Machine Learning model for Numeral Recognition

Harsha Halesh Patel, Hindu Shree C T, Jayanth S, Keerti D Kulkarni, Pushpa Mala S
Dept. of Electronics and Communication Engineering
Dayananda Sagar University, Bangalore, India
patelharsha2000@gmail.com, hindushreect@gmail.com, jayanth21099@gmail.com,
keertid130899@gmail.com, pushpasiddaraju@gmail.com

2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT) | 978-1-6654-2849-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/CONECCT52877.2021.9622607

***Abstract*— Numeral recognition from images of documents has drawn much more attention due to its vast applications involving processing bank cheque leaf, automatic number-plate recognition, identification of ID cards and zip codes. In a computer vision system, numeral recognition is a complex task since the numbers are not written or scripted accurately, as they differ in shape or size, due to which, feature extraction and segmentation of handwritten numerical script is very hard. The aim of the proposed methodology is to build and develop an application such that the model is trained using our own dataset. This ML model can recognize handwritten digits (0 to 9) either through camera or predict the digits drawn on the screen in real time. In this proposed work, optimized python codes are scripted using technologies such as Machine learning, python and its libraries like Scikit-learn, Open CV as well as SVM to develop a training and testing model to produce optimum results.**

Keywords— Digit recognition, Machine learning, Python libraries, SVM, Real time predictions.

I. INTRODUCTION

The challenges faced by numeral recognition methods have been considered extensively by the engineers and data analysts since the past few years and a number of pre-processing methods including the modern classification algorithms are being developed. However, the recognition of handwritten digits remains a challenge for all of us. The biggest trouble in predicting the digits is due to the disparity in size, shape, and misshaping of the numeral image because the handwritten digits are being written by various users whose writing style entirely differs from one individual to another. Building a database including all the standard samples of the dissipated digits is not an easy task. Manual digit recognition implies a system being able to differentiate and identify digits from any paper documents or any user inputs like touch-screens, laptops or computer systems, number plates of automobiles, digital entries in hand filled forms and other devices in real world scenarios.

Machine learning plays a vital role in artificial intelligence and computer technology. Numeral recognition has obtained a fair amount of reputation from the very birth of machine learning to a specialist who have been working on ML for many years. Building such a system requires correct knowledge of categorization of digits to rightly classify between various digits which can be only feasible with conventional training and testing. To achieve greater accomplishment in the area of numeral recognition, deep learning is rapid-progressing including other machine learning techniques. Nevertheless, CNN (Convolutional Neural Network) models are exorbitant to work on and resources get wasted when being used with less burdensome research problems. The total partitioning time can be reduced by reducing the total feature space used to train the model to find the appropriate handwritten digit classification model. This study is unique from other existing works as it discloses the efficiency of trained machine learning model in relation to the most accurate predictions of handwritten digits being classified. Through extensive literature research, we are aware of the limitations of the various reported models of CNN, as mentioned below. Therefore, we proposed an ML-based and python-based framework that uses SVM (Support Vector Machine) to train and evaluate data and achieved better performance levels compared to other CNN based methods.

II. RELATED WORK

Many researchers have effectively contributed in the field of hand written digit recognition. Haider A Alwzwazy et.al. [1] implemented convolutional neural network for classification of Arabic handwritten digits which consists of more than 45,000 samples. In this work, training and testing consumed less time, it was challenging with a large dataset being used. The model was used for real time applications and had superior accuracy. The limitations of this work is that the model can recognize only Arabic handwritten digits. Pawan Kumar Singh et.al. [2] established the model of invariant handwritten digit recognition for identifying digits written in five popular scripts of Indian subcontinent, namely, Indo-Arabic, Bangla, Devanagari, Roman, and Telugu. Was tested for the same. MLP (Multi-layer Perceptron) classifier was used which helped the model to attain high accuracy of 99.5%. But the model failed as Indo-Arabic numerals ‘2’ and ‘3’ were not recognized properly and the model parameters were not selected properly.

