

HAND WRITTEN DIGIT IMAGE CLASSIFICATION

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering Degree in

Computer Science and Engineering

by

CHANAMAREDDY MANIDEEP REDDY (REG.NO 37110124)

CHALUVADI SANJAY SAI NARESH (REG.NO 37110123)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

12B Status by UGC I Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119.

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC I 12B Status by UGC I

Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Mr. CHALUVADI SANJAY SAI NARESH(37110123)** and **Mr. CHANAMAREDDY MANIDEEP REDDY (37110124)** carried out the project entitled ‘**HAND WRITTEN DIGIT IMAGE CLASSIFICATION**’ under the supervision of **Dr. S. VIGNESHWARI, M.E., Ph.D.**, from November 2020 to March 2021.

Dr. S. VIGNESHWARI, M.E., Ph.D.,

INTERNAL GUIDE

HEAD OF THE DEPARTMENT

DECLARATION

I **CHANAMAREDDY MANIDEEP REDDY (Reg. No. 37110124)** and **CHALUVADI SANJAY SAI NARESH(37110123)** hereby declare that the Project Report entitled **“HAND WRITTEN DIGIT IMAGE CLASSIFICATION”** done by me under the guidance of **Dr. S. VIGNESHWARI, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean, School of Computing** and **Dr. S. VIGNESHWARI, M.E., Ph.D., and Dr. L. LAKSHMANAN, M.E., Ph.D., Heads of the Department, Department of Computer Science and Engineering** for providing me the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide **Dr. S. VIGNESHWARI, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of the project.

Training Certificate



W UNIVERSITY of WASHINGTON

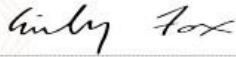
08/05/2020

**CHANAMAREDDY MANIDEEP
REDDY**

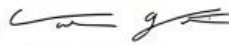
has successfully completed

**Machine Learning Foundations: A Case Study
Approach**

a MOOC from the University of Washington and offered through Coursera



Emily Fox
Amazon Professor of Machine Learning
Statistics



Carlos Guestrin
Amazon Professor of Machine Learning
Computer Science and Engineering

**COURSE
CERTIFICATE**



Verify at coursera.org/verify/PN7JB8GEF27Z
Coursera has confirmed the identity of this individual and
their participation in the course.

ABSTRACT

In this project, we have chosen to focus on recognizing handwritten digits available in the EMNIST database. The challenge in this project is to use basic Image Correlation, also known as Matrix Matching, techniques in order to maximize the accuracy of the handwritten digits recognizer

This project is about classifying the image consisting of hand written and printed digits or numbers. The number or character present in the image is to be identified and given in the output. This can be achieved through Machine Learning, EMNIST dataset and Convolutional Neural Network. If the digit or character is clumsy or look alike then a digit or character is assumed and its accuracy is calculated.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	1
	1.1 OBJECTIVES OF THE PROJECT	1
	1.2 MACHINE LEARNING OVERVIEW	1
	1.3 CNN OVERVIEW	1
	1.4 MNIST OVERVIEW	2
	1.5 EMNIST OVERVIEW	2
2	LITERATURE REVIEW	3
	2.1 LITERATURE REVIEW	3
3	AIM AND SCOPE OF THE PRESENT INVESTIGATION	5
	3.1 AIM	5
	3.2 SCOPE	5
4	EXPERIMENTAL MATERIALS AND METHODS	6
	4.1 ARCHITECTURE	6
	4.1.1 USER PROVIDING IMAGE AND DATASET	6
	4.1.2 PRE PROCESSING	7
	4.1.2.1 DATA CLEANING	7
	4.1.2.2 DATA INTEGRATION	8
	4.1.2.3 DATA REDUCTION	8
	4.1.2.4 DATA TRANSFORMATION	9
	4.1.3 TRAINING THE MODEL	9
	4.1.4 CLASSIFYING USING CNN	9
	4.1.5 OUTPUT	9
5	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	11
	5.1 PERFORMANCE ANALYSIS	11

	5.3 RESULTS	12
6	SUMMARY AND CONCLUSION	15
	6.1 SUMMARY	15
	6.2 CONCLUSION	15
	REFERENCES	16
	APPENDIX	18
	A. SOURCE CODE	18
	B. SCREENSHOTS	26

LIST OF FIGURES

NAME	DESCRIPTION	PAGE NO
FIG 4.1	ARCHITECTURE DIAGRAM	3
FIG 5.1	MNIST DATASET	11
FIG 5.2	CHARACTER DATASET	12
FIG 5.3	INPUT HANDWRITTEN IMAGE	12
FIG 5.4	OUTPUT PREDICTED VALUE	13
FIG 5.5	CHARACTER DATA VISUALIZATION	39
FIG 5.6	ACCURACY AND LOSS PERCENTAGE	39
FIG 6.1	CHARACTER OUTPUT	40
FIG 6.2	CHARACTER INPUT	40

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVES OF THE PROJECT

This project is basically to identify the digits and characters in an image given by the user through Machine Learning using datasets like MNIST and EMNIST and with the help of CNN and SVM. In case of clumsy and look alike digits or characters, an assumption is made and its accuracy is calculated.

The fully functional and verified project is to be given for the Traffic Police of our country,

to identify the vehicle numbers that violate the traffic rules and charge them fine. This will be a great inclusion for Police department and will save a lot of Time and Man power.

1.2 MACHINE LEARNING OVERVIEW

Machine Learning is a subfield of computer science, but is often also referred as predictive analytics or predictive modelling. Its goal and usage is to build new and/or leverage existing algorithms to learn data, in order to build generalizable models that give accurate predictions or to find patterns, particularly with new and unseen similar data.

At the outset of the Machine Learning project a dataset is usually split into two or three subsets. The minimum subsets are training and testing subsets. Once these datasets are created using the primary dataset, a predictive model is trained using the training data and then the models predictive accuracy is determined using the test data. Machine Learning leverages algorithms to automatically model and find patterns in data, usually with the goal of predicting some target output or response

1.3 CNN OVERVIEW

Convolutional Neural Network is a deep learning algorithm which can take in an input image assign importance to various aspects/objects in the image and able to differentiate one from the other. The preprocessing is required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand engineered, with enough training, CNN's have the ability to learn these characteristics.

Convolutional Neural Network have multiple layers.

- Convolutional layer
- Pooling layer
- Fully connected layer

The main objective of the convolutional layer is to obtain the features of the image by sliding them into smaller matrix. The pooling layer is used to retrain the most important aspect by reducing the feature maps. A fully connected layer interconnects every neuron in the layer to neuron from previous and next layer.

1.4 MNIST OVERVIEW

MNIST is an acronym that stands for the Modified National Institute of Standards and Technology dataset. It is a dataset of 60,000 small square 28x28 pixel greyscale images of handwritten single digits between 0 and 9. MNIST dataset was developed in 1998. In recent years many researchers made tests on the MNIST dataset but the best result was given in 2018 with a 0.18% error. The collection of images in the MNIST dataset is a combination of two NIST databases: Special Database-1 and Special Database-3. They consist of digits written by high employees and high school students of the UNITED STATE CENSUS BUREAU.

