

+

Handwritten Digits Classification System

A Project Report Phase - II

Submitted in partial fulfillment of the requirements for the award of Bachelor
of Engineering Degree in Electronics and Communication Engineering

By

G. BEJESH SRIRAM (Reg. No. 39130061)

M. R. AADITHYA (Reg. No. 39130002)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI, SALAI, CHENNAI - 600 119

APRIL - 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

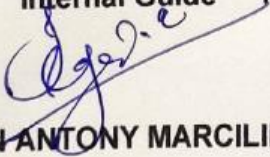
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

BONAFIDE CERTIFICATE

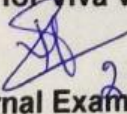
This is to certify that this Project Report is the bonafide work of **G. BEJESH SRIRAM** (Reg. no. **39130061**) and **M. R. AADITHYA** (Reg. no. **39130002**) who carried out the project entitled “HANDWRITTEN DIGITS CLASSIFICATION SYSTEM” under my supervision from November 2022 to April 2023.

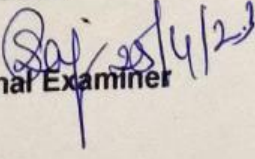
Internal Guide

Mr. L. JEGAN ANTONY MARCILIN, M.Tech.,

Head of the Department

Dr. T. RAVI, M.E., Ph.D.,

Submitted for Viva voice Examination held on 25/04/2023

 25/4/23
Internal Examiner

 25/4/23
External Examiner

DECLARATION

We, G. BEJESH SRIRAM (Reg. No. 39130061) and M. R. AADITHYA (Reg. No. 39130002) hereby declare that the Project Report entitled “Handwritten Digits Classification System” done by us under the guidance of Mr. L. JEGAN ANTONY MARCILIN, M.Tech., Department of Electronics and Communication Engineering, at SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

SIGNATURE OF THE CANDIDATE

DATE:25-04-23

PLACE: Chennai

1. 

2. 

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

We convey our thanks to **Dr. N.M. Nandhitha, M.E., Ph.D., Dean, School of Electrical and Electronics Engineering** and **Dr. T. RAVI, M.E., Ph.D., Head of the Department, Dept. of Electronics and Communication** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to our Project Guide Mr. L. JEGAN ANTONY MARCILIN, M.Tech., Assistant Professor, Department of Electronics and Communication Engineering, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to the Panel Members and also to all Teaching and Non-teaching staff members of the Department of Electronics and Communication who were helpful in many ways for the completion of the project.

We would like to extend our thanks to our parents who offered various support both directly and indirectly to complete the project with a successful outcome.

ABSTRACT

Machines cannot understand hand written words or digits, so it is necessary to make the machine to understand hand written notes. Here Machine learning Technique is employed. Many hand written notes are feed as sample images to the machine and make to learn the differences between those sample images. Then we feed a new set of images to the machine to identify based on the differences it learnt before. We use the above method, by feeding hand written Digits. There are many algorithms that are being used to Identify/ Classify the digits. Support Vector Machine, Gaussian Naïve Bayes, Decision Tree, Random Forest, K Nearest Neighbors, Stochastic Gradient Descent and many more. The difficulty in reading handwritten characters is mostly brought on by the wide variety of personal writing styles. In order to enhance the performance of a handwritten character recognition system, strong feature extraction is crucial. Due to its many applications, handwritten digit recognition is currently receiving a lot of attention in the field of pattern recognition systems. Character recognition technology may one day operate as the cornerstone for the establishment of a paperless society by processing and digitizing already existent paper records.

CONTENTS

CHAPTER No.	TITLE	PAGE No.
	Abstract	v
	List of Figures	ix
	List of Tables	xi
1	Introduction	1
2	Literature Survey	2
	2.1 Classifiers	15
	2.1.1 Support Vector Machines (SVM)	16
	2.1.2 Gaussian Naive Bayes	17
	2.1.3 Decision Trees	18
	2.1.4 Random Forest	19
	2.1.5 K Nearest Neighbors (KNN)	20
	2.1.6 Stochastic Gradient Descent	21
	2.1.7 Conclusion on Classifiers	22
3	Methodology	23
	3.1 Convolutional Neural Network	23
	3.1.1 Convolutional Layer	24
	3.1.2 Pooling Layer	25
	3.1.2.1 Max Pooling	25
	3.1.2.1 Average Pooling	25
	3.1.3 Fully Connected Layer	26
	3.2 Datasets Used	26

CONTENTS

CHAPTER No.	TITLE	PAGE No.
	3.3 Character Recognition Model	26
	3.3.1 Importing Libraries	27
	3.3.2 Reading Dataset	27
	3.3.3 Splitting of dataset	28
	3.3.4 Training data and Test data Split	28
	3.3.5 Reshaping the Training and Testing Data	28
	3.3.6 Shuffling the training data	28
	3.3.7 Visualizing the training data	29
	3.3.8 Reshaping for training	30
	3.3.9 Conversing to categorical	30
	3.3.10 Model Creation	30
	3.3.11 Model Summary	32
	3.3.12 Compile model	32
	3.3.13 Save model	33
	3.3.14 Creating Words Dictionary	33
	3.3.15 Prediction Process	33
	3.3.16 To Visualize the model	34
	3.4 Digits Recognition Model	35
	3.4.1 Processing the dataset	36
	3.4.2 Creating and Training a CNN	36
	3.4.3 Making Predictions	37

CONTENTS

CHAPTER No.	TITLE	PAGE No.
4	Results	39
	4.1 Digits Recognition Results	39
	4.2 Character Recognition Results	44
5	Conclusion	49
	References	50

LIST OF FIGURES

FIGURE No.	TITLE	PAGE No.
2.1	Image showing Numbers	2
2.2	Support Vector Machine Confusion Matrix	16
2.3	Confusion Matrix for Gaussian Naïve Bayes Classifier	17
2.4	Confusion matrix for Decision Tree	18
2.5	Confusion matrix for Random Forest	19
2.6	Confusion Matrix for K Nearest Neighbor.	20
2.7	Confusion Matrix for Stochastic Gradient Descent.	21
3.1	Block Diagram of Convolution Neural Network	23
3.2	Dataset used in Character recognition	27
3.3	Image Showing the Test Data.	29
3.4	Model Summary	32
3.5	Image Showing the EPOCH	32
3.6	Image Showing the Predictions.	34
4.1	Screen Shot of PyCharm Window	39
4.2	Model Training	39
4.3	Model Predicting Digit 0	40
4.4	Model Predicting Digit 1	40
4.5	Model Predicting Digit 2	41
4.6	Model Predicting Digit 3	41
4.7	Model Predicting Digit 4	42
4.8	Model Predicting Digit 5	42
4.9	Model Predicting Digit 7	43
4.10	Model Predicting Digit 8	43

LIST OF FIGURES

FIGURE No.	TITLE	PAGE No.
4.11	Model Predicting Char A	44
4.12	Model Predicting Char B	44
4.13	Model Predicting Char C	44
4.14	Model Predicting Char D	44
4.15	Model Predicting Char E	45
4.16	Model Predicting Char G	45
4.17	Model Predicting Char H	45
4.18	Model Predicting Char I	45
4.19	Model Predicting Char L	46
4.20	Model Predicting Char N	46
4.21	Model Predicting Char O	46
4.22	Model Predicting Char P	46
4.23	Model Predicting Char Q	47
4.24	Model Predicting Char R	47
4.25	Model Predicting Char T	47
4.26	Model Predicting Char V	47
4.27	Model Predicting Char W	48
4.28	Model Predicting Char X	48
4.29	Model Predicting Char Y	48
4.30	Model Predicting Char Z	48

LIST OF TABLES

TABLE No.	TITLE	PAGE No.
2.1	Support Vector Machines Report	16
2.2	Classification report for Gaussian Naïve Bayes	17
2.3	Classification report for Decision Tree	18
2.4	Classification Report for Random Forest	19
2.5	Classification Report for K Nearest Neighbor	20
2.6	Classification Report for Stochastic Gradient	21
2.7	List of Classifiers Accuracy	22
2.8	List of Classifiers F1-score	22

CHAPTER 1

INTRODUCTION

There are several ways that machine learning may minimize human effort to seeing handwritten numbers. An item or person can be recognized or distinguished by awareness based on prior knowledge or experience. Similar to this, digital recognition is nothing more than the identification or recognition of numbers in any document. The framework of digital recognition just needs the machine to prepare or interpret the digits. The ability of computers to read handwritten numbers from a variety of sources, including text messages, bank checks, documents, pictures, etc., is known as handwritten digit recognition.

Character recognition has existed since the 1980s. The handwritten digital recognition feature, which uses a separator, is extremely valuable and is used in many different contexts, such as online digital recognition on PC tablets, publishing zip codes, processing bank check rates, and scribbling numerical categories (such as tax forms). While attempting to tackle this problem, there are a number of challenges that must be addressed. Numbers written by hand may not always have the same size, thickness, or shape as those dictated by genes. The main objective was to use character matching to accurately identify handwritten numbers presented in the MINIST dataset for handwritten digital images (0–9).

A well-studied subject of handwriting recognition involves teaching models to detect pre-segmented handwritten numbers. In the domains of artificial intelligence that deal with data mining, machine learning, pattern recognition, and many other things, it is one of the most important subjects. Since the invention of image recognizing technology, methods such as machine learning have mostly been utilized to create artificial neural systems that can compete with humans in performance and surpass manually developed conventional recognizing systems.

CHAPTER 2

LITERATURE SURVEY

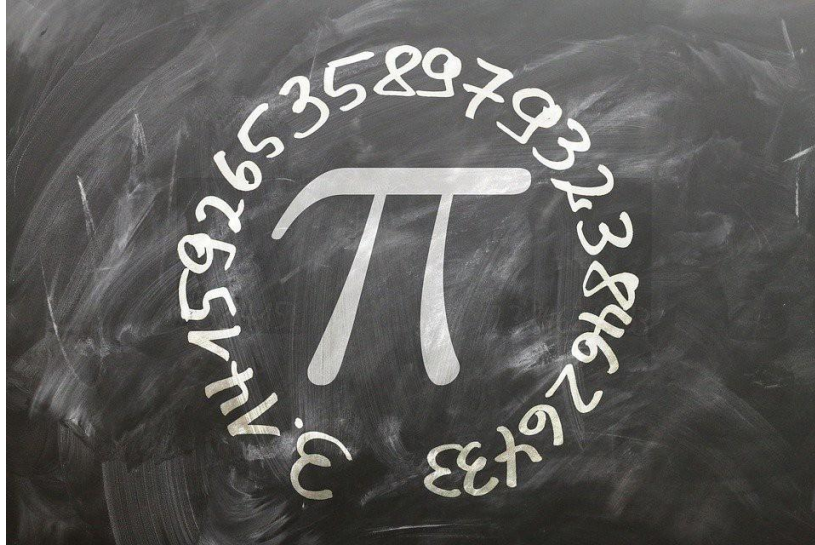


Fig:2.1: Image showing Numbers

Identifying the handwritten digits using different classification techniques is the intriguing machine learning topic known as "Handwritten Digit Recognition." Numerous techniques and algorithms can be used to recognize handwritten digits.

HANDWRITTEN ENGLISH CHARACTER AND DIGITS RECOGNITION [2]
2021 – Al-Mahmud, Asnuva Tanvin, Sazia Rahman.

The architecture of CNN employed in this study. Two versions of the model have been modified one to train on the MNIST dataset and the other to train on the combined dataset. There is a 28 by 28-pixel picture in the input layer. The first convolutional layer has a total of 32 filters which are of size 5X5 and uses ReLU function for activation. A second convolutional layer comprises 64 filters and also a ReLU function for activation, and a 7 x 7 sized window. Next, down sampling is performed using a second pooling layer with a 2 X 2 pool size and 2 stride.

Then, a flatten layer is used to transform the 2-D matrix into a 1-D array. In the CNN model for the MNIST dataset, 50% of the population drops out after the flatten layer. After that, fully-linked layer is built using 1024 nodes from the output. In order to further decrease overfitting, another 50% dropout is used. A completely linked layer is then implemented as an output layer. It has 10 output nodes, which may accept network results from 0 to 9.

HANDWRITTEN DIGIT RECOGNITION USING ROTATIONS [3] 2016 - Anca Ignat, Bogdan Aciobanitei.

Handwritten digit pictures are typically used to test the algorithms for recognition of digits and, more widely on machine learning techniques. To extract features from images with digits, straightforward image processing techniques including threshold binarization, image rotation, and Sobel filters for the edge, are utilized. Rotations are geometrical changes that include rotating the grid used to represent the picture by a specific angle before computing the pixel values on the new grid. To identify and highlight lines in digital pictures, Sobel filters, which are 3x3 first order derivative filters, are used.

FARSI HANDWRITING DIGIT RECOGNITION BASE ON CONVOLUTIONAL NEURAL NETWORKS [4] 2018 – Atefeh Dehghanian, Vahid Ghods.

A special dataset called the "HODA" is utilized in the work. This consists of pictures made with Farsi digit. 20,000 pictures are used as test images and remaining serve as training data for each class. They employed Python and TensorFlow, as well as the Keras library in a Jupyter notebook, to implement and train the CNN. The most important information about a Farsi digit is the top portion of the picture, which was exploited by cropping the image horizontally and using the samples for assessing the CNN model training. They might then examine the mistakes, which shows the amount of samples that are incorrectly identified, using a confusion matrix.