Kh Tohidul Islam et.al [3] implemented a multi-layer fully connected neural network with one hidden layer for handwritten digit recognition which used MNIST database. They were successful in extracting 28,000 digit images for training and 14,000 digits images for performing the test. The model achieved an accuracy of 99.60% with test performance. But the limitation is that the parameters of ANN are not optimized to obtain higher accuracies with low implementation time. Saqib Ali et.al. [4] employed convolutional neural network as classifier, incorporating deeplearning4j (DL4J) framework. The system successfully imparts average accuracy up to 99.21% which is higher than formerly proposed similar schemes. It also involves reduced computational time significantly for training and testing due to which the algorithm becomes efficient. However, more errors were encountered for some digits as small sized dataset was used.

Anchit Shrivastava et.al. [5] conducted a review on different methods being used for handwritten digit recognition and the results were observed and analyzed. The

978-1-6654-2849-1/21/$31.00 ©2021 IEEE

existing methods and techniques for handwritten digit recognition were reviewed and understood to analyze the most suitable and best method for digit recognition. Research was done to find the classifier ensemble with the highest accuracy. The review concludes that the classifier ensemble has high computational time which is a drawback. Further, Rohan Sethi and Ila Kaushik [6] proposed a system in which the vertical and horizontal projections methods are used for the purpose of segmentation. SVM is applied for recognition and classification, while Convex hull algorithm is applied for feature extraction in this work. Euclidean distance was used for the computation of best fit supervisory signal to the input vector. However, no rapid computation was done to decrease time and to increase efficiency and better results were not obtained. Hence, the proposed methodology is an attempt to achieve higher accuracy and to overcome the limitations of the existing techniques using ML and python-based frameworks.

III. METHODOLOGY

The proposed methodology is based on program driven and result oriented approaches. Technologies such as Machine learning, python and its libraries like Scikit-learn, Open CV (camera), pyscreenshot, os, PIL, Tkinter and joblib combined with the myriad functionalities of Python along with NumPy, Pandas are being used. Supervised learning technique, SVM is being used to develop the training and testing model to produce optimum results with high accuracy and least error.

This system receives a dataset containing images of handwritten digits, the model is trained to recognize and classify the handwritten digits (0 to 9) from the dataset. The image file is converted into a *.csv* file and this file is being split into train and test data under supervised learning. Later the model is fit on the training data to make predictions on it. Once training the model is completed, the trained model is saved and the saved model is used to create an executable file. After creating an executable file, an interactive software is developed which is a GUI(Graphical user interface) in which an interactive window is built where the digit (from 0-9) is drawn using a paint like tool. In addition to this, the model can also recognize any digit from 0-9 given as by the user as input through camera to get the real time predictions. The steps involved can be summarised as follows.

Steps:

- Python scripts are written and a dataset is being created using python libraries like pyscreenshot and open cv.
- The dataset from the images being created is converted to *.csv* files and the ML model is built using SVM and python libraries like pandas, numpy and open cv.
- The trained ML model is saved using python library joblib.
- A resource folder is created where the ML model is saved and a python code is written to build the tkinter window for giving user inputs in the form of images captured through camera, saving that image in the resource folder and predicting the output using the saved ML model.
- Pyinstaller module is used to create an executable file for the python code and testing of the executable file is done.
- An installer for the software is made using "inno setup compiler" and the app is installed from the installer.
- The app is tested and then predictions for the handwritten digits are made on ML model using the app. The flow diagram of our proposed work is shown in Fig.1.

