CSE 3200: SYSTEM DEVELOPMENT PROJECT

HANDWRITTEN CHARACTER AND DIGIT RECOGNITION

BY

ASNUVA TANVIN

ROLL: 1707005

SAZIA RAHMAN

ROLL: 1707012

SUPERVISOR

AL-MAHMUD

[illegible]
23/06/2021

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

JUNE 24, 2021

APPROVAL

This project report has been submitted for examination with the approval of our supervisor.

SUPERVISOR

AL-MAHMUD

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

CONTENTS

CHAPTER 1

INTRODUCTION

Handwritten character recognition system is one of the most sought-after systems in today's world. The problem of recognizing handwritten characters is a tough one to solve because every individual has a different handwriting pattern. But if this problem can be solved accurately then it will bring a revolutionary change in our lives by saving thousands of minutes that are wasted to digitalize handwritten documents. Keeping this motivation in mind we have attempted to develop a system that can covert texts from an image to editable digital data. Our developed system extracts characters as separate image from input image and our classifier tries to identify the characters in those images. Then our system provides the users the predictions as digital data in an editable document as well as an output image that shows the prediction value of each character of the input image. The user can edit the text if they want and save the output as text document.

1.1 Objectives

- To segment a text image into separate character images
- To recognize English capital letters and English digits
- To provide output to user in the form of an editable document

1.2 Overview

In this project, we have applied Convolutional Neural Networks (CNNs) for handwritten English capital letters and digit recognition. We have improved a previously developed CNN architecture [1] by hyperparameter tuning and by reducing the overfitting of the model. Experiments are evaluated on MNIST digit dataset and a merged dataset containing MNIST digit and A-Z handwritten alphabets, with recognition rates of 99.47% for MNIST and 98.94% for merged dataset respectively. In the user interface part, the text image taken from the user is processed through seven phases before using the trained model to classify characters of the text image. In post processing, predicted characters are rearranged as they were in the text image. The user receives an output image labeled with the predictions for each character of the image and an editable document that is the digitalized version of the text in the image. A user-friendly graphical interface is associated with the system to make it simple for user to operate the system.

1.3 Workflow Diagram

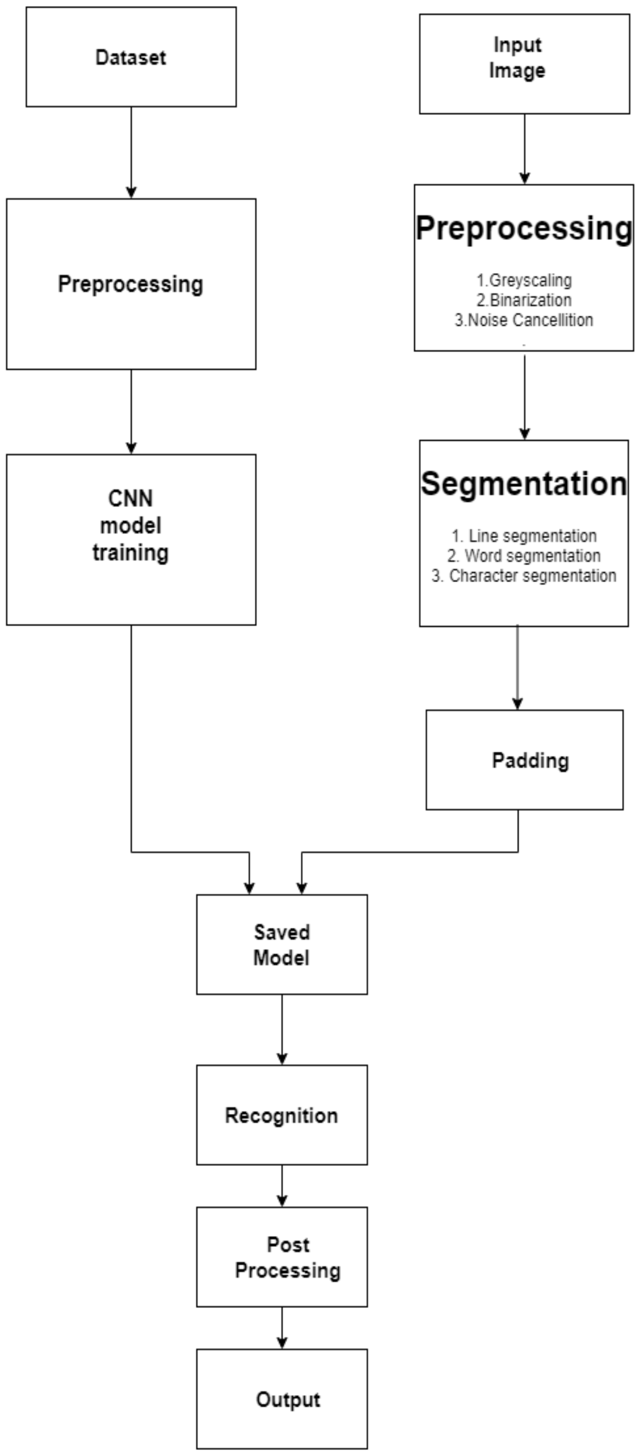


Fig 1: Workflow Diagram

CHAPTER 2

TOOLS AND LITERATURE REVIEW

This section provides information about the platforms, frameworks, libraries, datasets we have used and the hardware requirements to run and build this system. We also provide information about the previous attempts of various researchers to build a system like this.

2.1 Related Work

This section surveys related works on character recognition systems. For recognition of handwritten English characters, Pal and Singh have implemented the Multi-Layer Perceptron [5]. The features have been extracted and analyzed by comparing its features for character recognition using boundary tracing and its Fourier descriptors. It has been also analyzed to determine the number of hidden layers required to attain high accuracy. It has been reported with 94% accuracy of Handwritten English characters with very less amount of training time. Neves et al. have implemented a Support Vector Machine (SVM) based offline handwritten character recognition, which gave better accuracy compared to the Multi-layer perceptron classifier for the NIST SD19 standard dataset [6]. Although, MLP is suitable for segmentation of nonlinear separable classes, but it can easily trap into local minima. An implementation of deep neural network models has been established by Younis and Alkhateeb so that it can extract the important features without unnecessary pre-processing [7]. Such, a deep neural network has demonstrated an accuracy of 98.46% on MNIST dataset for handwritten OCR problem. A multilayer CNN using Keras with Theano has been implemented by Dutt and Dutt in [3] with accuracy of 98.70% on MNIST dataset. It provided better accuracy compared to SVM, K-Nearest Neighbor and Random Forest Classifier. A comparative study of three neural network approaches has been provided by Ghosh and Maghari [4]. The results showed that Deep Neural Network was the most efficient algorithm with accuracy of 98.08%. Another implementation of CNN is done by Jana and Bhattacharyya [1] on MNIST dataset resulting in 98.85% accuracy. This implementation took less training time compared to other implementations. Kishan, David and Femi designed a complete Optical Character Recognition System using CNN as classifier and EMNIST balanced dataset to train the CNN model [2]. The system can predict English alphabets, digits and 12 Tamil characters. Result showed 40 true positives and 8 false positives for the 48 characters that they tried to classified.