HANDWRITTEN DIGIT RECOGNITION BASED ON EXTENSION ENGINEERING [8] 2019 – Huimin Xiao, Chen Liu.

The identification of the enhanced matrix of correlation built upon the dependent functional theory and classification using the prior findings comes after analysis of the key elements of the classification objects. Assuming that P_0 is a subset of P_e , the degree to which each given P_0 p_x belongs to P_0 may be determined using the following approaches. All noise points beyond the digits are deleted, all pictures are converted to the same size, and the resulting handwritten digit images are created from the binary colour image created from the grey scale image produced by scanning (which only contains white and black). Numbers Physique Calculating the length of the line and the distance from the initial junction point to the last intersection point using characters.

IMPLEMENTATION OF FEATURE EXTRACTION AND MLP NEURAL NETWORK CLASSIFIER FOR FARSI HANDWRITTEN DIGIT RECOGNITION [9] 2020

- Marzieh Moradi, Mohammad Ali Poormina, Farbod Razzazi.

Three techniques were utilized in the feature extraction process. These tactics are described in the section below. With this feature extraction method, fifteen characteristics are extracted. Every extracted characteristic can be implemented by the FPGA using just basic addition and subtraction operations because they are all integers. These processes were carried out simultaneously on an FPGA. Comparing the density of black pixels on the top and bottom half, as well as the both side two-fourth of the picture, the binary image's geographic distribution of pixel values is shown using a statistical approach. For the purpose of grouping digits, this function is helpful. Some numbers' center vertical ray has two intersections. Some numerals only have one intersection with the middle ray, whereas others have more than two. Elastic Meshing Directional Feature Extraction is used to determine the spread of black pixels across each of the 4 directed segments that make up the full binary character picture. One hidden layer MLP neural network was employed in this research.

HANDWRITTEN DIGITS RECOGNITION USING MULTIPLE INSTANCE LEARNING [18] 2009 - Yuan hanning, Wang Peng.

Unlike MIL, occurrences of a bag in HeterMIL likely originate from several feature areas. A feature vector describes each feature space. Each feature vector was referred to as a modality. Similar to MIL, HeterMIL labels a bag favorably if it has at least one positive occurrence and a negative bag otherwise. Learning a concept from the training set is the task. The culmination of every decision made for each occurrence of each modality is the ultimate decision. The training dataset's instance label information is the key to the solution. HeterMIL operates under the same premise as MIL, i.e., a bag is labelled positively if it includes at least one positive instance, and a bag is labelled negatively if it does not. Therefore, it is evident from the label information of the examples in negative bags that all of the instances are negative. However, cases in positive bags have confusing label information.

A Proposed Framework for Recognition of Handwritten Cursive English Characters using DAG-CNN [5] 2019 - Bhagyasree P V, Ajay James, Chandran Saravanan.

Machines are educated in deep learning utilising enormous amounts of data and have considerable adaptability. Deep learning has produced better outcomes thanks to the use of neural networks. Even when employing the CNN for letter identification, also exists a potential that some alphabets, such as 'e, v, and u' will be incorrectly identified to 'l' among other things. A novel design termed DAG, or a network that is separated into numerous routes, is used to the fundamental CNN model in order to avoid these issues and increase accuracy. In a DAG, the output layer may receive characteristics from each layer directly and classify them. Both the inputs and outputs might be many. Due to the many connections between each layer, it is possible to use skip connections to link each layer directly to the final classifier layer such as all levels of characteristics may help the network run better. The overall number of features will also be quite huge because we are using features from every tier. Overfitting will become a concern as a result. The best layers that improve performance must be chosen to prevent this.

RECOGNITION OF CURSIVE ENGLISH HANDWRITTEN CHARACTERS [13]

2016 - Pritam Dhande, Reena Kharat.

There are two forms of segmentation, internal and exterior. External segmentation is the division of various literary units, such as sentences and phrases. Internal segmentation, which is employed in the identification of cursive handwriting, is the separation of letters. Numerous techniques are employed for segmentation, including histogram approaches, pixel counting approaches, water flow approaches, and methods based on vertical and horizontal projection. For extraction of characteristics, the convex hull algorithm, zoning technique, longest run characteristics, crossings and distances, geometrical and topographical visualizations, etc. HMM, NN, SVM, and other techniques are utilized for classification and recognition. One line from a paragraph is separated using the horizontal projection method. This creates the image's initial horizontal histogram. A rising section's average height is taken as a point. Every line is then separated from the image after the height of each rising part is verified to see if it is same or more than a specific point.

SINHALA HANDWRITTEN CHARACTER RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK [7] 2019 - Janotheepan Mariyathas, Vasanthapriyan Shanmuganathan, Banujan Kuhaneswaran

The pictures of the Sinhala characters are first scanned, and then preprocessing is done. Preprocessing in the sense of doing the work by cropping or chunking the photos into single characters, eliminating the noisy data, and classifying them into the proper character set. For CNN to train the model or algorithm, a sizable quantity of datasets is required. A great quantity of data must be created, hence data augmentation is being used to increase the dataset. Character images (image datasets) are not eligible for data augmentation because of their data nature. Here, each character is trained and tested using about 250-character pictures. In all, 110,000-character pictures ($250 * 434$) were employed for the development.

There are constraints. To treat all photos as having the same size after preprocessing, the data must be downsized to the same size. For the model, each image has been downsized to 100 x 100. Making all of the photos the same size will help the model avoid receiving additional noisy input and treat them all equally. The RGB effect (color effect) of the photographs is then removed by converting the enlarged images to grayscale images. because the identification process might be thrown off by such impacts. In this particular model, the pictures are enlarged using 4 convolutional layers and 4 max-pooling layers. Then, the pictures are transferred to the hidden levels; in this case, the suggested model uses 2 hidden layers. Two-by-two maximization was applied to all levels. Each layer utilized a Rectified Linear Unit (ReLU). where just two convolutional layers and two max-pooling layers are taken into account.

AN EFFICIENT BILINGUAL OPTICAL CHARACTER RECOGNITION
(ENGLISH-ORIYA) SYSTEM FOR PRINTED DOCUMENTS [14] 2020 -
Sanghamitra Mohanty, Himadri Nandini Dasbebartta, Tarun Kumar Behera

An OCR receives input from a camera or a scanner. The picture must then be binarized. The approach, which is the collection of dots making up the picture, is used to enhance the image. Through the use of both profiles of projection, the lines, words, and character are segmented. The extraction of characteristics comes after the segmentation of words and characters. A sentence has zones which are upper, medium, and lower. various machine-printed documents require various OCR techniques. Roman script will be incorrectly recognized by an OCR created for Oriya script, and vice versa. Chain code has produced feature extraction for each character. We have created integers for each aspect of each character using this chain code approach. That is, we have used a "Yes" or "No" rule. 0 or 1 indicates whether a character trait is present or absent. We can utilize an English OCR to assist with character recognition in English. English OCR can be utilized for recognition by enabling the route of a scanner. Dumping the scripts at the right location in an editor is important after they have been recognized by both OCR systems. Therefore, when delivering the characters to various OCR for recognition, we need to maintain track of their positions.

RECOGNITION OF ENGLISH MULTI-ORIENTED CHARACTERS [11] 2019 -
U. Pal, F. Kimura, K. Roy and T. Pal

For the tests, the majority of the 18232 characters come from books, periodicals, publications, advertising, and electronic copies. Digitization was done with a scanner with a flatbed. Greyscale in digital photographs is 300 dpi. We turned these pictures into two-colored photos by worldwide binarizing them utilizing a histogram method. Consider the 3×3 window that surrounds each item point in the image. The subject point is assumed to be a border point if any one of the four adjacent points is the backdrop point. Characters of varied shapes and orientations were used in the experiment. To validate the findings, a five-fold validation approach is performed. For the sake of developing and testing the system, the split dataset was divided into four parts. The proposed method generated accuracy of 98.34% on average. The recommended method's primary drawback will be the lack in ability to classify between letters like "b, q", "p, d", etc. The angular data of the character boundary points serves as the main basis for the features used.

OPTICAL CHARACTER RECOGNITION FOR ENGLISH HANDWRITTEN
TEXT USING RECURRENT NEURAL NETWORK [12] 2014 - R. Parthiban, R.
Ezhilarasi, D. Saravanan.

For the purpose of recognizing English handwritten text, this suggested framework uses a recurrent neural network. We are currently creating an OCR for English information that has been handwritten. The example set is therefore limitless. Similarly, different cases have different characteristics. It is hard for them to follow a comparable example because the handwriting exams are collected from different persons. In our technique, we have used a bottom-up approach, where we first focus on a specific case before moving on to the overall structure. Collecting transcribed data from multiple authors. Each person writes in a distinct way. All of those various papers have been gathered and scanned to create those databases. From these, physically crop each character and label each character as a unique picture. After that, optical character recognition is done with it.

Image resized to 30*30 pixels and converted to grayscale. Force pixels to change. Include a 2-pixel cushion on all sides. It is a 128 x 32 grayscale image. The characters appear exactly where they are expected to in the photo. The handwritten English documents in this proposed framework are scanned and optically recognized. With 90% accuracy, RNN algorithm output is obtained as written text. In order to further enhance the completed show, we must investigate the problem for better layout by developing an entirely new engineering for English content.

RECOGNITION OF OFF-LINE HAND PRINTED ENGLISH CHARACTERS, NUMERALS AND SPECIAL SYMBOLS [10] 2009 - Nisha Sharma, Bhupendra Kumar, Vandita Singh.

The training and testing data was made by obtaining texts from forty writers, every one produced a document that had examples of all twenty-six alphabets, ten numeric and eight unique characteristics. Upper case letters, lower case characters, and numbers were used almost 3517, 2340, and 1804 times, respectively. For the sake of training and testing, the gathered data has been split into a 60:40 ratio. By calculating the number of black pixels within a single area, the feature extraction method was first applied to every subdivided image. Zones were created inside the character picture, and features for the image were created using the data from each zone. Zones were created by dividing the character picture five times. Each of the five divisions has a mixed 1, 2, 3, 4 and 5 dimensions. Each character picture was thinned and normalized before receiving its geometric characteristics (distance, angle). The skeleton's center was estimated. The picture was then split into 3x3 zones according to the computed results. With MLP and Support Vector Machine, geometric, statistical, topological, and directional features were employed. SVM and neural networks have both been chosen for usage because of their great flexibility and rapidity in response to noise.

F-RATIO BASED WEIGHTED FEATURE EXTRACTION FOR SIMILAR SHAPE CHARACTER RECOGNITION [16] 2009 - T. Wakabayashi, U. Pal, F. Kimura and Y. Miyake

Producing 400-dimensional gradient features to compare the improvement in outcomes. After using the F-ratio to improve the handling of similarly shaped patterns. On the input picture, a 2X2 mean filtering is used. The gray picture is minimalized in step two such that the average scale for grey is 0 and the highest value is 1. Segmenting the normalized picture into 99 blocks. This block size, which compromises accuracy and complexity, is chosen based on the experiment. The picture is transformed into two tones using the Ostu. By doing this, the unneeded backdrop in the picture will be ignored. The image is then given a Roberts filter to create a gradient image. When Kimura evaluated the model for written by hand number recognition, the quadratic classifier yielded the best outcomes. They used similar-shaped letters and numbers from many scripts, which also includes many famous languages for the trial of the method. It took into consideration the 22546, 14650, 2220, 5638, 4820, 2690, and 20000 examples of the language characters, respectively, to obtain the pair of numerals with a similar form. In this research, they suggest a feature augmentation method to boost the identification of two characters with comparable shapes.

NEW PROBE INTO ENGLISH NETWORK LANGUAGE FEATURES VALUE IN THE CONTEXT OF ONLINE EDUCATION BASED ON CHARACTER RECOGNITION [6] 2021 - Bo Wang

The classifier is trained to distinguish the word area from the surrounding region. Words area feature extraction involves removing the feature vector from the character candidate region that can most accurately represent character features. This allows characters with the smallest differences between them to be grouped into one category while characters with the largest differences can be separated into two categories. The HOG feature is made up of calculations and counts of the local area of the image's gradient direction histogram.

The primary concept behind HOG feature extraction is that local targets' look and form may be described by the gradients or corner detection. In the realm of image recognition, HOG+SVM is often utilized, and the results are excellent. The CASIA-HWDB data collection, which totals 3755 frequent terms, was employed in this study. The development in English network language, exhibits a growing number of distinctive language features that set it apart from traditional written and oral languages.

These qualities include clear phrase construction, expressive emotion symbolism, epoch-defining new word use, and a greater quantity of new terms. Machine Learning has seen significant application recently in areas including voice recognition, picture segmentation, target identification, and classification of images, among others, and has shown impressive results. Convolutional neural networks, which have many good qualities that cannot be attained by conventional detection and classification methods, are a significant network structure in deep learning techniques.

