Handwritten Digit Recognizer

Kaiwalya Deshpande

Guided by: Abhishek Omkar Prasad

KIIT Robotics Society

School of Computer Engineering, KIIT University,
20051862@kiit.ac.in

Abstract- Optical Character Recognition (OCR) is a subfield of Image Processing which is concerned with extracting text from images or scanned documents. In this project, we have chosen to only focus on recognizing handwritten digits available in the MNIST database. The challenge in this project is to maximize the accuracy of the handwritten digit recognizer using machine learning.

Index Terms- Image Processing, Optical Character Recognition, Handwritten Digits, Machine Learning.

INTRODUCTION

It is easy for the human brain to process images and analyse them. When the eye sees a certain image, the brain can easily segment it and recognize its different elements. The brain automatically goes through that process, which involves not only the analysis of this images, but also the comparison of their different characteristics with what it already knows in order to be able to recognize these elements. There is a field in computer science that tries to do the same thing for machines, which is known as Image Processing.

Image processing is the field that concerns with analysing images so as to extract some useful information from them. This method takes images and converts them into a digital form readable by computers, it applies certain algorithms on them, and results in a better-quality image or with some of their characteristics that could be used in order to extract some important information from them.

Image processing is applied in several areas, especially nowadays, and several software have been developed that use this concept. Now we have self driven cars which can detect other cars and human beings to avoid accidents. Also, some social media applications, like Facebook, can do facial recognition thanks to this technique. Furthermore, some softwares use it in order to recognise the characters in some images, which is the concept of optical character recognition, that we will be discussing and discovering in this project.

One of the narrow fields of image processing is recognizing characters from an image, which is referred to as Optical Character Recognition (OCR). This method is about reading an image containing one or more characters, or reading a scanned text of typed or handwritten characters and be able to recognize them. A lot of research has been done in this field in order to find optimal techniques with a high accuracy and correctness. The most used algorithms that proved a very high performance are machine learning algorithms like Neural Networks and Support Vector Machine.

One of the main applications of OCR is recognizing handwritten characters. In this project, we will focus on building a mechanism that will recognize handwritten digits. We will be reading images containing handwritten digits extracted from the MNIST database and try to recognize which digit is represented by that image.

This approach is based on matrices manipulations, as it reads the images as matrices in which each element is a pixel. It overlaps the image with all the images in the reference set and find the correlation between them in order to be able to determine the digit it represents.

The goal of this project is to apply and manipulate the basic image correlation techniques to build program and keep polishing and enhancing in order to investigate to which extent it can get improved. This would allow us to fit the trained data in a Sequence Model and achieve maximum accuracy in order for the data to be viable and predicatable.

BASIC CONCEPTS/ TECHNOLOGY USED

Our primary objective to build a sustainable predictor of digits we need to have a bunch of data to train our model off of. MNIST dataset is the perfect dataset to offer us a solution for our requirements. The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support-vector machine to get an error rate of 0.8%. An extended dataset similar to MNIST called EMNIST has been published in 2017, which contains 240,000 training images, and 40,000 testing images of handwritten digits and characters. Hence the MNIST dataset serves the purpose sufficiently.

Now for the type of model to be used, we have decided to move forward with Sequential Model. Since it a sequential model is used for cases where we need to input or output large sequences of data such as text streams, audio clips, video clips, images, time-series data, it would be a perfect fit for our requirements. The Sequence Model will use a popular algorithm known as Recurrent Neural Networks Algorithm.

Recurrent Neural Network (RNN) is a Deep learning algorithm and it is a type of Artificial Neural Network architecture that is specialized for processing sequential data. RNNs are mostly used in the field of Natural Language Processing (NLP). RNN maintains internal memory, due to this they are very efficient for machine learning problems that involve sequential data. RNNs are also used in time series predictions as well. The main advantage of using RNNs instead of standard neural networks is that the features are not shared in standard neural networks. Weights are shared across time in RNN. RNNs can remember its previous inputs but Standard Neural Networks are not capable of remembering previous inputs. RNN takes historical information for computation.

Now that our dataset and data model are defined properly. We can construct a proper model out of this which can be used to predict the digit provided by the user. Another thing to keep in mind while fitting our model with dataset is the concept of overfitting/underfitting. Underfitting refers to a situation where our model isn't trained enough to properly predict output. This most often happens when we don't provide our model with enough data to fit. Overfitting is a concept in which our model is fed way more data than required, at this point the model starts to pick up noise and edge cases of the data and tries to find trends which don’t exist resulting in wrong predictions. We don’t need to worry about underfitting since our dataset has over 60,000 data entries, but overfitting on the other hand might be a concern, hence we drop about 15% of the data to avoid overfitting.

