Air Canvas with Handwriting Recognition using Computer-Vision and Deep Learning

RITURAJ SAURABH
M.C.A. III YEAR
Department of Computer Science
Pondicherry University
Pondicherry, India, 605014
saurabhrituraj04@gmail.com

T. SIVAKUMAR
Assistant Professor
Department of Computer Science
Pondicherry University
Pondicherry, India, 605014
tsivakumar@yahoo.com

Abstract**-- Handwriting recognition is one of the emerging fields in the computer vision and Artificial Intelligence domain. The main abilities of human are they can recognize any object or thing just by looking at it. The handwritten text can also be easily identified by a human, but not by a system until and unless it has been trained to do so. Different languages have different patterns within them to spot, and It's very difficult for the system to spot the text. During the text recognition, what we do is, we process the input image, extract the features, and classify the schema, train the system to acknowledge the text. My proposed plan is to develop an Air Canvas which can draw anything on it (text to be specific), by capturing the movement of fingertips in the air in front of a camera, and finally detect the text written by the user using his/her fingertips and convert it into digital text. My project is going to have mainly three modules: i) Air Canvas, ii) HTR System, iii) Web Application Development (using Python-Flask framework). For Air Canvas I'll be using Computer-Vision technology with the help of python's Open-CV library. For the HTR System, I'll be using Google-Cloud's Vision API.**

Keywords**—Python, Open-CV, NumPy, Flask, Google Cloud Vision API.**

I. INTRODUCTION

Have you ever wanted to draw your imagination by just waving your finger in the air? In this project, we are going to build an Air Canvas which can draw anything on it by just capturing the motion of a coloured marker with a camera. Here a coloured object at the tip of the finger is used as the marker. We will be using the computer vision techniques of **OpenCV** to build this project. The preferred language is Python due to its exhaustive libraries and easy to use syntax. Here Colour Detection and tracking are used in to achieve the objective. The colour marker is detected and a mask is produced. It includes the further steps of morphological operations on the mask produced which are Erosion and Dilation. Erosion reduces the impurities present in the mask and dilation further restore the eroded main mask. This is how the Air-Canvas module will be implemented.

The above Air-Canvas module will give an image output which will have the text written by the user in air. Our next and the ultimate goal of the project is to recognise the text written and convert it into digital (machine) text. And we are going to achieve this goal by the help of google-cloud's Vision API which is used for various needs and text detection and recognition is one of them.

The second module of the project i.e., Handwritten Text Recognition System will recognize the text written by the user and give the equivalent digital text as output.

The whole project will be bound together as a web application using Python-Flask framework. On executing the project, it will render a template having a button on it. After clicking the button OpenCV frames will be displayed tracking frame for object tracking and paint frame for displaying the text written. When user presses the 'S' button on his keyboard the text written on the Paint frame will be saved as an image to his project directory and it will call the HTR System module which will produce the digital text and will render a template showing the text output and will also speak the text using Python's Text-to-speech (pyttsx3) library.

II. RELATED WORK

Handwriting recognition, also known as optical character recognition, is performed for converting the handwritten document into digital form after the writing is completed by a user. The advantage of offline recognition is that it can be done even after many years, at any time after the completion of the document. The disadvantage is that it's not done in real time and thus it's not appropriate for immediate input.

There are numerous applications of offline handwriting recognition such as, reading postal addresses, bank check amounts, and forms etc. Also, OCR plays an important role in allowing image textual information entry for digital libraries, by digitization, image restoration, and other recognition methods.

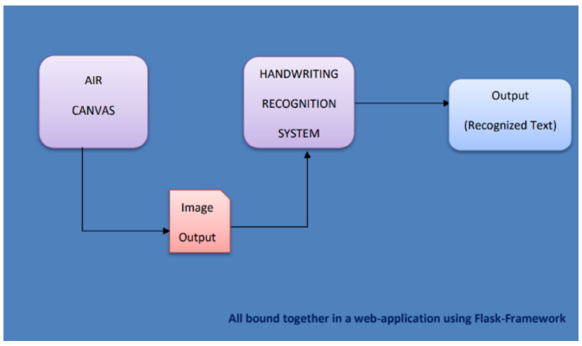
Offline handwriting systems generally go through four processes: acquisition, segmentation, recognition, and postprocessing. First, the handwriting that has to be recognized is digitized by the help of scanners or cameras. Second, segmentation of the image of the document takes place into lines, words, and individual characters. Third, for each character recognition is done using OCR techniques. Finally, errors are corrected using lexicons or spelling checkers and your job is done.