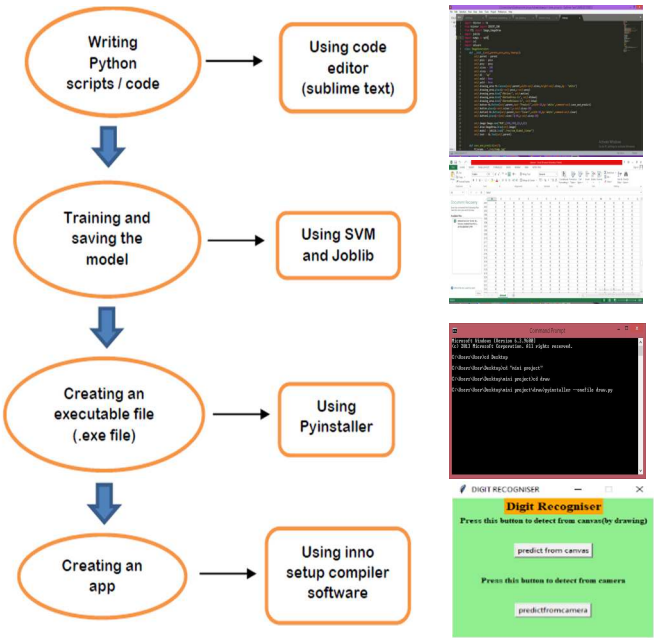

Figure 1. Proposed Model

The proposed work is divided into 4 stages: Building the dataset, data preparation, developing the ML model & prediction and App. These are discussed in detail.

A. Building the dataset

In the first stage, dataset of the handwritten digits from 0-9 are self-created. Data can be added to the dataset and updated whenever required and customized, so as to increase the accuracy of the model. For this, a folder called Original Images folder is created in which subfolders of each digits from 0 – 9 are made and the screenshots captured are saved in these folders using python libraries pyscreenshot, OS and Open CV. As example, samples of captured images of digit '0' and '1' stored in original images folder are shown in Fig.2.

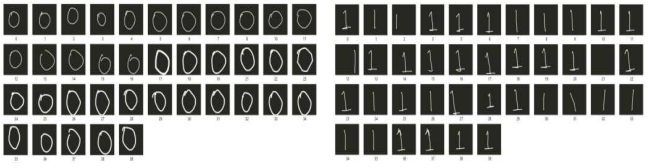

Figure 2. Samples of Captured Images of Digit '0' and '1'stored in Original Images Folder

The Py screenshot module is being installed first, and the python script is coded and saved as screen capture.py. By using a canvas app like paint, approximately 40 images are drawn on the screen for each digit. Further by using screen capture.py script, these images have been captured from top left corner of a screen and are saved in the Original Images folder.

B. *Data preparation*

This is the second phase and the process involves data collection, cleaning, and consolidation into a single file or data table that is used for analysis (Fig.3).

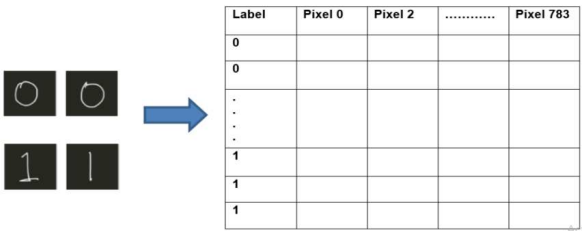

Figure 3. Data Preparation

The created / captured images are converted to *.csv* files which are used to train the ML model. All the images are iterated through using the *for* loop from the Python script which is saved in the Original Images folder to create the *.csv* File (Fig.4.) that is stored in the csv folder. Each image from original images folder is converted to a 28 X 28 pixel size by flattening them out.