2.2 Software Requirements:

- Language: Python
- Platform: Jupyter notebook, Spyder, Anaconda
- Libraries: tensorflow, Keras, OpenCV, Matplotlib, Tkinter, Numpy, Pandas, Seaborn

2.3 Hardware Requirements:

- Intel Core i5-7200U CPU
- 8 GB RAM
- Internal Graphics: Intel(R) HD Graphics 620
- Rendered Graphics: NVIDIA GeForce 930MX

2.4 Datasets for Training and Testing CNN Model

The MNIST dataset is a subset of NIST database. The MNIST dataset is a collection of 70,000 images of handwritten digits. The dataset is divided into 60,000 images for training and 10,000 images for testing. In the training and test label files, the label values are 0-9. All images have resolution of 28 X 28 and the pixel values are in the range 0-255(grey-scale). The background of digit is represented by 0 grey value (black) and a digit is represented by 255 gray value (white) as shown in figure 2. [11]

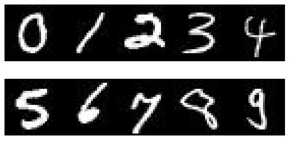


Fig 2: Sample images from MNIST dataset

A-Z Handwritten Alphabet dataset is also a subset of NIST database. It is a customized dataset owned by Sachin Patel and licensed as a public domain. This dataset is collected from Kaggle. The dataset contains 26 folders (A-Z) containing handwritten images in size 28 X 28 pixels, each alphabet in the image is center fitted to 20 X 20 pixel box and has padding in all four sides as shown in figure 3. Each image is stored as Grey-level. The dataset is a collection of 3,72,450 images of handwritten capital letters. [12]

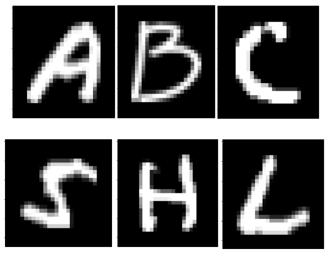


Fig 3: Sample images from A-Z handwritten capital letters dataset

For training our CNN model to recognize handwritten capital letters and digits we have merged MNIST dataset and A-Z handwritten capital letters dataset to create a custom dataset. The dataset contains grey scale images of 26 capital letters and 10 digits which makes 36 different categories. The total number of images in the dataset is 4,42,450 every one of them having a size of 28 X 28 pixels. The merged dataset has 36 labels. 26 label values are A-Z and remaining 10 label values are 0-9.

CHAPTER 3

SYSTEM ARCHITECTURE

Handwritten character and digit recognition is a translation problem of human writings into machine-editable text format. Our attempts to develop a system to solve this problem is described below.

3.1 Proposed Methodology

Recognizing text from real images become very complicated due to a wide range of variation in textures, backgrounds, font size and shading. The three basic steps of character recognition are segmentation, feature extraction and feature classification. The last two steps are where Convolutional Neural Networks (CNNs) come handy. CNN is designed to automatically and adaptively learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification. A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers, and a small grid of parameters called kernel, an optimizable feature extractor, is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. Therefore, CNN architecture is proposed for character recognition from the images of handwritten capital letters and digits. [8]

The proposed system manually performs character segmentation on images. The image is converted to binary format and then noise is removed to make the target information clear. Image is then segmented into lines, words and finally characters. All the characters of the image are then cropped and stored as separate images. Following that padding is added to all the images. These character images are classified with the help of a trained CNN classifier. Finally, results are rearranged as the image text and are given as output to the user.

3.2 Sequence Diagram

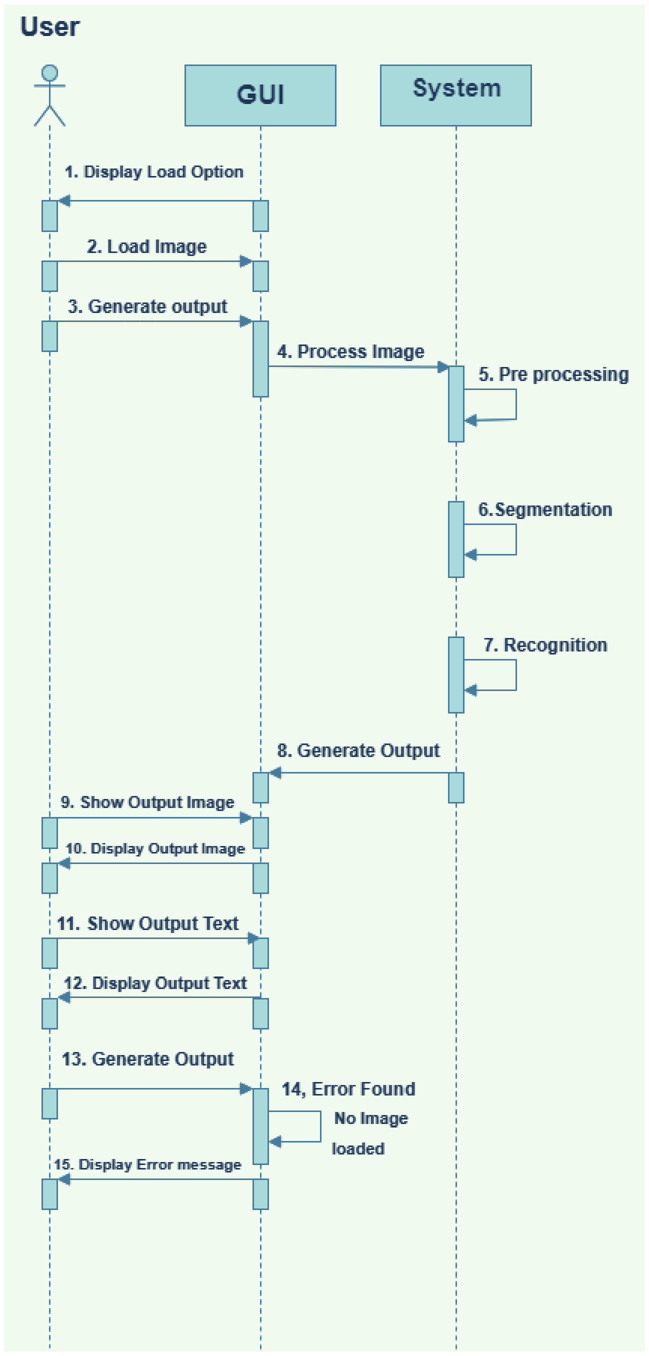


Fig 4: Sequence Diagram of the System

3.3 Image Segmentation Procedures

After finding the best model for classification the next steps were followed to process input text image so that it can be converted to separate character images that can be used as input to the classifier.