PROJECT COMPONENTS

For our Model training we will be using Python3 with Jupyter Notebook. While python doesn’t have the fastest interpreter it most certainly makes up with the access to the vast collection of libraries that it provides. According to engineers coming from academia and industry, deep learning frameworks available with Python APIs, in addition to the scientific packages have made Python incredibly productive and versatile. There has been a lot of evolution in deep learning Python frameworks and it’s rapidly upgrading.

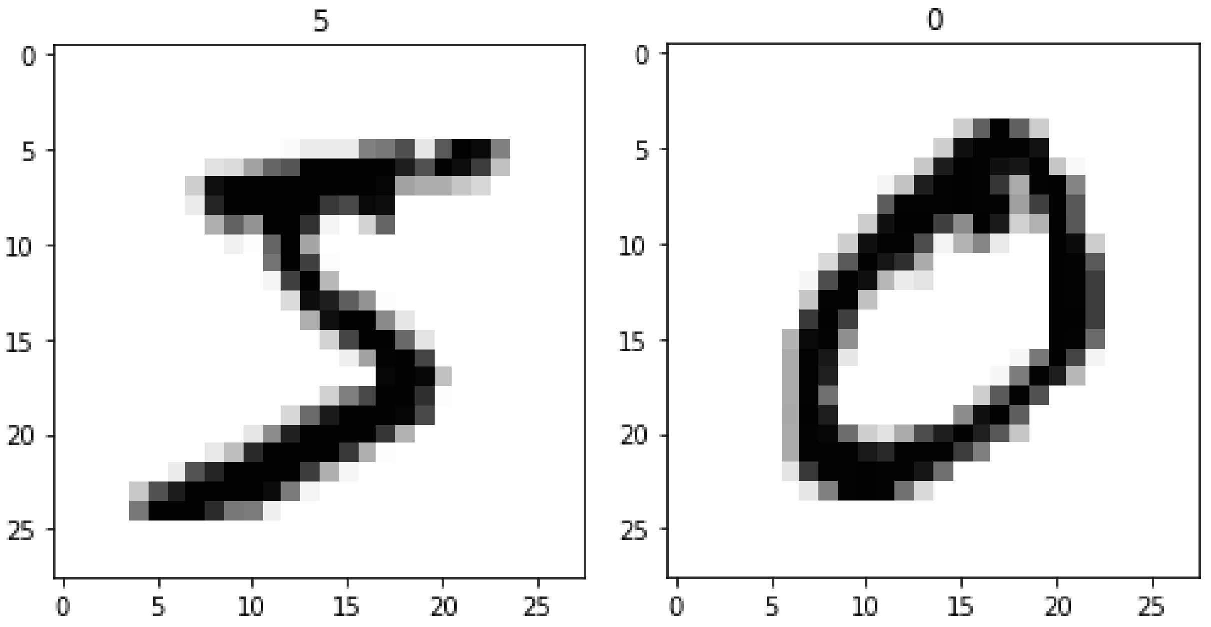
In terms of application areas, ML scientists prefer Python as well. When it comes to areas like building fraud detection algorithms and network security, developers leaned towards Java, while for applications like natural language processing (NLP) and sentiment analysis, developers opted for Python, because it provides large collection of libraries that help to solve complex business problem easily, build strong system and data application. Along with this the Jupyter Notebook, which is an open-source web application allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document.