1.5 CHARACTER DATASET OVERVIEW

The MNIST dataset is considered solved by most of the researchers so, Cohen et al. in April 2017 introduced the extension of MNIST dataset called EMNIST dataset which is also called as CHARACTER DATASET. According to Alejandro Baldominos, the authors of CHARACTER database stated that “The database labeling can be called into question”, describing MNIST as a non-challenging benchmark. Labeling schemata of CHARACTER datasets

- By_Class
- By_Merge
- Balanced
- Digits
- letters

CHAPTER 2

LITERATURE REVIEW

2.1 LITERATURE REVIEW

According to Levi and Hassner (2015), the CNN architecture would overcome the overfitting problem due to less number of images. According to Ashwini B. Patil (2020), the authors found that CNN learns powerful features even from weakly labeled data. Jmour et al. (2018) trained the CNN on image Net dataset for traffic sign classification systems. CNN is used to learn features and classify RGB-D image task. The authors presented the effect of mini-batch on the training model. The model was given 93.33% accuracy on the test set with a minimum batch size of 10. CNN for document image classification is presented in a paper by Kang et al. (2014). Many researchers used CNN with rectified linear units CNN are better than traditional methods. CNN is used for house number image digit classification in a paper published by Sermaneet et al. (2012).

K. Gaurav, Bhatia P. K. [2013], their journal deals with the various pre-processing techniques involved in the character recognition with different kind of images ranges from a simple handwritten form based documents and documents containing colored and complex background and varied intensities. In this, different preprocessing techniques like skew detection and correction, image enhancement techniques of contrast stretching, binarization, noise removal techniques, normalization and segmentation, morphological processing techniques are discussed. It was concluded that using a single technique for preprocessing, we can't completely process the image. However, even after applying all the said techniques might not possible to achieve the full accuracy in a preprocessing system.

Salvador España-Boquera et al [2011] , in this paper hybrid Hidden Markov Model (HMM) model is proposed for recognizing unconstrained offline handwritten texts. In this, the structural part of the optical model has been modelled with Markov chains, and a Multilayer Perceptron is used to estimate the emission probabilities.

Mohammed Z. Khedher, Gheith A. Abandah, and Ahmed M. Al Khawaldeh [2005] et al, their journal describes that Recognition of characters greatly depends upon the features used. Several features of the handwritten Arabic characters are selected and discussed. An off-line recognition system based on the selected features was built. The system was trained and tested with realistic samples of handwritten Arabic characters. Evaluation of the importance and accuracy of the selected features is made. The recognition based on the selected features give average accuracies of 88% and 70% for the numbers and letters, respectively. Further improvements are achieved by using feature weights based on insights gained from the accuracies of individual features.

Hu et al. (2015) proposed a five layer CNN architecture is experimented on several hyperspectral images directly in the spectral domain, which gave a better performance. CNN frameworks such as Caffe used to reduce training and testing time and achieved 90% accuracy on the MNIST dataset. According to Mansessi & Rozza paper presentation in 2018, they introduced two approaches to earn the different combination of base activation function namely identity function, ReLU, and tanh. Tang, y. (2013), proved that replacement of the softmax layer with SVM's is useful for classification of tasks. The experimentation took place on MNIST and CIFAR-10 datasets. Using SVM, cross-validation accuracy for testing phase was increased upto 68.9% while using softmax the accuracy was 67.6%.

A novel neural classifier limited receptive area (LIRA) is proposed by Kussal and Baidy (2004) for image recognition. It contains 3 neuron layers

- Sensor
- Associative layer
- Output layer

The proposed classifier tested on MNIST databases. The classifier showed 99.41% accuracy. The problem of handwritten digit recognition in optical character is addressed by Deng (2012).

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 AIM

The aim of the project is to identify the digits and characters present in an image given by the user.

3.2 SCOPE

It can be achieved through machine learning with the help MNIST and CHARACTER datasets and Convolutional Neural Network and SVM. The machine should take the image input and give the digits or characters in the image as output. In case of clumsy or look alike digits or characters the digit or character is predicted and its accuracy is shown as output.

The MNIST dataset, which is used to identify the digits is used for validating, testing, and training in the field of Machine Learning. A convolutional neural network is used to obtain the features of the image by dividing them into the smaller matrix. MNIST dataset was developed in 1998. CHARACTER dataset is an upgraded model of MNIST dataset.

CNN (Convolutional Neural Network) is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image. CNN need not be limited to only one Convolutional Layer. Conventionally, the first Convolutional Layer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would. There are two types of Pooling: Max Pooling and Average Pooling. **Max Pooling** returns the **maximum value** from the portion of the image covered by the Kernel. On the other hand, **Average Pooling** returns the **average of all the values** from the portion of the image covered by the Kernel. Adding a Fully-Connected layer is a cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

CHAPTER 4

EXPERIMENTAL MATERIALS AND METHODS

4.1 ARCHITECTURE

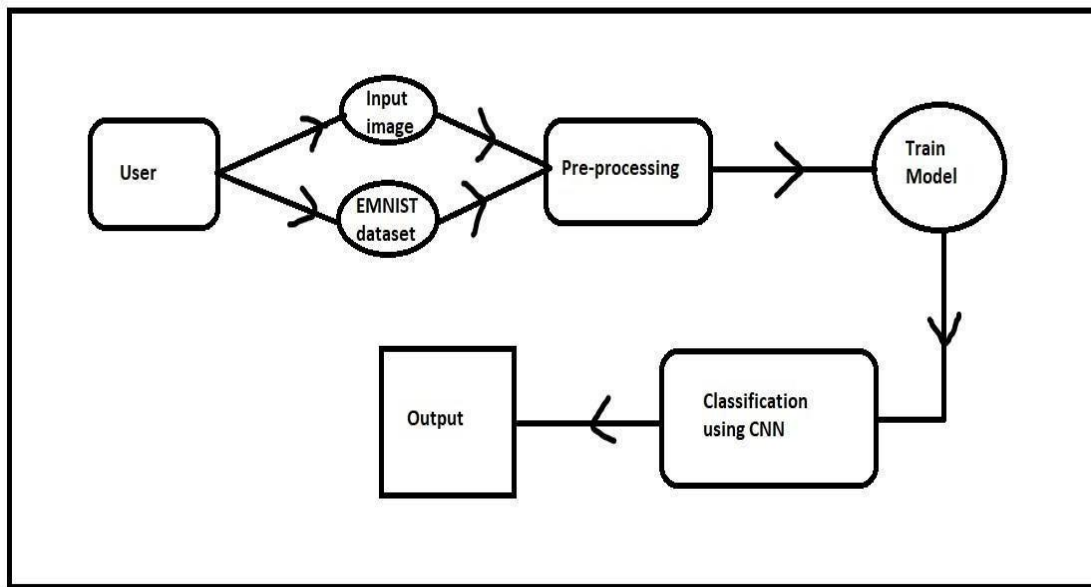


Fig 4.1 Architecture Diagram

This is the architecture diagram of the proposed project. The steps involved are

- User providing image and dataset
- Pre-processing
- Training the model
- Classifying using CNN
- Output

4.1.1 User providing Image and Dataset

The user provide the image in which characters are to be identified and EMNIST dataset code to the machine, for training the machine. Digitized/Digital Image is

initially taken as input. The most common of these devices is the electronic tablet or digitizer. These devices use a pen that is digital in nature. Input images for handwritten characters can also be taken by using other methods such as scanners, photographs or by directly writing in the computer by using a stylus.

