"Hand Writing Recognition (English & Digit)"

Dissertation Manuscript

Submitted to the Faculty of the Engineering

Comilla University at Cumilla

in Partial Fulfillment of the

Requirements for the

Image Processing-based lab

by

Meheniger Alam

Department of Information & Communication Technology

Comilla University

May, 2023

CERTIFICATE OF ACCEPTANCE

This is to confirm the completion of the project report titled **"Hand Writing Recognition (English & Digit)"** by Meheniger Alam as a partial fulfilment of the Image Processing-based lab requirement. The dissertation was completed under my direction and as a proof of genuine work successfully completed. Her performance in this project period has been satisfactory.

I wish her every success in life.

Supervisor:

.....

Alimul Rajee

Lecturer

Department of Information and Communication Technology

Comilla University

Cumilla.

ACKNOWLEDGEMENT

Completing the dissertation could not be possible without the help of many hands and positive thoughts and prayers. I would like to express my sincere gratitude to my esteemed supervisor sir Alimul Rajee (Lecturer, Dept. of ICT, Comilla University) for his unwavering support and encouragement. I would also like to express my gratitude to my colleagues and classmates from the 10th batch of ICT at Comilla University for their collaboration and assistance in collecting the necessary data and preparing this. I acknowledge that some errors may have been made in the dissertation, and I take full responsibility for these mistakes.

Thanking you,

Meheniger Alam

Department of Information and Communication Technology

Comilla University

Cumilla.

ABSTRACT

The field of handwriting recognition was concerned with the development of algorithms and systems that were capable of reading handwritten text and converting it into digital forms. Recently, there'd been a lot of interest in using handwriting recognition systems in different ways, like document management, taking digital notes, and signing autographs. This project aimed to create a handwriting recognition system that used machine learning to accurately recognize handwritten text. It used a big dataset of handwritten texts to train the system, and it used advanced image processing to preprocess the data. The system used a CNN to get features out of the images and turn them into text characters. It had also used a language model to make the system more accurate by using contextual info to tell the difference between similar characters. It had been tested on a bunch of real-world situations to see how well it worked. In our proposed method we used the MNIST dataset along with A-Z handwritten characters dataset & CNN algorithm. And the result compared with the results of KNN, SVM and DCNN algorithms results. The accuracy obtained by this project was about 98%.

LIST OF CONTENTS

ABBREVIATIONS

List of Abbreviations

BND	Bangla Handwritten Digit Recognition
BHC	Bangla Handwritten Character
CNN	Convolutional Neural Network
CER	Character Error Rate
CPU	Central Processing Unit
CTC	Connectionist Temporal Classification
CMATER	Center for Microprocessor Applications for Training Education and Research
DLA	Diffusion Limited Aggregation
DP	Deep Learning
DCNN	Deep Convolutional Neural Network
DCANN	Deep Convolutional Auto encoder Neural Network
DLA	Diffusion Limited Aggregation
EMNIST	Extended Modified National Institute of Standards and Technology Database
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HTR	Handwriting Recognition Technology
HWDB	Hindustan Hierarchical Database
HMM	Hidden Markov Models
HWR	Handwriting Recognition
ISI	Indian Statistical Institution
IAM	Identity and Access Management
KNN	Knowledge-Based Network
KNN	K-Nearest Neighbors
MLP	Machine Learning Pattern

MLP	Multi-Layer Perceptron
MNIST	Modified National Institute of Standards and Technology Database
NLP	Natural Language Processing
NIST	National Vulnerability Dataset
NN	Neural Network
OPENCV	Open Source Computer Vision Library
OCR	Open Character Recognition
RNF	Random Forest
RNN	Recurrent Neural Network
SVM	Support Vector Machine
SPR	Statistical Pattern Recognition
SPAR	Statistical Pattern Analysis
UCM	Universal Church of Man
WER	Word Error Rate

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

Introduction

Hand writing recognition, the technology that allows computers or digital devices to recognize and interpret handwritten text. It involved analyzing and interpreting handwritten text images, identifying the characters or words present in image, and converting them into machine-readable text. It could be used in a variety; of applications, such as digitizing handwritten notes or documents, recognizing handwritten addresses on envelopes, converting handwritten text into typed text in text editors or email clients, and even recognizing handwritten signature for authentication purposes. It had applications in many fields, including education, finance, healthcare, and law enforcement.

1.1 Introduction

Handwriting recognition, sometimes known as HWR or HTR, the capacity of a computer to get and decipher coherently written input by which a computer receives and interprets intelligible handwritten data from a variety of sources, including paper documents, photographic images, touchscreen displays, and other electronic devices Formatting, splitting letters, and finding the most relevant words all handled by the handwriting recognition system.

Automatic handwriting character recognition acknowledgment had numerous scholarly and marketable interface. Dealing with the enormous variety of writing styles of various authors presents the fundamental challenge in handwritten character identification. The most challenge in manually written character acknowledgment was to bargain with the colossal assortment of penmanship styles by diverse scholars. Moreover, a few complex penmanship scripts contain distinctive styles for composing words. Depending on the dialect, characters were composed disconnected from each other in a few cases (e.g., English, Bangla, and Arabic). In a few other cases, they were cursive and some of the time characters are related to each other. This challenge had been as of now recognized by numerous analysts within the field of NLP.

Besides, English comprises of numerous comparative molded characters like as Bangla. In a few cases, a character varies from its comparative one with a single dab or stamp. Besides, English too

contained a few extraordinary characters with proportionate representation of vowels. Transcribed digit acknowledgment included distinguishing 10 characters, i.e., 0-9, but the input was delicate to the natural clamor. Input was sensitive to environmental noise.

The most popular approach to constructing the handwritten digit recognizer was using a MLP, KNN or support vector machine SVM. Different other procedures were created utilizing diverse methods with MLP structure. Arbitrary Woodland, SVM, KNN, and other machine learning procedures had been created to recognize written by hand digits. Profound learning strategies like CNN had the most noteworthy precision when compared to the foremost commonly utilized machine learning calculations for manually written character acknowledgment. Design acknowledgment and large-scale image classification were both done with CNN. Penmanship character acknowledgment might be acquired about field in computer vision, manufactured insights, and design acknowledgment. It could be claimed that a computer application that conducts handwriting recognition had the capacity to procure and recognize characters in photos, paper documents, and other sources, and change over them to electronic or machine-encoded shape. Profound learning could be a well-known field of machine learning that employments progressive structures to memorize high-level deliberations from information. Concurring to references, the accessibility of innovation CPUs, GPUs, and difficult drives, among other things, machine learning calculations, and expansive information, such as MNIST manually written digit information sets were all variables in profound learning's success.

Deep learning was currently gaining popularity as a method for extracting and learning to recognize deep patterns. It could produce patterns from a given dataset at a deep learning level. It was a remarkable algorithm with a variety of libraries to recognize patterns in photographs and classify them. The CNN was one of the more effective deep learning algorithms, and it performed well at image classification, image recognition, pattern identification, feature extraction, and other tasks. However, in our project, we used the packages Tensorflow, Keras, Matplotlib, and Numpy to create our system.

1.2 Problem Statements

Character was difficult to define, and there was to universally accept single definition the words was used to refer to both the inner different writing method of a person researchers had developed various taxonomies of character related character trained by machine learning for use in different domains or contexts.

1.3 Objectives

1.3.1 Broad Objectives

The main objective of this study was to evaluate the classification of the general regression neural network for character Recognition. Sub objectives contained select the ideal model. Training and testing the input data recognized the character.

1.3.2 Specific Objectives

Hand written character acknowledgment was more challenging compared with the printed shapes of character due to the taking after reasons:

• Handwritten characters composed by distinctive journalists were not as it were no indistinguishable but too change completely different perspectives such as estimate and shape;

• Numerous varieties in composing styles of person character made the acknowledgment assignment troublesome;

• The similitudes of distinctive character in shapes, the covers, and the interconnects of the neighboring characters encourage complicate the character acknowledgment issue.

1.4 Findings

In our project on Handwriting Recognition, we had used the CNN with some python frameworks and also used the A-Z English alphabet consisting of 5 vowels and 21 consonant words as well as

MNIST datasets. This study had reported high accuracy rates of up to 98% in recognizing isolated handwritten characters, while recognizing full sentences and paragraphs still presents challenges.

1.5 Chapter Summary

To sum up, the wide range of writing styles and unique characteristics of handwritten characters made it difficult to categorize them.

1.6 Keywords

Isolated hand writing recognition, MNIST, Python library, Tensorflow, Keras, Matplotlib, Numpy.

1.7 Paper Type

Project Report

1.8 Chapter Overview

This study had been documented in five chapters. Chapter one reviews the research problem and the goals of the study. Chapter tow reviewed the definition of character pattern recognition and its kinds and its components and its applications. Chapter three on the study methodology review in. Chapter four on the artificial neural networks contained. Chapter five contained the results and conclusion and recommendations.

CHAPTER TWO

Literature Survey

In this literature review, we explored the current techniques and technologies for handwriting recognition. We delved into the various approaches used, their strengths and limitations, and their applications in different domains. Additionally, we examined the challenges that were associated with handwriting recognition, including variations in handwriting styles, character, segmentation, and noise in input data. We also discussed about the datasets, results and other limitations from the previous experiments by different author. We hoped to provide a comprehensive overview of the current landscape of handwriting recognition research, highlight the most promising approach, and identify areas that require further investigation.

2.1 Background of Study

Developing a handwriting recognition system required a deep understanding of image processing, feature extraction, machine learning, language modeling, and testing and evaluation. With the right approach and expertise, it was possible to create an accurate and reliable system for recognizing handwritten text. The key areas must be studied: the dataset used for the model, there may be so many datasets such as Banglalekha-Isolated, MNIST, IAM, CVL, RIMES, NumtaDB, EMNIST used by different researchers on their projects. Again there different types of algorithms like as CNN, SVM, MLP, DL, KNN, DLA, CTC along with the frameworks known as Tensorflow, Keras, Matplotlib, Panda and Numpy were used in different researches and they gave different accuracy which were much better than the previous proposed methods.

2.2 Theoretical Background

This section provided the most up-to-date information relevant to the use of machine learning for character recognition. Tandra et al. employed a CNN with the Bangladesh-Isolated dataset, resulting in a training loss of.0108 and a validation loss of .0153. The training accuracies reached 99.87, and the validation accuracy was 99.50%, respectively. In this paper, the authors intended to

address the issue of implicit feature extraction, proposing a method that automatically selected and extracted features from character images regardless of the individual encoding and spacing. Additionally, they had considered fifty classes of base letters (11 vowels and 49 consonants) and ten classes of digits, with the aim of proposing a method that would have a higher overall accuracy than current methods. [3]

I Khandokar et al. found that the average accuracy increases as the number of training images increases. With 200 training images, the average accuracy was 65.32%, and with 1000 training images, the accuracy was 92.91%.