Offline handwriting recognition system's accuracy is lesser than online systems because for offline systems only spatial information is available, while for online systems both spatial and temporal information is available.

Google-Cloud's Vision API is a handwriting recognition vendor using which developers easily integrate vision detection features within applications, which also includes labelling of images, detection of faces and landmarks, optical character recognition (OCR), and also tagging of explicit content.

III. PROPOSED MODEL

The overall structure of the project is going to be a web-application using flask framework in python, which includes two modules, Air canvas and HTR System respectively. It will be clearer from the diagram below, which is the **system design** of the project:


AIR CANVAS
HANDWRITING RECOGNITION SYSTEM
Output (Recognized Text)
Image Output
All bound together in a web-application using Flask-Framework

For the first half of the project i.e., **Air Canvas** which is used to detect the text pattern drawn in air using fingertips (or a blue pointed object) we need to build an application using open-cv and NumPy libraries in python which draws the text on the Paint Window by tracking the movement of the object and gives an image output which is the text written by you on white background i.e., the Paint window.

As stated already, here Colour Detection and tracking are used in order to achieve the objective. The colour marker is detected and a mask is produced. It includes the further steps of morphological operations on the mask produced which are Erosion and Dilation. Erosion reduces the impurities present in the mask and dilation further restores the eroded main mask.

Algorithm for Air Canvas:

1. Start reading the frames and convert the captured frames into HSV colour space (Easy for Colour Detection).
2. Prepare the Canvas frame and Put CLEAR ALL button on it.
3. Find the mask of the coloured marker (blue object).
4. Pre-process the Mask with Morphological operations (Eroding and Dilation).
5. Detect the Contours, Find the centre co-ordinates of largest contour and keep storing them in the array for successive frames (Array for drawing points on canvas).
6. Finally draw the points stored in an array on the frames and Canvas.

After executing this module, it gives the text written on the white background i.e., Paint Window in the form of image output, and also calls the HTR System module which takes the image as input and detects as well as recognizes the text written using Google-cloud Vision API.

The second module i.e., The **HTR System,** as stated already will take the image (Paint Window) as input and with the help of Google cloud's vision API will give the desired output which will then be returned to the flask application and it will render the result template along with the text output.

Now let's talk about **Google Cloud's Vision API**. So basically, Google released the API to help people, industry, and researchers to use their functionalities. Google Cloud's Vision API has powerful *machine learning models* pre-trained through REST and RPC APIs. You will be able to detect objects and faces, read printed or handwritten text, and integrate useful metadata into your image catalogue.

The Google Cloud Vision API enables developers to create vision-based machine learning applications based on object detection, OCR, etc. without having any actual background in machine learning.

The Google Cloud Vision API comprises extremely complex machine learning models used for image recognition and assembles it in a simple REST API interface. It enables you with a broad selection of tools to extract textual data on the images with a single API request.

It uses a model which trained on a large dataset of images, similar to the models used to power Google Photos, so there is no need to develop and train your own custom model.

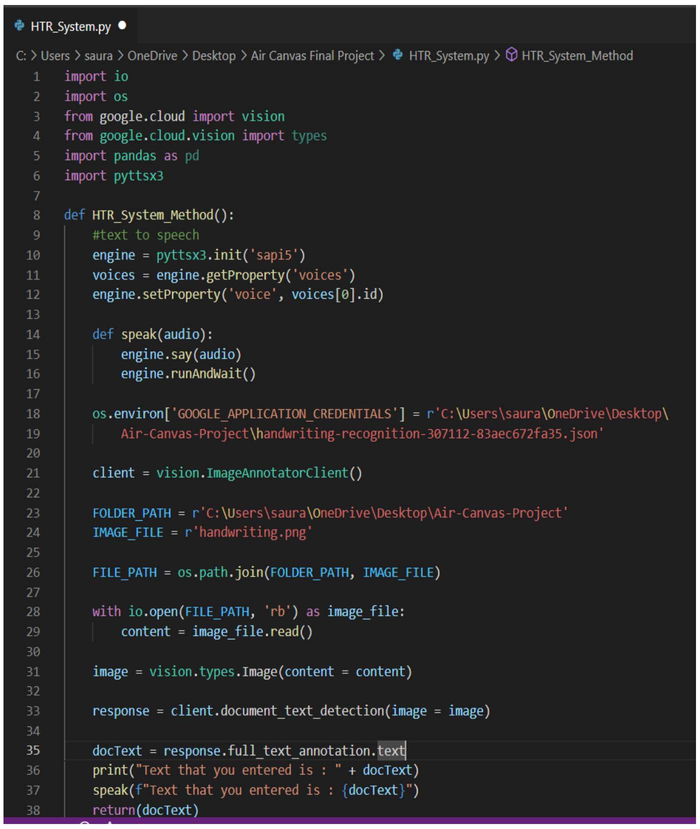
The part of the API that interested us for this project is the OCR part. **Optical Character Recognition** or OCR is a technology where characters are recognized and detected inside an image. Most of the time Convolutional Neural Networks (CNN) are trained on a very large dataset of characters and numbers in different types and colors.