A ROBUST APPROACH FOR OFFLINE ENGLISH CHARACTER RECOGNITION [15] 2017 - Suman Avdhesh Yadav, Smita Sharma, Shipra Ravi Kumar

The study provides an ANN-based approach to enhance the accuracy of offline handwritten alphabets. Correct feature selection improves recognition precision and requires less computing work. The steps for doing the suggested task are as follows, The feed forward neural network technique is used to execute training on selected samples in the training database. The trained picture is saved in the database as a template and will be used to match patterns with test photos. A test picture is created using a scanned image. A decision is reached on the chosen character. The cropping of a certain character occurs. 6. Noise in the clipped portion is removed by pre-processing. The trained database's recognized character is displayed by the recognition. Pre-processing aims to improve the hand-printed data's quality. That more exactly implies that the character is changed so that they are more like the class to which they belong.

A scanner is used to assist in the capture of images. The photograph was scanned at 300 dpi since this resolution often produces noise-free, high-quality images. The picture is sent to a module that changes the RGB image to a greyscale format once it has been scanned and saved. Additionally, it transforms the image to a binary format and deletes any objects with fewer than 30 pixels. The network is ready to be trained once it has been initialized and the training input space has been set up. The algorithm TRAINING_ROUTINE uses the training routine. In this study, we present an offline English character recognition system based on artificial neural networks to recognize English alphanumeric characters. When the English alphabets are taught and tested separately, our suggested system produces exceptional results, but when they are processed together, it produces good results. The suggested system offers a shorter training period and lower training costs when compared to previous handwriting recognition methods like HMM and fuzzy membership function-based approach. Our technology is also less expensive to compute with and simpler to use.

RECOGNIZING ENGLISH CURSIVE USING GENERATIVE ADVERSARIAL NETWORKS [17] 2015 - Xinrui Yu and Jafar Saniie.

This this paper GAN transfers the original script's style to the intermediate font, and the typical OCR technique is used on the intermediate font to provide the recognition result. The third line's machine font is only a representation of the encoded text. In this procedure, a choice of the intermediary typeface is crucial. To create an intermediary script that is identical to the primary script, the GAN must be trained using a dataset that contains quite a few instances of calligraphy letters and their corresponding computer-generated text, in image format. Following the creation of the scripts, the final recognition phase is carried out by a traditional OCR algorithm that is easily accessible. According to the they were capable of recognizing the calligraphy letters that were previously unrecognizably.

HANDWRITTEN ENGLISH WORD RECOGNITION BASED ON CONVOLUTIONAL NEURAL NETWORKS [1] 2012 - Aiquan Yuan, Gang Bai, Po Yang, Yanni Guo, Xinting Zhao.

Each of the sampling point values in the initial data points were not on an identical scale, thus they were scaled to achieve a height in the 1-199 range with a width in the same ratio. The strokes are also rounded off at the same time. With this in mind, the neighbors of the present sample point in the 2-D picture are impacted. Because sample strokes are angled, it is required to align all sample slant angles to the same angle. Over-segmentation makes it such that one segment generally only makes up a portion of a character. For later character identification, it is crucial to combine numerous neighboring segments into a single character. Review demonstrates that excessive classification is manageable. A few characters are divided into more than three parts, which is a difficulty that cannot be overlooked while creating character hypotheses. One character like this may be recognized as two characters using the grouping methods of G1, G2, and G3 above.

Comparisons on KNN, SVM, BP and the CNN for Handwritten Digit Recognition[21] 2020 - Wenfei Liu, Jingcheng Wei, Qingmin Meng

One of the very straightforward yet useful machine learning algorithms is KNN (K-Nearest Neighbor). Relying upon the geometrical measuring idea, classification and regression are achieved by computing the gap between distinct characteristic values in the model. SVM, a component of the supervised learning algorithm, plays a significant role in engineering practice. The two fundamental theories are the foundation of statistical learning theory and have clear benefits in the solution of high-dimensional feature space classification issues and non-linear classification problems. The SVM algorithm makes use of the kernel function to progressively make the transition from a low-dimensional feature space to a high-dimensional feature space easier.

The intricate neural network of the brain served as the inspiration for the information processing model known as the ANN (Artificial Neural Network). Simply described, this model consists of several layers of neurons. The critical parameters between neurons must be regularly adjusted in order to obtain the best model output outcomes. To address the issue of how to modify the weight parameters, the ANN model must be paired with the appropriate methods.

Bangla Handwritten Digit Recognition Using an Improved Deep Convolutional Neural Network Architecture [7] - Chandrika Saha, Rahat Hossain Faisal and Md. Mostafijur Rahman

The LeNet-5 design served as inspiration for the D-CNN architecture employed in this study. There are 7 levels in this architecture. The network receives the input photos in batches of 50 at a time. All of the photographs have been resized to be 32×32 pixels, and they are all RGB or true colour images. Adam optimizer uses prior parameter values to provide a smooth learning process. This output is combined with a bias value, and the reLU activation function is then used.

The following layer receives this finished product after that. The bias value is likewise initialized at random. After the completely linked layer, the categories are predicted using a SoftMax output unit. A probability value is assigned to each of the ten output classifications that adds up to 1. The selected prediction belongs to the class with the highest likelihood. In this study, for isolated Bangla handwritten digits, a 7 layered model is recommended.

Graph-based Handwritten Digit String Recognition [8] 1995 - Alexander Filatov, Alexander Gitis, Igor Kil

The creation of a graph of the input picture is the first step in the recognition process. The next step is to separate the numeric string into individual symbols. Without applying the recognition procedure, it is obviously not always feasible to locate the proper segmentation variation. Each subgraph of the input graph is compared to prototypes once the subgraphs that are purportedly corresponding solitary characters were then selected.

Those area in the graph of heavy pen traces (such as those caused by forward-backward pen motions) and shattering and body-crossing pen-marking locations. The graph's edges represent to those of the pen. Such a graph, however, isn't necessarily appropriate for later identification. With this method, it is possible to represent handwritten symbols completely using a relatively small number of prototypes for instance, our system utilizes less than two prototypes per sign.

2.1 CLASSIFIERS

There are many classifiers that are in use in many areas of image recognition. The Classification-Report and Confusion-Matrix will give the accuracy and other parameters that determine the consistency of the classifier. Here some of the famous classifiers are taken and their performances are discussed.

2.1.1 Support Vector Machines (SVM)

SVM is one of the most popular Supervised learning algorithms, which is used for Classification as well as Regression problems. Accuracy for the SVM Classifier is 0.9688. The Classification Report that it produces looks like this

Table 2.1: Support Vector Machines Report

Classification report for classifier SVC(gamma=0.001):				
	precision	recall	f1-score	support
0	1.00	0.99	0.99	88
1	0.99	0.97	0.98	91
2	0.99	0.99	0.99	86
3	0.98	0.87	0.92	91
4	0.99	0.96	0.97	92
5	0.95	0.97	0.96	91
6	0.99	0.99	0.99	91
7	0.96	0.99	0.97	89
8	0.94	1.00	0.97	88
9	0.93	0.98	0.95	92
accuracy			0.97	899
macro avg	0.97	0.97	0.97	899
weighted avg	0.97	0.97	0.97	899

The Confusion Matrix is as follows:

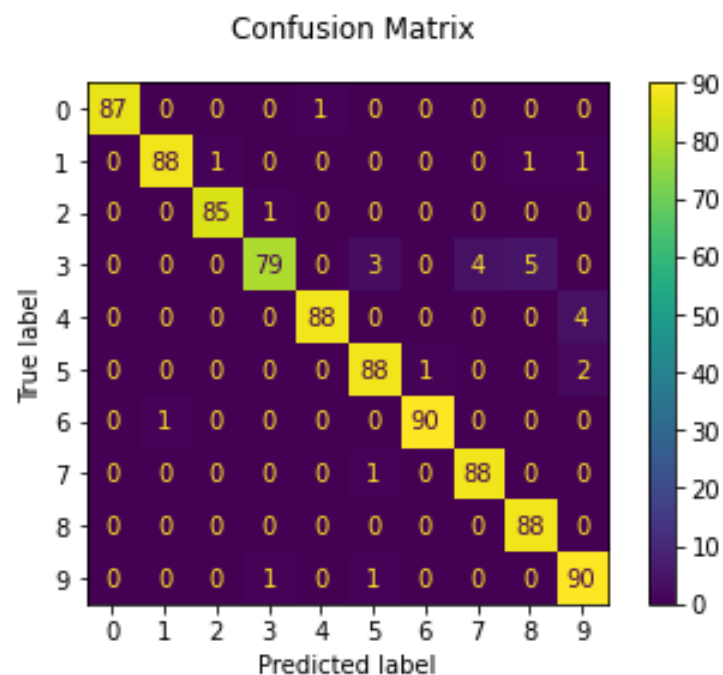


Fig:2.2: Support Vector Machine Confusion Matrix

2.1.2 Gaussian Naïve Bayes

They are a group of Bayesian-based classification algorithms. It is a family of algorithms rather than a single technique. This Classifier's accuracy was determined to be 0.80. When it comes to identifying handwritten digits, the Gaussian Naive Bayes Classifier performs worse than the SVM Classifier. The Confusion Matrix and Naive Bayes Classifier's Classification Reports are as follows:

Table 2.2: Classification Report for Gaussian Naive Bayes Classifier

Classification report for classifier GaussianNB():					
	precision	recall	f1-score	support	
0	0.98	0.95	0.97	88	
1	0.81	0.74	0.77	91	
2	0.87	0.84	0.85	86	
3	0.88	0.79	0.83	91	
4	1.00	0.73	0.84	92	
5	0.70	0.81	0.76	91	
6	0.96	0.99	0.97	91	
7	0.65	0.81	0.72	89	
8	0.61	0.76	0.68	88	
9	0.77	0.66	0.71	92	
accuracy			0.81	899	
macro avg	0.82	0.81	0.81	899	
weighted avg	0.82	0.81	0.81	899	

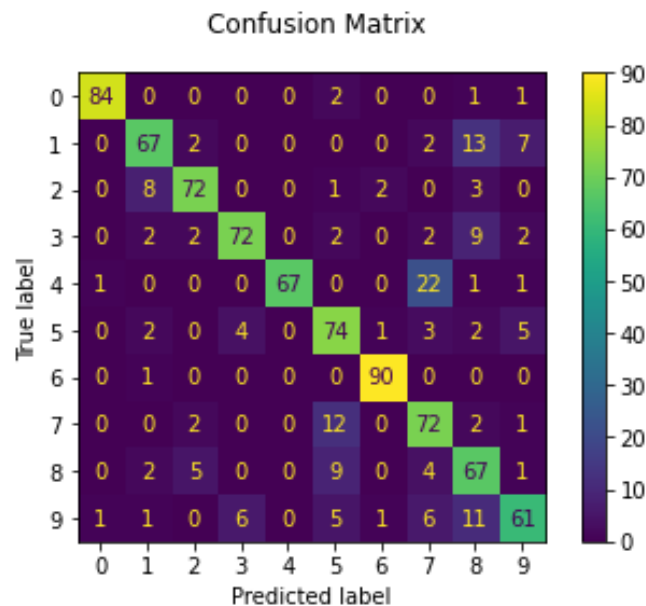


Fig:2.3: Confusion Matrix for Gaussian Naïve Bayes Classifier

2.1.3 Decision Tree

Decision tree is a Supervised learning technique that can be used for both classification and regression problems. The accuracy score for the Decision Tree Classifier was much worse, 0.7352. Classification Report and Confusion Matrix are as follows.

Table:2.3: Classification Report for Decision Tree

Classification report for classifier DecisionTreeClassifier():				
	precision	recall	f1-score	support
0	0.91	0.91	0.91	88
1	0.74	0.59	0.66	91
2	0.93	0.78	0.85	86
3	0.65	0.75	0.69	91
4	0.72	0.80	0.76	92
5	0.59	0.73	0.65	91
6	0.86	0.82	0.84	91
7	0.86	0.63	0.73	89
8	0.62	0.66	0.64	88
9	0.63	0.68	0.66	92
accuracy			0.74	899
macro avg	0.75	0.74	0.74	899
weighted avg	0.75	0.74	0.74	899

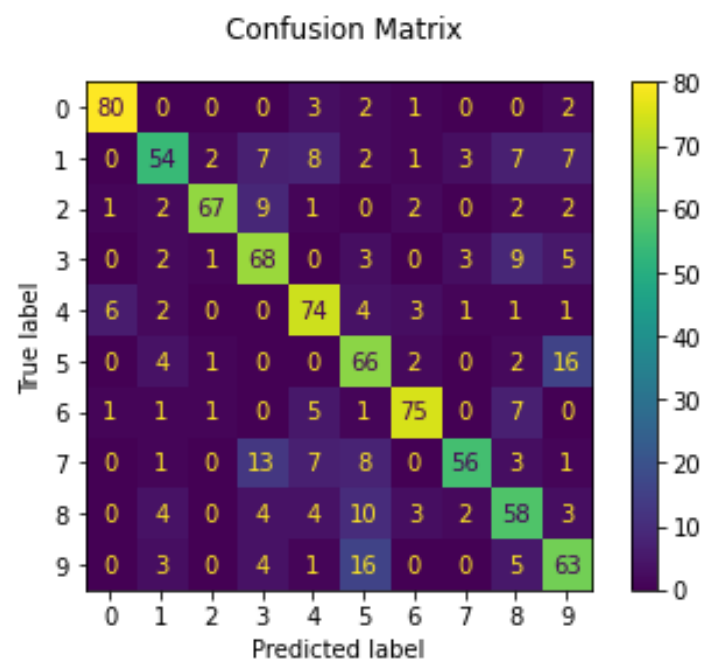


Fig:2.4: Confusion matrix for Decision Tree

2.1.4 Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It is based on the concept of ensemble learning. The Random Forest Classifier's accuracy score is 0.7530, which is almost as high as the Decision Tree Classifier's.