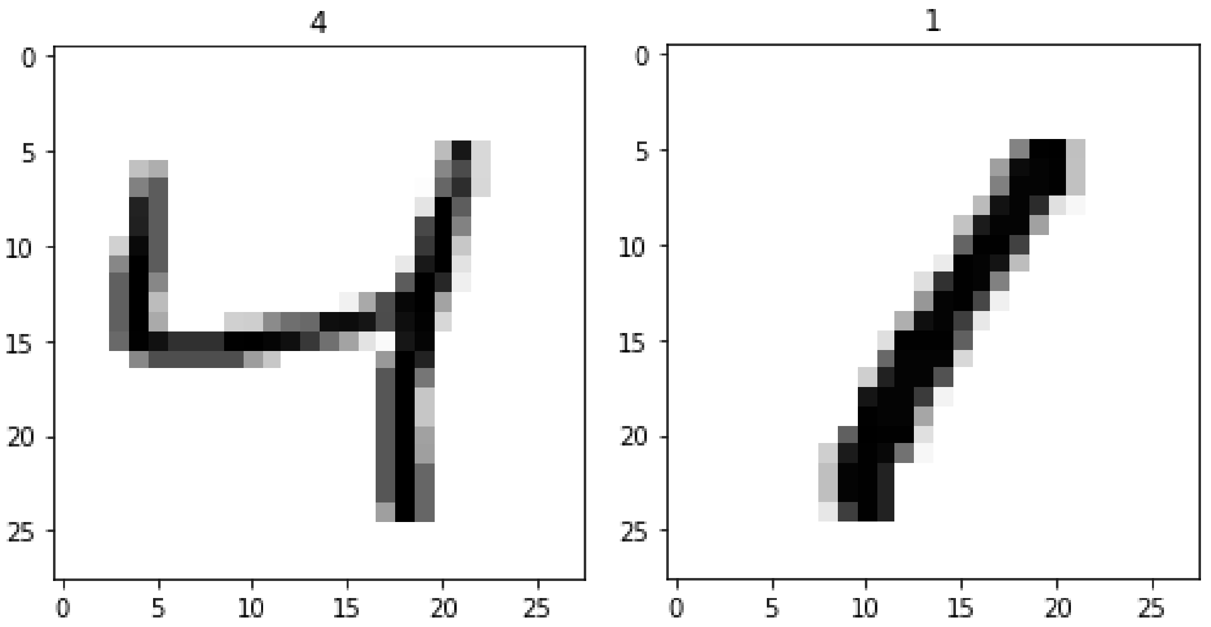
And for our UI we will be using Pygame. While there are many other great modules to be used with/without Python such as Tkinter. I opted for pygame since it is convenient and I have already used it in some of my previous projects so it will be easy for me to use all the functionalities provided by pygame without having to spend unnecessary amount of time going through documentations. Apart from personal choice pygame does end up as a pretty decent option with its ease of use functions.

PROPOSED MODEL / ARCHITECTURE / METHODOLOGY/ MODEL TOOL

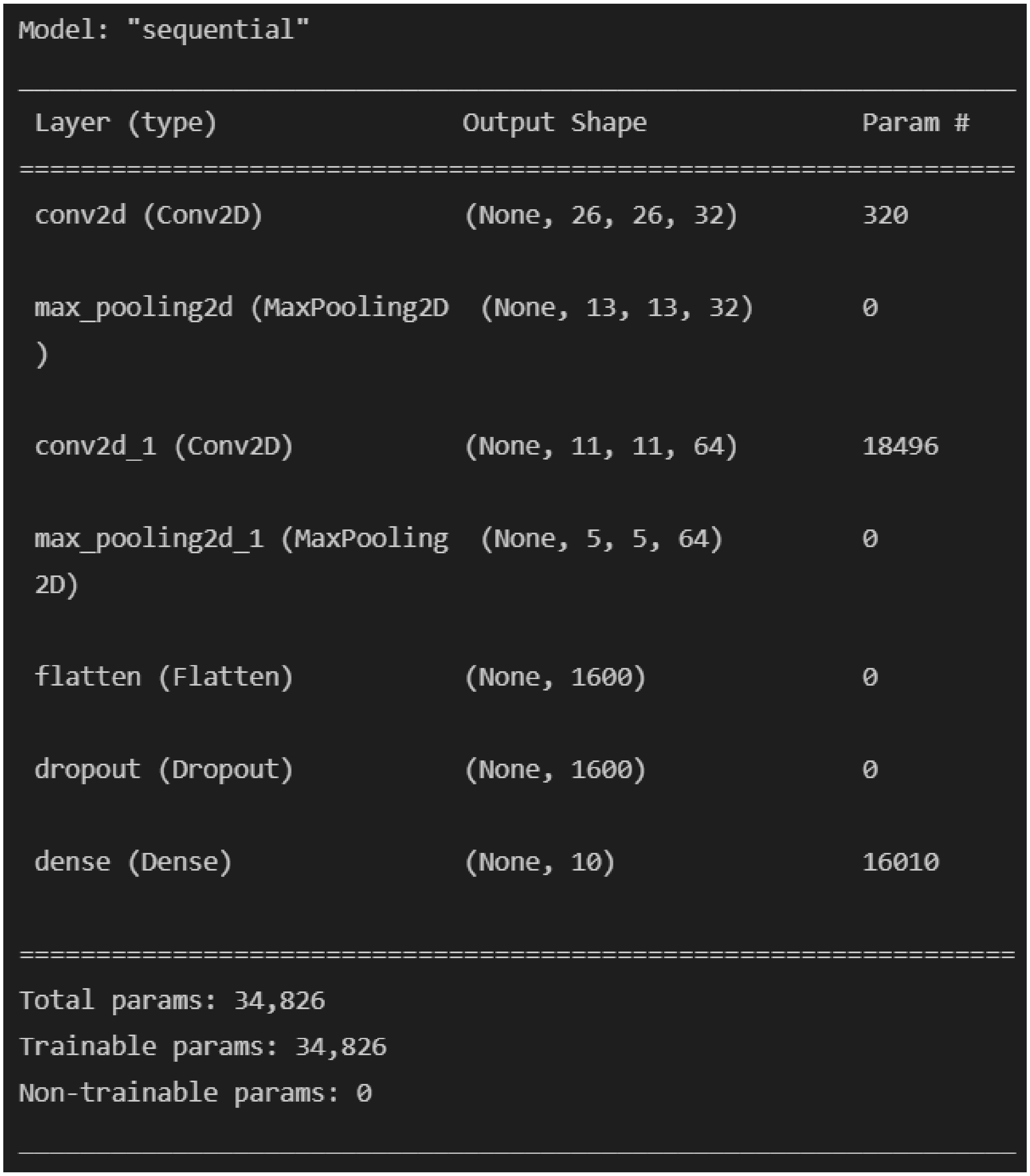
In this project we import the MNIST dataset using Keras library. After importing the data we first get familiarized with the data a little bit. Understand how the dataset stores data and what kind of data awaits us with the dataset etc. Cleaning the data takes up a major amount of time considering that this is the most crucial step of the whole process. Fortunately, the MNIST dataset is already processed and clean and hence it is a pretty neat way to get entry level experience with machine learning.