4.1.2 Pre-processing

Data pre-processing is used in machine learning to make input data easier to work with. Pre-processing is the basic phase of character recognition and it's crucial for good recognition rate. The main objective of pre-processing steps is to normalize strokes and remove variations that would otherwise complicate recognition and reduce the recognition rate. These variations or distortions include the irregular size of text, missing points during pen movement collections, jitter present in text, left or right bend in handwriting and uneven distances of points from neighbouring positions. To make the process easier, the data pre-processing is divided into four stages.

- Data Cleaning
- Data Integration
- Data Reduction
- Data Transformation

4.1.2.1 Data Cleaning

Data Cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.

This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results. There are several methods for cleaning data depending on how it is stored along with the answers being sought.

Data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set's accuracy without necessarily deleting information.

For one, data cleaning includes more actions than removing data, such as fixing spelling and syntax errors, standardizing data sets, and correcting mistakes such as empty fields, missing codes, and identifying duplicate data points. Data cleaning is considered a foundational element of the data science basics, as it plays an important role in the analytical process and uncovering reliable answers.

4.1.2.2 Data Integration

Data integration is the process of combining data from different sources into a single, unified view. Integration begins with the ingestion process, and includes steps such as cleansing, ETL mapping, and transformation. Data integration becomes increasingly important in cases of merging systems of two companies or consolidating applications within one company to provide a unified view of the company's data assets. The later initiative is often called a data warehouse.

Probably the most well-known implementation of data integration is building an enterprise's data warehouse. The benefit of a data warehouse enables a business to perform analyses based on the data in the data warehouse. This would not be possible to do on the data available only in the source system. The reason is that the source systems may not contain corresponding data, even though the data are identically named, they may refer to different entities.

4.1.2.3 Data Reduction

Data Reduction is the process of reducing the amount of capacity required to store data. Data reduction can increase storage efficiency and reduce costs. Storage vendors will often describe storage capacity in terms of raw capacity and effective capacity, which refers to data after the reduction. Data reduction can be achieved several ways. The main types are data deduplication, compression and single-instance storage. Data deduplication, also known as data dedupe, eliminates redundant segments of data on storage systems. It only

stores redundant segments once and uses that one copy whenever a request is made to access that piece of data. Data dedupe is more granular than single-instance storage. Single-instance storage finds files such as email attachments sent to multiple people and only stores one copy of that file. As with dedupe, single-instance storage replaces duplicates with pointers to the one saved copy.

4.1.2.4 Data Transformation

Data transformation is the process of converting data from one format to another, typically from the format of a source system into the required format of a destination system. Data transformation is a component of most data integration and data-management tasks, such as data wrangling and data warehousing. Data transformation is the process of converting data from one format to another. The most common data transformations are converting raw data into a clean and usable form, converting data types, removing duplicate data, and enriching the data to benefit an organization. During the process of data transformation, an analyst will determine the structure, perform data mapping, extract the data from the original source, execute the transformation, and finally store the data in an appropriate database.

Transformed data is usable, accessible, and secure to benefit a variety of purposes. Organizations may transform data to make it compatible with other types of data, move it into the appropriate database, or combine it with other crucial information. Organizations benefit from transforming data by gaining insights into vital operational and informational internal and external functions. In addition, data transformation makes it possible for organizations to transform data from a storage database to the cloud to keep information moving.

4.1.3 Training the model

The goal of training is to create an accurate model that answers our questions correctly most of the time. But in order to train a model, we need to collect data to train on. Training usually takes between 2-8 hours depending on the number of files and queued models for training.

4.1.4 Classifying using CNN

The image is classified through the different layers of CNN like convolution, pooling and fully connected layers. When input image is presented to CNN system, its features are extracted and given as an input to the trained classifier like artificial neural network or support vector machine. Classifiers compare the input feature with stored pattern and find out the best matching class for input.

4.1.5 Output

The output is the digit or the character present in the image provided by the user. In case of clumsy or look alike character or digit, an assumption is made and its accuracy is calculated. Language information can increase the accuracy obtained by pure shape recognition. For handwriting input, some shape recognizers yield a single string of characters, while others yield a number of alternatives for each character, often with a measure of accuracy for each alternative.

CHAPTER 5

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

5.1 PERFORMANCE ANALYSIS

Image processing in general can cause the opposition of some people from different societies. The reason behind this is because this field can be further developed for malicious intentions, perhaps in the military field to be able to target certain people, and many other similar possibilities. That is why if image processing is not well managed and used with a wise mentality and good intentions, it could represent a threat to societies.

The output of this project is a program that can read images of an unknown handwritten digits and be able to identify. Therefore, it can be useful in the future to identify images containing handwritten digits. Moreover, it provides some techniques that could be further developed in order to be able to recognize all types of characters in images or scanned documents.

Image processing needs to be taken into consideration by all the authorities, as they need to enforce some laws on its uses and applications to ensure that none of them could be used against the well-being and serenity of humanity.

5.2 DISCUSSION

Implementing this technology on a large scale could evidently bring about several repercussions. If we were to begin with an impact of this approach, the most obvious one is the social effect. Recognizing handwritten digits, if not used with good intentions, could lead to the manifestation of several social issues. First of all, this method could lead to the increase of theft in societies. That is to say, if someone was to take a picture of a code or pin number of a private account, which maybe written by the owner, or available in a scanned document, the image could be used, even if it is not of a high resolution as it could be taken by a camera located far away from the original document, in order to be able to segment it and extract the code from it. This problem could appear in several other cases.

5.3 RESULTS

The screenshot displays an Excel spreadsheet titled 'mnist_train.csv'. The spreadsheet contains 20 rows and 24 columns. The columns are labeled DQ, DR, DS, DT, DU, DV, DW, DX, DY, DZ, EA, EB, EC, ED, EE, EF, EG, EH, EI, EJ, and EK. The rows are numbered 1 through 20. The data is as follows:

	DQ	DR	DS	DT	DU	DV	DW	DX	DY	DZ	EA	EB	EC	ED	EE	EF	EG	EH	EI	EJ	EK
1	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	51	159	253	159	50	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
4	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
6	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	145	255	211	31	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
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	189	190	0	0	0	0
11	0	0	0	0	42	118	219	166	118	118	6	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	23	197	253	252	208	19	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	58	242	221	143	17	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 5.1 MNIST dataset

This figure shows the representation of MNIST dataset and its values stored.