The Classification Report and Confusion Matrix are as follows:

Table 2.4: Classification Report for Random Forest

Classification report for classifier RandomForestClassifier(max_depth=2, random_state=0):

	precision	recall	f1-score	support
0	0.87	0.99	0.93	88
1	0.90	0.42	0.57	91
2	0.66	0.80	0.72	86
3	0.55	0.82	0.66	91
4	0.91	0.82	0.86	92
5	0.74	0.62	0.67	91
6	0.87	0.99	0.93	91
7	0.70	1.00	0.82	89
8	0.92	0.26	0.41	88
9	0.74	0.82	0.77	92
accuracy			0.75	899
macro avg	0.79	0.75	0.73	899
weighted avg	0.79	0.75	0.73	899

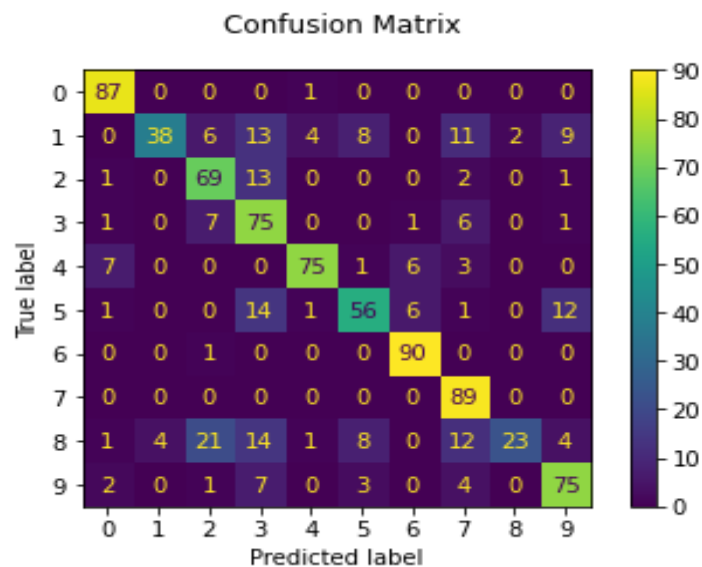


Fig:2.5: Confusion matrix for Random Forest

2.1.5 K Nearest Neighbors (KNN)

This is one of the simplest Machine Learning algorithms based on Supervised Learning Technique. It is also called lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. The accuracy is found to be 0.9555. For a KNN-based Classifier, the Classification Report and Confusion Matrix is as follows.

Table 2.5: Classification Report for K Nearest Neighbor

Classification report for classifier KNeighborsClassifier(metric='euclidean'):				
	precision	recall	f1-score	support
0	0.99	1.00	0.99	88
1	0.95	0.98	0.96	91
2	0.98	0.93	0.95	86
3	0.89	0.90	0.90	91
4	1.00	0.95	0.97	92
5	0.96	0.98	0.97	91
6	0.99	1.00	0.99	91
7	0.95	1.00	0.97	89
8	0.95	0.90	0.92	88
9	0.91	0.92	0.92	92
accuracy			0.96	899
macro avg	0.96	0.96	0.96	899
weighted avg	0.96	0.96	0.96	899

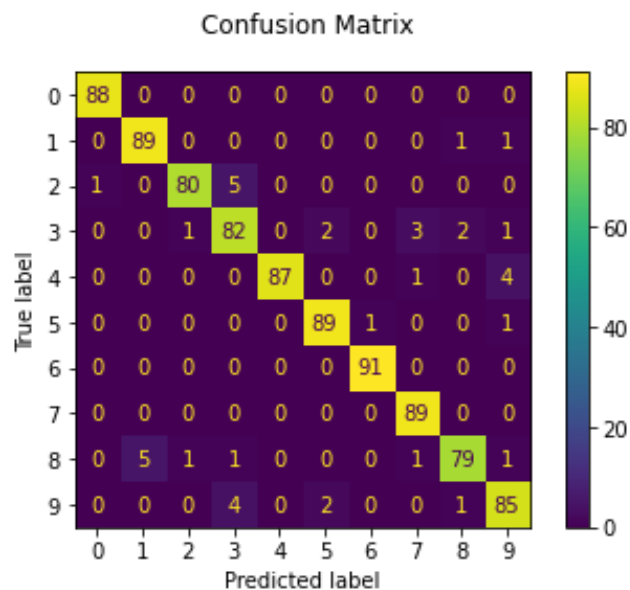


Fig:2.6: Confusion Matrix for K Nearest Neighbor

2.1.6 Stochastic Gradient Descent

It is a generic optimization algorithm capable of finding optimal solutions to a wide range of problems. The accuracy rating of the stochastic gradient descent algorithm is 0.8932.

The Classification Report and the Confusion Matrix are provided below:

Table 2.6: Classification Report for Stochastic Gradient Descent

Classification report for classifier SGDClassifier(max_iter=5):				
	precision	recall	f1-score	support
0	0.91	0.98	0.95	88
1	0.93	0.70	0.80	91
2	0.92	0.97	0.94	86
3	0.99	0.82	0.90	91
4	0.86	0.90	0.88	92
5	0.75	0.95	0.84	91
6	0.96	0.99	0.97	91
7	0.98	0.97	0.97	89
8	0.91	0.73	0.81	88
9	0.80	0.93	0.86	92
accuracy			0.89	899
macro avg	0.90	0.89	0.89	899
weighted avg	0.90	0.89	0.89	899

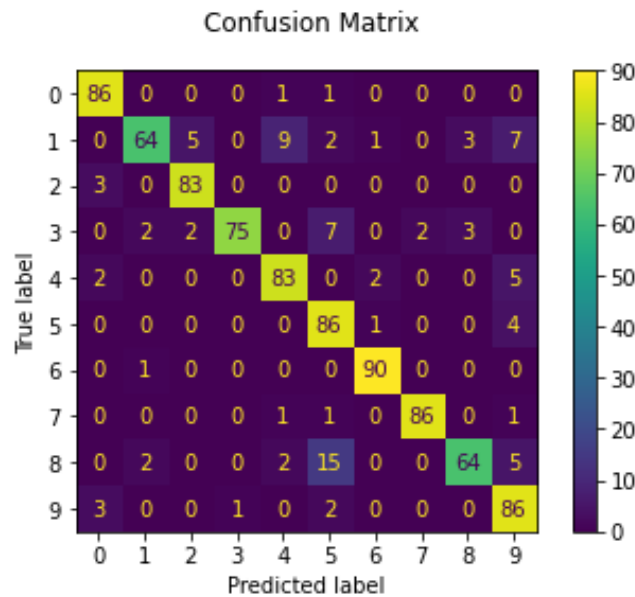


Fig:2.7: Confusion Matrix for Stochastic Gradient Descent

2.1.7 CONSLUSION ON CLASSIFIERS

Above discussed are some of the fundamental classification techniques for handwritten digit recognition to get started. The SVM classifier was the most accurate in terms of accuracy score, whereas Decision Trees were the least accurate.

Table 2.7: List of Classifiers Accuracy

Classifiers	Accuracy Score
SVM	0.9688
KNN	0.9555
Stochastic Gradient Descent	0.8932
Naïve Bayes	0.8075
Random Forest	0.7532
Decision Trees	0.7352

Even comparing the F1- scores, the SVM stands at the top.

Table 2.8: List of Classifiers F1-score

Classifiers	F1-Score
SVM	0.92 - 0.99
KNN	0.90 - 0.99
Stochastic Gradient Descent	0.81 - 0.97
Naïve Bayes	0.68 - 0.97
Random Forest	0.41 - 0.93
Decision Trees	0.64 - 0.91

Therefore, drawing the conclusion that the SVM classifier performed the best comparing the accuracy score and F1-score to other classifiers. For this reason, it is frequently employed to solve picture recognition issues as well.

CHAPTER 3

METHODOLOGY

3.1 CONVOLUTION NEURAL NETWORK

CNN uses deep learning to automatically classify the input. CNN has developed a strong grasp on categorizing pictures for computer vision over the years, and it is now being utilized in healthcare sectors as well. CNN is therefore shown to be a reliable deep learning technique for automating prediction from start to finish. CNN essentially takes 'useful' data dynamically from the input, which makes it incredibly simple. When giving all types of inputs, convolutional neural networks perform better than other common neural networks.

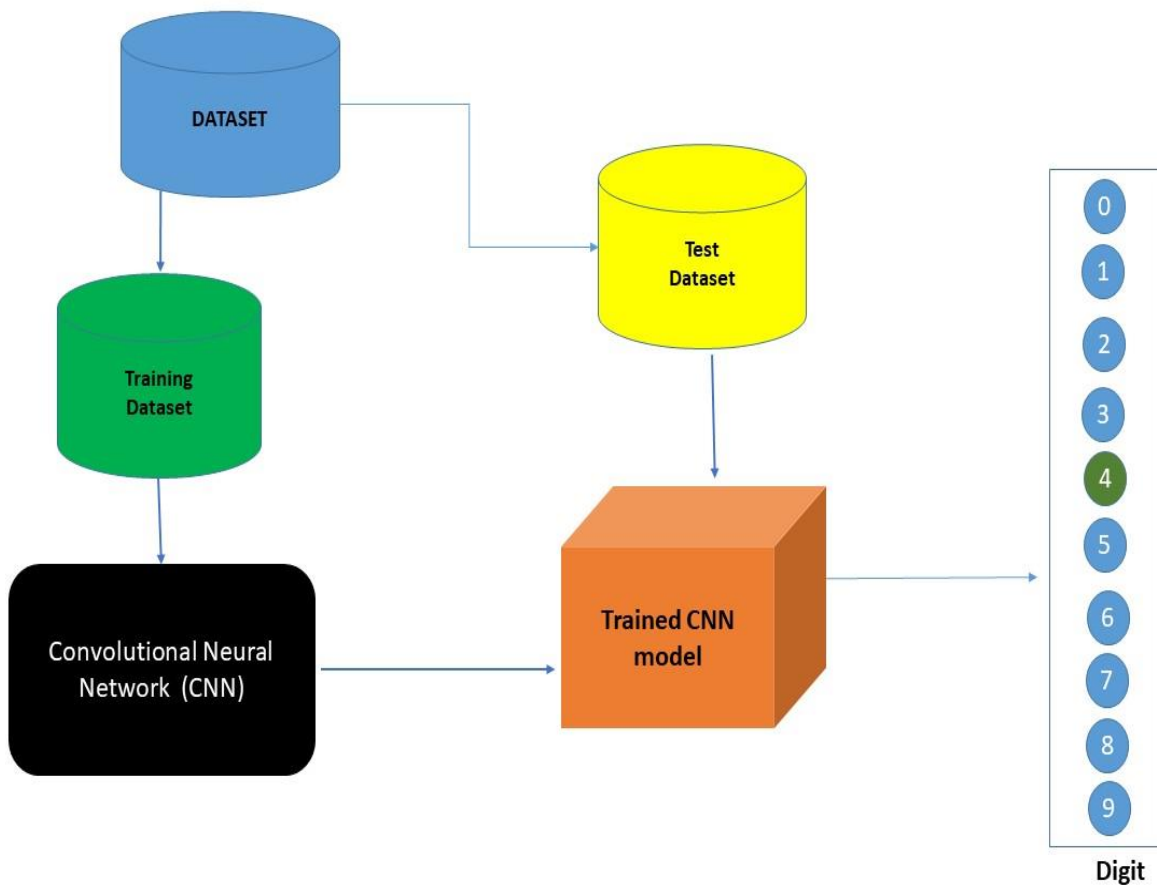


Fig:3.1: Block Diagram of Convolution Neural Network

A CNN model consists of three primary layers,

- Convolutional Layer
- Pooling layer(s)
- Fully connected layer.

The convolutional layer will be the convolutional networks first layer. The fully-connected layer is the last layer, even though convolutional layers, further convolutional layers, or pooling layers, might come after it. Each layer of the CNN makes it more complex, allowing it to recognize bigger portions of the image. Early layers highlight fundamental components like colour and boundaries. The bigger features or forms of the item are first seen when the visual data moves through the CNN layers, and eventually the desired object is recognized.

3.1.1 Convolutional Layer

The central component of a CNN is the convolutional layer, which is also where the majority of processing takes place. It needs input data, a filter, and a feature map, among other things. Assume that the input will be a colour image that is composed of a 3D pixel matrix. As a result, the input will have three dimensions—height, width, and depth—that are analogous to RGB in a picture. Additionally, we have a feature detector, sometimes referred to as a kernel or filter, which will traverse through the image's receptive fields and determine if the feature is there. Convolution describes this process.

A two-dimensional (2-D) array of weights serving as the feature detector represents a portion of the picture. The filter size, which also controls the size of the receptive field, is normally a 3x3 matrix, however they can vary in size. Following the application of the filter to a portion of the picture, the dot product between the input pixels and the filter is determined. The output array is then supplied with this dot product. Once the kernel has swept through the whole picture, the filter moves by a stride and repeats the operation. A feature map, activation map, or convolved feature is the ultimate result of the series of dot products from the input and the filter.