When we print out the data we get something like this :





With our dataset properly separated in training and testing sets, we can begin with our model training. But before that we will need to standardize the data with us, so we normalize the data as a 32 bit float between 0 to 1 both included and convert out y to one hot vectors. Now we can truly begin with model training, So we initialize a sequential model and add all the other properties to the model that would fit our needs. After the initialization our model should look something like this :



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten (Flatten)           (None, 1600)              0

 dropout (Dropout)           (None, 1600)              0

 dense (Dense)               (None, 10)                16010

=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
_________________________________________________________________
```

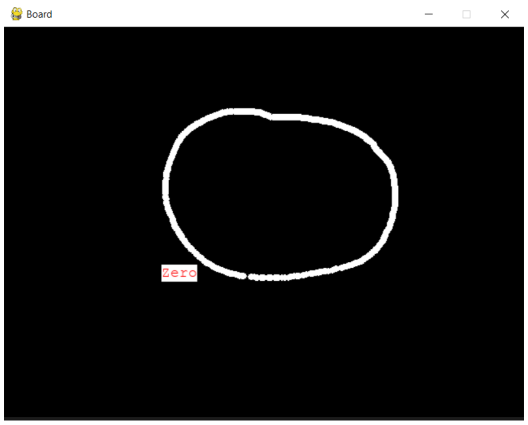
Now the only thing remaining is fitting the data into the model, which although a time-consuming process is handled by the system itself. After the fitting is completed, we compile and save the model externally so that we have easy access to it in future without having to go through the same process every-time the application is run.

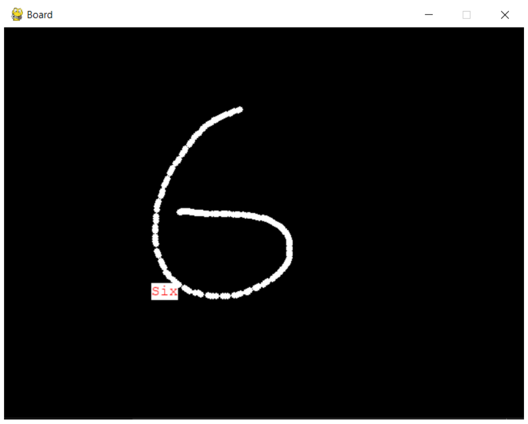
IMPLEMENTATION AND RESULTS

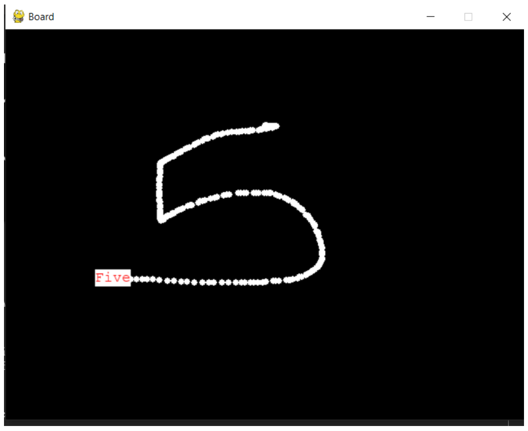
Once we have the model saved, we can check from the test cases that our model is a pretty good representation of the dataset by checking its confusion matrix and classification report.

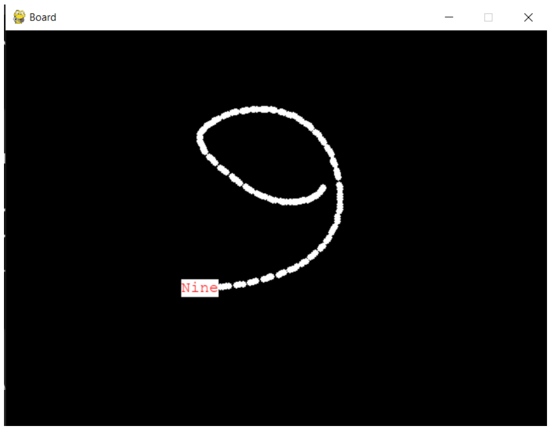
This model can be used in our app developed using pygame to get digit recognized which is received as input from the user. Our app is a pretty basic display with a blank screen where the user can draw any drawing which is later taken as an input image by pygame and reshaped to the desired dimensions being (28,28) and then sent to the model to predict the output and the most probable digit returned from the prediction is again reflected on the screen.

The result vary a little bit per digit but overall accuracy can be described to be pretty accurate:









PERFOMANCE ANALYSIS

The performance of our app is pretty good. In terms of accuracy our model boasts an accuracy count of 0.9907 with the loss of only 0.0810.

SOCIETAL IMPACT AND FUTURE SCOPE

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.

For example, thieves or other people with malicious incentives, could use this technique in order to get some important codes and keys of important ciphers, so as to serve political or personal needs. These kinds of images can also be extracted from social media accounts or from phones. Nowadays, with the increase of technological advancements, the use of social media has become very popular, and people started sharing important information through private direct messages, which could also include these types of codes and ciphers. If someone is spying on these people or could penetrate their private accounts in a way or another, they could get them and extract the information they need in order to serve their personal motives. In addition to that, recognizing handwritten digits could be used in a bad way so as to break the captcha code. This code is used in order to make sure that the entity who is accessing a certain website or is trying to get a benefit from a certain service is a human being and not a computer. The captcha is usually put as an image to make it hard for computers to read it, which will make sure that only human beings can be granted access. However, if digits recognition is used in this case by computers, then they would be able to appear as humans.