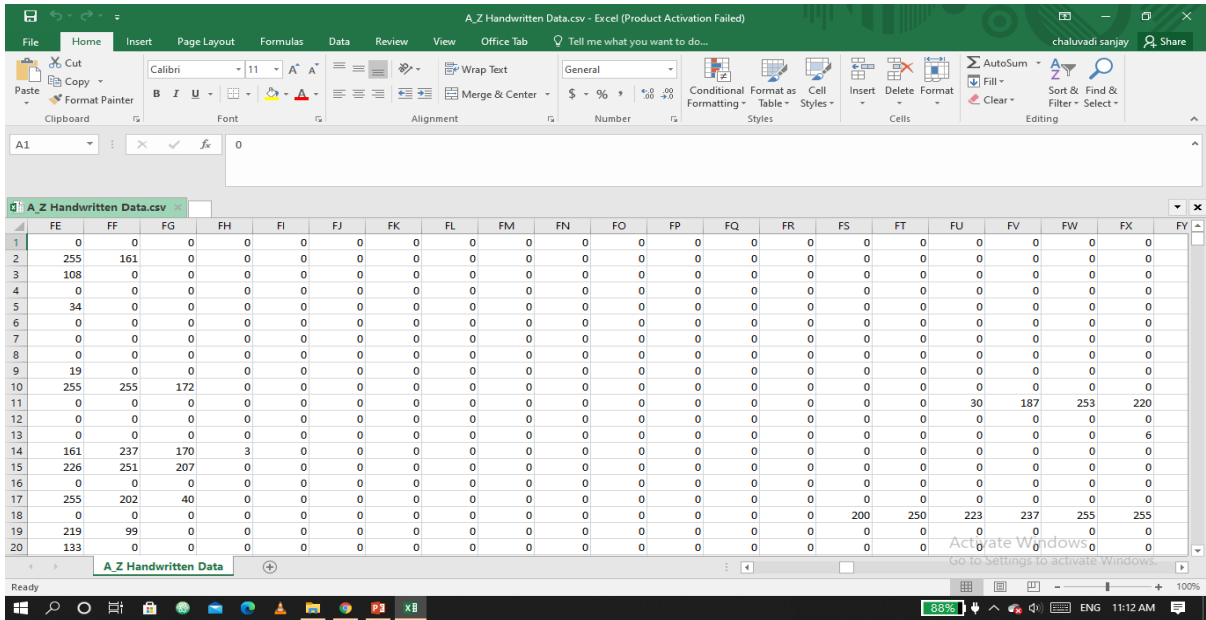


Fig 5.2 Character dataset

This figure shows the representation of Character dataset and its values stored.

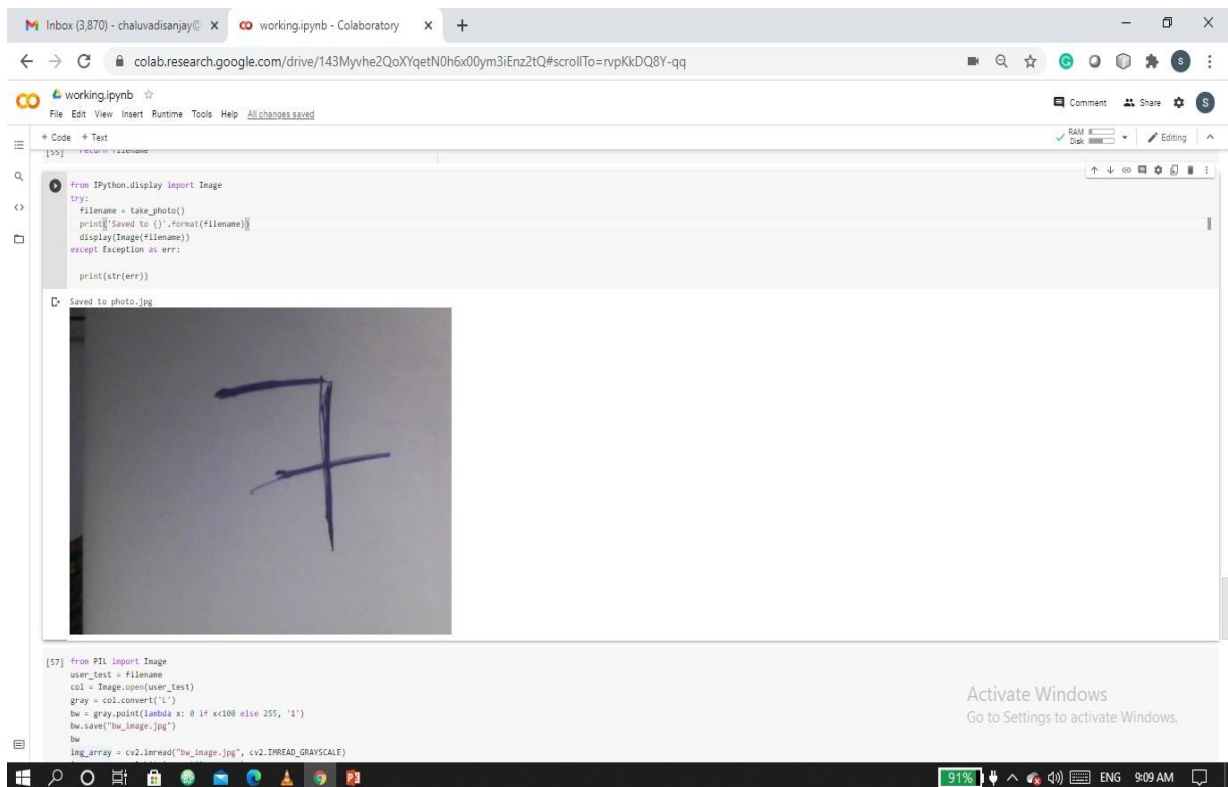


Fig 5.3 Input Hand-written digit

This figure shows the Input image with Hand-Written digit “7” on it.