In this study, the CNN was used in conjunction with the NIST datasets. The handwritten characters in the NIST dataset were presented as images. The training and testing sets were divided into different training sets. The training was carried out with different numbers of images, and the testing was conducted to determine the CNN accuracy. The primary focus of the study was to see if the CNN could recognize the characters from the NIST dataset with high accuracy. [4]

The primary goal of deep learning was to generate equivalent or superior learning effects based on limited domain knowledge or expertise. In order to achieve this, Yintong Wang ,Wenjie Xiao and Shuo Li employed the CNN and RNN algorithm with the datasets of IAM, Bentham (International Humanist Association), BH2M (International Humanist Institute for Migration and Refugees), HIT-MW (International Telecommunication Hydrological Institute), and HWDB (HWDB1.1, HWDB2, and HWDB2.2).In their model they found the accuracy of the CNN mode lwas 97.04% and in RNN 96.07%. According to the authors, this trend was likely to continue in future work, and researchers will be looking for ways to extract and recognize handwritten text more effectively. [5]

In order to demonstrate the accuracy and recognition rate of the SPR method, Professor Sathish and Ms. Babiker presented a proposal for a SVM. This model was based on the SPAR method, which was based on MNIST data sets, CENPARMI data sets, UCOM data sets, and IAM data sets. The HCR method, based on the SVM, was found to be 94% accurate and had a higher recognition rate than existing methods. Additionally, the proposed model was tested in a training section, which included various stylized letters and digits, to ensure a higher level of accuracy. The test results showed that 91% of the characters from the documents were correctly identified. [6]

Omer Othman Ahmed conducted an analysis on a variety of handwritten scripts and characters, including English, Arabic, and other languages. Each text was composed of a collection of symbols, referred to as letters, each of which had its own basic shape. The analysis was conducted with the assistance of Tensor flow, Lightbench, Numby, Theano, Keras, Pandas, Scipy, Scikit-learn, Python, and other Machine Learning algorithms. After approximately 50 epochs, the CER was estimated to be 10.72%, the WER to be 26.45%, and the Word Accuracy to be 73.55%. [2]

According to Md. Ismail Hossain ET all had proposed that Bangla Writing had significantly less no. of samples (21,234 words) with less data diversity (5,470 unique words). BN-HTRD had a large no. of samples (108,147) with large data diversity (23,115 unique words). On their paper A. Synthetically Generated Isolated Printed Characters on their teacher model was an isolated character prediction model. The training dataset for the teacher model was generated synthetically by using the Text Recognition Data Generator module from Python Package Index. And they found an accuracy of 95%. A list of various Bangla fonts was fed to the package, along with a set of graphemes. For each grapheme in the grapheme set, arbitrarily 160 isolated character images were generated using random augmentations. B. Handwritten Word Datasets In their experiments, they had used two handwritten word level datasets to train their student model. The Bangla Writing dataset has 21,234 Bangla handwritten words which obtained an accuracy of 97%. Additionally, the BN-HTRD dataset has 108,147 Bangla handwritten words. [7]

Rabia Karakaya and Serap Kazan proposed that the SVM accuracy was 90%, the Decision Tree making was 87%, the RNA was 97%, the ANN was 97%, the KNN was 98%, and the KMA was 98%. This analysis was conducted using the MNIST dataset with the SVM algorithm, the Decision Tree, the Random forest, the ANN, the KNN, and the KMA. [8]

In their paper of Ritik Dixit, Rishika Kushwah and Samay Pashine, the authors of the paper mentioned that, upon implementation of all the models, SVM achieved the highest accuracy on the training data, while CNN achieved the highest accuracy of 98.03% with the MNIST datasets and NIST datasets. To address the novel challenge of non-negative image generation, the authors proposed a framework that could combine the strengths of modern image generation methods with the advantages of the human visual system, taking advantage of the degrees of freedom available to it. This framework was compatible with nearly all existing image to image translation methods, allowing for its application to a broad range of tasks. Specifically, the authors claim that, first, they

incorporate existing image- to-image translation models into the framework, resulting in a proposal image that met the requirement, but was not feasible in a non-negativity generation setting. Subsequently, the proposal image could be fine-tuned by adding non-Negative Light to the input image, leaving perceptually unchanged. [10]

LeNet 4 was the most accurate method in this paper, getting 99.3%. It's the fastest method too, with an operational/actual recognition time of 0.05 milliseconds. MLP was also used to create SVM, which stands for Analysis of Machine Learning Algorithms for Character Recognition: A Case Study on Handwritten Digit Recognition. This paper was written by Mr Owais Moolabta Khanday, and it showed a bunch of different feature extraction techniques. The methods were compared against other methods in the literature, and the boosted one got the best results, getting 99.03%, and the other 98.75%, respectively. [9]

This paper provided a comprehensive overview of the BND dataset, which was initially used as a foundation for the development of an OCR system in Bengal. Prior to the publication of this paper, BND digit recognition was not standardized due to the lack of large and impartial datasets. To rectify this problem, a large, impartial dataset was used for BND digit recognition, known as NumtaDB. The use of the NumtaDB dataset posed a number of difficulties, as most of the images were not processed or augmented. Consequently, a variety of preprocessing techniques were used to process the images. CNN was used as the classification model for the classification of the images. A study was conducted to strengthen the analysis, comparing the network's performance against MNIST dataset and EMNIST dataset, which was conducted in 2018 among 57 participating teams, in which the deep convolutional neural network model was ranked 13th with a test accuracy of 92.72%..[11]

Even though Bangla was an official language in Bangladesh and some Indian states with more than 200 million people speaking it, it's still not up to scratch when it came to recognizing handwritten characters. In a paper published in the Journal of Neural Network Science, S M Hakim and Azaduzzaman showed a 9-layer convolutional neural network model that can recognize 60 handwritten Bangla characters (10 numbers plus 50 basic characters). They used a set of 6000 images from Bangladesh as a training data set and then created a new set of 6000 images for cross-validation. The model was able to recognize 60 Bangla characters with a 99.44 percent accuracy

on the set of 6000 images and a 95.16 percent accuracy on the test set. They also showed some impressive results for recognizing Bangla numbers separately. [12]

Md Zahangir alom et al. demonstrated that several well-known DCNNs performed well for the recognition of handwritten Bangla characters (i.e., digits, alphabet, and special characters). According to the experiment results, DenseNet was the best solution for classifying handwritten Bangladesh numbers, alphabets and Special characters. In particular, DenseNet achieved a 99.13% recognition rate for handwritten Bangladeshi numbers, a 98.31% recognition rate for the handwritten Bangladeshi alphabet, and a 98.18% recognition rate for special characters using. [13]

Dr. Md Shopon, Dr. Nabeel Muhammad and Dr. Md Anowurul Abedin presented a paper in which they demonstrated that an unsupervised, pre-training approach with auto encoder and deep ConvNet could be used to recognize handwritten Bangladeshi digits, ranging from 0-9 digits. The two datasets used in the study were CMATERDB (3.1) 3.1 and an ISI dataset. This paper examined four distinct combinations of the two datasets, with two experiments conducted against the authors' own training and test images, and two experiments cross-validated against the datasets. Out of the four experiments, one of the proposed approaches achieved 99.50%, which was considered to be the highest accuracy for the recognition of handwritten Bangladeshi numbers. This ConvNet model was trained with 19.313 images of the ISI handwritten character datasets and tested with CMATERDB datasets. [14]

A dataset of 114988 handwritten Bangladeshi digits was used to perform the Bangla Handwritten Number Recognition using DCNN by Md. Hadiuzzaman Bappy et al. The database was generated by randomly selecting samples from a larger database (85596 in the case of NumtaDB; 23392 in the case of ISI; and 6000 in the case of CMATERdb) for each of the ten digit classes. NumtaDB consists of six large datasets collected from different sources. The datasets were compiled from pre-defined database sheets of individuals of different ages, genders, and educational backgrounds. As mentioned above, they used three datasets for training and test set:The NumtaDB dataset contains 72044 and 13552 samples with an accuracy of 97.06% respectively the ISI dataset contains 19337 and 3986 samples with 97.09% accuracy and the CMATERdb dataset contains 4000 and 2000 samples with 98.00% accuracy respectively. [16]

CNN aimed to reduce the images into a form without losing any features. In this paper, the author introduced an Autoencoder with a Deep CNN, which we call DConvAENNet for recognizing BHC. A total of 22 experiments were performed on the three-character datasets (BanglaLekha-Isolated, CMATERdb 3.1, Ekush). All attempts acquired satisfying results up to 90% accuracy for the recognition of Bangla handwritten numerals, vowels, consonants, compound characters, modifiers, and all characters set unitedly. Using this supervised and unsupervised learning technique, their proposed DConvAENNet model achieved 95.21% on BanglaLekha-Isolated for 84 classes, 92.40% on CMATERdb 3.1 for 238 classes, and 95.53% on Ekush for 122 classes. Most errors in our model were caused due to the similarity and high curvature nature of the BHC sets is showed in a paper of Md Ali Azad ET all. [15]

In order to begin the analysis of handwritten characters, Shahrukh Ahsan and his colleagues selected the 'BanglaLanguage-Isolated' dataset to take the initial samples. The dataset included a wide range of samples, from simple to modern handwritten and cursive. For implementation, 200 samples were taken for individual characters. The model was able to accurately identify a single character at up to 94% accuracy, and all characters at an average of 91%. This method was more efficient than existing methods for Bengali handwriting recognition. On the basis of their study, Ahsan and his team developed a different approach, which involved using face mapping to enhance the recognition of Bengali characters. This approach was successful in distinguishing different characters, and the results were more efficient than anticipated with a basic machine learning technique. To implement the proposed method, the Python library was used, which included the Scikit-Learn library, as well as the Pandas and Matplotlib libraries. [17]

The results of a study conducted by Dr. Aditya Jain and Dr. Pawan Singh revealed that the field of Artificial Intelligence was rapidly expanding, particularly in the areas of computer vision. According to the study, the chances of a human error in computer vision were approximately 3%. This suggested that computers were now more proficient than humans in distinguishing and examining photographs. It was a remarkable feat that computers, which were once small pieces of technology, were capable of understanding the world around them in ways that were previously unimaginable. There was no doubt that machine learning will have a significant impact on the world in the near future with an accuracy near 97%. [18]

Anil Chandrasekumar Matcha's study found that after training for around 50 epochs, the CER was 10.72% and the WER was 26.45%, resulting in Word Accuracy of 73.55%. The model was able to accurately predict the characters, but it didn't work very well in some cases. He used a CTC algorithm using IAM and CVL data sets, as well as RIMES data. He divided the handwriting recognition methods into two categories: online and offline. Online methods used a digital pen and stylus, so you can see the stroke information and pen location while writing, but you can't see the strokes or directions. Offline methods, on the other hand, only recognize the text after it's written, with some background noise added from the source, like paper. [19]

Inertial and force sensors had been developed by Tsige Tadesse Alemayoh ORCID ET. All into a novel digital pen. Handwriting data was collected from six subjects and segmented into a dataset of 36 alphanumeric characters (10 numerals and 26 small Latin). The dataset was then restructured into a 2D array of 18 × 200 virtual images. The dataset was then trained with four neural network models using deep-learning methods: ViT, CNN, LSTM and DNN .The ViT model outperformed the other three models in terms of validation accuracy, achieving 99.05%. The accuracy was determined using LSTM, CNN, DNN, and ViT algorithms. [24]

When the application was launched, a window will appear in which the user can write a digit. Upon clicking on the recognize button, the algorithm will recognize the digit with a probability percentage indicating how accurately the digit corresponds to the original one. In the example provided, the user had written digit 1 and the algorithm recognizes it with 17% accuracy using the CNN algorithm. Additionally, the author has used MNIST (which contains images of digits ranging from 0 to 9 in greyscale), 60000 training images (which are very large) and approximately 10000 test images (which are small squares with 28 x 28 pixels). All of these images were developed by the author Amrutha K. [23]

Sania Firdous used various English handwritten words like Cursive, Block Writing, Text-To-Speech, etc. to help people who struggle to read on-screen text. However, she found various feature-based classification techniques for Offline handwritten character recognition. After experimenting with various methods, she proposed an optimum character recognition technique which had accuracy of 20%. The way to do this was to use heuristics and AI to break down a handwritten word. The outputs obtained using the suggested character recognition system were satisfactory. [25]

For these DL models, it was important not to treat them as black boxes, but rather to gain an understanding of why predictions were made in this way. The author had already explained some of the reasons for failure spaces in the models that led us to use ensemble learning. In this section they were going to use a LIME interpretation for the time series data.