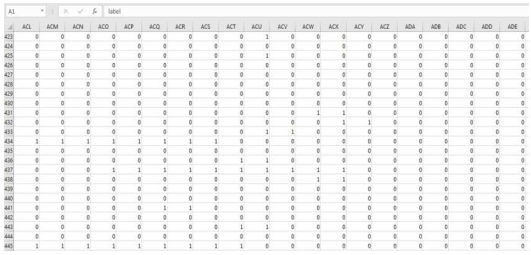

Figure 4. *csv* File

C. *Building Machine Learning model*

In this phase (Fig.5.), a ML model using support vector machine's Support Vector Classifier (SVC) is being built. Support vector machine is a part of scikit learn library. Sklearn is imported along with SVC, which is a part of SVM. The *.csv* file of our dataset consists of a column named 'label' with digits and subsequent columns comprises of pixel values of images. We fit the model with 'x' as the first column, i.e. labels and 'y' as the remaining columns, i.e. pixel values of corresponding images. We split the whole dataset into train and test set, using x_train and y_train variables to store the labels and pixel values. To achieve this, the trained model is saved using joblib library and is used in the software. Images of data from Original Images Folder that we created using pyscreenshot module are fed to the SVC to train it. Data is retrieved from *.csv* File with separated labels and features. Training and testing data are split apart and it is made sure that correct feature or label combination is available during training.

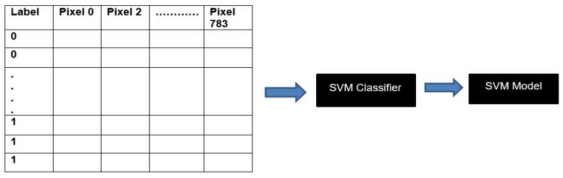

Figure 5. Building the ML Model

SVM model Parameters considered here:
SVM with a linear kernel is applied to classify our data. The regularization parameter, C, is set at 2.

D. *Prediction phase and App*

This stage (Fig.6.) involves prediction of the digit with the software and is adopted in three scenarios.

Firstly, when the software is run, the camera opens up and starts capturing images with minimum delay. This captured image is converted into an array of pixel values. These values will be fit into the trained model to predict the digit. The captured images are converted into pixels to get the vector representation of that image which could be identified by the SVM. Then, every grayscale image is converted to black and white images. We use a presser threshold to get the values corresponding to the handwritten digit pixels using Open CV library. Thus, the converted image will be fed into the trained model to get the predictions.

Secondly, the software can be used without the camera for recognition of the digit in real time, wherein the digit is drawn on the screen using canvas on tkinter GUI. We import tkinter and create a window which includes a canvas with the option for the user can to draw the digit. The digit drawn on the canvas will be captured, saved and fed into our model with a click on 'predict' button. Here, the digit written on canvas will be detected based on pixel values.

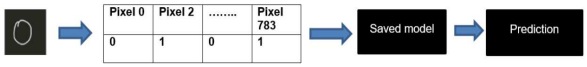

Figure 6. Prediction of Digit

Thirdly, for detection of a handwritten digit by taking the photo of the digit written or using image data and saving it. The location of the saved image is fed into the pop-up box that will appear when the user clicks on 'predict from image' button. We use OS library to feed that image into the model.

IV. RESULTS AND DISCUSSION

Support Vector Machines (SVM) is being used in our experiments for the classification. After building the dataset, preparing and pre-processing data, building ML model using machine learning, python and its libraries, an app is developed for the prediction of the digit drawn on the screen or shown through the camera. An Intel (R) Core(TM) i5-8250U CPU @ 1.80 GHz with an Installed memory (RAM) of 8.00 GB, 64-bit Operating system , x64-based processor and Windows 10 has been used for the proposed work.

The App class was first created and adopted for building the GUI. Any digit from 0 – 9 is drawn and or by capturing the mouse event with a button, the predict_digit()

function is triggered and the corresponding digit is predicted accurately. The real time prediction of digits drawn on the canvas are shown in Fig 7. For the prediction of digits through camera, a new file is first created in an interactive window which is built using Tkinter library that comes in the Python standard library. Then a function called 'predict_digit()'is being created that takes the image as input and then uses the trained model to predict the digit.