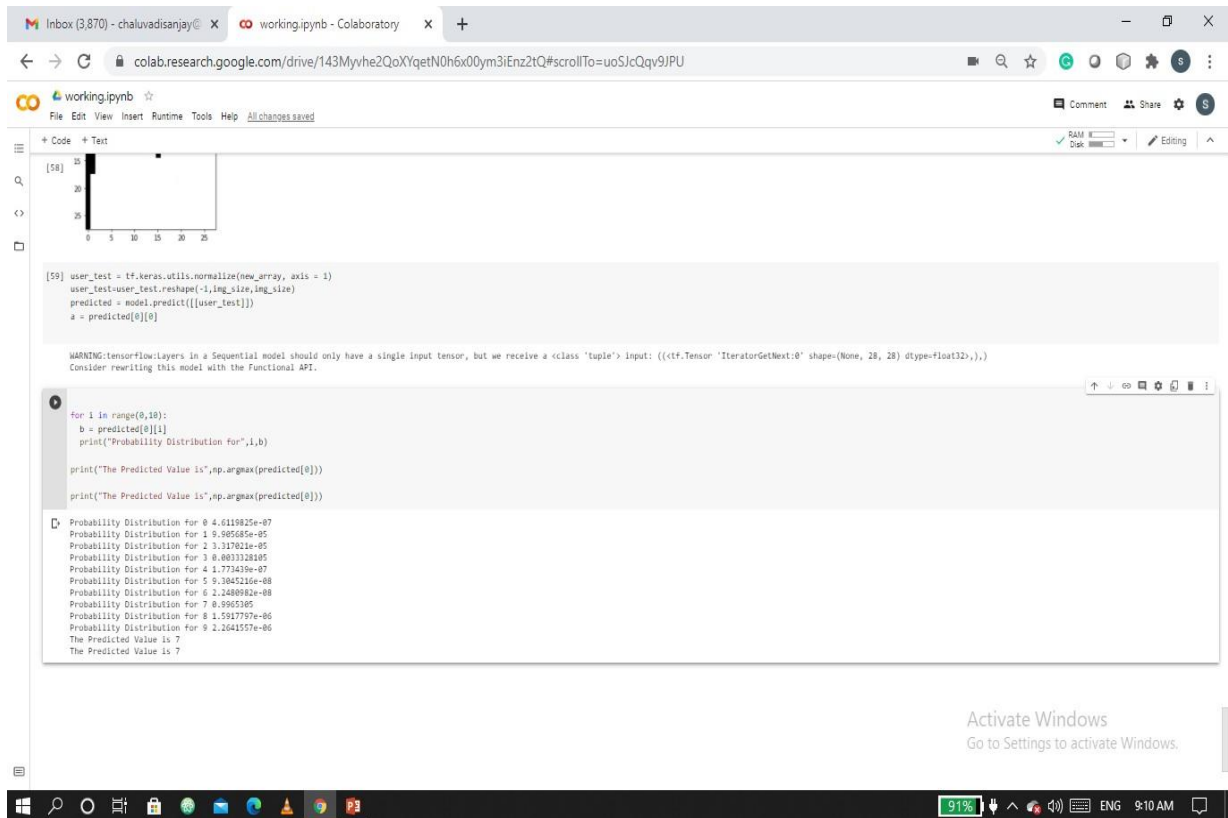


Fig 5.4 Output displaying the Predicted Value

This figure shows that the Predicted value as “7” which is same as the input Value.

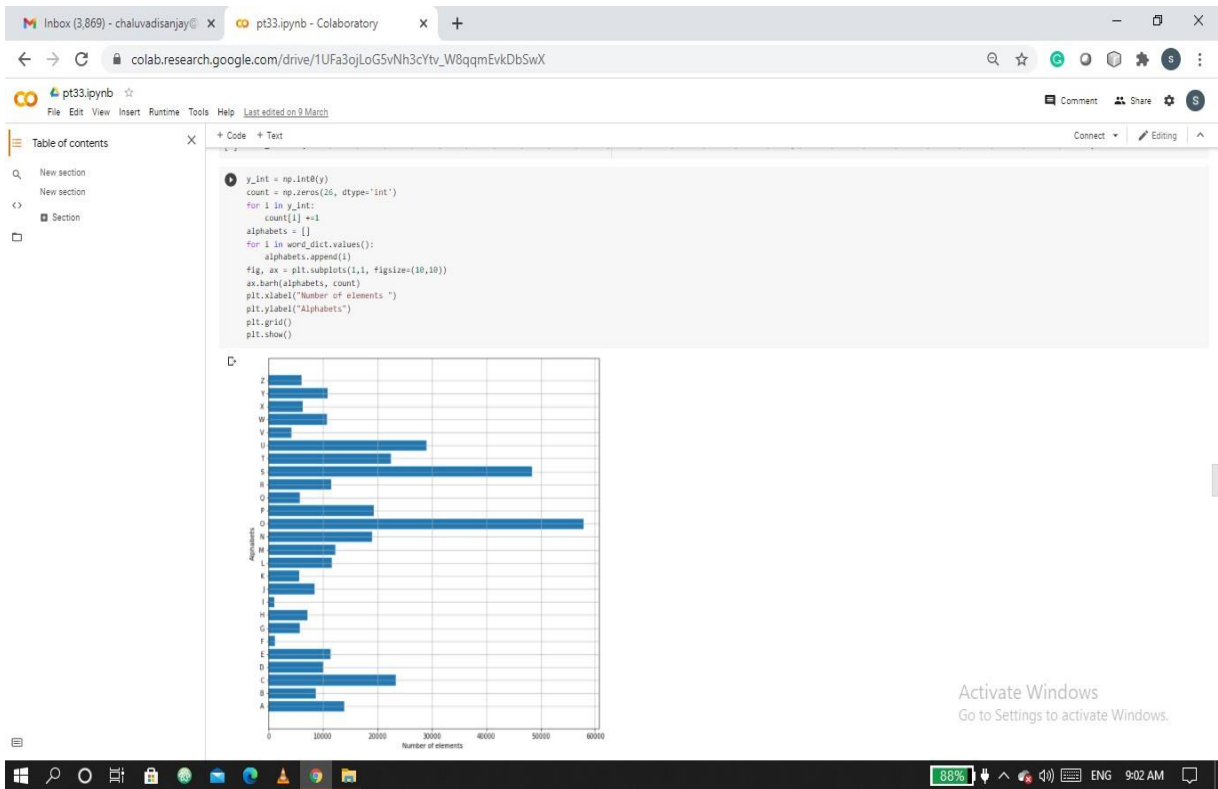


Fig 5.5 Character data visualization

This figure shows the visualization of the Character data to the number of elements.