3.3.1 Pre-processing

The image captured by the camera is a color image, which contains a huge amount of information. The content of the picture can be simply divided into the foreground information and the background information. In order to allow the computer to recognize text faster and better, the color image needs to be processed so that in the picture only foreground information and background information are left. We simply define the foreground information as white and the background information as black, so we aim to obtain a binary image.

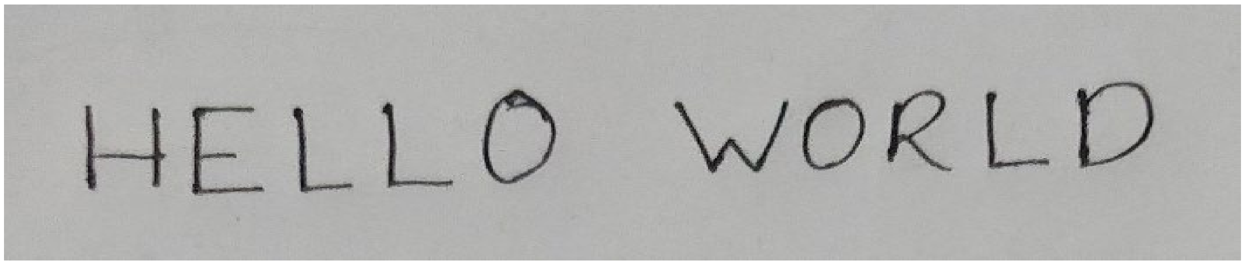


Fig 5: Raw Image

3.3.1.1 Grayscaling

Grayscaling is the process of converting an image from other color spaces such as RGB, CMYK, HSV etc. to shades of gray. It varies between complete black and complete white. Greyscaling is done to reduce dimensions of an image. In an RGB image there are three dimensions whereas in a greyscaled image there is only one dimension. Using a greyscaled image helps to reduce model complexity.

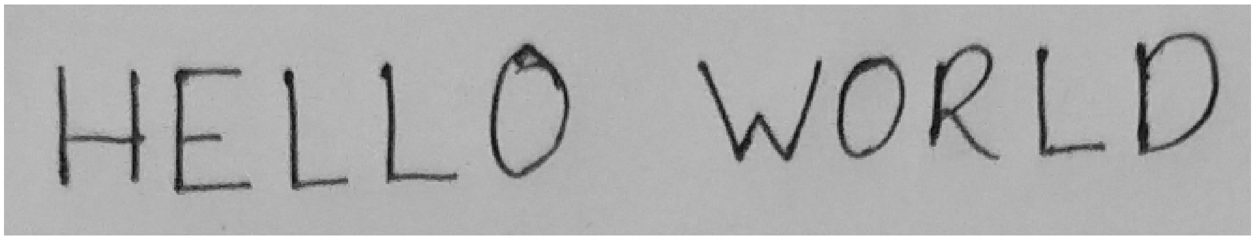


Fig 6. Greyscaled Image

3.3.1.2 Binarization

Image binarization is the process of taking a grayscale image and converting it to black-and-white, essentially reducing the information contained within the image from 256 shades of gray to two: black and white, a binary image. This is sometimes known as image thresholding. The process of binarization works by finding a threshold value in the histogram – a value that effectively divides the histogram into two parts, each representing one of two objects (or the object and the background). In this system we have used adaptive thresholding algorithm which determines the threshold value for a pixel based on a small region around it. So we get different thresholds for different regions of the same image which gives better results for images varying illumination.

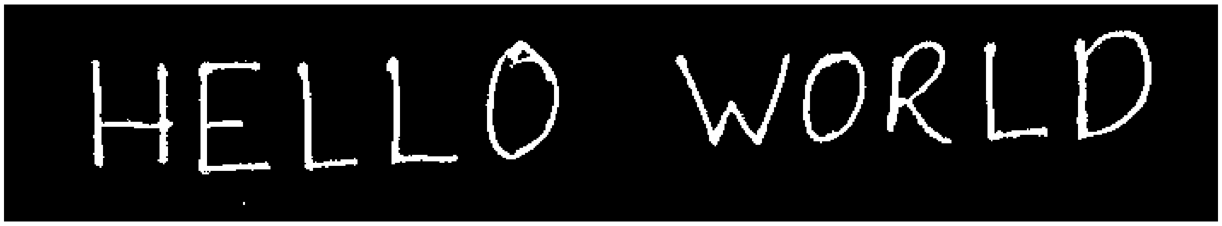


Fig 7. Binary Image

3.3.1.3 Noise Removal

In this phase morphological operations are done on the binary image. Two basic morphological operations are Erosion and Dilation. The erosion operation is useful for removing small white noises. The dilation operation increases the white object area after erosion so the object in the image does not shrink. We have performed closing operation which is another name for Dilation followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object.

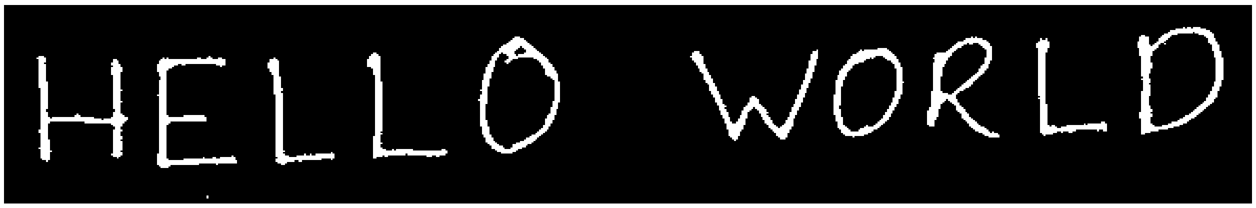


Fig 8: Image after noise removal

3.3.2 Segmentation

Segmentation phase consist of following steps:

3.3.2.1 Line Segmentation

For the preprocessed image, number of pixels with a pixel value of 0 in each row is counted and the rows that have the most pixels with 0 pixel values are considered as empty lines. In an ideal pre-processed image only the target foreground information is white, and the background part is completely black. In that case, rows full of pixels with intensity values equal to 0 can be found. But because the image noise cannot be completely removed, the pre-processed image is mixed with a lot of noise. So resulting information will usually not contain any rows that only have 0 as pixel intensity values. Therefore, a threshold value needs to be set, and the portions where most pixels have intensity value lower than the threshold value are considered empty lines.