In order to use the Google Cloud Vision API, you will need to login to your google account, create a project, or select an existing project, then enable Cloud Vision API. You will also need to create a service account key and save its json file to your local drive following the instruction on Google Cloud.

Note: *A **service account** is a special type of Google account intended to represent a non-human user that needs to authenticate and be authorized to access data in Google APIs.*

Basically, you can imagine it as an RSA key (encrypted key to communicate with high security between machine via the internet) with which you can connect to Google services (API, GCS, IAM…). Its basic form is a json file.

After this the location of json file that has the service account key has to be specified, and then the following Python script can be used to feed the handwritten image to Google Cloud Vision API to extract text from it.


HTR_System.py ●
C: > Users > saura > OneDrive > Desktop > Air Canvas Final Project > HTR_System.py > HTR_System_Method

```python
import io
import os
from google.cloud import vision
from google.cloud.vision import types
import pandas as pd
import pyttsx3

def HTR_System_Method():
    #text to speech
    engine = pyttsx3.init('sapi5')
    voices = engine.getProperty('voices')
    engine.setProperty('voice', voices[0].id)

    def speak(audio):
        engine.say(audio)
        engine.runAndWait()

    os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = r'C:\Users\saura\OneDrive\Desktop\
        Air-Canvas-Project\handwriting-recognition-307112-83aec672fa35.json'

    client = vision.ImageAnnotatorClient()

    FOLDER_PATH = r'C:\Users\saura\OneDrive\Desktop\Air-Canvas-Project'
    IMAGE_FILE = r'handwriting.png'

    FILE_PATH = os.path.join(FOLDER_PATH, IMAGE_FILE)

    with io.open(FILE_PATH, 'rb') as image_file:
        content = image_file.read()

    image = vision.types.Image(content = content)

    response = client.document_text_detection(image = image)

    docText = response.full_text_annotation.text
    print("Text that you entered is : " + docText)
    speak(f"Text that you entered is : {docText}")
    return(docText)
```

After successful execution of this module i.e., **The HTR System**, it gives the text as output as well as *speaks* the text loud using python's text-to-speech (**pyttsx3**) library.

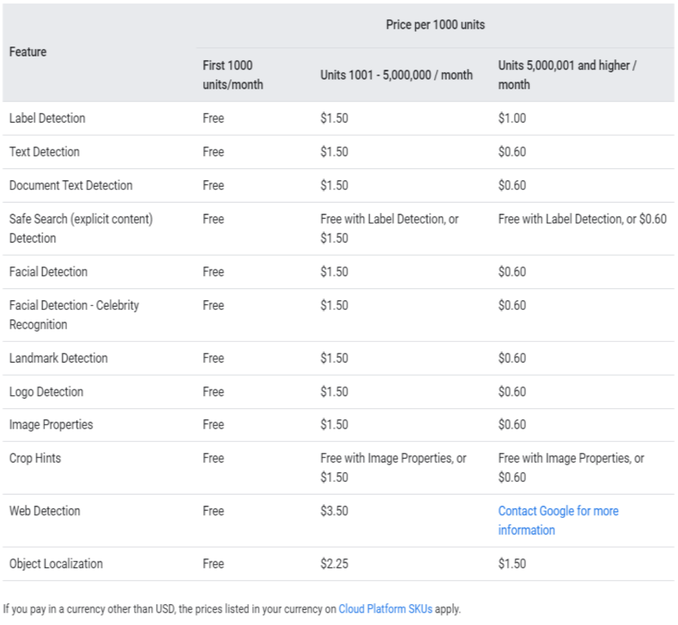
Here is a guide, i.e., a video tutorial by a data science enthusiast *Jie Jenn* on you tube for the configuration and setup of Google-cloud Vision API before using it on our project and also another tutorial for handwritten text extraction and detection from an image.