Following each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) correction to the feature map, introducing nonlinearity to the model. As was previously noted, the first convolution layer may be followed by another convolution layer. Because the subsequent layers will be able to see the pixels in the previous layers' receptive fields when this happens, the CNN's structure may then become hierarchical. Let's use the case of trying to detect whether a bicycle is there in an image as an example. The bicycle may be viewed as a collection of components. It is made of a frame, pedals, wheels, and handlebars. The bicycle's individual parts, each of which represents a lower-level pattern in the neural network and the bicycle as a whole, a higher-level pattern, produce a feature hierarchy inside the CNN.

3.1.2 Pooling Layer

Down sampling, also known as pooling layers, reduces the number of parameters in the input and does dimensionality reduction. The pooling operation sweeps a filter across the whole input similarly to the convolutional layer, with the exception that this filter lacks weights. Instead, the kernel populates the output array by applying an aggregation function to the values in the receptive field. There are principally two forms of pooling:

3.1.2.1 Max pooling

The filter chooses the input pixel with the highest value to transmit to the output array as it advances across the input. As a side note, this method is applied more frequently than typical pooling.

3.1.2.2 Average pooling

The average value inside the receptive field is determined as the filter advances over the input, and it is then sent to the output array.

The pooling layer loses a lot of information, but it also offers the CNN a number of advantages. They lessen complexity, increase effectiveness, and lower the danger of overfitting.

3.1.3 Fully-Connected Layer

The full-connected layer is what its name suggests it to be. As was already noted, partly linked layers do not have a direct connection between the input image's pixel values and the output layer. A node in the layer above it is directly connected to every node in the output layer of the fully-connected layer, in contrast. Based on the characteristics that were retrieved from the preceding layers and their various filters, this layer conducts the classification operation. FC layers often utilise a SoftMax activation function to categorise inputs properly, yielding a probability ranging from 0 to 1. Convolutional and pooling layers typically use ReLu functions.

3.2 DATASET USED

The dataset that is being used here is the MNIST digits classification dataset. Keras is a deep learning API written in Python and MNIST is a dataset provided by this API. This dataset consists of 60,000 training images and 10,000 testing images. It is a decent dataset for individuals who need to have a go at pattern recognition. The name of this dataset is A-Z Handwritten Alphabets in.csv file. This dataset is basically a csv file which contains 372450 images of alphabets of 28×28 size.

3.3 CHARACTER RECOGNITION MODEL.

This is a deep learning project, or machine learning project, in which we will use TensorFlow and Keras to build a Convolutional neural network (CNN) model that can recognize handwritten letters, or the English alphabets from A to Z. A significant portion of the dataset on which we will train our model includes pictures of the English alphabet.

3.3.1 Importing libraries.

When necessary, we shall import the libraries that we have installed on our system. To begin with, we are merely importing NumPy, pandas, and matplotlib in order to preprocess and visualize our data.

```
import NumPy as np
import pandas as pd
import matplotlib.pyplot as plt
```

3.3.2 Read Dataset.

Now, we'll use the pandas `read_csv()` function to read our dataset from its csv file. As the csv file is quite huge and it will be easier to read it as float, we will read the data as data type float32.

```
data = pd.read_csv('A_Z Handwritten Data.csv').astype('float32') data.head(10)
```

```
In [3]: data = pd.read_csv('A_Z Handwritten Data.csv').astype('float32')
data.head(10)
```

```
Out[3]:
```

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.639	0.640	0.641	0.642	0.643	0.644	0.645	0.646	0.647	0.648
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

10 rows × 785 columns

Fig:3.2: Dataset used in Character recognition

3.3.3 Splitting of Data.

The data was divided into X and Y. a character picture will be present in X, and a label for that image will be present in Y.

```
X = data.drop('0',axis = 1)
y = data['0']
```

3.3.4 Training data and Testing data.

To obtain training (x_train, y_train) and testing (x_test, y_test) data, let's run train_test_split on X and Y.

```
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

3.3.5 Reshaping the Training and Testing data.

NumPy is used to reshape our training and testing picture data so that we could display it as an image instead of the 784 columns of pixel data that they were originally included in the csv file. We thus change them to 2828 pixels.

```
x_train = np.reshape(x_train.values, (x_train.shape[0], 28,28))
x_test = np.reshape(x_test.values, (x_test.shape[0], 28,28))
print("Shape of Training data: ", x_train.shape)
print("Shape of Testing data: ", x_test.shape)
```

3.3.6 Shuffle the training data.

Shuffling the data after reshaping it so that we may see training on scrambled characters, so that they are random.

```
shuffle_data = shuffle(x_train)
```

3.3.7 Visualizing the training data.

Let's see the training data now and make sure we were able to correctly render them as images.

```
import cv2
fig, axes = plt.subplots(3,3, figsize = (10,10))
axes = axes.flatten()
for i in range(9):
    _, shu = cv2.threshold(shuffle_data[i], 30, 200,
                           cv2.THRESH_BINARY)
    axes[i].imshow(np.reshape(shuffle_data[i], (28,28)), c map="Greys")
plt.show()
```

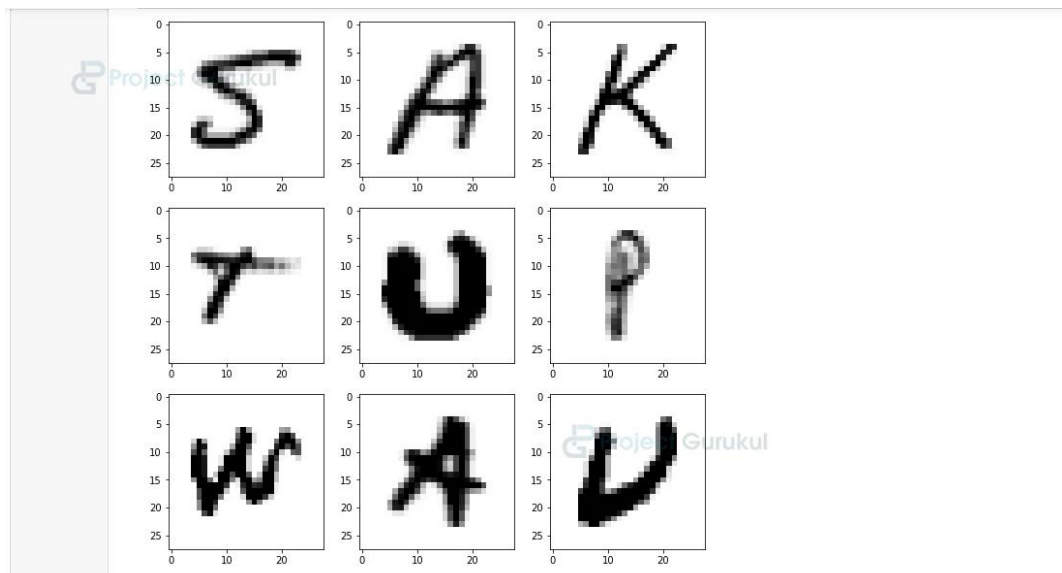


Fig:3.3: Image showing the test data

3.3.8 Again Reshaping data.

In order to use this data for training our model, we reshape our picture data, so that it is suitable for training the model.

```
x_train = x_train.reshape(x_train.shape[0],x_train.shape[1],
x_train.shape[2],1)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2],1)
print("New shape of training data: ", x_train.shape)
print("New shape of testing data: ", x_test.shape)
```

3.3.9 Converting to-categorical.

Now we must use the `to_categorical` function provided by Tensorflow, Keras, to transform our single float data into categorical values.

```
import tensorflow
from tensorflow.keras.utils import to_categorical
y_training = to_categorical(y_train, num_classes = 26, dtype='int')
y_testing = to_categorical(y_test, num_classes = 26, dtype='int')
```

3.3.10 Model Creation.

In order to categorise photos, we will be building a Convolutional Neural Networks (CNN) model, which is highly well-liked since it extracts the characteristics of the images using a number of hidden layers, or as we may say, a number of layers of filters.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

The model we have created consist of,

- A total of three convolutional layers (Conv2D) with 64, 64, and 64 layers each are employed, and then MaxPool layers are used to cut down on the amount of features extracted.
- After flattening the layers, we built two dense layers with a total of 128,265 layers each.
- Finally, we created our output layer, which is likewise a dense layer with a total of 128,265 layers and uses SoftMax as an activation function.

```
model = Sequential()
model.add(Conv2D(64 , (3, 3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPool2D(2, 2))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPool2D(2, 2))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Flatten())
model.add(Dense(128,activation ="relu"))
model.add(Dense(256,activation ="relu"))
model.add(Dense(26,activation ="softmax"))
```

3.3.11 Model Summary.

```
In [16]: model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 26, 26, 64)	640
max_pooling2d_3 (MaxPooling2)	(None, 13, 13, 64)	0
conv2d_4 (Conv2D)	(None, 11, 11, 64)	36928
max_pooling2d_4 (MaxPooling2)	(None, 5, 5, 64)	0
conv2d_5 (Conv2D)	(None, 3, 3, 64)	36928
max_pooling2d_5 (MaxPooling2)	(None, 1, 1, 64)	0
flatten_1 (Flatten)	(None, 64)	0
dense_3 (Dense)	(None, 128)	8320
dense_4 (Dense)	(None, 256)	33024
dense_5 (Dense)	(None, 26)	6682
Total params: 122,522		
Trainable params: 122,522		
Non-trainable params: 0		

Fig:3.4: Model Summary

3.3.12 Compile model.

For training on our image and checking accuracy that it is classifying images to its correct alphabet or not, we compile the model and check the accuracy.

```
model.compile(optimizer = Adam(learning_rate=0.001),  
loss='categorical_crossentropy', metrics=['accuracy'])  
history = model.fit(x_train, y_training, epochs=5,  
validation_data = (x_test,y_testing))
```

```
Epoch 1/5  
9312/9312 [=====] - 816s 66ms/step - loss: 0.2159 - accuracy: 0.9384 - val_loss: 0.1639 - val_accu  
acy: 0.9568  
Epoch 2/5  
9312/9312 [=====] - 470s 50ms/step - loss: 0.1125 - accuracy: 0.9686 - val_loss: 0.1166 - val_accu  
acy: 0.9693  
Epoch 3/5  
9312/9312 [=====] - 445s 48ms/step - loss: 0.0981 - accuracy: 0.9730 - val_loss: 0.0964 - val_accu  
acy: 0.9742  
Epoch 4/5  
9312/9312 [=====] - 430s 46ms/step - loss: 0.0907 - accuracy: 0.9752 - val_loss: 0.1089 - val_accu  
acy: 0.9718  
Epoch 5/5  
9312/9312 [=====] - 425s 46ms/step - loss: 0.0859 - accuracy: 0.9771 - val_loss: 0.1150 - val_accu  
acy: 0.9722
```

Fig:3.5: Image showing the EPOCH process

3.3.13 Save the model.

The model has now been developed, and we have also trained it. The model has to be saved for use in the following stage, therefore we are doing so by

```
model.save(r'handwritten_character_recog_model.h5')
```

3.3.14 Creating Words Dictionary.

This word dictionary's values are matched to its key. The model outputs the key from 0 to 25 and the relevant alphabet is outputted from the words dictionary.

```
words = {0:'A', 1:'B', 2:'C', 3:'D', 4:'E', 5:'F', 6:'G', 7:'H', 8:'I', 9:'J', 10:'K', 11:'L',  
12:'M', 13:'N', 14:'O', 15:'P', 16:'Q', 17:'R', 18:'S', 19:'T', 20:'U', 21:'V', 22:'W', 23:'X',  
24:'Y', 25:'Z'}
```

3.3.15 Prediction Process.

Let's test the model by guessing the alphabets that are shown in a picture, which are handwritten and converted into a .png format.

```
fig, axes = plt.subplots(3,3, figsize=(8,9))  
axes = axes.flatten()  
for i,ax in enumerate(axes):  
    image = np.reshape(x_test[i], (28,28))  
    ax.imshow(image, cmap="Greys")  
    pred = words[np.argmax(y_testing[i])]  
    ax.set_title("Prediction: "+pred)  
    ax.grid()
```