In this interpretation, each time series was split into 20 slices. Each slice had its importance determined by the algorithm. Since they were using MTS data we are going to analyze each of these 13 channels separately. We will look at the top 30 signals as well as the slices when it comes to importance. A green bar indicates that this part of the data impacted the model output positively. A red bar indicates the part of the data that impacted the model output negatively. All 13 channels of raw data had an importance bar with an overplayed importance bar. The dark hue of the data quantifies the significance of the segmentation of the data for the classification of the data according to the findings of Hildy Azimi et al. Using a variety of characteristics datasets, the authors used the DL algorithm to determine the accuracy of the L-WD dataset, as well as the L-WI dataset, the C-WD dataset, and the C-WI dataset. And the accuracy was near 97% for each dataset. [26]

This text describes the development of a unique character prediction model, the teacher model, by Md. Hossain ET all. To train the student model, the training dataset for this model was generated synthetically from the Python Package Index text recognition data generator module. The package was fed a list of various Bengali fonts, as well as a set of graphs. For each graph in the graph set, an arbitrary number of isolated character images was generated using random augmentation. The two handwritten word-level datasets were used to train the model. The dataset for Bangla Writing contained 21,234 Bengali handwritten words, while the dataset for Bangla Reading had 108,147 Bengali handwritten words. It had an accuracy of 94%. It was found that Bangla Writing had a significantly lower number of samples, with a greater number of data diversity, in contrast to the dataset for Bengali Reading, which had a large number of samples with a greater diversity of data, in this case 5,470. Additionally, LILA- was used for Bengali handwriting recognition. [27]

IAM was a popular benchmark used for handwriting recognition. WikiText was a set of 103 images used to make synthetic images. Free Form Answers was a proprietary dataset with 13K samples, Answers2 is 16K words and lines taken from handwritten test responses, and Names is a private dataset with 22K names and identifiers. This dataset was used in the DNN algorithm by Sumeet

S. Singh and Sergei Karayev. When the model was trained on all the datasets and tested on the Free Form Answers dataset, it got the error rate down to 7.6%, which was lower than the best cloud API's 14.4%. It only takes about 4 seconds to download a set of 2500x2200 pcs, 456 characters, and 1165 lines without any compression, which was when the model was pruned, distilled, or quantized. [32]

The CNN was utilized to compare and evaluate the difference in precision between the handwritten Geez numerical models. The training and validation accuracy was measured at 30 different epochs using different convolution layers and batch size 32 in each case. In the paper, the Deep Learning method is used by Muhammed Ali Nur et al., using a dataset of 52.400 instances and Python's OpenCV. In this paper, the proposed model for Geez digits is to use the CNN. DNNs, which had recently been used in lots of machine learning and pattern recognition applications, are used for Geez digits but hadn't been used for Ethiopic scripts yet. They used a dataset of 51.952 images of Geez digits from 524 people to train and evaluate our CNN model. By using the CNN, we can significantly improve the performance of some machine-learning methods. The proposed CNN model had a 96.21% accuracy and a 0.2013 loss. Compared to previous studies of Geez handwritten number recognition, this one was able to get better accuracy with the new CNN model. [34]

The official information for the IAM dataset was that it consists of about 655 people's handwriting samples, which were further divided into sentences collection, words collection, along with their traits details to correct our prediction results. In this case, they were unable to extract the entire dataset because of the limitations of our system. However, they were able to get around 4899 dataset samples out of which 1000 were female data sample, and 3899 were male. Only 552 were left-handed people in the ratio 220 female and 332 male left-handed people, and all the rest were right-handed data. Training was done with 3233 dataset and testing was done with about 900-1000 dataset in each phase. Moving on to the Real-Time dataset, here writing samples were collected from 2900 pupils. 1900 were male pupils and 1000 female pupils, 2600 right-handed, and 300 left-handed. For male gender prediction training, 2900 data were trained and 2000 data were tested. For women, 2700 data were used for training and 2300 for testing. For men, 2800 data were used to train and 2420 for testing. For women with left-handedness, 980 data were used for training and 300 for testing. For combined prediction for men with right-handedness, 2690 data were used for

trainings and 1473 data were used for tests. For left-handed men, 529 data were used for trains and 180 for tests. For right-handed women, 2430 data were issued for trainings and 1155 were used for training. Finally, for left-handed women, 740 data were issued for training and 280 data were used for testing. Compared to previous works, the data used in this study was large enough that direct implementation could be performed. However, the extracted data and real-time data was not enough to compare it to previous works. To expand their data, they used online data augmentation. This is because there was no need for extra storage space as the data was processed simultaneously while execution was taking place. The entire study was conducted by Ayushi Agarwal and Mala Sarwat. [35]

This character recognition model was designed to be trained to recognize handwritten English alphabetical sequences. The alphabet was divided into 25 segments, and the Neural Network architecture was primarily designed to accommodate the processing of 25 input bits. The parameters of the network used for training were as follows: number of intermediate layers (hidden) = 2 rate of learning = 0.05 number of unit in intermediate layers (hidden) = 25 number of unit in first layers (input) = 25 initial weight = random [0, 1]. The transfer function used for intermediate layer 1 is "Logging" and the transfer function for intermediate layer 2 is "Tensing". This study was conducted by Prof. Doke Kanchan Kiran and ET. All. [36]

According to Nghia Hieu Nguyen ET. All, since Transformer OCR had the best results in their experiments on both the word-level and the line-level dataset, they focused on analyzing the common mistakes that were encountered by Transformer OCR. After observing the test set and their respective predictions, they conducted the following two types of errors of Transformer OCR: the incorrect prediction of number characters, and the misunderstanding of scrawled handwritten characters. For the first mistake, since the domain of UIT-HWDB dataset is daily newspapers, the occurrence of number characters was lower than that of alphabet characters, which means that the model doesn’t learn well enough to make the correct decision when faced with these numerical characters. Stated differently, the frequency of number characters was about 0.065 in the word trainset and about 0.0066 in the word test set. However, due to the low frequency of number characters on both the train set and the word test set, this error was typically not the main factor that reduced the performance of the transformer OCR method. The most common type of error was misinterpretation of handwritten characters, which was the primary difficulty in recognizing

hand-written images and the primary issue of our dataset. Transformers OCR was unable to make an acceptable prediction for images that contained scrawled handwritten characters. Humans found these images to be difficult to read, however, this did not mean that reading these images was impossible. The only method of reading these words was to interpret them together with the meaning of the line. Therefore, they recommended improving the Transformers OCR or general deep learning methods for this challenge by finding a way to exploit the meaning of the phrase to enable models to have better inference. [38]

Jorge Sueiras and co. came up with a new way to recognize isolated handwritten words using a neural network architecture. It's a combination of a deep neural network and an encoder-decoder, which they call sequence to sequence. The goal was to recognize the characters and compare them to their neighbors to figure out if any given word is worth recognizing. Their model suggests a new way to get the right visual features out of a word image, which was to use a Horizontal sliding window to extract the image patches. They also used a LENet-5 architecture to recognize the characters. The extracted features are then encoded using a sequence to sequence architecture and then the sequence was decoded from the handwritten text. They tested their model on two handwriting databases, IAM & RIMES, to see if they could get better parameterization. The results were way better than what's currently known about handwriting models. Without any language model or closed dictionary, the word error rate was 12.7%, and the RIMES was 6.6%. [39]

Marcus Georgi et al. proposed a wearable input system that allows users to interact with 3-dimensional handwriting recognition. By writing text in the air, the system was able to capture the user's handwriting gestures wirelessly with the help of motion sensors. The sensors were attached to the user's back and used to detect the presence of handwriting. The two-stage approach was based on the use of a SVM to identify the data segments that contained handwriting. The second stage was the use of a HMM to generate text representation from motion sensor data. The HMMs were then used to model individual characters and combined them with word models. The system was able to continuously recognize arbitrary sentences based on a freely defined vocabulary and obtained accuracy of 98.7%. Additionally, a statistical language model was employed to improve recognition performance and to limit the search space. It was demonstrated that continuous gesture recognition combined with inertial sensors was possible for gesture dictionaries that were many times larger than those found in traditional systems. In the first trial, the spotting algorithm was

evaluated on a real-world dataset that included day-to-day activities. In the second trial, the nine users were presented with the results of a handwritten sentence recognition experiment. Finally, the final system was evaluated on a smaller but still realistic dataset. [40]

This paper proposed a novel approach to the recognition of handwritten Bangla numbers using multi-layered CNN. The dataset used to train the CNN model is a large, standard, and unbiased dataset containing 85,000+ digit images. Upon training the model, the model achieved a training accuracy (99.66%) and validation accuracy (98.96%) for the recognition of Bangla digits, demonstrating that the model was not over- or under-fitted and was suitable for the detection of unseen data in the real world. A significant focus was placed on the pre-processing of digits and splitting of a number, as multilayered CNN alone is not sufficient for achieving a high degree of accuracy. A smaller dataset of 200 number images with 2-5 digit numbers was tested against the proposed model, and the validation accuracy was 88%, which was higher than any other proposed models. [41]

The recognition approaches performed with various methods chosen by Yasser Baiker Hamdan, Prof. Sathi in artificial neural networks, statistical methods etc. and to solve nonlinearly separable problems. This research article consists of various approaches to comparison and recognition of the handwriting characters from image documents. In addition, the research paper compares statistical approach of SVM classifier network method with statistical method, template matching method, structural pattern recognition method, and graphical method. It proved that Statistical SVM was performing well for OCR system that wss providing a good result when configured with the machine learning approach. Recognition rate was higher than the other methods mentioned in the research paper. This study demonstrated that Statistical SVM performs satisfactorily for OCR systems that were delivering a satisfactory outcome when configured with a machine learning approach. The test results of the proposed model were 91% of accuracy for recognition of the characters from documents. [42]

Based on the test data provided by EMINST, Shubham Sanjay Mor et all's model was predicted with 87.1% accuracy. This accuracy could be further improved in future research by further pre-processing of the dataset and addition of new hyper parameters in keras model (the lower the loss the better) (unless, of course, the model has already been over fitted to train data). The model had been calculated on our train data as well as test data. Typically, for neural networks the loss was

negative log- likelihood and RSS for both classification and regression. So, naturally, their goal was to lower the loss function value in relation to the model parameters. Their keras model had a loss of 0.68 in the first epoch of training and 0.31 in the second. As mentioned in the model chapter, the classifier consists of 2 Convolution layers and 1 Maxpooling layers with a drop out layer and 512 units dense layer. They tried to fine tune the classifier by adding a couple of back-to-back convolution layers, but this did not improve the accuracy of the model. [43]

2.3 Literature Summary

Table I- Table on Literature Summary

No.	Dataset	Algorithm	Result
1	IAM	DNN	7.6% error rate compared to 14.4% for the best cloud API. On a single CPU thread, inference takes an average of 4.6 seconds [32]
2	BanglaLekha-Isolated	SVM	Accuracy of up to 94% for one character, and an overall average of 91% [17]
3	IAM, CVL, RIMES	CTC	Word accuracy was 73.55% since the CER was 10.72% and the WER was 26.45% [19]
4	MNIST	CNN	It was accurately identified as 1 by the writing with 17% [23]
5	36 alphanumeric characters	LSTM,CNN, DNN, ViT	99.05% validation accuracy [24]
6	Different shaped letters of English , Arabic and others	Python, Tensorflow , MLs	The CER was 10.72%, the WER was 26.45%, and the Word Accuracy was therefore 73.55% after training for around 50 epochs [2]
7	Cursive or block writing English handwritten words	KNN,SVM	The proposed Character Recognition system's findings were judged to be satisfactory [25]
9	Human handwriting style, Color.	DNN	There were roughly 0.065 numerical characters in the UIT-HWDB-word train set and around 0.0066 in the UIT-HWDB-word test set [38]
10	IAM, RIMES	NN	Accuracy of 6.6% in RIMES and 12.7% in IAM [39]
11	DW ,DS	SVM	WER = (2 + 1 + 1)/6 · 100 = 66% .Because the hypothesis might be longer than the reference, the WER might be greater than 100% [40]
12	NumtaDB	CNN	Training accuracy was 99.66%, validation accuracy was 98.96%, and training loss was 0.0163% [37]
13	CMATERDB and ISI	Autoencoder , DCNN	The greatest recorded result on this dataset to date for the CMATERDB was an accuracy rating of 99.50% [14]
14	Banglalekha-Isolated, Prepared	DCNN	For 10 digits, 99.6% accuracy was achieved on the prepared dataset, and 99.82% accuracy was achieved on the BanglaLekha-Isolated dataset [12]