3.3.2.2 Word Segmentation

After line segmentation words in each line are separated. First, average character width is calculated. If the space between two consecutive characters is greater than the average character width, it can be assumed that the two characters belongs two different words and the word segmentation is performed by separating those two characters as two different words.

3.3.2.3 Character Segmentation

The contour detection method is used to segment each character in the word and contour detection is performed using the help of findContours function encapsulated in the OpenCV library. Contours refer to a curve that has the same color or density and connects all continuous points. The work of detecting contours is very useful for shape analysis and object detection and recognition. In order to improve the accuracy of contour detection, before contour detection, the picture must first be binarized.

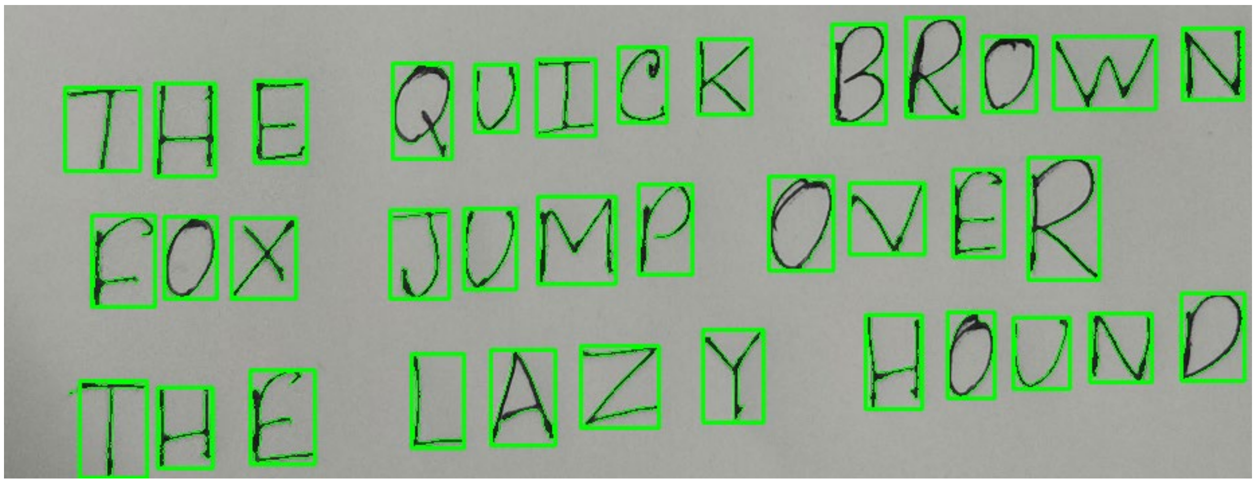


Fig 9: Image after line, word and character segmentation

After segmentation characters from the image are cropped as separate images and saved in a folder named “result”.

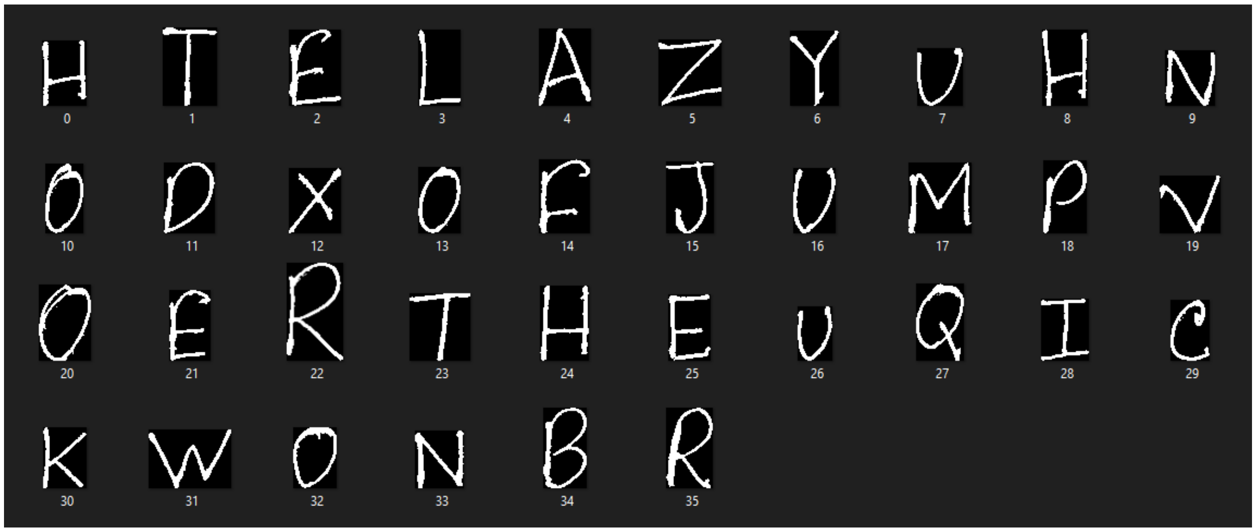


Fig 10: Cropped image saved in a folder

3.3.3 Preparing for Classification

The input images need to have the same distribution as images that were used to train the CNN model. For this reason, we resize the images to 28 X 28 and add padding to them.

3.3.3.1 Padding

Segmented character images are taken and resized as 20 X 20 pixel images. Then 4 pixels with intensity value equal to 0 are added to each side of the image making them a 28 X 28 pixel images.

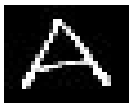


Fig 11: An image after adding padding

3.4 Developing CNN Model for Feature Extraction and Classification

The processed images are now ready to be used as inputs to a trained model that can recognize the characters in them. We have chosen CNN as feature extractor and classifier for this system. We have modified the CNN model proposed here [1] and developed two separate models for two separate datasets. One model that fits the MNIST dataset and another model that fits the merged dataset described in chapter 2.4.

3.4.1 Convolutional Neural Network Architecture

The system has two parts: the image segmentation part and the classification part.

The first phase of the system creates a separate image for every character in the text image. Grayscaling, thresholding and noise removal operations are done on input image. Then image is segmented first by lines, then by words and finally by characters. Segmented character images are resized to 20 X 20 pixels' images. 4 pixels with 0 intensity value are added to each side of each image. The resulted images are sized 28 X 28 pixels at the end of the first part.