LINK –

1) Configuration and Setup of Vision API: (221) Google Vision API in Python (Part 2): Configuration and Setup - YouTube

2) Detection and extraction of text from image (Handwritten): (221) Google Vision API in Python (Part 4): Detect and Extract Text (Handwriting) - YouTube

Let's have a look on pricing of Vision API which charges minimal cost and gives unbeatably accurate results.



Feature	Price per 1000 units		
	First 1000 units/month	Units 1001 - 5,000,000 / month	Units 5,000,001 and higher / month
Label Detection	Free	$1.50	$1.00
Text Detection	Free	$1.50	$0.60
Document Text Detection	Free	$1.50	$0.60
Safe Search (explicit content) Detection	Free	Free with Label Detection, or $1.50	Free with Label Detection, or $0.60
Facial Detection	Free	$1.50	$0.60
Facial Detection - Celebrity Recognition	Free	$1.50	$0.60
Landmark Detection	Free	$1.50	$0.60
Logo Detection	Free	$1.50	$0.60
Image Properties	Free	$1.50	$0.60
Crop Hints	Free	Free with Image Properties, or $1.50	Free with Image Properties, or $0.60
Web Detection	Free	$3.50	Contact Google for more information
Object Localization	Free	$2.25	$1.50

If you pay in a currency other than USD, the prices listed in your currency on Cloud Platform SKUs apply.

For example, if your application made 3600 images with text-detection requests in a month, your cost would be:

- $0 for 1000 requests.
- $4.50 for remaining 2600 requests. (For exactly 2000 requests the cost would be 2 * $1.50 = $3 and for remaining 600 request it again goes into 3rd cycle so the overall cost comes up to be 3 * $1.50 = $4.50).

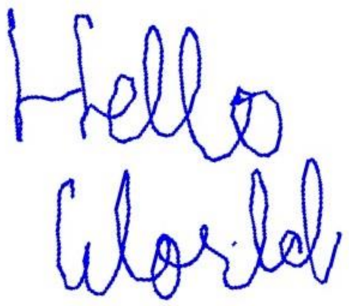
IV. EXPERIMENTAL ANALYSIS

Before coming to Google-Cloud's Vision API, I analysed few pretrained models for text-recognition. In terms of accuracy, I found this Vision API to be the most accurate solution for handwritten text-recognition in case of my project.

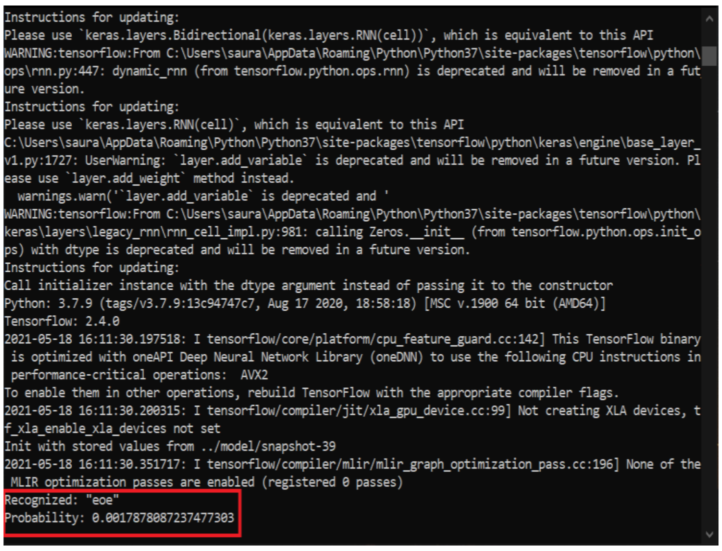
1) The first model which I used was trained on **IAM dataset**. The **IAM Handwriting Database** contains forms of handwritten English text which can be used to train and test handwritten text recognizers and to perform writer identification and verification experiments.

The first system that I used i.e., The Handwritten Text Recognition (HTR) system was implemented with TensorFlow and trained on the IAM offline dataset. This already trained Neural Network (NN) model recognizes the text contained in the images of segmented words.