No.	Dataset	Algorithm	Result
15	NumtaDB	DCNN	Compared to other biased datasets, the huge and unbiased NumtaDB dataset achieved testing accuracy of 92.72%, which was an excellent result [11]
16	MNIST	CNN	With more training photos, the average accuracy improves. The accuracy of the 200 training images was 65.32%, while the accuracy of the 1000 training images was 92.91% [4]
17	MNIST CENPARMI, UCOM, IAM	SVM	When compared to existing methods, the SVM-based HCR method provides 94% accuracy and a good recognition rate [42]
18	ISI, CMATERdb , NumtaDB	DCNN	Achieved recognition accuracy of 96.02% for NumtaDB, 99.10% for CMATERdb, and 98.44% for the ISI dataset, respectively[13]
19	CMATERdb 3.1	DCANN	Proposed DConvAENNet Numbers (10), Basic Characters (50), Compound Characters (171), and All Characters (238) 99.33%, 96.33%, 90.15%, and 92.40% respectively [15]
20	OnHW -chars	KNN, SVM.	DL models improved the accuracy by 7.01%, while ML models increased it by 23.56% [31]
21	EMNIST	CNN	A prediction accuracy of 87.1% was made by the model. With the first epoch, the loss with the Keras model during training started at 0.68 and ended at 0.31 [43]
22	Real world dataset	Heuristic or greedy searching algorithm	A higher reduction of the feature number (up to 70%) implies a reduction of the recognition rate of less than 2%. It was possible to lower the number of features by up to 30% without any loss in terms of recognition rate [33]
23	52,400 instances, Python’s OpenCV	DNN	The proposed model's accuracy was 96.21%, with a loss of 0.2013 [34]
24	CMATERdb 3.1.1	DCNN	Achieved a recognition rate of 98.18% for special character recognition using DenseNet, 99.13% for handwritten Bangla numbers, and 98.31% for handwritten Bangla alphabet [37]

CHAPTER THREE

Methodology and Modeling

The methodology for a handwriting recognition project involved several stages, including data collection, preprocessing, feature extraction, and classification. The first step was to collect a dataset of handwritten images. The next step was to preprocess the dataset by removing any noise or artifacts that might interfere with the recognition process. Techniques like edge detection, contour analysis, and texture analysis can all be used in feature extraction. Techniques like neural networks, support vector machines, and decision trees can all be used in classification. The trained model could then be used to recognize new handwritten characters and texts. The modeling aspect of a handwriting recognition project involved selecting an appropriate algorithm or model architecture to perform the classification task. There were several popular models for handwriting recognition, including: CNN, KNN, SVM, DNN and so on.

3.1 Image Recognition Methods

3.1.1 Steps of Developing the Model

1. **Read the data:** The dataset was being read and the first 10 images were being printed .

2. Splitting into images and their labels: Split the read data into images and their corresponding labels. Since the column "0" contains the labels, we removed the '0' column from the read data frame and used it in y to form the label.

3. **Reshaping the Data in the CSV file:** In the segment above, we were using **train_test_split()** to divide the data into the training and testing datasets. Additionally, we were reshaping the train and test image data, which were initially presented in the CSV file as 784 columns of pixel data, so that they could be displayed as an image. We then changed it to 2828 pixels. The labels were all present as floating point values. We transformed all of the labels from their original form as floating point values to integer values before mapping the integer values to the characters in a dictionary called **word_dict** ().

4. Plotting the numbers: Only the distribution of alphabets was described here. First, we converted the labels to integer values and add the list to the list according to the label. This list showed the number of images in the dataset for each alphabet. Now we created a list - an alphabet containing all the characters dictionary function. Data shuffle: At this point, we had changed the order of some of the train set's images.

5. Data Shuffling: Shuffled some images of the train set. The shuffling was done with the **shuffle ()** function so that we could display random images. Then we created 9 graphics in 3×3 format and show threshold images of 9 alphabets.

3.1.2 Data Reshaping

We modified the train and test image data sets to be incorporated into the model. New shape of the train data (297960, 28, 28, and 1), new shape of the train data (74490, 28, 28, and 1). It was then necessary to modify the train and test image datasets in order to incorporate them into the model. The train data had changed from a float to a categorical data. This was achieved by the CNN model, which took the input of labels and converts the output into a probability vector.

3.1.3 CNN

One of the most used techniques for handwriting recognition was CNN. The image must first go through pre-processing before being fed into the convolutional neural network. The pre-processing steps were as follows:

1) Input the image you wanted to recognize

2) Did editing or twisting. The objective was that the image portion that did not need to be recognized was misplaced

3) Set the image size. All images must be the same size [20]

Figure 1 shows an illustration of Handwriting Recognition Diagram

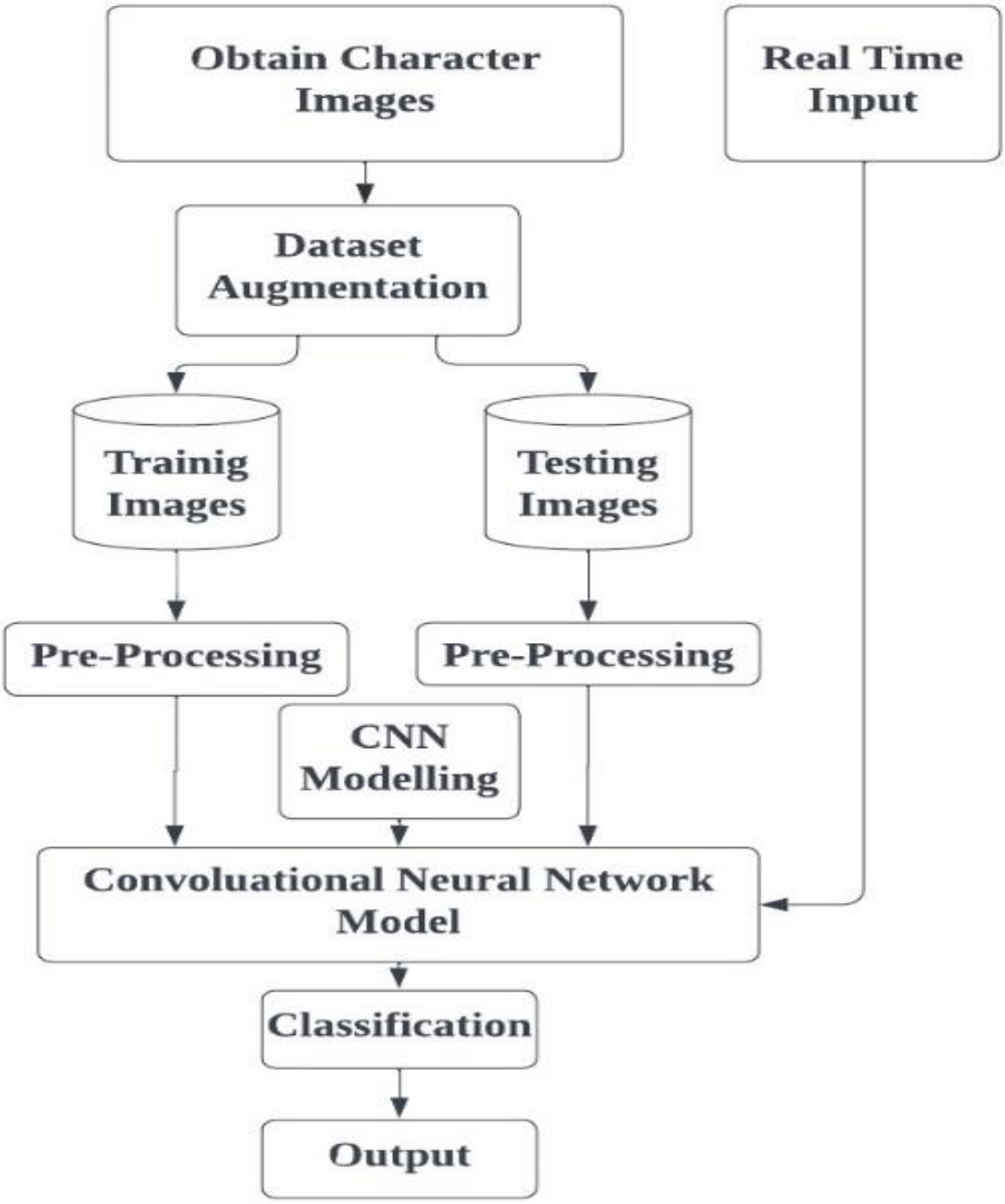

Figure 1: Diagram of Hand Writing Recognition

Incorporating an additional layer into a CNN architecture could have a significant impact on the accuracy of the image detection. Convolutional layer, sub-sampling layer, and fully connected

layer were the three layers that make up CNN in general. But another layer, such as the soft max layer, could also be added. Each layer was connected to the one before it. The soft max level improves the accuracy of image recognition. A CNN architecture example without an additional layer was shown in Figure 2. Depending on the situation, CNN might or might not use the same number of layers. Differences in recognizable handwritten languages also affected the layers and number of layers used. Additional layers inserted into the CNN were optional. Adding an extra layer to the CNN had an effect. For example, adding a soft max layer to the CNN resulted in higher handwriting recognition accuracy than a CNN without the soft max layer. What layers were employed and how many layers were used were also influenced by variations in identifiable handwriting language. On CNN, further layers could be introduced at your discretion. There would be a result if a new layer was added to CNN. For instance, if the soft max layer was added to CNN, the handwriting would change.

A CNN was typically composed of three layers:

1) Convolutional Layer :

Convolutional layer was the introductory layer that built a CNN. The convolution procedure was carried out in this layer. The input picture feature was extracted by this image's convolution procedure. The final convolution layer maintained the spatial location and grayscale information of the convolution feature map.

2) Sub-sampling layer :

Pooling layer (Subsampling). Used to transform an input feature into a statistically-resulted representation of its surrounding features, with the resulting feature size much smaller than the previous feature. Max pooling was mostly used in CNN's subsampling.

3) Fully-Connected Layer :

As a classifier on CNN, this layer might be a CNN design comprising of input layer, hidden layer, and output layer.

4) Soft-max Layer :

On CNN, the soft max layer comes last. A Soft Max Layer was used to represent the output in the form of probabilities. Quite helpful for classifying. The soft max layer was used to classify characters. The soft max function had a value between 0 and 1. The class with the maximum value would be selected as the class for the image, while the smaller value meant not including the main image to be recognized.

