Handwritten Text Detection using Optical character Recognition model

Hari Om Swarup S A, Vennalla V

Department of Computer Science Engineering, PES University Bangalore, India, swarupsaofficial@gmail.com, vennillareddy@gmail.com

Mentor: Ms Divya Prabha K N

Assistant Professor, Department of Computer Science Engineering, PES University, Bangalore, India, divyaprabha@pes.edu

Abstract **- The goal of this project is to develop an interface for capturing and translating handwritten text using optical recognition (OCR). This technology is commonly used for machine learning and pattern recognition. This project could be achieved by using python programming language.**

Keywords – Machine Learning, Computer Vision, Artificial Intelligence.

I. INTRODUCTION

Text is very important in our daily essential life. Text can be found on sign boards, bank statements, on vehicle registration number and many more. Detecting the text in an image and reading is required in this fast-developing world. Such as converting handwritten notes on device to text. Scanning vehicle number on toll gates, etc. This is possible through Optical character Recognition (OCR).

Although optical character recognition can be challenging to accurately identify text in different handwriting styles. We are trying to minimize the problem by training the datasets and through Convolutional Neural Network.

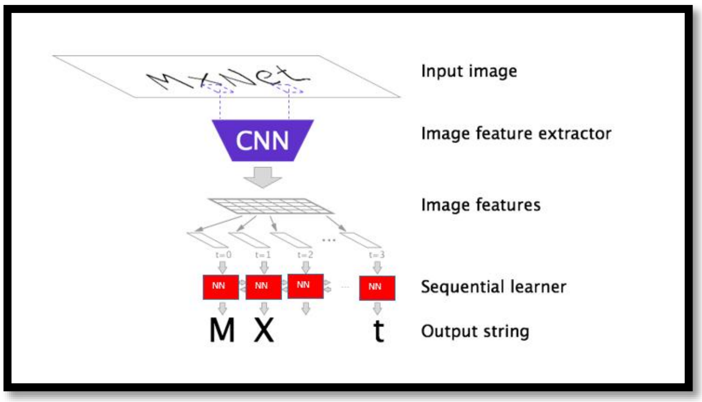


Figure 1: Analogy of Optical character Recognition Model

II. CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Network is a class of deep neural algorithms often used to analyze visual imagery. A convolutional neural network (CNN) receives an image, extracts features using filters, and mainly for image processing, classification, and segmentation.

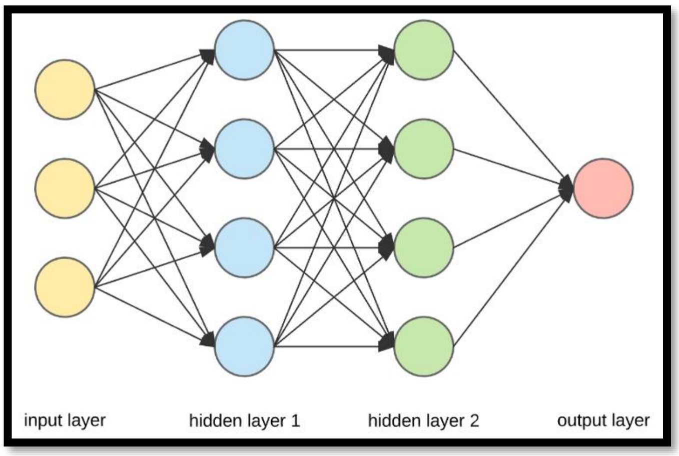


Figure 2: Convolutional Neural Network Diagram

The filters used in CNN are described here below.

1. **Kernel** - The kernel is a programming algorithm that is used to extract the features from images. It does so by moving the data over the input region and performing the dot product with its sub-region.

2. **Pooling** - The pooling layers are used to consolidate the features learned from the convolutional layer output. They are ideal for minimizing the number of learning parameters.

3. **Stride** - Stride is the number of pixels shifts over the input matrix. The kernel moves the pixel on image depending on the value of strides.

4. **Padding** - Padding is an integral part of the image processing process. It adds a layer of pixels to an image.

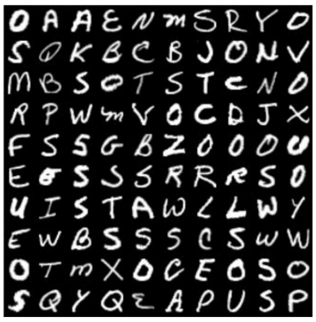
5. **Flattening** - flattens the multidimensional data into a 1-dimensional array, where the next layer is the data.