The second part is where images are classified using a CNN model. The CNN architecture proposed in this project is a modified version of the CNN architecture proposed here [1]. The model has two modified versions: one that is modified to train MNIST dataset and the another that is modified to train the merged dataset. The architectural description is given below:

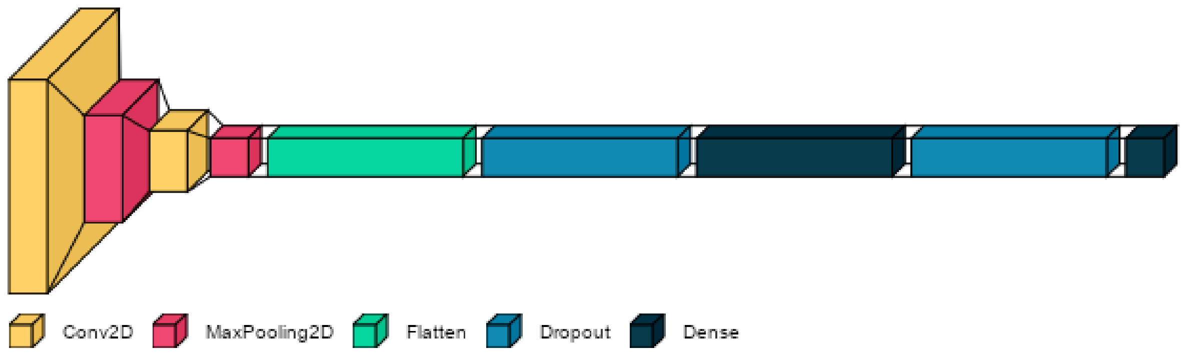


Fig 12: The proposed CNN architecture

The CNN architecture has one input layer, one output layer and 9 hidden layers. The input layer is an image of size 28 X 28. At first, 32 filters of window sized 5 X 5 with a ReLU activation function for nonlinearity are used in first convolutional layer. After that, a pooling layer is used to perform a down-sampling operation using a pool size 2 X 2 with stride by 2. As a result, the image volume is reduced. After that, 64 filters with window size 7 X 7 with a ReLU activation function for nonlinearity is used in another convolutional layer. Then, another pooling layer is used to perform down-sampling operation using a pool size 2 X 2 with stride by 2. Then there is a flatten layer to convert the 2D output matrix to a 1D array.

After the flatten layer the CNN architecture for MNIST dataset is followed by Dropout of 50%. Then a fully connected layer is used with 1024 output nodes. Again another dropout of 50% is used to reduce overfitting. Lastly, a fully connected layer with 10 output nodes is placed to receive network results for 10 digits (0-9).

In case of the architecture for merged dataset (A-Z capital letters + MNIST) the flatten layer is followed by a dropout of 10%. Then a fully connected layer is used with 1024 output nodes. Again another dropout of 10% is used to reduce overfitting. Lastly, a fully connected layer with 36 output nodes is placed to receive network results for 10 digits (0-9) and 26 capital letters (A-Z). The architecture is shown in figure 12.

3.5 Post Processing

In the post processing phase the predictions of the segmented characters were rearranged as they were in the input image.

CHAPTER 4

GRAPHICAL USER INTERFACE FOR THE SYSTEM

A GUI (graphical user interface) is a system of interactive visual components for computer software [10]. Graphical User Interface of this system is designed in a user friendly way using Tkinter framework. The interface has all the features needed to take an image as input to the system, generate output and show it to the user.

At first the GUI will show options to load image from any folder of the computer. User can load input image from their device using **Browse** button. User can exit the application by pressing **Exit** button.

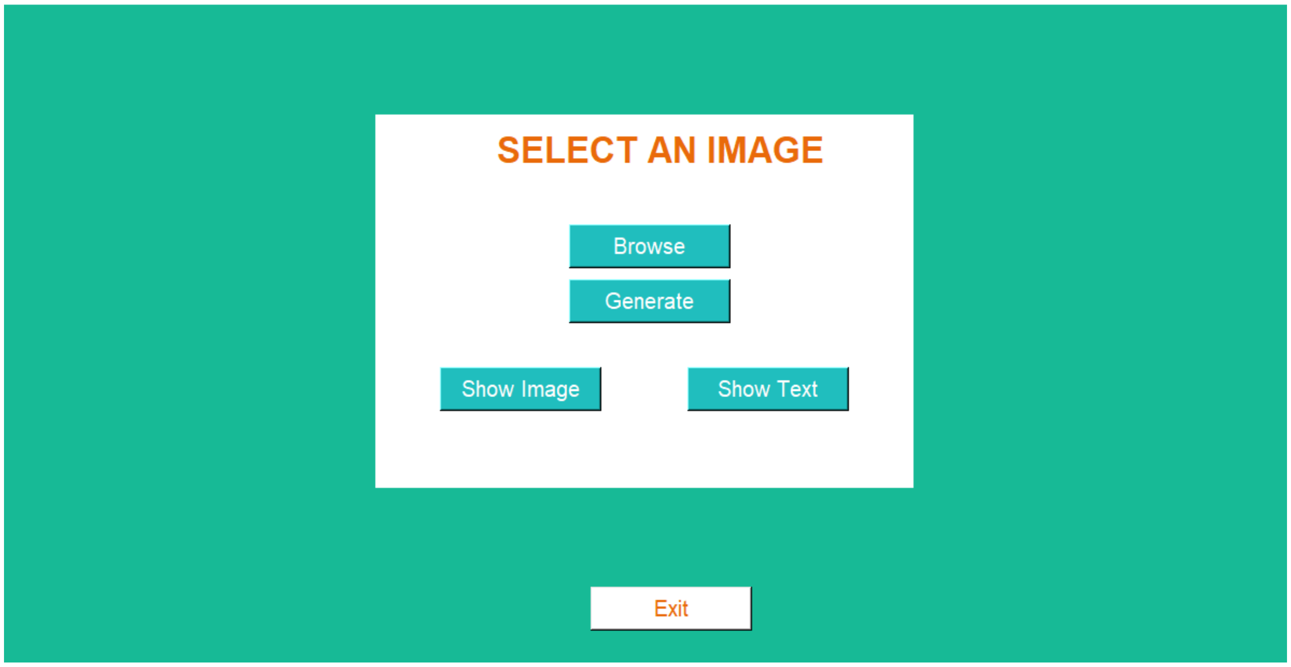


Fig 13: **Browse** button will show **Loaded** when input image is loaded.

After loading the input image user need to press Generate button to generate output. During this process a simple progress bar will show the progress of the process. After finishing the process, the Generate Button will show Generated text.