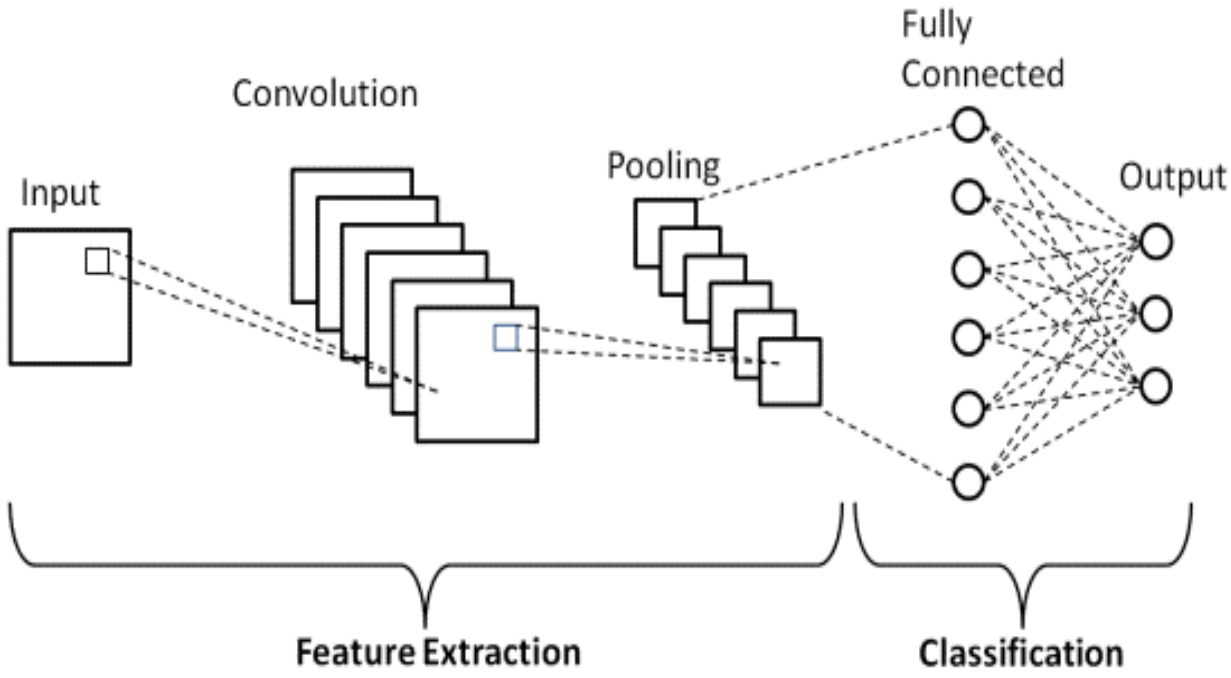

Figure 2: CNN Architecture

Source: Techiepedia [44]

3.2 Experiments

3.2.1 Pseudocode-

The pseudocode of our proposed model given below-

1: Algorithm CNN
2: input: dataset, dataset true labels
3: output: score of Parallel-CNN trained model on
test dataset
4: let f be the featureset 3d matrix
5: for i in dataset do
6: let fi be the featureset matrix of sample i
7: for j in i do
8: vj ← vectorize (g, w)
9: append vj to fi
10: append fi to f
11: ftrain, ftest, ltrain, ltest— split feature set and labels
into train subset and test subset
12: M ← Parallel-CNN (ftrain, ltrain)
13: score ← evaluate (i, ltesi, M)
14: return score

3.3 Project Perquisites

Following were the prerequisites for this project:

1. Python - version 3.7.4
2. IDE - Jupyter

Required frameworks used in the project-

1. Numpy -version 1.16.5
2. CV2 or openCV - version 3.4.2
3. Keras - version 2.3.1
4. Tensorflow (TensorFlow is used Keras in backend and for some image preprocessing) - version 2.0.0
5. Matplotlib - version 3.1.1

6. Pandas - version 0.25.1

3.4 Python Frameworks

Some of the majorly used libraries in python for machine learning were discussed below-

1. NumPy -

The NumPy library was developed shortly after Numeric, an older library, and it was used to handle multi-dimensional data and sophisticated mathematical operations. NumPy was an extremely quick computing library that could handle a wide range of tasks and features, from basic algebra to Fourier transforms, shape manipulations, and random simulations.

2. Pandas -

The complete name for Pandas was Python data analysis library. This allowed us to create multidimensional structures by updating records for numerical data and time series. It used data frames and collecting to simultaneously display 3-dimensional and 2-dimensional data. Additionally, it provided alternatives for quickly indexing huge statistics in large datasets. It was highly known for its skill in reshaping records, dealing with missing records, and merging data and filter data. Pandas could be veritably salutary and rapid-fire with huge datasets.

3. Keras -

Keras was an advanced deep learning API written in Python for neural networks. It helped in computing multiple background networks and made the implementation of neural networks smooth. As a high-level TensorFlow API, Keras was a robust and easy-to-use open-source Python framework for creating deep learning models.

4. Matplotlib -

Matplotlib could be a library utilized in Python for graphical outline to recognize the records some time recently moving it to data-processing and preparing it for Machine learning capacities. It used object-oriented APIs to obtain graphs and associated visualizations using Python GUI toolkits. A similar UI to MATLAB was also provided by Matplotlib so users can carry out operations similar to those found in MATLAB. This open-source, free package offers multiple extension interfaces that extend the matplotlib API to various other libraries. In machine learning, there are four

essential sorts: administered learning, unsupervised learning, semi-supervised learning, and strengthening learning. [18]

5. OpenCV -

OpenCV was an open source computer vision and machine learning computer program library. OpenCV was erected to supply a common frame for computer vision operations and to quicken the use of machine recognition within the marketable particulars. [22]

6. TensorFlow -

TensorFlow was a comprehensive open source machine learning framework. TensorFlow was a versatile system for managing all aspects of a machine learning system. [21]

3.5 Project Modeling

3.5.1 Data Acquisition

A. Synthetically Generated Isolated Printed Characters

Our model was a specialized character prediction model. The training dataset for the model was synthetically generated using the Text Recognition Data Generator module from the Python Package Index. A list of various English fonts was entered into the package along with a series of numbers. For each grapheme in the grapheme set, randomly separated character images were generated using random additions.

B. Datasets of handwritten words

Two handwritten word-level datasets were used to train our model in our experiments: the MNIST dataset of 372450 Alphabet images of 28×2, all present in the form of a CSV train CSV format, and the A-Z English Character dataset. For orienteer-dataset evaluation protocol, we had picked this dataset for training, and the other for testing and also repeated the process by changing the places of the datasets. Random augmentations were used to create randomly isolated character images for every grapheme in the grapheme set. We chose this dataset for training and the other for testing in our inter-dataset assessment strategy.

3.6 MNIST Datasets

The MNIST dataset was utilized to construct this application. This dataset consisted of images of digits ranging from 0 digits to 9 digits, all rendered in a greyscale format. Both training and testing images were included in the dataset, with the training images comprising approximately 60000 large-scale training images and the testing images comprising approximately 10000 smaller-scale testing images. Each training image was composed of a small square (28 x 28 pixels) and a testing image was a handwritten image of an individual digit. The following was how these datasets images appeared.

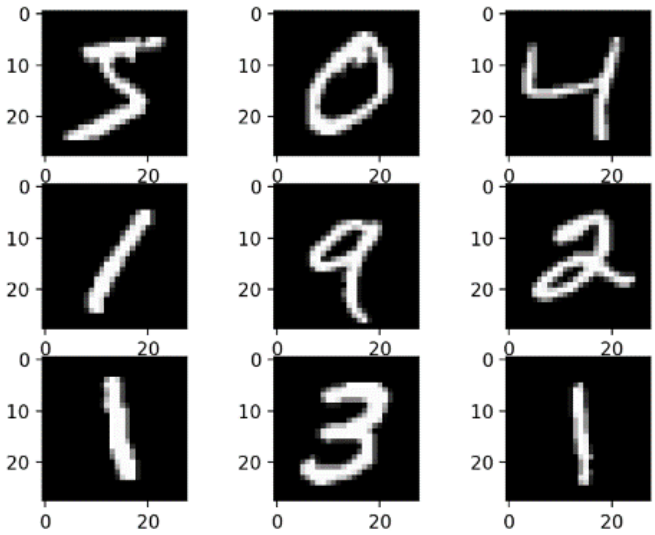

Figure 3: Image of MNIST dataset

Source: Machine Learning Mastery [45]

3.7 Recognition System

The CNN method for recognizing images goes through the following stages:

1. Pre-processing: The image was resized during pre-processing; if it was too large, the calculation would be expensive, or if it was too small, it would be challenging to adapt to big

networks. To get the standard size, larger images were cropped, and padding was added to smaller images.

2. Generating datasets: If an open source dataset was not available for handwriting character recognition, it must be built into a new dataset, but if a dataset was available, an existing dataset could be used.

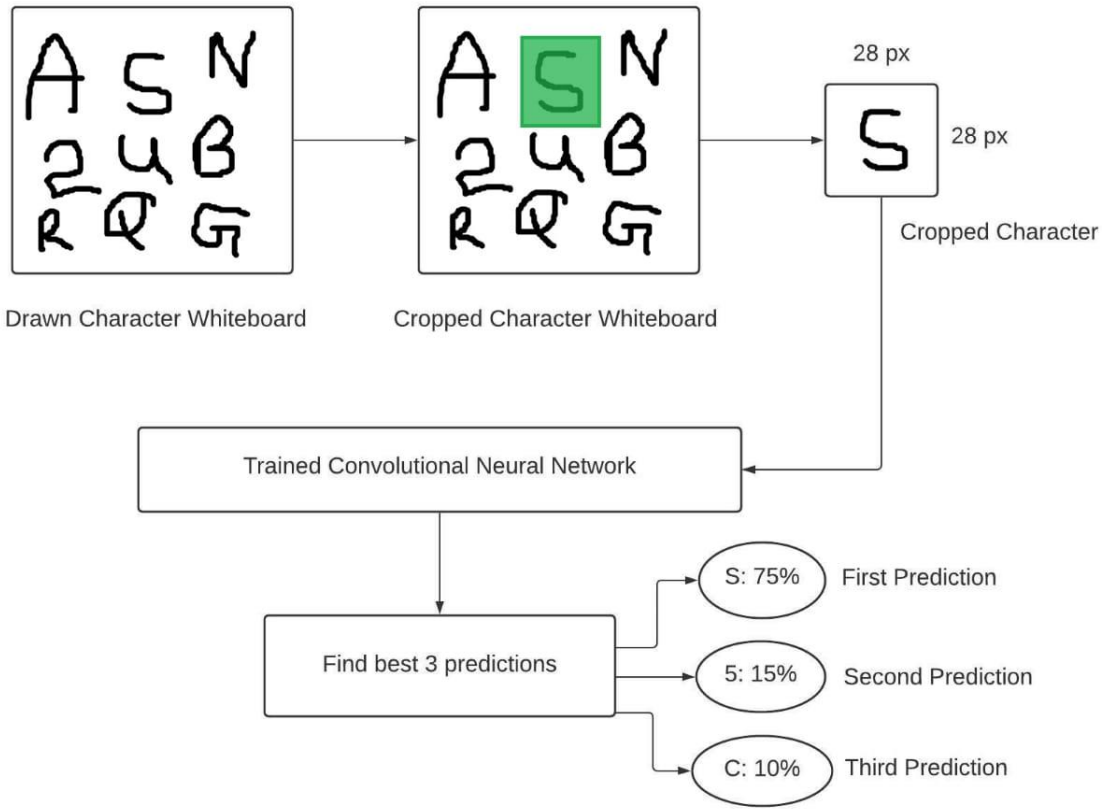

Figure 4: Procedure of the proposed project

3. Determine the final data: A large dataset was required to train a CNN. To achieve this, the resulting images were modified and altered to produce a large number of variations.