II. TRAINING MODEL

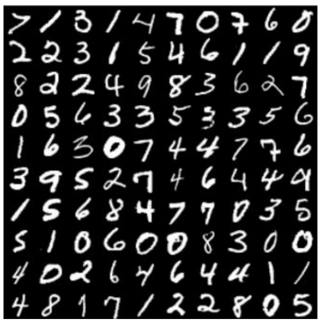
A. Loading Data

To train a model, datasets are needed to train in order to classify images; the first step in the process is to load data.

Datasets have been chosen from the Kaggle website, which contains all the capital alphabets starting from A – Z as in Figure 3.a and python's TensorFlow module, which contains numbers ranging from 0 – 9 as in Figure 3.b where each character and number contains about 1000 images each.



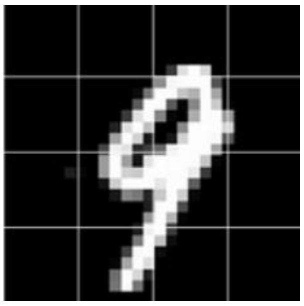
a)



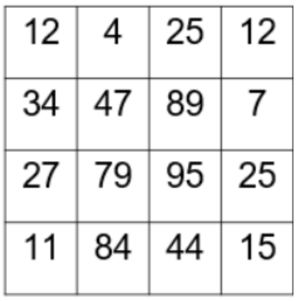
b)

Figure 3: a) Kaggle A – Z Dataset b) MNIST 0 – 9 Dataset

Image is a multidimensional array that consist of small pixels where each pixel value ranges from 0-255. And the label along the Image represents the character or number it is. Then the array of values is appended in a list.



a)



12	4	25	12
34	47	89	7
27	79	95	25
11	84	44	15

b)

Figure 4: a) A Sample Image of Number 9
b) Image to Number representation in Array

B. Splitting Data

To predict images, Dataset has been divided into the Training dataset, which is used to fit the machine learning model, and testing dataset, which is used to evaluate the fit of the machine learning model. In our project, we have considered 80% of datasets in the training dataset and the rest 20% for testing.

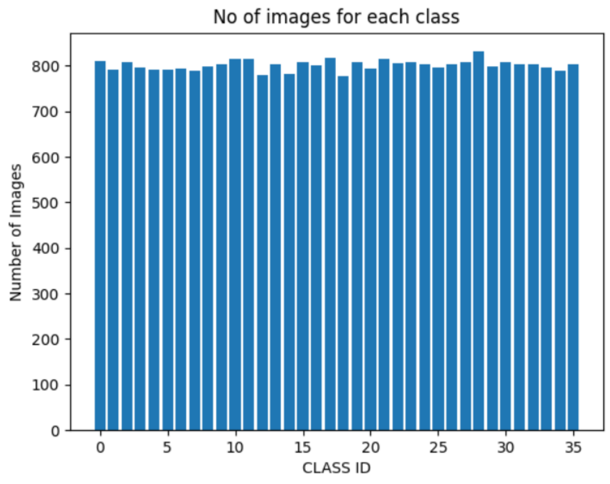
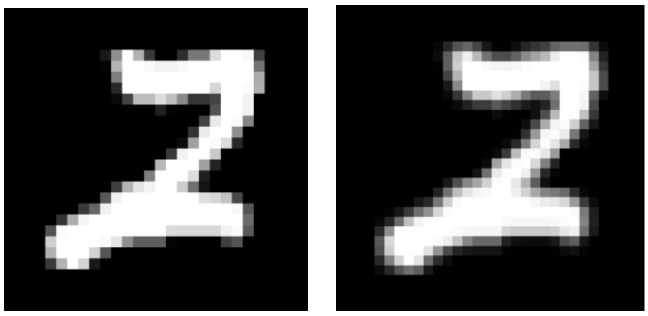


Figure 5: Distribution of images of each Alphabet and Numbers in Training set.

0 -26 Class ID being A -Z Alphabets and 27 – 35 Class ID being 0 – 9 Numbers Respectively

C. Image Improvisation

To achieve better accuracy in the training model, improvisation includes resizing the image from 28*28 pixels into 32*32 pixels. Processing the image through histogram equalization, which is used to improvise the image's contrast, makes the dark portion darker and the brighter portion brighter.



a) b)

Figure 6: a) Letter Z in 28 * 28 pixels before transformation b) Letter Z in 32 * 32 pixels after transformation

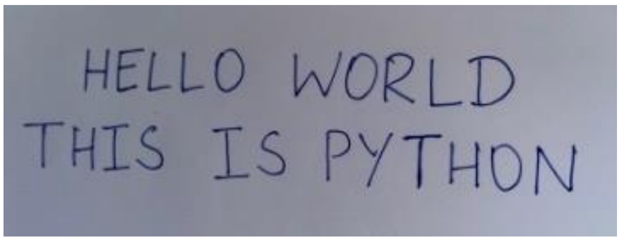
D. Model Training

The last step is to train the model. So, the images are processed into CNN in batches of images. This process continues until all images in the training dataset have been completed. This process is done multiple times to get more accuracy in the trained model. After the training process, it is saved in a file, which can predict the characters in the image.