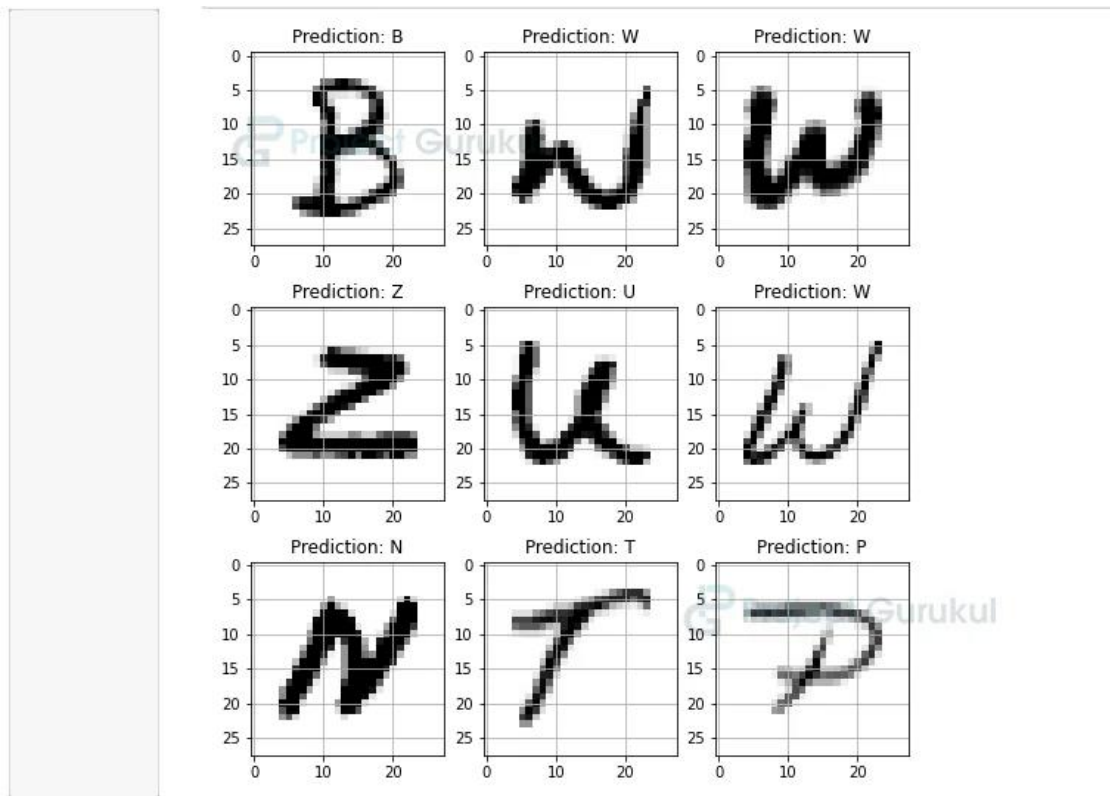


Fig:3.6: Image showing the Prediction

This demonstrates that our model is accurate and effective since it properly predicts the alphabets that are present in a picture.

3.3.16 To visualize our model on Customize image.

Let's see how well our model predicts whether the alphabet in a given image is present by using a bespoke image. By just setting the image's path in this code, we can also see the forecast.

```
image = cv2.imread('G.png')
image_copy = image.copy()
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image = cv2.resize(image, (400,440))
image_copy = cv2.GaussianBlur(image_copy, (7,7), 0)
gray_image = cv2.cvtColor(image_copy, cv2.COLOR_BGR2GRAY)
```

```

_, img_thresh = cv2.threshold(gray_image, 100, 255,
cv2.THRESH_BINARY_INV)
final_image = cv2.resize(img_thresh, (28,28))
final_image = np.reshape(final_image, (1,28,28,1))
prediction = words[np.argmax(model.predict(final_image))]
cv2.putText(image, "Prediction: " + prediction, (20,410),
cv2.FONT_HERSHEY_DUPLEX, 1.3, color = (0,255,0))
cv2.imshow('ProjectGurukul handwritten
character recognition ', image)

while (1):
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows()

```

3.4 DIGITS RECOGNITION MODEL.

After loading the necessary libraries, load the MNIST dataset as shown below,

```
(X_train, y_train), (X_test, y_test) = keras.datasets.mnist.load_data().
```

The four numbers `x_train`, `y_train`, `x_test`, and `y_test` are returned when the Keras API is invoked. Python is the language in use here. For authoring and running the Python code, we use "Pycharm." Pycharm is our choice since it makes all modules accessible with simplicity. This dataset gives four values in the same order as described above, as we previously explained. Additionally, the training and test datasets are represented by the names `x_train`, `y_train`, `x_test`, and `y_test`. To see how the dataset is split into training and test, look at the image above.

3.4.1 PROCESSING THE DATASET

To increase the quality of the data, processing, cleaning, and correction must be done. A dataset that is scaled, all-numerical, and devoid of null values is the optimal kind of dataset for CNN to learn from. Therefore, we will take certain actions in this section to make sure that our dataset is ideal for a CNN model to learn from. The 0th picture with values between 0-255 is obtained by writing `X_train[0]` (0 is black, while 255 is white). The result is a two-dimensional matrix (but we won't know what the handwritten digit `X_train[0]` stands for, of course). Write `y_train[0]` to find out, and you'll receive the output 5 as a result. In this dataset's training, the 0th picture corresponds to the number 5.

So, let's scale this training and test datasets as shown below:

```
X_train = X_train / 255
X_test = X_test / 255
```

After scaling, we should convert the 2-d matrix to a 1-d array by using this:

```
X_train = X_train.reshape(-1,28,28,1)   #training set
X_test = X_test.reshape(-1,28,28,1)    #test set
```

Now that the dataset is looking good, it's time that we create a Convolutional Neural Network.

3.4.2 CREATING AND TRAINING A CNN

Let's create a CNN model using the TensorFlow library. The model is created as follows:

```
convolutional_neural_network = models.Sequential([
    layers.Conv2D(filters=25, kernel_size=(3, 3), activation='relu',
    input_shape=(28,28,1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
```

```
layers.Flatten(),  
layers.Dense(64, activation='relu'),  
layers.Dense(10, activation='softmax')
```

An input layer, two hidden layers, and then an output layer make up the CNN model that was previously developed. In the simplest words possible, activation functions are in charge of deciding whether or not to proceed. Numerous neurons make up a deep neural network like CNN. Based on activation functions, neurons flare up and the network advances. We employ the activation function known as "relu" because it is so widely used.

After the model has been developed, it must be assembled and fitted. The model will scan the dataset and recognize the relationships as it is being fitted. The technique will be repeated as many times as necessary for learning. In this case, we have 10 epochs defined. The CNN model will pick up new skills and make mistakes along the way. The loss value for each is a representation of the penalty that the model incurs for each error (i.e., incorrect prediction). In other words, at the conclusion of the previous epoch, the model should provide as little loss and as much precision as feasible

3.4.3 MAKING PREDICTIONS

To evaluate the CNN model so created you can run,

```
convolutional_neural_network.evaluate(X_test, y_test).
```

It is time to use the test dataset to see how well the CNN model will perform.

```
y_predicted_by_model = convolutional_neural_network.predict(X_test).
```

The `convolutional_neural_network` model will be used in the aforementioned code to create predictions for the test dataset, which will then be stored in the `y_predicted_by_model` dataframe. It will be possible to generate a probability score for each of the 10 possible digits. The model's forecast is the class with the highest probability score. For instance, `y_predicted_by_model` will show you what the digit in the first row of the test set is.

The output will be something like this:

```
array([3.4887790e-09, 3.4696127e-06, 7.7428967e-07, 2.9782784e-08,  
6.3373392e-08, 6.1983449e-08, 7.4500317e-10, 9.9999511e-01, 4.2418694e-  
08, 3.8616824e-07], dtype=float32).
```

It is really difficult to identify the output class label with the highest probability score, let's write another code:

```
np.argmax(y_predicted[0])
```

And with this, it will output one of the ten digits as output (0 to 9).

CHAPTER 4

RESULTS

4.1 DIGITS RECOGNITION RESULTS

This is a Screenshot of the python code using TensorFlow and MNIST dataset in the PyCharm IDE.

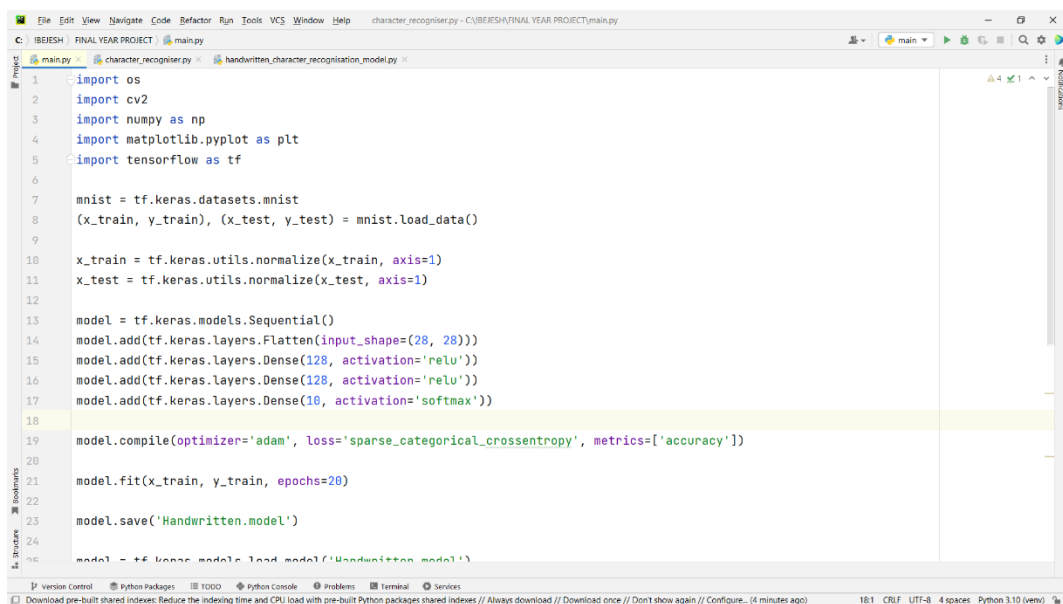


Fig:4.1: Screen Shot of PyCharm Window

Following image is when the Model is training.



Fig:4.2: Model Training

The following images show the predictions from the model created for the Hand written Digits in image format and fed into the model.

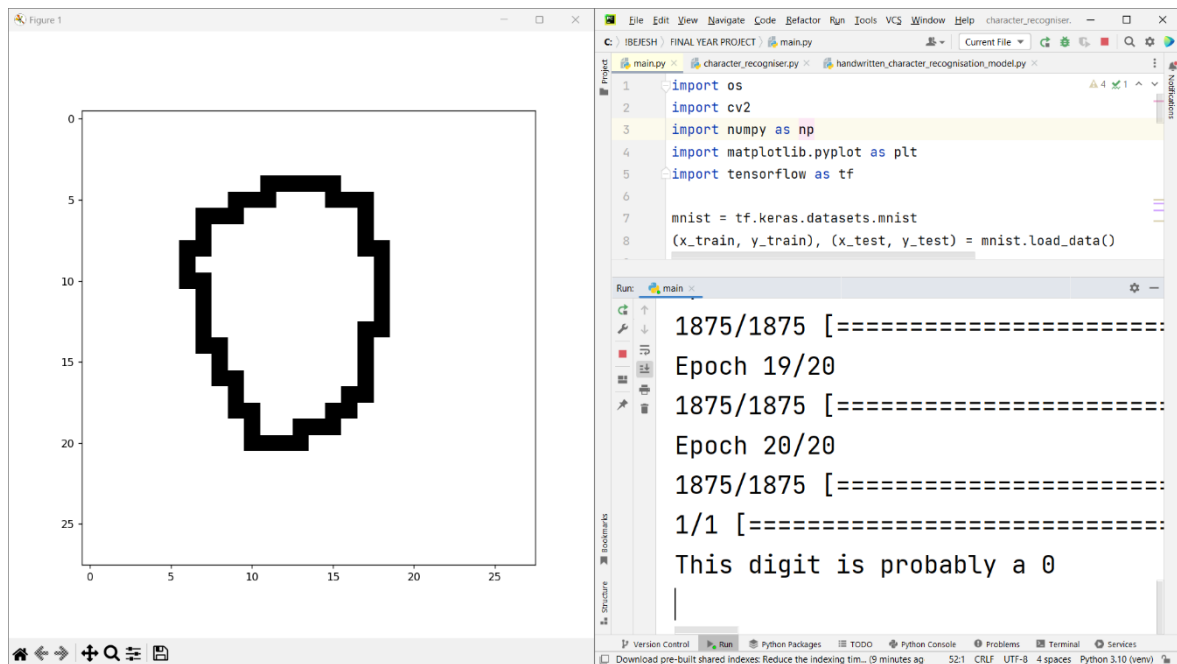


Fig:4.3: Model Predicting Digit 0

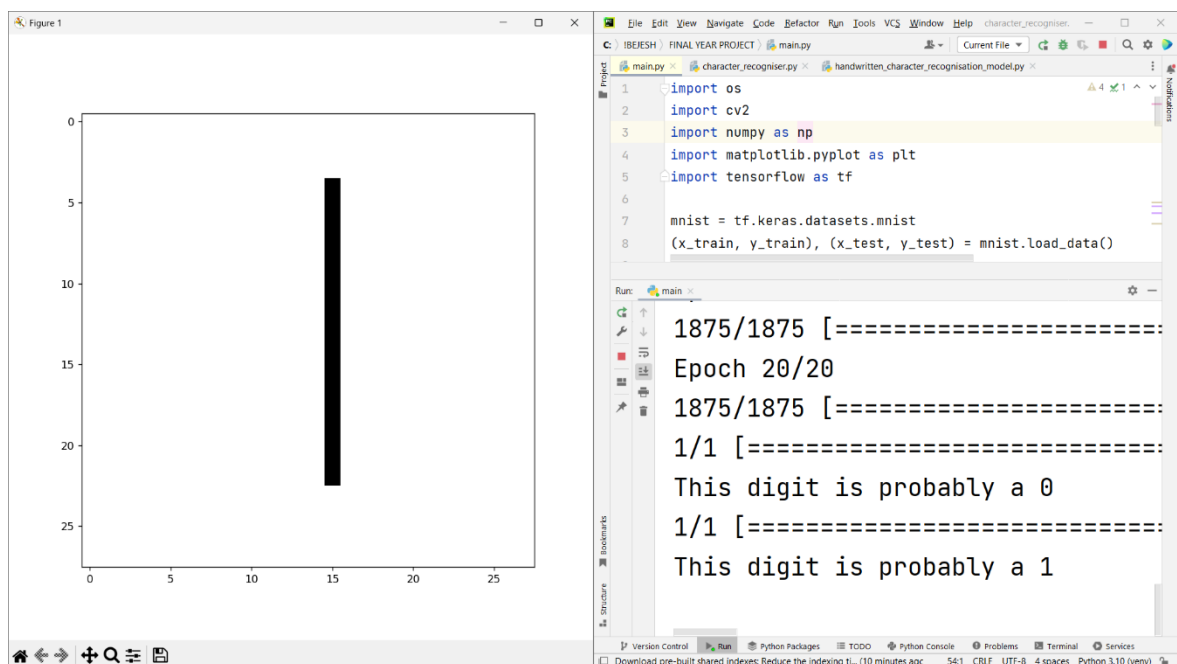


Fig:4.4: Model Predicting Digit 1

The below two images represent the result produced by the model when it has the left displayed image as input.