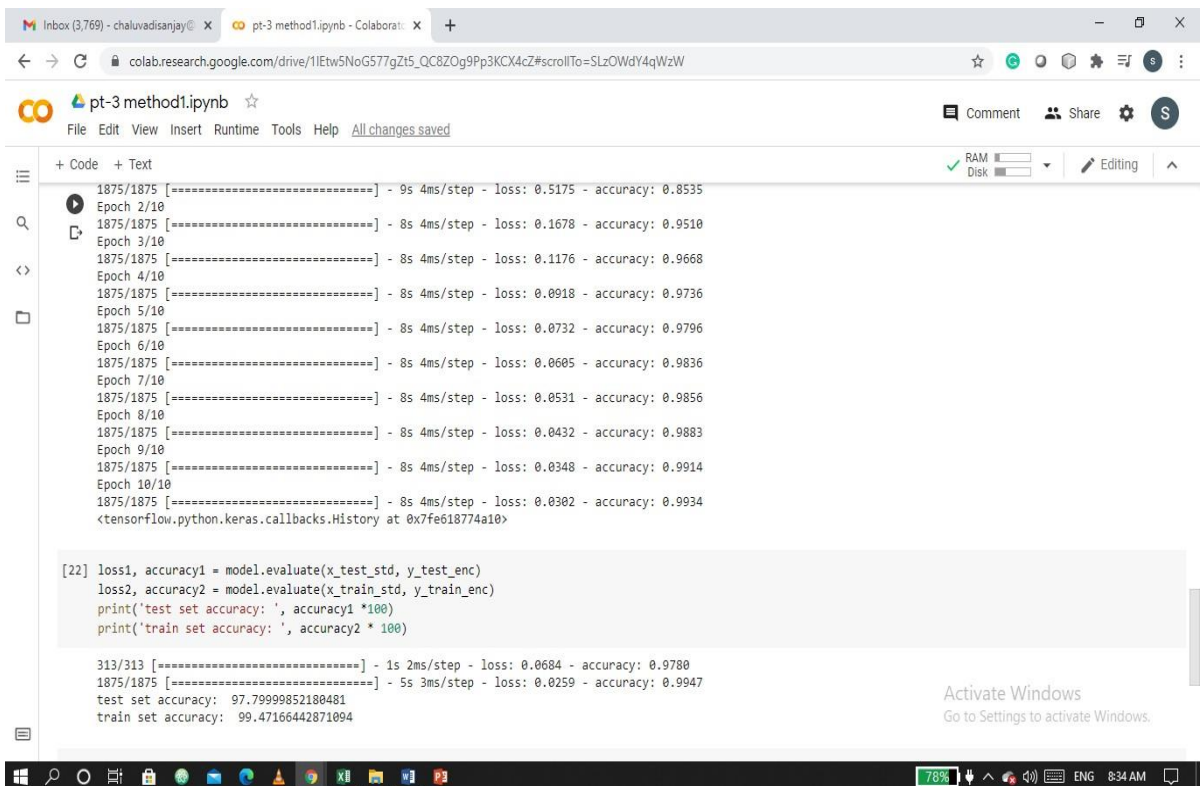


Fig 5.6 Accuracy and Loss Percentage

This figure shows the calculated Accuracy and Loss Percentage.

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 SUMMARY

Let us now discuss about the summary in this section, the major problems I figured out while making this project and I could see that this project can be used for good as well as bad. Its usage depends on person using it. The main idea of developing this project is providing this to police department for identifying the vehicle numbers violating the traffic rules and help police in catching them or identifying them so that a challan can be sent to the owner of the vehicle.

6.2 CONCLUSION

Optical Character Recognition is a very broad field concerned with turning an image or a scanned document containing a set of characters into an encoded text that could be read by machines. In this project, we have attempted to build a recognizer for handwritten digits using the MNIST dataset. The challenge of this project was to be able to come up with some basic image correlation techniques, instead of some sophisticated algorithms, and see to what extent we can make this mechanism accurate. We have tried several versions and kept trying to improve each one in order to reach a higher performance rate. The future steps that to go for would be having a closer look at the results of all the versions in order to find new rules. By extracting and implementing them, we will be able to enhance the performance of these versions. Moreover, it would be good if we could make some modifications to both the reference set and the rules in order to make our program more general and able to identify both typed and handwritten digits.

REFERENCES

1. Boser, B. E., Guyon, I. M. & Vapnik, V. N., 1992. A training algorithm for optimal margin classifiers. Proceedings of the fifth annual workshop on Computational learning theory, p. 144.
2. Burges, C., 1998. A tutorial on support vector machines for pattern recognition. Knowledge Discovery Data Mining, 2(2), p. 1–43.
3. Byun, H. a. S.-W. L., 2002. Applications of support vector machines for pattern recognition: A survey.. Pattern recognition with support vector machines. Springer, Berlin, Heidelberg,, pp. 213-236.
4. Cavalin, P.R.; Britto, A.S.; Bortolozzi, F.; Sabourin, R.; Oliveira, L.E.S. An implicit segmentation-based method for recognition of handwritten strings of characters. In Proceedings of the 2006 ACM Symposium on Applied Computing, Dijon, France, 23–27 April 2006; pp. 836–840.
5. Cireşan, D.C.; Meier, U.; Schmidhuber, J. Multi-column deep neural networks for image classification. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3642–3649
6. Cireşan, D.; Meier, U.; Gambardella, L.M.; Schmidhuber, J. Deep, big, simple neural nets for handwritten digit recognition. Neural Comput. 2010, 22, 3207–3220.
7. Cohen, G.; Afshar, S.; Tapson, J.; van Schaik, A. EMNIST: An extension of MNIST to handwritten letters. arXiv 2017, arXiv:1702.05373.
8. D.E. Rumelhart, G. H. R. W., 1986. Learning representations by back-propagation errors. Nature, 323(9), p. 533–536.
9. Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web].
10. F. Kimura, K. T. S. T. Y. M., 1987. Modified Quadratic Discriminant Functions

and the Application to Chinese Character Recognition. IEEE Trans. Pattern Anal., 9(1), p. 149–153.

11. Haykin, S., 1999. Neural Networks. s.l.: Prentice Hall Inc. Knerr, S. L. P. a. G. D., 1990. Single-layer learning revisited: a stepwise procedure for building and training a neural network. Neurocomputing. Springer, Berlin, pp. 41-50.

12. IEEE Signal Processing Magazine, 29(6), 141-142.

13. Meier, U.; Cireşan, C.; Gambardella, L.M.; Schmidhuber, J. Better Digit Recognition with a Committee of Simple Neural Nets. In Proceedings of the 2011 International Conference on Document Analysis and Recognition, Beijing, China, 18–21 September 2011; pp. 1250–1254.

14. Sato, I.; Nishimura, H.; Yokoi, K. APAC: Augmented Pattern Classification with Neural Networks. arXiv 2015, arXiv:1505.03229.

15. Shivam S. Kadam, Amol C. Adamuthe* , and Ashwini B. Patil (2020). Journal of Scientific Research. Volume 64, issue 2.

16. Simard, P.; Steinkraus, D.; Platt, J.C. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In Proceedings of the 7th International Conference on Document Analysis and Recognition, Edinburgh, UK, 3–6 August 2003; Volume 2, pp. 958–963.

17. Singh, S.; Paul, A.; Arun, M. Parallelization of digit recognition system using Deep Convolutional Neural Network on CUDA. In Proceedings of the 2017 Third International Conference on Sensing, Signal Processing and Security, Chennai, India, 4–5 May 2017; pp. 379–383.

18. Wan, L.; Zeiler, M.; Zhang, S.; LeCun, Y.; Fergus, R. Regularization of neural networks using DropConnect. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; Volume 28, pp. 3-1058–3-1066.

19. Yang, J.; Yu, K.; Huang, T. Supervised translation-invariant sparse coding. In Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 3517–3524.