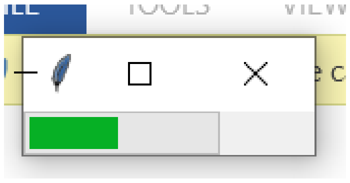


Fig 14: Progress Bar

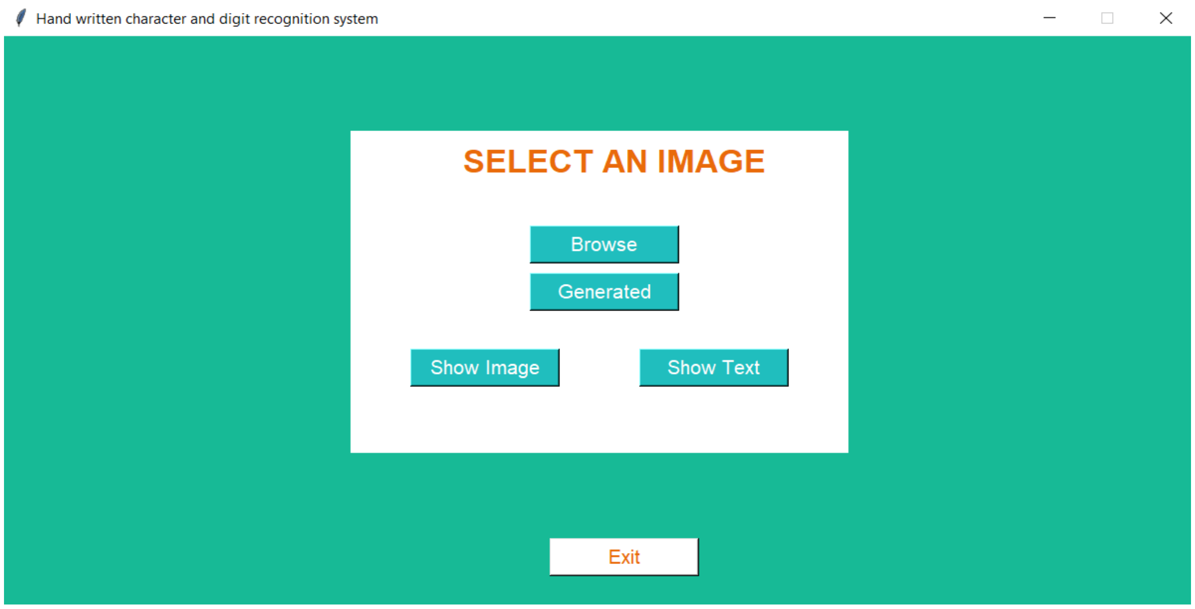


Fig 15: GUI after image has been processed

After generating output user can check output image by pressing Show Image button and an editable text file in a pop up window by pressing Show Text button.

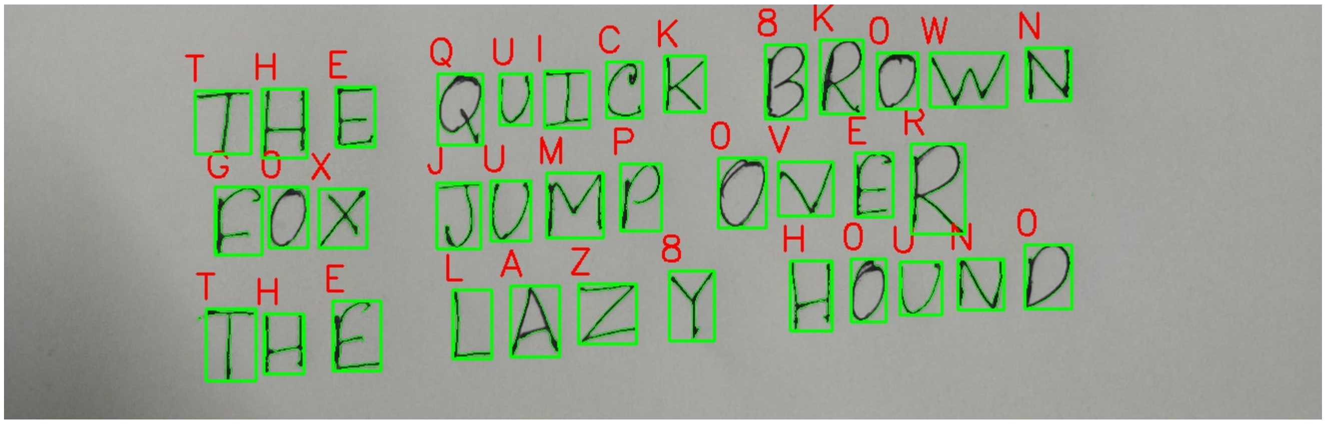
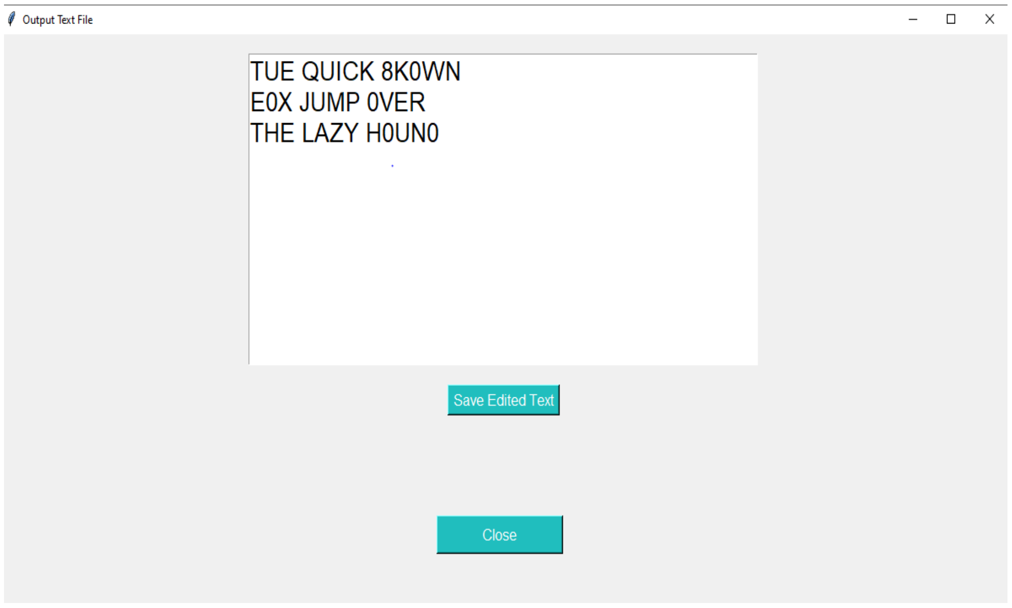