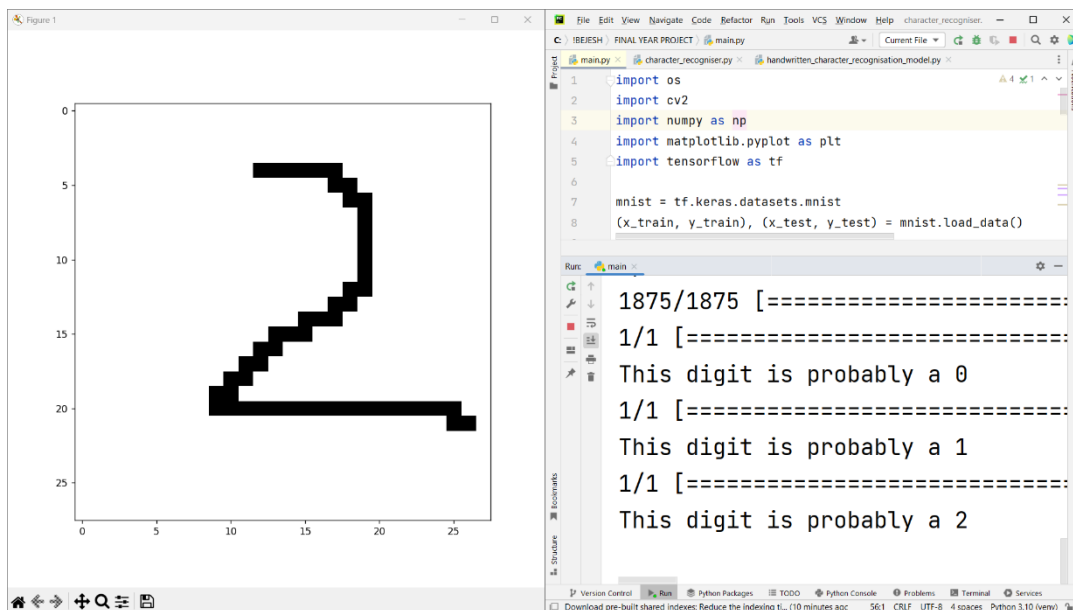


Fig:4.5: Model Predicting Digit 2

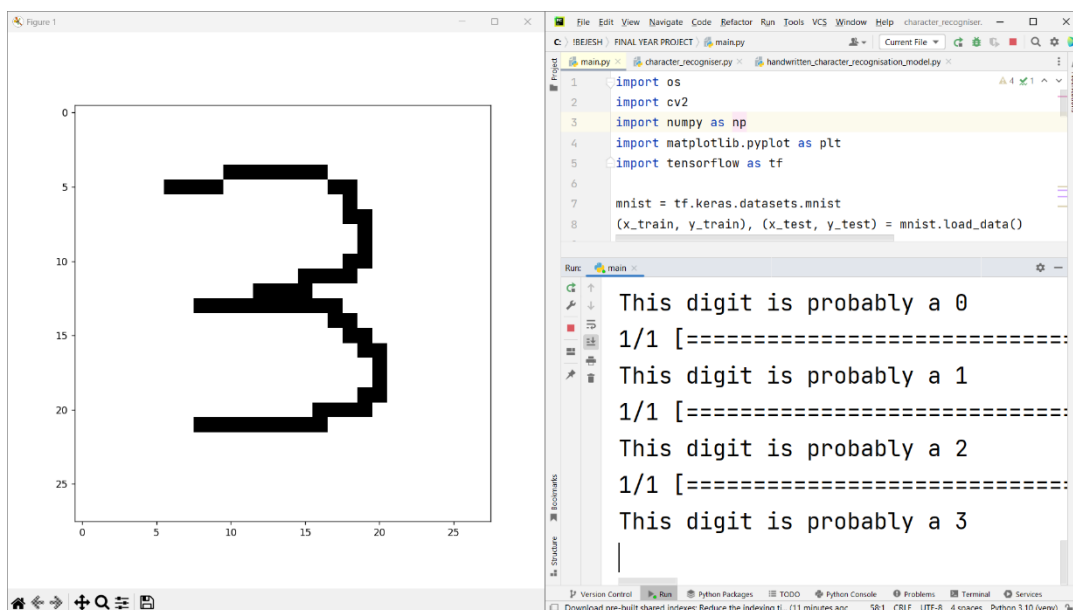


Fig:4.6 Model Predicting Digit 3

When the model is inputted with the Handwritten number 4 and 5, it can be seen that the model has correctly predicted the number.