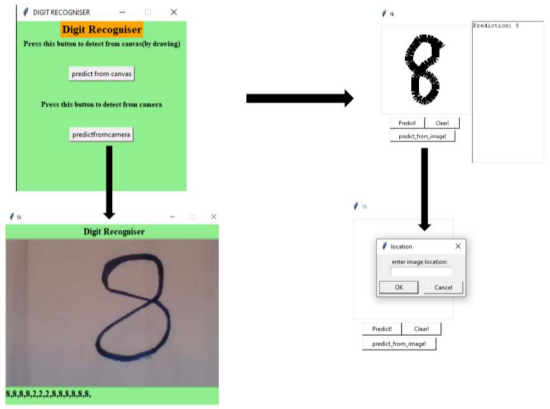

Figure 7. Procedural Flow of App

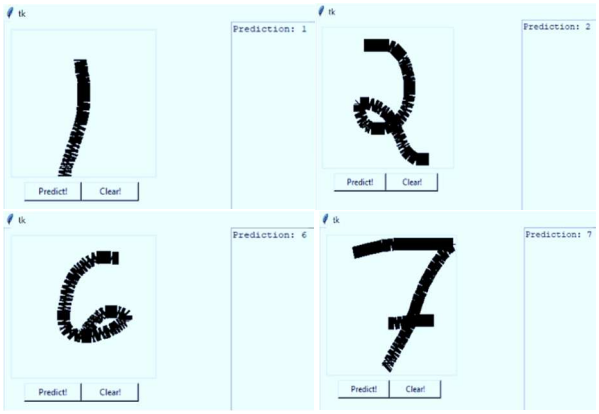

Figure 8. Real Time Prediction of Digits Drawn on Canvas in the App

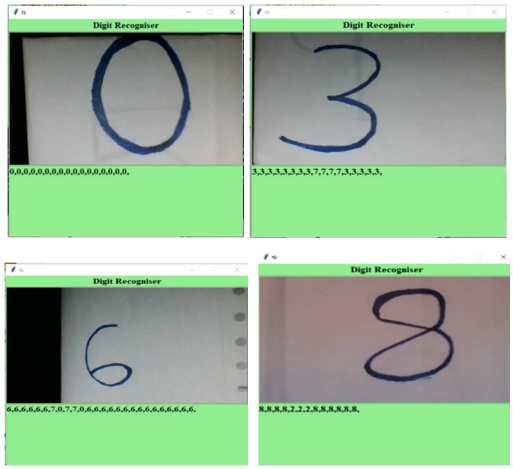

Figure 9. Real Time Prediction of Digits through Camera

Open CV camera is being used to capture real time images and predict a digit from the image. An executable file is created using pyinstaller for the python code created by using the command: "pyinstaller –onefile filename.extention". An installer for our software is being made using "inno setup compiler". The app is then installed from the installer and is tested. Predictions of the handwritten digits are made on ML model using the App. Fig.8 depicts the results of the real time predictions made by the digit recognition app through camera.

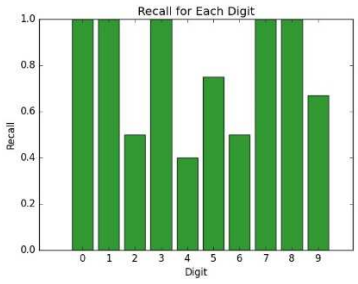

Figure 10. Recall for Each Digit

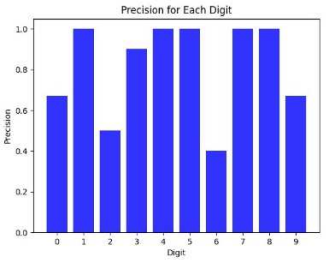

Figure 11. Precision for Each Digit

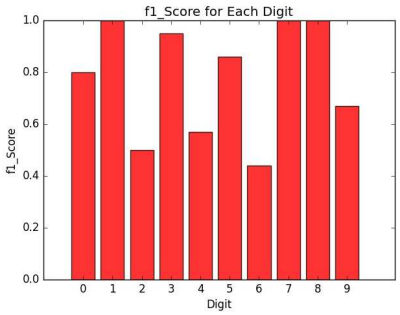

Figure 12. f1_Score for Each Digit

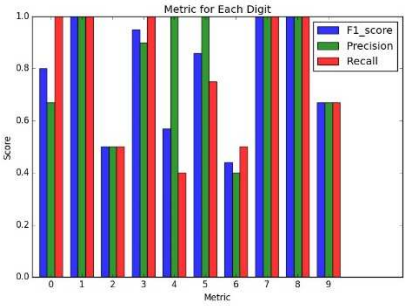

Figure 13. Metric for Each Digit

Fig.10. represents the Recall values for each digit from 0 – 9. It is observed that the Recall value obtained range from 0.4 to 1. Fig. 11.represents the precision value for each digits from 0 – 9. The precision value obtained ranges from 0.4 to 1. Further, it is observed that the f1_Score