Furthermore, these issues could increase the lack of trust in societies, therefore, affect the social behaviour of people. As a consequence of the bad uses and applications of this technique, people in different societies could become more suspicious and doubtful of everyone around them. If people notice that this kind of methods could be used in spying and theft, then there will no longer be trust inside societies and the problem of wariness will end up being elevated. This problem will affect the behaviours of these people towards each other.

Yet, all of the consequences of this technology aren’t negative. We get many quality-of-life changes to improve somethings as well. Such as the handwriting keyboards being available in mobile and tablets and applications such as google lens make our life just that bit easier. It gets easier to edit, read, search online with handwritten text and much more.

This also lets methods of using pen and paper to write stuff down not go obsolete, with the rise in popularity of typing we can see a steady decline in the use of traditional writing methods. While which of the two is better is a subjective argument it can still be said that we are seeing more and more of the documents typed rather than written, the introduction of online mode of conducting classes and other meetings certainly didn’t help the situation. Thanks to the handwriting recognition anyone who is still comfortable with writing the documents has the luxury of sharing the document in any desired format, be it handwritten or typed without the increase of manual labour.

CONCLUSION

In conclusion I can say that the handwritten digit recognizer is a success. The primary motive of the project was to get acquainted with machine learning technology practically applied in the day to day lives with the potential of future expansion. With the given possible application this app has it won't be a point of argument to say that the project is definitely a part of practical application of Machine Learning and has a rising usage ahead of it. The project itself is satisfying its purpose with a great deal of accuracy and hence can be properly presented to the audience as a successful means of predicting the digits.

There are still many shortcomings and a huge room for improvement in this project, such as its difficulty in recognizing the digit 1 with a good success or broadening the scope with the inclusion of English alphabets and special characters to begin with.

REFERENCES

[1] http://www.aui.ma/sse-capstone-repository/pdf/spring-018/HANDWRITTEN%20DIGITS%20RECOGNITION.pdf

[2] https://data-flair.training/blogs/python-deep-learning-project-handwritten-digit-recognition/

[3] https://www.youtube.com/watch?v=L2cAjgc1-bo

[4] https://en.wikipedia.org/wiki/MNIST_database