4. Classification: The given input image was classified using the Soft max layer, which was the CNN end layer.

5. Testing: The test module was associated with the test image. The test images were obtained by randomly dividing the augmented dataset. [20]

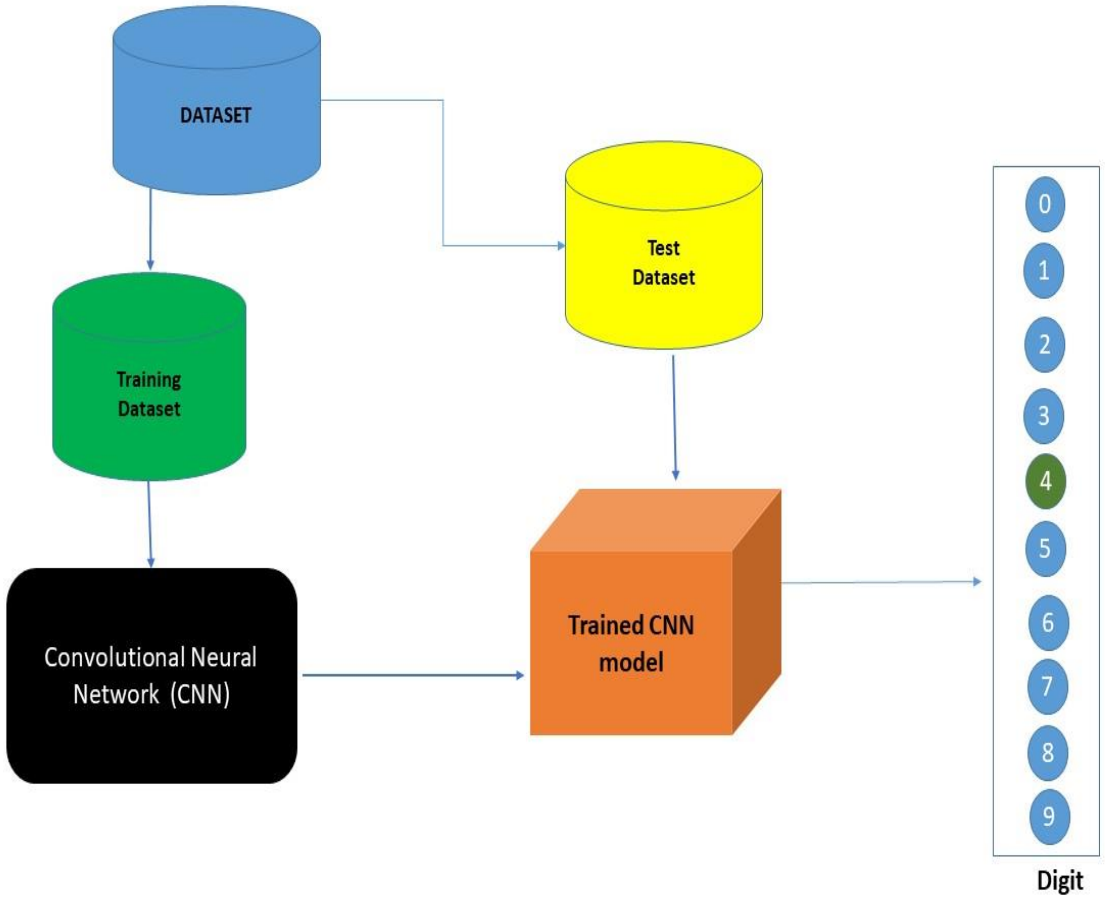

Figure 5: End-to-End CNN

3.8 Console

A console traditionally refers to a computer terminal where a user may input commands and view output such as the results of inputted commands or status messages from the computer. The console is often connected to a remote computer or computer system that is controlled from the console. [46] In our project we used console for the input command of our proposed data's.

Figure 6: A matrix denoting the image of 9 in console

CHAPTER FOUR

Results and Findings

The task of handwriting character recognition involved the accuracy of recognizing and classifying new handwritten characters in a dataset of handwritten characters. The results of such a project could be measured using the F1 score, which combined precision and recall measures. Recent advances in deep learning algorithms, such as CNN and RNNs, have enabled these models to achieve high-quality results on a variety of handwriting recognition benchmarks, including MNIST datasets.

4.1 Experimental Analysis

CNN provided the best accuracy rates for both datasets, even when cross-testing, showing the usefulness of unsupervised training for this purpose. Tables I and II showed previously published results for A-Z handwriting and MNIST digit data sets. Table III showed the accuracy rates for different experiments. For MNIST digit data set, our CNN configuration was pretty close to 100% accuracy at 98.87%. In fact, out of all the ConvNet data sets, our reported result came in second. When you train a CNN model on the tiny A-Z handwriting dataset and test it on the larger MNIST dataset, you would still get 98.55% accuracy, which was pretty close to some of the previously published results from Table II. To show how useful supervised pre-training was, we could also see that CNN gives 0.32% better results than SCMA, even with the same data augmentation.

CNN provided the highest accuracy rates for both datasets, including cross-data sets, demonstrating the effectiveness of unsupervised prior training for this purpose. Table I and II summaries the previously published results for A-Z hand written words and MNIST digit datasets. Table III summarizes the error rates obtained in various experiments. For the MNIST digit data set, our CNN configuration provided comparable results at 98.87%.In fact, among all, we also applied our proposed model for the previous dataset, which was used in some research papers. We had used the same preprocessing, and deep CNN model for both datasets. However, we got better results for MNIST than for A-Z English Alphabets. This was because MNIST contains more

complex images, which were difficult to recognize. Last but not least, we applied the same procedure to MNIST and found that MNIST digit recognition was 99.70% accurate, while MNIST state of the art was 99.79%.Finally, we found that EMNIST digit and letter recognition was also 99.79% accurate, with 47 balanced letter classes. Table IV illustrates the improved accuracy of our proposed CNN model compared to an earlier work on an EMNIST digit and balanced dataset using a linear and OPIUM classifier.

4.2 Evaluation

To figure out how well the spotting algorithm works, we looked at how many times we'd run it on each of the samples in the test data set with different window sizes and overlap. As we said in 3, the spotting algorithm was biased towards recognizing handwriting, so we expect it to have a high recall and a low precision. Since we had a lot more non-handwritten data than handwriting data, we also looked at the specificity, which told us how many non-handwritten samples we'd classified correctly. This number was related to the number of times we'd seen no handwriting and no handwriting had been spotted. We could get really high recall values up to 98%, which was possible with different combinations of window size and shift width. For example, if you had a small shift width between two windows, you could get the highest recall because the combined classifier, CCOMB, includes all the windows that a sensor sample was in. On the other hand, you could get a really small shift width and a really high recall if you had a really big shift between window .

4.3 Combined Evaluation

Table II: Previous Works on A-Z Handwritten Alphabet

Algorithm	Accuracy
CNN [31]	98.47%
CNN [40]	99.66%
SVM [4]	98.08%
DCNN [37]	99.13%

Table III: Previous Works on MNIST Dataset

Work	Accuracy
CNN [29]	96.80%
KNN [30]	96.91%
SVM [15]	98.18%

We observed mainly three results from our experiment. These were training accuracy, validation accuracy, and testing accuracy. After 10 epochs we got this result. Table IV shows our training, validation result as well as the testing result for un-weighted average accuracy.

Table IV: Accuracy Comparison among Different Algorithms

Algorithm	Training	Validation	Testing
CNN	99.47%	98.61%	98.97%
KNN	94.03%	92.65%	92.71%
SVM	93.05%	90.85%	91.45%
DCNN	97.59%	97.05%	97.13%

The processing time of training was also measured. It took 12 seconds per epoch. As there are total 10 epochs, the total training time was 10×12=120 seconds.

The accuracy vs. epoch graph in Figure 6 shows that the difference between train and validation accuracy was fairly small when trained with 50% dropout. The result was more generalized as a result.

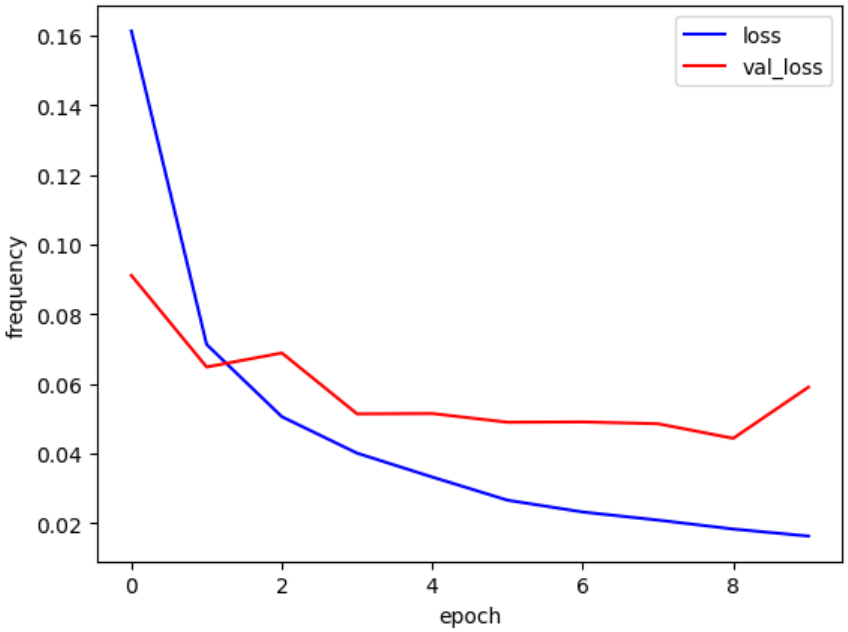

Figure 7: Frequency vs. epoch graph of CNN model on MNIST datasets

As illustrated in Figure 7, using 10% dropout, we could reduce overfitting and get 99.47 percent train accuracy and 98.61 percent validation accuracy.

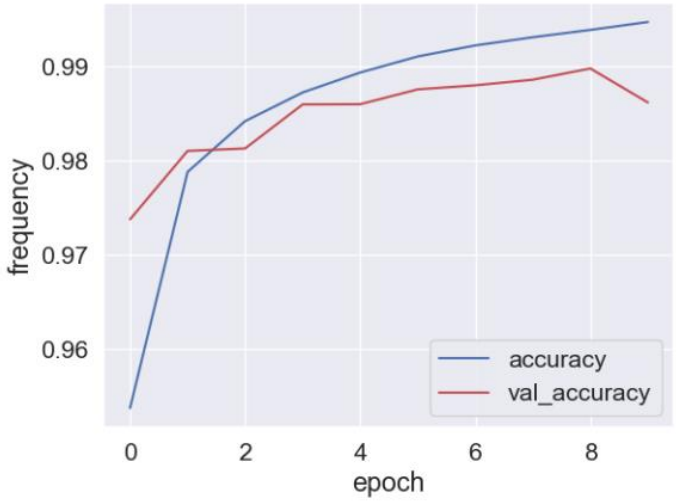

Figure 8: Frequency vs. epoch graph on CNN model with two different merged datasets

CHAPTER FIVE

Conclusions and Recommendations

5.1 Conclusion

In this paper, we showed how unsupervised pre-training with an auto encoder could be used as a stepping stone to supervised training for English handwritten digit recognition. We evaluated this method using three distinct training settings on two separate datasets of Standard English characters to show its efficacy. The suggested approach yields accuracy rates for the test images of the A-Z Handwritten Alphabets dataset that were comparable to earlier research. For the A-Z Written by hand Letter set, it accomplishes a precision rate of 99.50%, which was the leading detailed result on this dataset so distant. Separated from demonstrating the utility of unsupervised pre-training within the setting of English digit acknowledgment, our project came about too demonstrate that such pre-training could be valuable indeed when the information sets were autonomously and indiscriminately collected. In every experiment, the demonstrate with CNN gives superior exactness rate than the demonstrate with as it were ConvNet. The suggested method yields cutting-edge results for the MNIST dataset and excellent results for the A-Z Hand Written dataset also. It was worth saying that past considers on English manually written digit acknowledgment once in a while detailed on these two datasets together. Future research about this project could investigate whether pre-training on bigger datasets, which were not particular to English, can offer assistance accomplish way better came about to be essentially valuable.