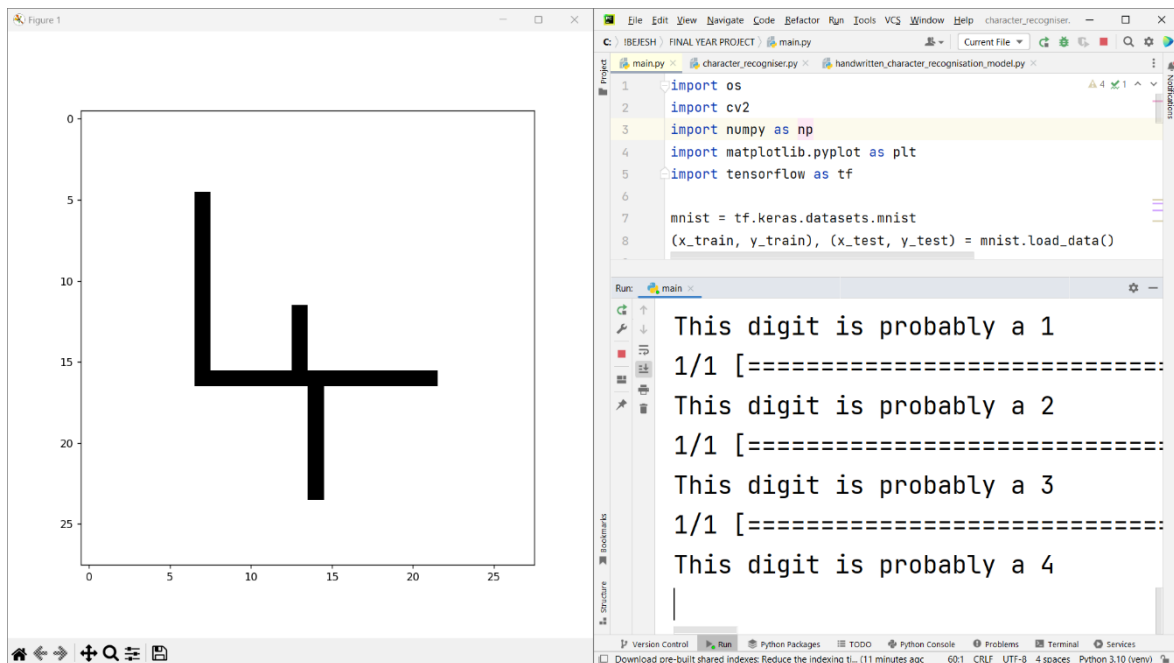


Fig:4.7: Model Predicting Digit 4

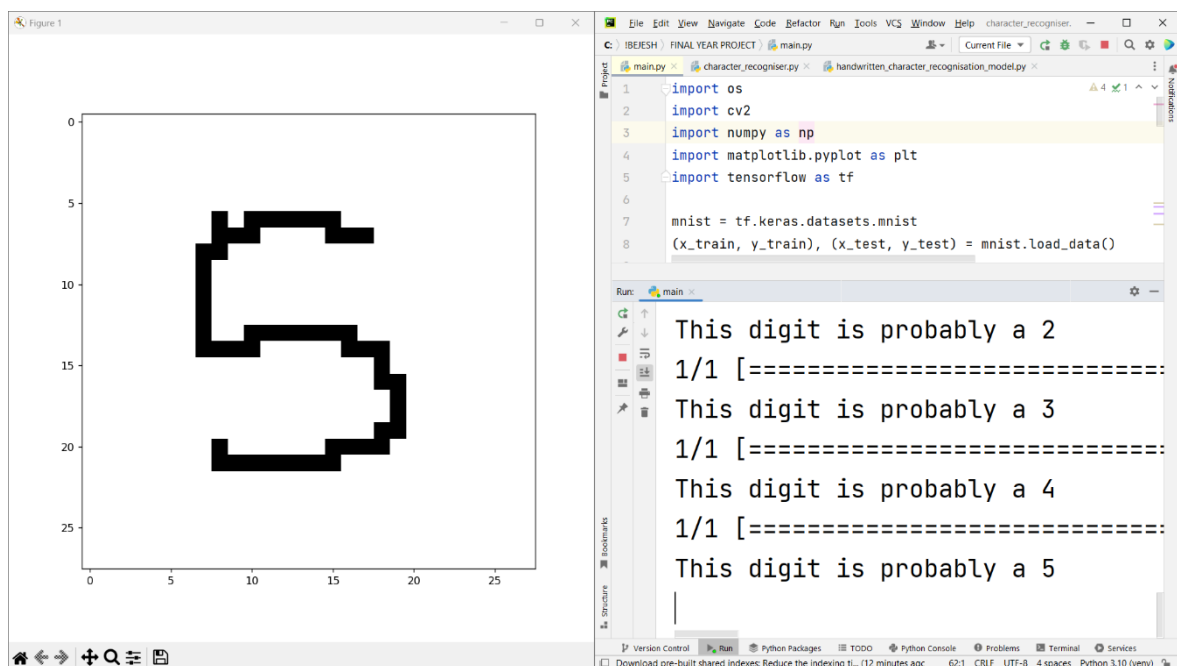


Fig:4.8: Model Predicting Digit 5

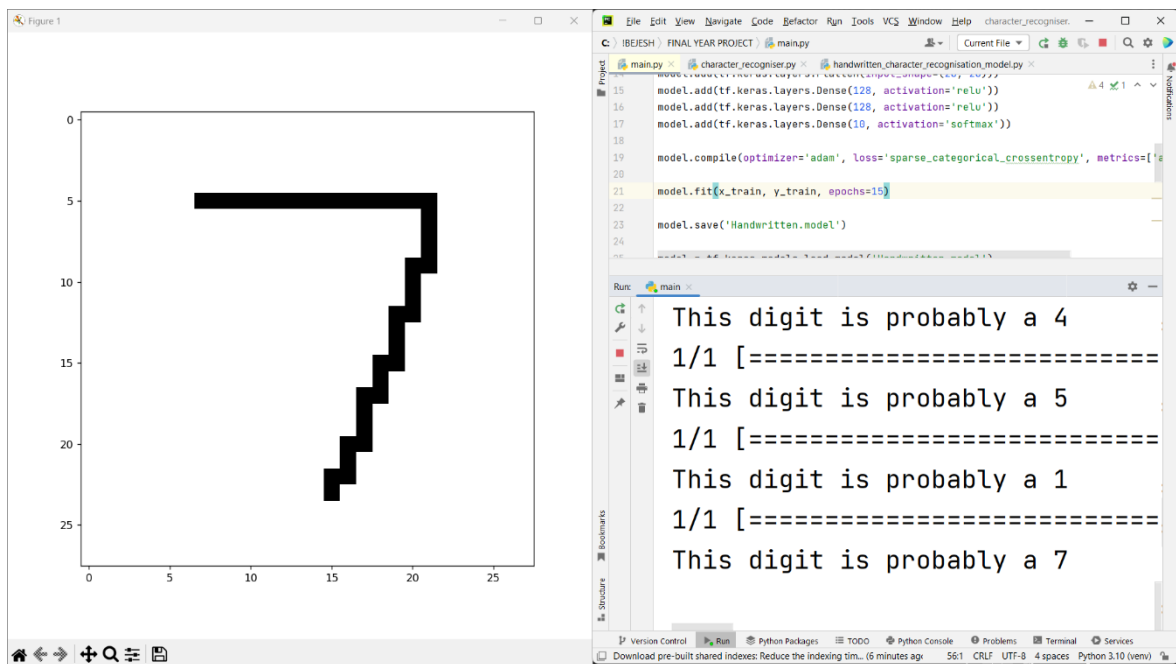


Fig:4.9: Model Predicting Digit 7

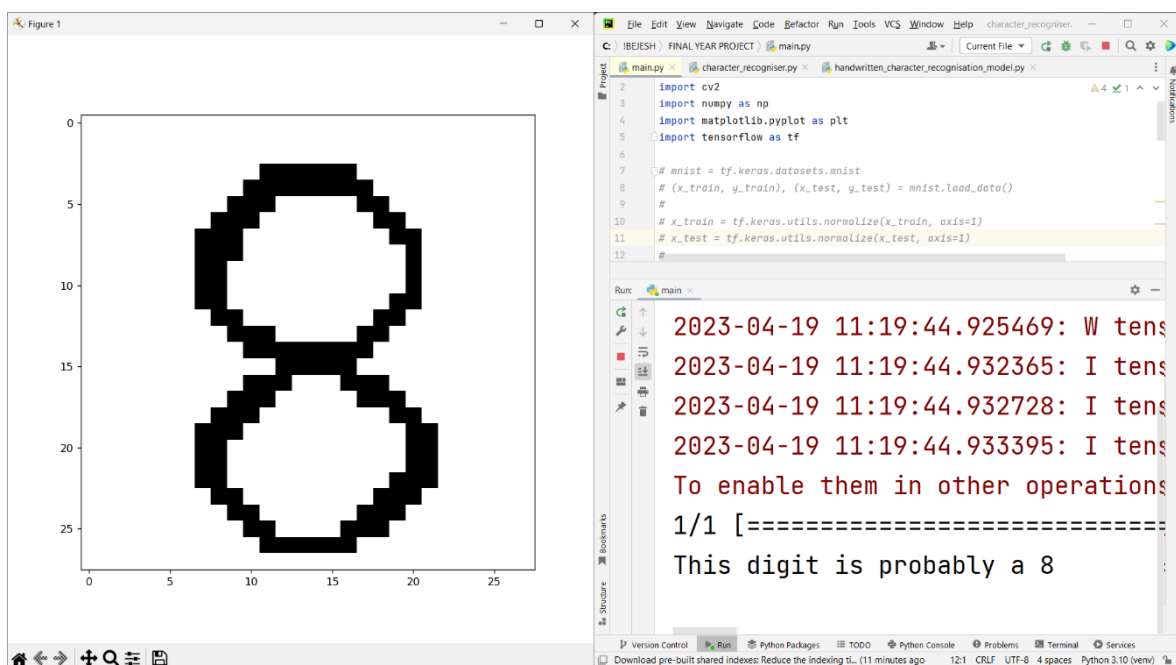


Fig:4.10: Model Predicting Digit 8

4.2 CHARACTER RECOGNITION RESULTS

The Following image represents the prediction made by the Character recognition model when the model is fed with the Handwritten Characters in a (.png) image format.



Fig:4.11: Model Predicting Char A.

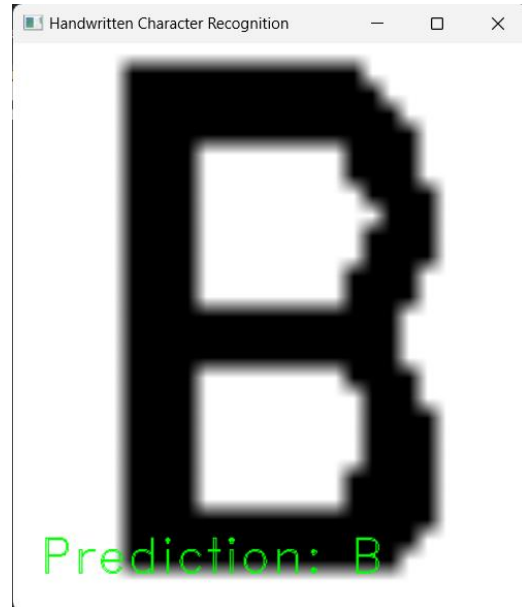


Fig:4.12: Model Predicting Char B.



Fig:4.13: Model Predicting Char C



Fig:4.14: Model Predicting Char D.

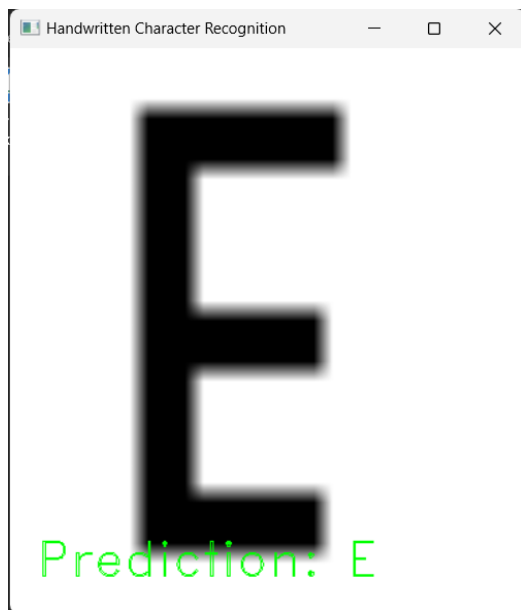


Fig:4.15: Model Predicting Char E



Fig:4.16: Model Predicting Char G



Fig:4.17: Model Predicting Char H



Fig:4.18: Model Predicting Char I

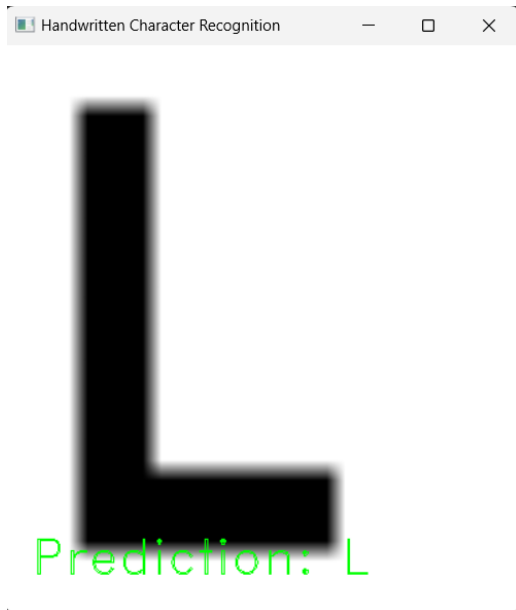


Fig:4.19: Model Predicting Char L



Fig:4.20: Model Predicting Char N



Fig:4.21: Model Predicting Char O

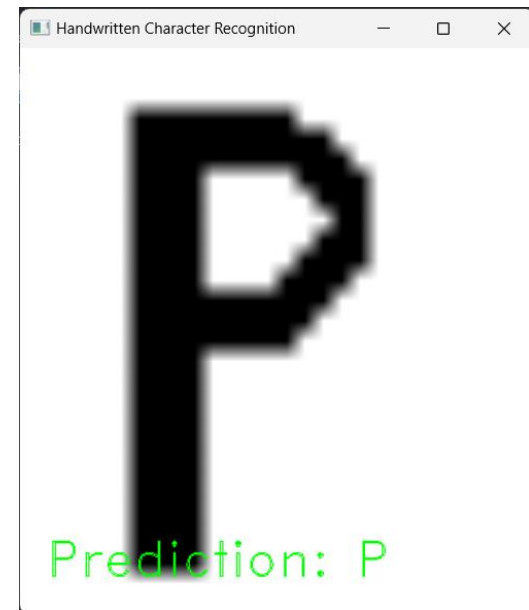


Fig:4.22: Model Predicting Char P



Fig:4.23: Model Predicting Char Q



Fig:4.24: Model Predicting Char R



Fig:4.25: Model Predicting Char T



Fig:4.26: Model Predicting Char V



Fig:4.27: Model Predicting Char W



Fig:4.28: Model Predicting Char X



Fig:4.29: Model Predicting Char Y



Fig:4.30: Model Predicting Char Z

CHAPTER 5

CONCLUSION

The working and importance of Convolutional Neural Network is discussed here. To produce predictions of handwritten numbers from 0 to 9 and characters from A to Z, a well-known dataset called MNIST was used, the dataset was decontaminated, resized, and shaped. TensorFlow is used for creating the machine learning project and successfully created. It is done by developing a CNN model that is incredibly precise and effective at identifying the letters and numbers contained in a given picture. After training the CNN model, predictions were made using the trained model, which had a high level of success in guessing the alphabets and digits.

REFERENCES

- [1] Aiquan Yuan, Gang Bai, Po Yang, Yanni Guo, Xinting Zhao, "Handwritten English Word Recognition based on Convolutional Neural Networks" DOI 10.1109/ICFHR.2012.210
- [2] Al-Mahmud; Asnuva Tanvin; Sazia Rahman, "Handwritten English Character and Digit Recognition", ISBN:978-1-6654-2363-2 DOI: 10.1109/ICECIT54077.2021.9641160 IEEE
- [3] Anca Ignat; Bogdan Aciobanitei, " Handwritten Digit Recognition Using Rotations", ISBN:978-1-5090-5707-8 DOI: 10.1109/SYNASC.2016.054
- [4] Atefeh Dehghanian; Vahid Ghods, "Farsi Handwriting Digit Recognition Based on Convolutional Neural Networks", ISBN:978-1-5386-9450-3 DOI: 10.1109/ISCBI.2018.00022 IEEE
- [5] Bhagayasree P V, Ajay James, Chandran Saravanan, "A Proposed Framework for Recognition of Handwritten Cursive English Characters using DAG-CNN" DOI: 10.1109@ICIICT1.2019.8741412
- [6] Bo Wang, "New Probe into English Network Language Features Value in the Context of Online Education based on Character Recognition", DOI: 10.1109/ICICCS51141.2021.9432183
- [7] Chandrika Saha, Rahat Hossain Faisal and Md. Mostafijur Rahman, "Bangla Handwritten Digit Recognition Using an Improved Deep Convolutional Neural Network Architecture", 978-1-5386-9111-3/19/\$31.00 ©2019 IEEE
- [8] Alexander Filatov, Alexander Gitis, Igor Kil, "Graph-based Handwritten Digit String Recognition", 0-8186-7128-9/95 \$4.00 © 1995 IEEE
- [9] Janotheepan Mariyathas, Vasanthapriyan Shanmuganathan, Banujan Kuhaneswaran, "Sinhala Handwritten Character Recognition using Convolutional Neural Network", DOI: 978-1-6654-1475-3 978-0-73 81 -4403-0/20/\$31.00 ©2020 IEEE
- [10] Huimin Xiao; Chen Liu, "Handwriting digit recognition based on extension engineering", ISBN:978-1-4244-2722-2 DOI: 10.1109/CCDC.2009.5192803
- [11] Marzieh Moradi; Mohammad Ali Poormina; Farbod Razzazi, "FPGA Implementation of Feature Extraction and MLP Neural Network Classifier for Farsi Handwritten Digit Recognition", ISBN:978-1-4244-5345-0 DOI: 10.1109/EMS.2009.13 IEE
- [12] Nisha Sharma, Bhupendra Kumar, Vandita Singh, "Recognition of Off-line Hand printed English Characters, Numerals and Special Symbols", DOI : 978-1-4799-4236-7/14/\$31.00 c 2014 IEEE

- [13] Pal U, Kimura F, Roy K, Pal T, "Recognition of English Multi-oriented Characters", DOI : 0-7695-2521-0/06, 2006 IEEE
- [14] Parthiban R, Ezhilarasi R, Saravanan D, "Optical Character Recognition for English Handwritten Text Using Recurrent Neural Network", 2020 IEE
- [15] Pritam Dhande, Reena Kharat, "Recognition OfCursive English Handwritten Characters", DOI : 978-1-5090-4257-9/17/\$31.00 ©2017 IEEE
- [16] Sanghamitra Mohanty, Himadri Nandini Dasbebartta, Tarun Kumar Behera, "An Efficient Bilingual Optical Character Recognition (English-Oriya) System for Printed Documents" DOI 10.1109/ICAPR.2009.49
- [17] Suman Avdhesh Yadav, Smita Sharma, Shipra Ravi Kumar, "A Robust Approach For Offline English Character Recognition", 978-1-4799-8433-6/15/\$31.00©2015 IEEE
- [18] Wakabayashi T, Pal U, Kimura F, Miyake Y, "F-ratio Based Weighted Feature Extraction for Similar Shape Character Recognition", DOI 10.1109/ICDAR.2009.197
- [19] Xinrui Yu and Jafar Saniie, "Recognizing English Cursive Using Generative Adversarial Networks", 978-1-7281-5317-9/20/\$31.00 ©2020 IEEE
- [20] Yuan Hanning, "Handwritten digits recognition using multiple instance l earning", ISBN:978-1-4799-1282-7 DOI: 10.1109/GrC.2013.6740445
- [21] Wenfei Liu, Jingcheng Wei, Qingmin Meng, "Comparisions on KNN, SVM, BP and the CNN for Handwritten Digit Recognition", 978-1-7281-6521-9/20/\$31.00 2020 IEEE