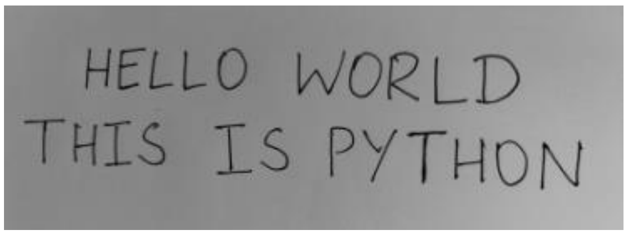
III. PREDICTING IMAGES

A. Fetching Images and processing

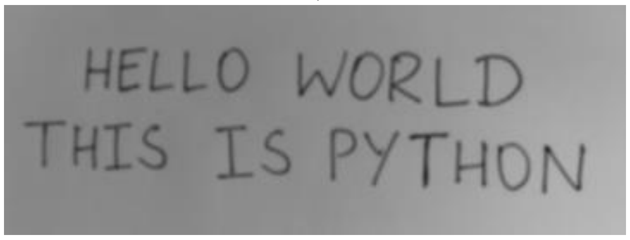
Images can be added by camera or scanner. Image files that work are '.png', '.jpeg' or '.jpg'. The image is converted from RGB (Red, Blue, Green) to Gray Scale. Then the image is processed through Gaussian blur, a filter to blur images. Then processed through canny to detect edges. As shown in figures 7.



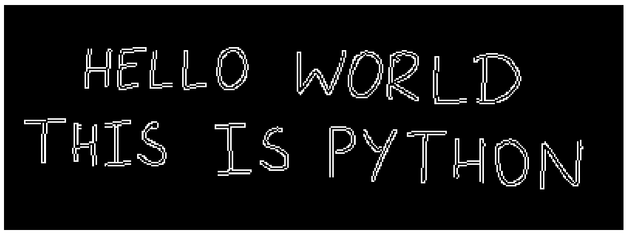
a)



b)



c)



d)

Figure 7:

a) Image input in RGB
b) Image from RGB to Gray Scale
c) Blurred Image
d) Image edges

B. Detecting Letters in the Image

When an image is processed through canny to detect edges, we also find the area where characters are present in the image. Then the selected area is cropped, and the trained model predicts the character present in the cropped image. The one which matches with a higher probability that the letter is predicted and the letter is printed.

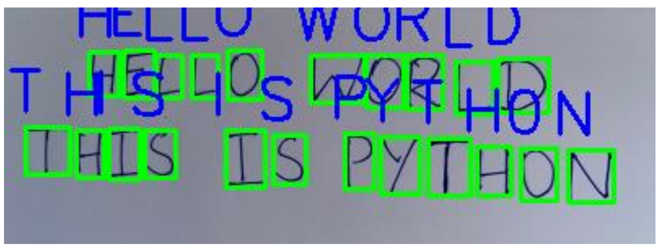


Figure 8: Predicted letters in the image

IV. PYTHON MODULES

1. **OpenCV** – An open-source module used to solve computer vision problems such as understanding and analyzing digital images by the computer and processing the images.

2. **NumPy** – A module to store a list of numbers maintained in minimal memory. Working with NumPy also includes easy-to-use functions for mathematical computation on the array data set.

3. **TensorFlow** – This module is a part of the Machine Learning Library. It provides a framework for developing deep neural network-based applications.

4. **Imutils** – Module used for image processing functional package to do image rotation, translation, blur amount detection, resizing, etc.

V. SCOPE

This technology is beneficial since it saves time without retyping documents. It is a standard method of digitalizing handwritten texts to be electronically edited. It is used in Identifying Number Plates for vehicles, Samsung S pen, Amazon Textract, and many more in a real-time application.

IV. CONCLUSION

In this paper, we presented a system that can recognize the characters in an image using machine learning and a Convolutional Neural network algorithm to train various alphabet letters and numbers models. Though the method of training model might be easy, getting higher accuracy is difficult. In order to achieve higher accuracy, many datasets are needed with multiple iterations in training the images. We hope to improve the project by adding more datasets with small letter alphabets in the next project.

VII. REFERENCES

[1] STN-OCR: A single Neural Network for Text Detection and Text Recognition by Christian Bartz Haojin Yang Christoph Meinel Hasso Plattner Institute, University of Potsdam Prof.-Dr.-Helmert Strabe 2-3 14482 Potsdam, Germany -2017

[2] OCR: Handwriting recognition with OpenCV, Keras, and TensorFlow - PyImageSearch

[3] Convolutional Neural Network - Javatpoint