5.2 Future Goal

Handwriting recognition technology had come a long way in recent years, but there was still room for improvement. Here were some potential areas for future work in handwriting recognition: improved accuracy, multilingual support, real-time recognition, and integration with other technologies, recognition of mathematical symbols and equations, support for low-resource languages.

REFERENCES

References

1. Md Zahangir Alom, Paheding Sidike, Mahmudul Hasan, Tarek M. Taha, Vijayan K. Asari (2018) ; "Handwritten Bangla Character Recognition Using the State-of-the-Art Deep Convolutional Neural Networks"; Computational Intelligence and Neuroscience, vol. 2018; https://doi.org/10.1155/2018/6747098

2. M. Nakagawa, K. Akiyama, T. Oguni, and N. Kato (2019); "Handwriting Based User Interfaces Employing On-Line Handwriting Recognition"; https://doi.org/ 10.1142/978981 2797650_0057

3. Das, T.R., Hasan, S. Jani, Md. R Tabassum, F and Islam, Md I. (2021); “Bangla Handwritten Character Recognition Using Extended Convolutional Neural Network”; Journal of Computer and Communications; https://doi.org/10.4236/jcc.2021.93012

4. Khandokar (2021); “Handwritten Character Recognition using Convolutional Neural Network”; J. Phys.; https://doi.org/10.1088/1742-6596/1918/4/042152

5. Yintong Wang (2021); "Offline Handwritten Text Recognition Using Deep Learning: A Review"; J. Phys. ; https://doi.org/10.1088/1742-6596/1848/1/012015

6. Hamdan, Y. B. & Sathesh, A. (2021); “Construction of Statistical SVM based Recognition Model for Handwritten Character Recognitio”; Journal of Information Technology and Digital World ; https://doi.org/10.36548/jitdw.2021.2.003

7. Song-Yang Cheng, Yu-Jie Xiong, Jun-Qing Zhang & Yan-Chun Cao (2022) ; " Handwriting and Hand-Sketched Graphics Detection Using Convolutional Neural Networks"; https://doi.org/ 10.1007/978-3-030-59830-3_31

8. Mst Shapna Akter, Hossain Shahriar, Alfredo Cuzzocrea, Nova Ahmed, Carson Leung (2019) ; "Handwritten Word Recognition using Deep Learning Approach: A Novel Way of Generating Handwritten Words"; https://doi.org/10.48550/arXiv.2303.07514

9. Owais Mujtaba Khanday, Dr. Samad Dadvandipor (2021); "Analysis of Machine Learning Algorithms for Character Recognition: A Case Study on Handwritten Digit Recognition";

Indonesian Journal of Electrical Engineering and Computer Science; Vol. 21, https://doi.org/10.11591/ijeecs.v21.i1. pp574-581

10. Ali, S., Shaukat, Z., Azeem, M. (2019); "An Efficient and Improved Scheme for Handwritten Digit Recognition Based on Convolutional Neural Network"; https://doi.org/10.1007/s42452-019-1161-5

11. S. M. Azizul Hakim and Asaduzzaman (2019); "Handwritten Bangla Numeral and Basic Character Recognition Using Deep Convolutional Neural Network," International Conference on Electrical, Computer and Communication Engineering (ECCE); https://doi.org/10.1109/ECACE.2019.8679243

12. Odeh, A., Odeh, M., Odeh, H., Odeh, N. (2022); “Hand-written text recognition methods: Review Study”; Revue d'Intelligence Artificielle; Vol. 36 ; https://doi.org/10.18280/ria.360219

13. Paheding Sidike,Mahmudul Hasan,Tarek M. Taha,and Vijayan K. Asari(2018); "Handwritten Bangla Character Recognition Using the State-of-the-Art Deep Convolutional Neural Networks"; https://doi.org/10.1155/2018/6747098

14. M. Shopon, N. Mohammed and M. A. Abedin (2016); "Bangla handwritten digit recognition using autoencoder and deep convolutional neural network," International Workshop on Computational Intelligence (IWCI); https://doi.org/10.1109/IWCI.2016.7860340

15. M. A. Azad, H. S. Singha and M. M. H. Nahid (2020); "Bangla Handwritten Character Recognition Using Deep Convolutional Autoencoder Neural Network" ; 2nd International Conference on Advanced Information and Communication Technology (ICAICT); https://doi.org/10.1109/ICAICT51780.2020.9333472

16. Md. Hadiuzzaman Bappy, Md. Siamul Haq, Kamrul Hasan Talukder (2019); "Bangla Handwritten Numeral Recognition using Deep Convolutional Neural Network”; https://doi.org/10.53808/KUS.2022.ICSTEM4IR.0166-se

17. Shahrukh Ahsan , Shah Tarik Nawaz , Talha Bin Sarwar , M. Saef Ullah Miah , Abhijit Bhowmik (2022); "A Machine Learning Approach for Bengali Handwritten Vowel Character Recognition"; IAES International Journal of Artificial Intelligence (IJ-AI); Vol. 11, No. 3; https://doi.org/ 10.11591/ijai.v11i3.pp1143-1152

18. Darmatasia and M. I. Fanany (2017); "Handwriting Recognition form Document using Convolutional Neural Network and Support Vector Machines (CNN-SVM)" ; 5th International Conference on Information and Communication Technology (ICoIC7); https://doi.org/ 10.1109/ICoICT.2017.8074699

19. A. Srivastava and P. Singh (2019); “Handwritten Digit Image Recognition Using Machine Learning” ; Journal of Informatics Electrical and Electronics Engineering(JIEEE); Vol. 03; https://doi.org/10.54060/JIEEE/003.02.003

20. A. A. Ahmed Ali, M. Suresha and H. A. Mohsin Ahmed (2019); "Different Handwritten Character Recognition Methods: A Review," Global Conference for Advancement in Technology (GCAT), https://doi.org/ 10.1109/GCAT47503.2019.8978347

21. Saniya Firdous (2022); "Handwritten Character Recognition"; International Journal for Research in Applied Science & Engineering Technology (IJRASET); Volume 10; https://doi.org/10.22214/ijraset.2022.42114

22. Tsige Tadesse AlemayohORCID, Masaaki Shintani, Jae Hoon Lee ORCID and Shingo Okamoto (2022); "Deep-Learning-Based Character Recognition from Handwriting Motion Data Captured Using IMU and Force Sensors"; https://doi.org/10.3390/s22207840

23. Asha K, Krishnappa H K (2022); "Handwriting Recognition using Deep Learning based on Convolutional Neural Network"; https://doi.org/10.35940/ijrte.D7811.118419

24. Ayşe Ayvaci Erdoğan, A. E. Tümer (2021);"Deep Learning Method for Handwriting Recognition"; https://doi.org/10.51354/mjen.852312

25. Dibyakanti Mahapatra, Chandrajit Choudhury, R. Karsh (2020); "Handwritten Character Recognition Using KNN and SVM Based Classifier over Feature Vector from Autoencoder"; International Conference ; https://doi.org/10.1007/978-981-15-6315-7_25

26.P. Ghadekar, S. Ingole and D. Sonone (2018); "Handwritten Digit and Letter Recognition Using Hybrid DWT-DCT with KNN and SVM Classifier" ; Fourth International Conference on Computing Communication Control and Automation (ICCUBEA); https://doi.org/ 10.1109/ICC UBEA.2018.8697684

27. A. Baldominos, Y. Sáez, P. Isasi (2019); “A Survey of Handwritten Character Recognition with MNIST and EMNIST"; https.//doi.org/10.3390/APP9153169

28. A. K. Agrawal, A. K. Shrivas and V. k. Awasthi (2021); "A Robust Model for Handwritten Digit Recognition using Machine and Deep Learning Technique," 2nd International Conference for Emerging Technology (INCET); https://doi.org/10.1109/INCET51464.2021.9456118

29. Al-Mahmud, A. Tanvin and S. Rahman (2021); "Handwritten English Character and Digit Recognition," International Conference on Electronics, Communications and Information Technology (ICECIT); https://doi.org/10.1109/ICECIT54077.2021.9641160

30. K. Upender and V. S. K. Pasupuleti (2021); "Real Time Handwritten Digits Recognition Using Convolutional Neural Network," International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE); https://doi.org/10.1109/ICA CITE5 1222.2021.9404615

31. Savita Ahlawat,Amit Choudhary, Anand Nayyar, Saurabh Singh and Byungun Yoon (2021); "Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)"; https://doi.org/10.3390/s20123344

32. Sparsh Kotriwal, Nitesh Pradhan, Vijaypal Singh (2019); “Handwritten Character Recognition using Deep Neural Network"; https://doi.org/10.1145/3484824.3484901

33. Agarwal, A., Saraswat, M. (2022); “Analyzing Various Handwriting Recognition Phenomenon for Predicting Gender, Age and Handedness”; vol 1738; Springer; https://doi.org/10.1007/978-3-0

34. Jana, R., Bhattacharyya, S. (2019); “Character Recognition from Handwritten Image Using Convolutional Neural Networks”; vol 922; Springer; https://doi.org/10.1007/978-981-13-6783-0_3

35. Shashank, R., Adarsh Rai, A., Srinivasa Pai, P. (2022); “Handwritten Character Recognition Using Deep Convolutional Neural Networks”; vol 1386; Springer; https://doi.org/10.1007/978-981-16-3342-3_21

36. Abir, B.M., Mahal, S.N., Islam, M.S., Chakrabarty, A. (2019); “Bangla Handwritten Character Recognition with Multilayer Convolutional Neural Network”; vol 39; Springer; Singapore. https://doi.org/10.1007/978-981-13-0277-0_13

37. Rabeya Basri, Mohammad Reduanul Haque, Morium Akter, Mohammad Shorif Uddin (2020); "Bangla handwritten digit recognition using deep convolutional neural network"; Proceedings of the international conference on computing advancements; https://doi.org/10.1145/3377049.3377077

38. R. Jadhav, S. Gadge, K. Kharde, S. Bhere and I. Dokare (2022); "Recognition of Handwritten Bengali Characters using Low Cost Convolutional Neural Network," Interdisciplinary Research in Technology and Management (IRTM); https://doi.org/10.1109 /IRTM54583.2022.9791802

39. Chakraborty, P., Islam, A., Abu Yousuf, M., Agarwal, R., Choudhury, T. (2022); “Bangla Handwritten Character Recognition Using Convolutional Neural Network. vol 132; Springer; https://doi.org/10.1007/978-981-19-2347-0_56

40. K. K. Rabbi, A. Hossain, P. Dev, A. Sadman, D. Z. Karim and A. A. Rasel (2022); "KDANet: Handwritten Character Recognition for Bangla Language using Deep Learning," 25th International Conference on Computer and Information Technology (ICCIT); https:// doi.org/ 10.1109/ICCIT57492.2022.10054708

41. S. Ahmed, F. Tabsun, A. S. Reyadh, A. I. Shaafi and F. M. Shah (2019); "Bengali Handwritten Alphabet Recognition using Deep Convolutional Neural Network," International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2); https://doi.org/ 10.1109/IC4ME247184.2019.9036572

42. AKM Shahariar Azad Rabby , Sadeka Haque , Sheikh Abujar , Syed Akhter Hossain (2021); "EkushNet: Using Convolutional Neural Network for Bangla Handwritten Recognition"; https://doi.org/10.1016/j.procs.2018.10.437

43. Akm Shahariar Azad Rabby , Sadeka Haque , Sanzidul Islam (Md.) , Sheikh Abujar , Syed Akhter Hossain (2019); "BornoNet: Bangla Handwritten Characters Recognition Using Convolutional Neural Network"; https://doi.org/10.1016/j.procs.2018.10.426

