generalization and minimal overfitting. LetterNet surpasses the project's accuracy target of 95% and achieves the lowest loss among all models, highlighting its effectiveness for letter recognition tasks.

Figure 4.2: LetterNet Performance on EMNIST Dataset

4.3 BinaryNet Model Performance

The BinaryNet model is a binary classification model designed to distinguish between uppercase letters and digits. It was trained on a subset of the EMNIST dataset that includes only digits and uppercase letters, aligning with the project's character set requirements. The training process spanned 20 epochs, and the performance is depicted in Figure 5.3.

The loss plot shows a consistent decrease in both train and test loss, stabilizing at 0.0132. The accuracy plot demonstrates effective learning, with train and test accuracy converging at 99% (0.99). The steady improvement in accuracy and reduction in loss indicate strong training and generalization performance. BinaryNet exceeds the project's accuracy goal of 95%, making it a reliable component for the two-step classification process.

Figure 4.3: BinaryNet Performance on EMNIST Dataset

4.4 Overall Model Performance

All three models—BinaryNet, DigitNet, and LetterNet—exceeded the project's accuracy target of 95%, with loss values below 0.02. LetterNet achieved the lowest loss at 0.0015, while BinaryNet and DigitNet both reached an impressive 99% accuracy. These results demonstrate the models' reliability for handwritten character

recognition tasks. The strong generalization performance ensures their suitability for processing real exam papers, as evidenced by the consistent convergence of train and test metrics across all models.

4.5 Application Results

The practical application of the system was tested on real exam papers, focusing on extracting course codes, exam numbers, and student numbers. Figures 5.4 to 5.6 showcase the system's performance through the GUI, highlighting the recognition results for these fields.

- **Student Number Recognition**: Figure 5.4 shows the recognition of a student number "12531103". The system correctly identified all digits, with the processed images and classifications (e.g., "Type: Digit, Result: 1") confirming the accuracy of DigitNet.
- **Exam Number Recognition**: Figure 5.5 illustrates the recognition of an exam number "C9170051". The system accurately classified the mix of letters and digits (e.g., "Type: Letter, Result: C" and "Type: Digit, Result: 9"), demonstrating the effectiveness of the two-step classification process involving BinaryNet, LetterNet, and DigitNet.
- **Course Code Recognition**: Figure 5.6 displays the recognition of a course code "STATS313F". The system successfully identified the combination of letters and digits, with classifications such as "Type: Letter, Result: S" and "Type: Digit, Result: 3", further validating the models' performance in a real-world scenario.

Figure 5.4: Student Number Recognition Result

Figure 5.5: Exam Number Recognition Result

Figure 5.6: Course Code Recognition Result

Chapter 5

Discussion

5.1 Strengths

The developed handwritten character recognition (HCR) system demonstrates several key strengths. First, after model optimization, all three models—DigitNet, LetterNet, and BinaryNet—achieved high accuracy, exceeding the project's target of 95%. This ensures reliable recognition of digits (0-9) and uppercase letters (A-Z), critical for processing exam papers. Second, the user-friendly graphical user interface (GUI) simplifies PDF uploads and manual region selection, making the system accessible to educators and administrators. Finally, the robust preprocessing pipeline effectively handles noise, such as lines and background artifacts in scanned documents, ensuring clean input data for classification.

5.2 Challenges

Despite the system's successes, several challenges were encountered during development and testing:

1. **Initial Model Accuracy Below Target**: Early iterations of the deep learning models, using a softmax activation function, achieved accuracy below the desired 95% threshold, indicating suboptimal performance.
2. **Course Code Detection Issues**: Detecting the course code "STAT S313F" proved difficult due to noise and spacing issues, leading to character merging (e.g., "S3" being interpreted as a single entity).
3. **Misclassification Errors**: Some misclassification errors occurred, such as the uppercase letter "S" being recognized as the digit "5," highlighting limitations in the BinaryNet model's ability to distinguish between similar-looking characters.

5.3 Addressing the Challenges

To overcome these challenges, the following solutions were implemented or are proposed:

- **Improving Deep Learning Model Accuracy**: The initial low accuracy issue was addressed by switching from softmax to logits with cross-entropy loss. This change boosted the accuracy of all models above 95%. DigitNet and BinaryNet achieved 99% accuracy, while LetterNet reached 98% with the lowest loss (0.0015).
- **Enhancing Course Code Sequence Detection**: To improve the detection of course codes like "STAT S313F," refinements to the contour detection algorithm are proposed to better handle spaces between characters. Additionally, advanced noise reduction techniques, such as Gaussian blur, will be applied to prevent character merging. Figure 6.1 illustrates the improvement in course code detection before and after applying these enhancements, showing clearer separation of characters in "STAT S313F."

  Figure 6.1: Improvement in Course Code Detection

- **Reducing Misclassification Errors**: To address misclassifications (e.g., "S" as "5"), BinaryNet will be retrained with a larger dataset of handwritten letter samples to improve its ability to differentiate between digits and letters. Additionally, post-processing rules will be implemented to enforce expected course code formats (e.g., ensuring the first four characters are letters in "STAT S313F"), further reducing errors.

5.4 Implications and Future Improvements

The solutions implemented and proposed will enhance the system's reliability for real-world applications, such as exam paper processing in educational settings. The high accuracy and robust preprocessing already make the system a valuable tool, but addressing the remaining challenges will ensure even greater precision. Future work could also explore incorporating additional character sets (e.g., lowercase letters) and further optimizing the GUI for scalability across different document types.

References