APPENDIX

A. SOURCE CODE

```
import tensorflow as tf
import keras
from tensorflow.keras.callbacks import TensorBoard
import time
import numpy as np
import matplotlib.pyplot as plt
import cv2
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print("Training Data Shape is {}".format(x_train.shape))
print("Training Labels Shape is {}".format(y_train.shape))
print("Testing Data Shape is {}".format(x_test.shape))
print("Testing Labels Shape is {}".format(y_test.shape))
print("Sample Training Data is {}".format(x_train[0]))
print("Sample Training label is {}".format(y_train[0:5]))
for i in range(0,20):
    plt.imshow(x_train[i], cmap = plt.cm.binary)
    plt.show()
x_train = tf.keras.utils.normalize(x_train, axis = 1)
x_test = tf.keras.utils.normalize(x_test, axis = 1)
print("Training Data after normalizing is {}".format(x_train[0]))
print("Testing Data after normalizing is {}".format(x_test[0]))
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128,activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(128,activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(10,activation=tf.nn.softmax)) # so
ftmax for probability distribution
model.compile(optimizer = "adam" , loss = 'sparse_categorical_cros
sentropy' , metrics = ['accuracy'] )
model.fit(x_train,y_train,epochs = 3 )
```

```

predictions = model.predict([x_test])
val_loss, val_acc = model.evaluate(x_test, y_test)
print(val_loss, val_acc)
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
    js = Javascript('''
        async function takePhoto(quality) {
            const div = document.createElement('div');
            const capture = document.createElement('button');
            capture.textContent = 'Capture';
            div.appendChild(capture);

            const video = document.createElement('video');
            video.style.display = 'block';
            const stream = await navigator.mediaDevices.getUserMedia({vi
deo: true});

            document.body.appendChild(div);
            div.appendChild(video);
            video.srcObject = stream;
            await video.play();

            google.colab.output.setIframeHeight(document.documentElement
.scrollHeight, true);

            await new Promise((resolve) => capture.onclick = resolve);

            const canvas = document.createElement('canvas');
            canvas.width = video.videoWidth;
            canvas.height = video.videoHeight;
            canvas.getContext('2d').drawImage(video, 0, 0);
            stream.getVideoTracks()[0].stop();
            div.remove();
            return canvas.toDataURL('image/jpeg', quality);
        }
    ''')
    display(js)
    data = eval_js('takePhoto({})'.format(quality))
    binary = b64decode(data.split(',')[1])

    with open(filename, 'wb') as f:
        f.write(binary)
    return filename
from IPython.display import Image
try:
    filename = take_photo()
    print('Saved to {}'.format(filename))
    display(Image(filename))
except Exception as err:

```

```

    print(str(err))
from PIL import Image
user_test = filename
col = Image.open(user_test)
gray = col.convert('L')
bw = gray.point(lambda x: 0 if x<100 else 255, '1')
bw.save("bw_image.jpg")
bw
img_array = cv2.imread("bw_image.jpg", cv2.IMREAD_GRAYSCALE)
img_array = cv2.bitwise_not(img_array)
print(img_array.size)
plt.imshow(img_array, cmap = plt.cm.binary)
plt.show()
img_size = 28
new_array = cv2.resize(img_array, (img_size,img_size))
plt.imshow(new_array, cmap = plt.cm.binary)
plt.show()

user_test = tf.keras.utils.normalize(new_array, axis = 1)
user_test=user_test.reshape(-1,img_size,img_size)
predicted = model.predict([[user_test]])
a = predicted[0][0]

for i in range(0,10):
    b = predicted[0][i]
    print("Probability Distribution for",i,b)

print("The Predicted Value is",np.argmax(predicted[0]))

print("The Predicted Value is",np.argmax(predicted[0]))
from keras.datasets import mnist

(train_data,train_target),(test_data,test_target)=mnist.load_data(
)
print(train_data.shape)
print(test_data.shape)
print(train_target.shape)
print(test_target.shape)
from matplotlib import pyplot as plt

plt.imshow(train_data[0],cmap='gray')
plt.show()
train_target[0]
from keras.utils import np_utils

new_train_target=np_utils.to_categorical(train_target)
new_test_target=np_utils.to_categorical(test_target)

print(train_target[:10])
print(new_train_target[:10])
new_train_data=train_data/255
new_test_data=test_data/255

```

```

from keras.models import Sequential
from keras.layers import Dense, Flatten

model=Sequential()

model.add(Flatten(input_shape=(28,28)))

model.add(Dense(512,activation='relu'))
model.add(Dense(256,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(10,activation='softmax'))

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
model.fit(new_train_data,new_train_target,epochs=20)
from matplotlib import pyplot as plt

plt.plot(model.history.history['loss'])
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.show()
plt.plot(model.history.history['accuracy'])
plt.xlabel('# epochs')
plt.ylabel('accuracy')
plt.show()
import matplotlib.pyplot as plt
import cv2
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from keras.optimizers import SGD, Adam
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras.utils import to_categorical
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
data = pd.read_csv(r"A_Z Handwritten Data.csv").astype('float32')
print(data.head(10))
X = data.drop('0',axis = 1)
y = data['0']
train_x, test_x, train_y, test_y = train_test_split(X, y, test_size = 0.2)
train_x = np.reshape(train_x.values, (train_x.shape[0], 28,28))
test_x = np.reshape(test_x.values, (test_x.shape[0], 28,28))
print("Train data shape: ", train_x.shape)
print("Test data shape: ", test_x.shape)
shuff = shuffle(train_x[:100])
fig, ax = plt.subplots(3,3, figsize = (10,10))
axes = ax.flatten()
for i in range(9):

```

```

_, shu = cv2.threshold(shuff[i], 30, 200, cv2.THRESH_BINARY)
axes[i].imshow(np.reshape(shuff[i], (28,28)), cmap="Greys")
plt.show()
word_dict = {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'}
y_int = np.int0(y)
count = np.zeros(26, dtype='int')
for i in y_int:
    count[i] +=1
alphabets = []
for i in word_dict.values():
    alphabets.append(i)
fig, ax = plt.subplots(1,1, figsize=(10,10))
ax.barh(alphabets, count)
plt.xlabel("Number of elements ")
plt.ylabel("Alphabets")
plt.grid()
plt.show()
train_X = train_x.reshape(train_x.shape[0],train_x.shape[1],train_
x.shape[2],1)
print("New shape of train data: ", train_X.shape)
test_X = test_x.reshape(test_x.shape[0], test_x.shape[1], test_x.s
hape[2],1)
print("New shape of test data: ", test_X.shape)

train_yOHE = to_categorical(train_y, num_classes = 26, dtype='int'
)
print("New shape of train labels: ", train_yOHE.shape)
test_yOHE = to_categorical(test_y, num_classes = 26, dtype='int')
print("New shape of test labels: ", test_yOHE.shape)
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu'
, input_shape=(28,28,1)))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'
, padding = 'same'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'
, padding = 'valid'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Flatten())
model.add(Dense(64,activation = "relu"))
model.add(Dense(128,activation = "relu"))
model.add(Dense(26,activation = "softmax"))
model.compile(optimizer = Adam(learning_rate=0.001), loss='categor
ical_crossentropy', metrics=['accuracy'])
history = model.fit(train_X, train_yOHE, epochs=1, validation_dat
a = (test_X,test_yOHE))
model.summary()
model.save(r'model_hand.h5')
print("The validation accuracy is :", history.history['val accurac
y'])

```

```

print("The training accuracy is :", history.history['accuracy'])
print("The validation loss is :", history.history['val_loss'])
print("The training loss is :", history.history['loss'])
fig, axes = plt.subplots(3,3, figsize=(8,9))
axes = axes.flatten()
for i,ax in enumerate(axes):
    img = np.reshape(test_X[i], (28,28))
    ax.imshow(img, cmap="Greys")

    pred = word_dict[np.argmax(test_yOHE[i])]
    ax.set_title("Prediction: "+pred)
    ax.grid()
    img = cv2.imread('hii.png')
    img_copy = img.copy()
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (400,440))
    predcted=[2]
    whaimg_copy = cv2.GaussianBlur(img_copy, (7,7), 0)
    img_gray = cv2.cvtColor(img_copy, cv2.COLOR_BGR2GRAY)
    _, img_thresh = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY_INV)
    img_final = cv2.resize(img_thresh, (28,28))
    img_final = np.reshape(img_final, (1,28,28,1))
    from google.colab.patches import cv2_imshow
    img_pred = word_dict[np.argmax(model.predict(img_final))]
    cv2.putText(img, "Dataflair ", (20,25), cv2.FONT_HERSHEY_TRIPLEX, 0.7, color = (0,0,230))
    cv2.putText(img, "Prediction:" + img_pred, (20,410), cv2.FONT_HERSHEY_DUPLEX, 1.3, color = (255,0,30))
    cv2_imshow(img)