44. Sei Balaji (2020); Binary Image Classifier CNN using Tensorflow; Techiepedia

45. Jason Brownlee (2019); Deep Learning for Computer Vision ; Machine Learning Mastery

46. Margaret Rouse (2014); techopedia

APPENDIX

Supplementary Code

Supplementary Code

```
import cv2
import numpy as np
from keras.models import Sequential
from keras.layers import Conv2D, Activation, MaxPooling2D, Flatten, Dense, Dropout,
BatchNormalization

def clear_whiteboard(display):
    wb_x1, wb_x2, wb_y1, wb_y2 = whiteboard_region["x"][0], whiteboard_region["x"][1],
whiteboard_region["y"][0], whiteboard_region["y"][1]

    display[wb_y1-10:wb_y2+12, wb_x1-10:wb_x2+12] = (255, 255, 255)

def setup_display():
    title = np.zeros((80, 950, 3), dtype=np.uint8)
    board = np.zeros((600, 650, 3), dtype=np.uint8)
    panel = np.zeros((600, 300, 3), dtype=np.uint8)
    board[5:590, 8:645] = (255, 255, 255)

    board = cv2.rectangle(board, (8, 5), (645, 590), (255, 0, 0), 3)
    panel = cv2.rectangle(panel, (1, 4), (290, 590), (0, 255, 192), 2)
    panel = cv2.rectangle(panel, (22, 340), (268, 560), (255, 255, 255), 1)
    panel = cv2.rectangle(panel, (22, 65), (268, 280), (255, 255, 255), 1)

    cv2.line(panel, (145, 340), (145, 560), (255, 255, 255), 1)
    cv2.line(panel, (22, 380), (268, 380), (255, 255, 255), 1)

    cv2.putText(title, "       " +  window_name,(10, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.2,
(255, 255, 255), 2)
    cv2.putText(panel, "Action: ", (23, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255,
255), 1)
    cv2.putText(panel, "Best Predictions", (52, 320), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(255, 255, 255), 1)
    cv2.putText(panel, "Prediction", (42, 362), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255,
255), 1)
    cv2.putText(panel, "Accuracy", (168, 362), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,
255, 255), 1)
```

```
    cv2.putText(panel, actions[0], (95, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
action_colors[actions[0]], 1)

    display = np.concatenate((board, panel), axis=1)
    display = np.concatenate((title, display), axis=0)

    return display

def setup_panel(display):
    action_region_pt1, action_region_pt2 = status_regions["action"]
    preview_region_pt1, preview_region_pt2 = status_regions["preview"]
    label_region_pt1, label_region_pt2 = status_regions["labels"]
    acc_region_pt1, acc_region_pt2 = status_regions["accs"]

    display[action_region_pt1[1]:action_region_pt2[1],
action_region_pt1[0]:action_region_pt2[0]] = (0, 0, 0)
    display[preview_region_pt1[1]:preview_region_pt2[1],
preview_region_pt1[0]:preview_region_pt2[0]] = (0, 0, 0)
    display[label_region_pt1[1]:label_region_pt2[1], label_region_pt1[0]:label_region_pt2[0]] =
(0, 0, 0)
    display[acc_region_pt1[1]:acc_region_pt2[1], acc_region_pt1[0]:acc_region_pt2[0]] = (0, 0,
0)

    if crop_preview is not None:
        display[preview_region_pt1[1]:preview_region_pt2[1],
preview_region_pt1[0]:preview_region_pt2[0]] = cv2.resize(crop_preview, (crop_preview_h,
crop_preview_w))

    if best_predictions:
        labels = list(best_predictions.keys())
        accs = list(best_predictions.values())
        prediction_status_cordinate = [
            ((725, 505), (830, 505), (0, 0, 255)),
            ((725, 562), (830, 562), (0, 255, 0)),
            ((725, 619), (830, 619), (255, 0, 0))
        ]
        for i in range(len(labels)):
            label_cordinate, acc_cordinate, color = prediction_status_cordinate[i]

            cv2.putText(display, labels[i], label_cordinate, cv2.FONT_HERSHEY_SIMPLEX, 0.8,
color, 2)
            cv2.putText(display, str(accs[i]), acc_cordinate, cv2.FONT_HERSHEY_SIMPLEX, 0.8,
color, 2)

        for i in range(len(labels), 3):
            label_cordinate, acc_cordinate, color = prediction_status_cordinate[i]
```

```
        cv2.putText(display, "_", label_cordinate, cv2.FONT_HERSHEY_SIMPLEX, 0.8, color,
2)
        cv2.putText(display, "_", acc_cordinate, cv2.FONT_HERSHEY_SIMPLEX, 0.8, color,
2)

    cv2.putText(display, current_action, (745, 120), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
action_colors[current_action], 1)

def arrange_crop_rectangle_cordinates(cor1, cor2):
    if cor1 is None or cor2 is None:
        return

    result = ()
    if cor1[1] < cor2[1]:
        if cor1[0] > cor2[0]:
            result = ( (cor2[0], cor1[1]), (cor1[0], cor2[1]) )
        else:
            result = (cor1, cor2)
    else:
        if cor2[0] > cor1[0]:
            result = ( (cor1[0], cor2[1]), (cor2[0], cor1[1]) )
        else:
            result = (cor2, cor1)
    return result

def mouse_click_event(event, x, y, flags, params):
    if current_action is actions[1]:
        whiteboard_draw(event, x, y)
    elif current_action is actions[2]:
        character_crop(event, x, y)

def whiteboard_draw(event, x, y):
    global left_button_down, right_button_down

    wb_x1, wb_x2, wb_y1, wb_y2 = whiteboard_region["x"][0], whiteboard_region["x"][1],
whiteboard_region["y"][0], whiteboard_region["y"][1]

    if event is cv2.EVENT_LBUTTONUP:
        left_button_down = False
    elif event is cv2.EVENT_RBUTTONUP:
        right_button_down = False
    elif wb_x1 <= x <= wb_x2 and wb_y1 <= y <= wb_y2:
        color = (0, 0, 0)
        if event in [cv2.EVENT_LBUTTONDOWN, cv2.EVENT_LBUTTONUP,
cv2.EVENT_RBUTTONDOWN, cv2.EVENT_RBUTTONUP, cv2.EVENT_MOUSEMOVE]:
```

```
        if event is cv2.EVENT_LBUTTONDOWN:
            color = (0, 0, 0)
            left_button_down = True
        elif left_button_down and event is cv2.EVENT_MOUSEMOVE:
            color = (0, 0, 0)
        elif event is cv2.EVENT_RBUTTONDOWN:
            color = (255, 255, 255)
            right_button_down = True
        elif right_button_down and event is cv2.EVENT_MOUSEMOVE:
            color = (255, 255, 255)
        else:
            return

        cv2.circle(display, (x, y), 10, color, -1)
        cv2.imshow(window_name, display)

def character_crop(event, x, y):
    global bound_rect_cordinates, lbd_cordinate, lbu_cordinate, crop_preview, display,
best_predictions

    wb_x1, wb_x2, wb_y1, wb_y2 = whiteboard_region["x"][0], whiteboard_region["x"][1],
whiteboard_region["y"][0], whiteboard_region["y"][1]

    if wb_x1 <= x <= wb_x2 and wb_y1 <= y <= wb_y2:
        if event is cv2.EVENT_LBUTTONDOWN:
            lbd_cordinate = (x, y)
        elif event is cv2.EVENT_LBUTTONUP:
            lbu_cordinate = (x, y)

        if lbd_cordinate is not None and lbu_cordinate is not None:
            bound_rect_cordinates = arrange_crop_rectangle_cordinates(lbd_cordinate,
lbu_cordinate)
        elif lbd_cordinate is not None:
            if event is cv2.EVENT_MOUSEMOVE:
                mouse_move_cordinate = (x, y)
                mouse_move_rect_cordinates = arrange_crop_rectangle_cordinates(lbd_cordinate,
mouse_move_cordinate)
                top_cordinate, bottom_cordinate = mouse_move_rect_cordinates[0],
mouse_move_rect_cordinates[1]

                display_copy = display.copy()
                cropped_region = display_copy[top_cordinate[1]:bottom_cordinate[1],
top_cordinate[0]:bottom_cordinate[0]]
                filled_rect = np.zeros((cropped_region.shape[:]))
                filled_rect[:, :, :] = (0, 255, 0)
                filled_rect = filled_rect.astype(np.uint8)
```

```
            cropped_rect = cv2.addWeighted(cropped_region, 0.3, filled_rect, 0.5, 1.0)

            if cropped_rect is not None:
                display_copy[top_cordinate[1]:bottom_cordinate[1],
top_cordinate[0]:bottom_cordinate[0]] = cropped_rect
                cv2.imwrite("captured/filled.jpg", display_copy)
                cv2.imshow(window_name, display_copy)

        if bound_rect_cordinates is not None:
            top_cordinate, bottom_cordinate = bound_rect_cordinates[0], bound_rect_cordinates[1]
            crop_preview = display[top_cordinate[1]:bottom_cordinate[1],
top_cordinate[0]:bottom_cordinate[0]].copy()
            crop_preview = np.invert(crop_preview)
            best_predictions = predict(model, crop_preview)
            display_copy = display.copy()
            bound_rect_cordinates = lbd_cordinate = lbu_cordinate = None
            setup_panel(display)
            cv2.imshow(window_name, display)
    elif event is cv2.EVENT_LBUTTONUP:
        lbd_cordinate = lbu_cordinate = None
        cv2.imshow(window_name, display)

def load_model(path):
    model = Sequential()

    model.add(Conv2D(32, (5, 5), input_shape=(28, 28, 1), activation="relu"))
    model.add(BatchNormalization())

    model.add(Conv2D(32, (5, 5), activation="relu"))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(2, 2))
    model.add(Dropout(0.25))

    model.add(BatchNormalization())
    model.add(Flatten())

    model.add(Dense(256, activation="relu"))
    model.add(Dense(36, activation="softmax"))

    model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
    model.load_weights(path)

    return model

def predict(model, image):
```

```
    labels = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
'W', 'X', 'Y', 'Z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = cv2.resize(image, (28, 28))
    image = image / 255.0
    image = np.reshape(image, (1, image.shape[0], image.shape[1], 1))
    prediction = model.predict(image)
    best_predictions = dict()

    for i in range(3):
        max_i = np.argmax(prediction[0])
        acc = round(prediction[0][max_i], 1)
        if acc > 0:
            label = labels[max_i]
            best_predictions[label] = acc
            prediction[0][max_i] = 0
        else:
            break

    return best_predictions

left_button_down = False
right_button_down = False
bound_rect_cordinates = lbd_cordinate = lbu_cordinate = None
whiteboard_region = {"x": (20, 632), "y": (98, 656)}
window_name = "Live Cropped Character Recognition"
best_predictions = dict()
crop_preview_h, crop_preview_w = 238, 206
crop_preview = None
actions = ["N/A", "DRAW", "CROP"]
action_colors = {
    actions[0]: (0, 0, 255),
    actions[1]: (0, 255, 0),
    actions[2]: (0, 255, 192)
}
current_action = actions[0]
status_regions = {
    "action": ((736, 97), (828, 131)),
    "preview": ((676, 150), (914, 356)),
    "labels": ((678, 468), (790, 632)),
    "accs": ((801, 468), (913, 632))
}
model = load_model("../models/best_val_loss_model.h5")

display = setup_display()
```

```
cv2.imshow(window_name, display)
cv2.setMouseCallback(window_name, mouse_click_event)
pre_action = None

while True:
    k = cv2.waitKey(1)
    if k == ord('d') or k == ord('c'):
        if k == ord('d'):
            current_action = actions[1]
        elif k == ord('c'):
            current_action = actions[2]
        if pre_action is not current_action:
            setup_panel(display)
            cv2.imshow(window_name, display)
            pre_action = current_action
    elif k == ord('e'):
        clear_whiteboard(display)
        cv2.imshow(window_name, display)
    elif k == 27:
        break

cv2.destroyAllWindows()
```