Fig 16: Output Image

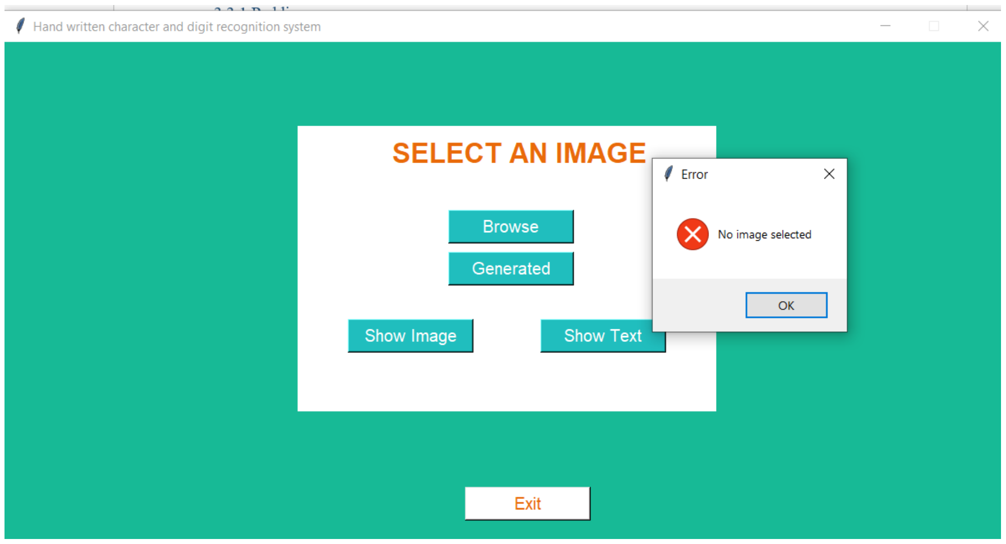


TUE QUICK 8K0WN
E0X JUMP 0VER
THE LAZY H0UN0

Fig 17: Editable Text Output

In the text document user can edit and save edited document in the same file.

If the user wants to generate output without loading input image an error message will pop up on a window



No image selected

Fig 18: Error Message

When generating output, the GUI automatically saves the output image as output.jpg and a text file as HCDROUTPUT.txt file in the home directory. This output files will remain saved even after exiting the GUI.

CHAPTER 5

EXPERIMENTAL RESULTS

Experiments were conducted to improve the accuracy of the proposed model [1] on MNIST dataset. Learning rates were changed as part of hyperparameter tuning. Dropout layers were added and their values altered to reduce overfitting as much as possible. The number of epochs are 30. The dataset is split into 3 parts. 80% of the data is used for training, 10% for validation and 10% for testing. The results are shown in table 1 and the improved accuracy compared to other works are shown in table 2. The accuracy vs epoch graph is shown in figure 19.

Table 1: Experimental Results of training CNN model with MNIST dataset

Dropout Layer 1	**Dropout Layer 2**	**Training Accuracy(%)**	**Validation Accuracy(%)**	**Test Accuracy(%)**	**Learning Rate**
None	None	99.87	99.17	98.94	0.001
None	None	99.96	99.25	99.26	0.0001
None	0.2	99.94	99.16	99.26	0.0001
0.2	0.2	99.94	99.22	99.27	0.0001
0.4	0.4	99.74	99.40	99.41	0.0001
0.5	0.5	99.57	99.51	99.47	0.0001

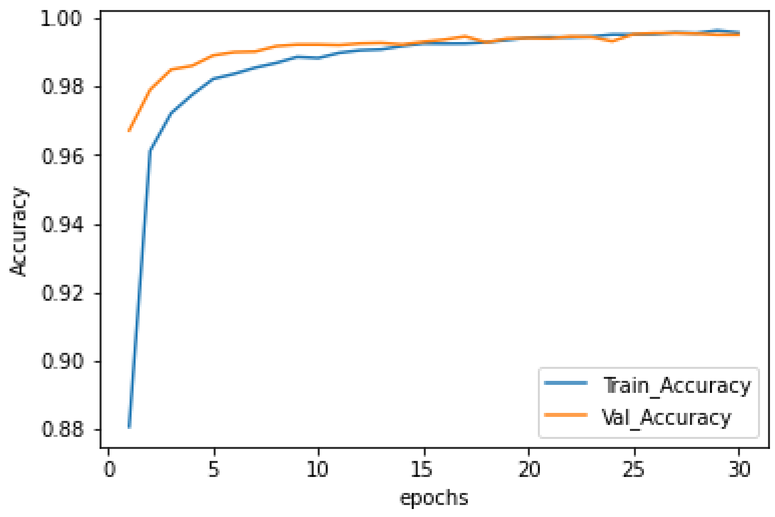


Fig 19: accuracy vs epochs graph of CNN model with two dropouts of 50% trained on MNIST dataset

Table 2: Accuracy comparison with different CNN architectures

Author	Method	Accuracy (%)
Younis and Alkhateeb [7]	CNN	98.46
Dutt and Dutt [3]	CNN (Keras + Theano)	98.70
Ghosh and Maghari [4]	DNN	98.08
Jana and Bhattacharyya [1]	CNN	98.85
Proposed CNN	CNN	99.47

Our experiments have resulted in an architecture that achieved 99.47% accuracy on MNIST dataset. We have also managed to reduce overfitting of the model as train and validation accuracies are very close. Compared to other works done on this subject before, our results are significantly better.

We extended our work by enlarging our dataset. We combined A-Z capital letters dataset with MNIST and modified the proposed model [1] to perform better on this dataset. The number of epochs are 20. The dataset is split into 3 parts: 80% of the data is used for training, 10% for validation and 10% for testing. We managed to achieve 98.94% accuracy by modifying the architecture and reducing overfitting as much as possible. The experimental results are shown in table 3.

Table 3: Experimental Results of CNN model on merged dataset

Dropout Layer 1	**Dropout Layer 2**	**Train Accuracy(%)**	**Validation Accuracy(%)**	**Test Accuracy (%)**	**Learning Rate**
None	None	96.67	98.84	98.84	0.001
0.1	0.1	99.33	98.92	98.94	0.001
0.2	0.2	98.99	98.90	98.91	0.001

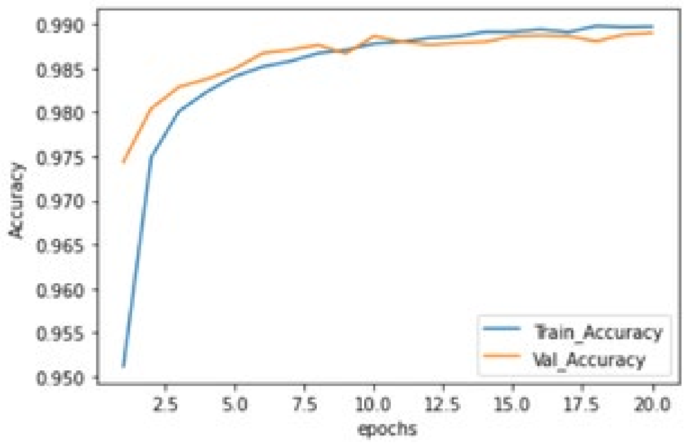


Fig 20: accuracy vs epochs graph of CNN model with two dropouts of 20% trained on merged dataset

The experimental results show that the model has least variance when dropout layers are valued 0.2. The accuracy vs epoch curve is shown in figure 20. However, test accuracy is highest when dropouts are valued 0.1. Increasing the values of the dropouts further resulted in validation accuracy being higher than train accuracy and a decrease in test accuracy.