```

```
import requests
```

```

print ('Starting to Download!')
url='https://user-images.githubusercontent.com/40036314/48663507-57567a00-eab7-11e8-822e-ac80eace819b.jpg'
r = requests.get(url)

```

```
filename = '2.jpg'
```

```

with open(filename, 'wb') as out_file:
    out_file.write(r.content)

```

```
print("Download complete!")
```

```

import cv2
import numpy as np
import pytesseract

```

```

from PIL import Image

# Path of working folder on Disk

def get_string(img_path):
    # Read image with opencv
    img = cv2.imread(img_path)

    # Convert to gray
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Apply dilation and erosion to remove some noise
    kernel = np.ones((1, 1), np.uint8)
    img = cv2.dilate(img, kernel, iterations=1)
    img = cv2.erode(img, kernel, iterations=1)

    # Write image after removed noise
    cv2.imwrite("removed_noise.png", img)

    # Apply threshold to get image with only black and white
    #img = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH
    _GAUSSIAN_C, cv2.THRESH_BINARY, 31, 2)

    # Write the image after apply opencv to do some ...
    cv2.imwrite(img_path, img)

    # Recognize text with tesseract for python
    result = pytesseract.image_to_string(Image.open(img_path))

    # Remove template file
    #os.remove(temp)

    return result

print ('--- Start recognize text from image ---')
print (get_string(filename))

print ("----- Done----- ")

```

B. SCREENSHOTS

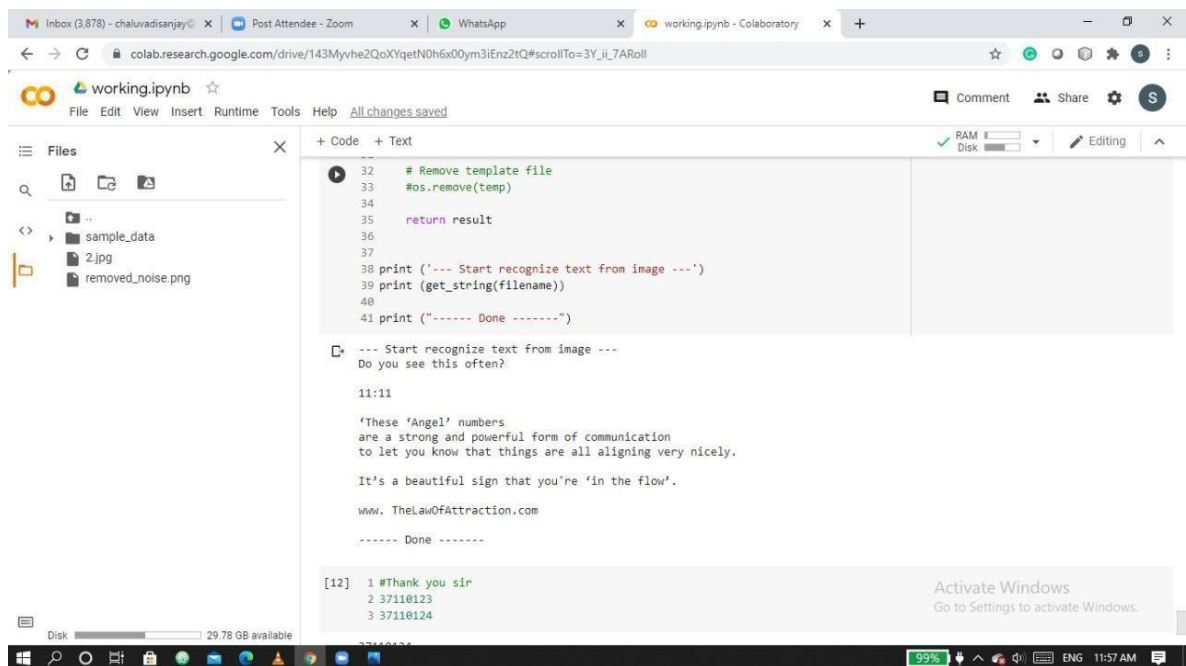


Fig 6.1: Character output

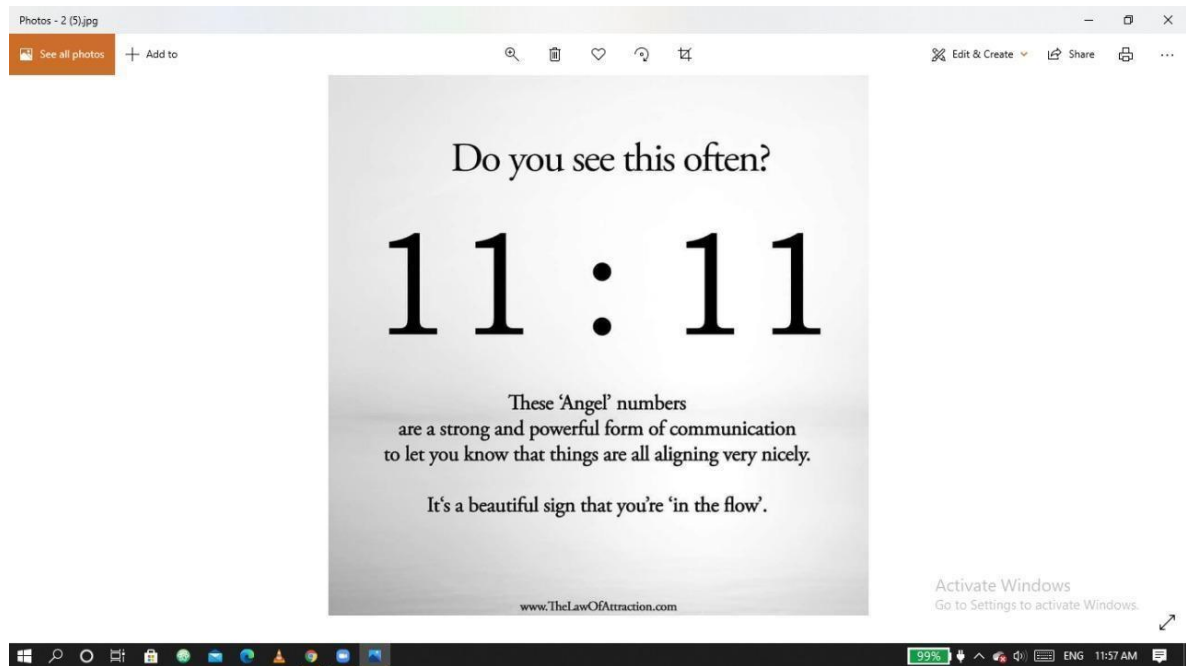


Fig 5.2: Character input