(Fig.12.) ranges from 0.44 to 1 for each digit from 0 – 9. Fig. 13. depicts the consolidated graphical representation for f1_score, precision and recall respectively for each of the digits 0-9. The performance is summarized in Table.I.

TABLE I. PERFORMANCE SUMMARY

Digits	Precision	Recall	f1_Score
0	0.67	1	0.8
1	1	1	1
2	0.5	0.5	0.5
3	0.9	1	0.95
4	1	0.4	0.57
5	1	0.75	0.86
6	0.4	0.5	0.44
7	1	1	1
8	1	1	1
9	0.67	0.67	0.67

The Application performance is depicted in the Table II. It is observed that as the user taps on the app icon, the first screen experiences a delay of 1 – 2 seconds. The digit recognition app does not use more resources than required and thus does not consume high amount of battery life. The app consumes 53 MB of phone memory and on implementing certain functionalities does not use additional memory.

TABLE II. SYSTEM PERFORMANCE

Name	CPU	Memory	Disk
Digit Recognition App	0.4%	53.1	0Mb/s

V. CONCLUSION

There is an immense requirement for machine learning technologies in today's world. This would be an understatement as artificial intelligence and ML are slowly taking over even the conventional roles that were thought to be solely human. In this work we have proposed a numeral recognition model as a contribution to the very basic needs of our human psyche to replace conventional jobs by machines to make all our lives simpler. Throughout the millennia humans have slowly replaced our hands with tools, the tools by machines, or machines with complexed power tools. Therefore, the proposed model would be able to recognize the introduced digit according to the well-made formulations adopting. the values in the datasets. The proposed model has achieved an accuracy of 84%. As a further study, more datasets of digits as well as the alphabets can be added to make this model detect more than one digit in a single frame. Further optimization of the parameters of this model by adopting efficient training models to obtain higher accuracy could be developed.

REFERENCES

[1] Haider A Alwzwazy, Hayder M Albehadili, Younes S Alwan, Naz E Islam, "Handwritten Digit Recognition Using Convolutional Neural Networks", International Journal of Innovative Research in Computer and Communication, Vol 4, Issue 2, February 2016, pp: 3480 – 3485.

[2] Pawan Kumar Singh, Ram Sarkar, Mita Nasipuri, "A Study of Moment Based Features on Handwritten Digit Recognition", Applied Computational Intelligence and Soft Computing, Vol 16, Issue 11, March 2016, pp: 1-17.

[3] Kh Tohidul Islam , Ghulam Mujtaba , Dr. Ram Gopal Raj , Henry Friday Nweke, "Handwritten digits recognition with artificial neural network" International Conference on Engineering Technologies andTechnopreneurship, September 2017.

[4] Saqib Ali, Zeeshan Shaukat, Muhammad Azeem, Zareen Sakhawat, Tariq Mahmood, Khalil ur Rehman,"Classifying Handwritten Digit Recognition Using CNN and PSO", SN Applied Sciences, Vol 8, Issue 8, August 2019, pp: 98-110.

[5] Rohan Sethi, Ila Kaushik, "Hand Written Digit Recognition using Machine Learning", 9th IEEE International Conference on Communication Systems and Network Technologies, April 2020, pp: 49-54.

[6] Hanmandlu M, Murthy, "Fuzzy model based recognition of handwritten numerals", Pattern Recognition Society, Vol 40, Issue 6, June 2007, pp:1840–1854.