The system when tested with a text image that contained all 26 capital letters and 10 digits. The outputs are shown below:

Table 4: System Test Results

INPUT	**CORRECT OUTPUT NUMBER**	**INCORRECT OUTPUT NUMBER**
26 Capital Letters (A-Z)	23	3
10 digits (0-9)	7	3

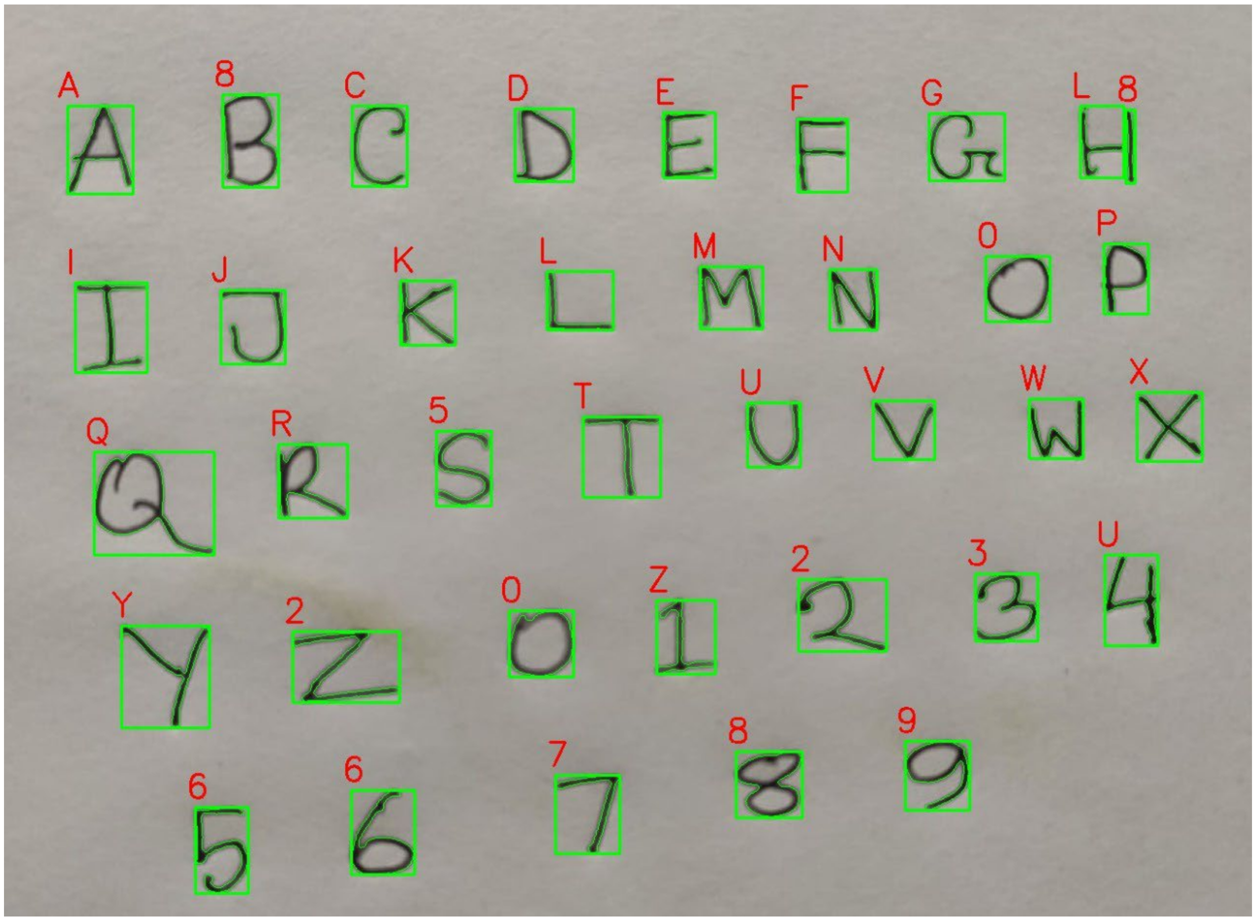


Fig 21: System Test Results

CHAPTER 6

CONCLUSION

This system can effectively recognize English digits and English capital letters, display them on screen for users, save output in both image and text format. The system has been tested with various images that contain different handwriting. In most of the cases the system can identify the handwritten characters, digits, line gaps and word gaps. Nevertheless, human handwriting patterns differ significantly and it is simply not possible to faultlessly detect every character or line gap or word gap. This system also fails in many cases but as it gives the user an editable document as output the user can correct the systems mistakes.

6.1 Limitations

- System can only recognize handwritten English capital letters and digits.
- System cannot always correctly recognize all characters, line gaps and word gaps.
- System requires characters in the input image to be greater than 50 pixels to be able to segment them into separate images.

6.2 Future Work

This system can only identify English capital letters and English digits. In future we aim to improve the system so that it can also classify small English letters. We also aim to improve accuracy of the classifier and add more features for users.

References

[1] Ranjan Jana, Siddhartha Bhattacharyya (2019), "Character Recognition from Handwritten Image Using Convolutional Neural Networks", Springer Nature Singapore Pte Ltd. 2019, DOI: https://doi.org/10.1007/978-981-13-6783-0_3

[2] S Anandh Kishan, J Clinton David, P Sharon Femi (2018), "Handwritten Character Recognition Using CNN", Published in: Volume 5| Issue 3, Published Paper ID: IJRAR1903931

[3] Dutt A, Dutt A (2017) "Handwritten Digit Recognition Using Deep Learning", Int J Adv Res Comput Eng Technol 6(7): 990-997

[4] Ghosh MMA, Maghari AY (2017), "A comparative study on Handwritten Digit Recognition using Neural Networks". In: International Conference on Promising Electronic Technologies, pp 77-81

[5] Pal A, Singh D (2010) "Handwritten English Character Recognition using Neural Network". Int J Comput Sci Commun 1(2):141–144

[6] Neves RFP, Filho, ANGL, Mello CAB, Zanchettin C (2011) "A SVM based off-line Handwritten Digit Recognizer". In: International conference on systems, man and cybernetics, IEEE Xplore, Brazil, 9–12. pp 510–515

[7] Younis KS, Alkhateeb, AA (2017) "A new implementation of Deep neural networks for Optical Character Recognition and Face Recognition". In: Proceedings of the new trends in information technology, Jordan, pp. 157–162

[8] https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

[9] https://github.com/imane0897/simple_OCR

[10] https://www.computerhope.com/jargon/g/gui.htm

[11] https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format

[12] http://yann.lecun.com/exdb/mnist/