When it comes to accuracy on the image that I obtained from my Air Canvas module, given an example as follows:



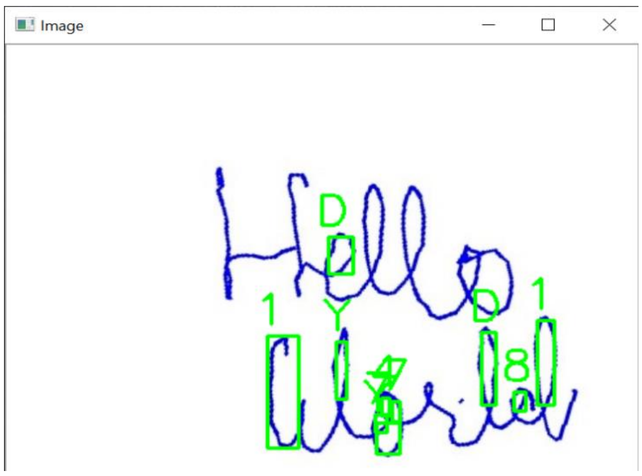
For the above image file, the accuracy of the system was approximately **0.0017**, as shown in the screenshot below, which is very less and that's why I started looking for another appropriate model for my project.

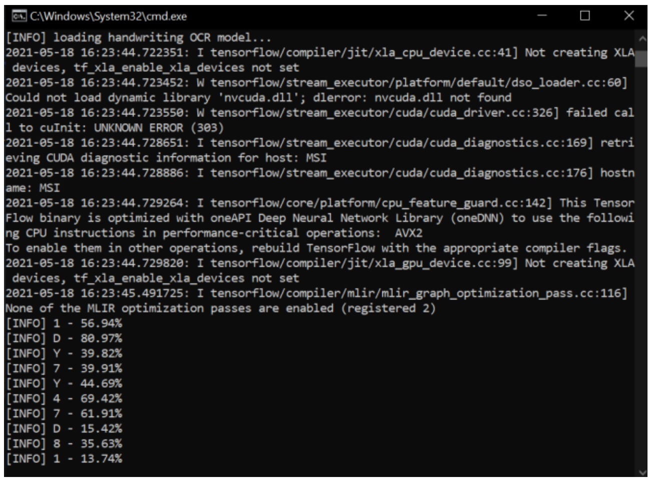


```
Instructions for updating:
Please use `keras.layers.Bidirectional(keras.layers.RNN(cell))`, which is equivalent to this API
WARNING:tensorflow:From C:\Users\saura\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\
ops\rnn.py:447: dynamic_rnn (from tensorflow.python.ops.rnn) is deprecated and will be removed in a fut
ure version.
Instructions for updating:
Please use `keras.layers.RNN(cell)`, which is equivalent to this API
C:\Users\saura\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\base_layer_
v1.py:1727: UserWarning: `layer.add_variable` is deprecated and will be removed in a future version. Pl
ease use `layer.add_weight` method instead.
  warnings.warn('`layer.add_variable` is deprecated and '
WARNING:tensorflow:From C:\Users\saura\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\
keras\layers\legacy_rnn\rnn_cell_impl.py:981: calling Zeros.__init__ (from tensorflow.python.ops.init_o
ps) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Python: 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
Tensorflow: 2.4.0
2021-05-18 16:11:30.197518: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary
 is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in
 performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-05-18 16:11:30.200315: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, t
f_xla_enable_xla_devices not set
Init with stored values from ../model/snapshot-39
2021-05-18 16:11:30.351717: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:196] None of the
 MLIR optimization passes are enabled (registered 0 passes)
Recognized: "eoe"
Probability: 0.0017878087237477303
```

2) Coming to the second model which I tried on the image obtained by the first module of my project, this model was trained on **MNIST dataset**.
For this system, they used python, OpenCV and sklearn to run classification and read the dataset. They used MNIST dataset for training and evaluation for classification.
MNIST dataset is used for evaluating machine learning models on handwritten digit classification problems. The dataset was built from a number of scanned document dataset available from the National Institute of Standards and Technology (NIST).
Each image in this dataset is a 28 by 28-pixel square. There are 70000 images in the dataset that are used for training and evaluate the system. In this proposed system they have used three classification algorithms for the recognition which are Support Vector Machine (SVM), K-Nearest Neighbour and Multi-layer Perceptron Neural Network (MLP).

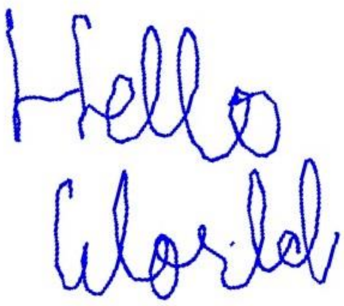
But when it comes to the results and the accuracy, this model also failed to give satisfying and accurate results as shown below in the screenshots :

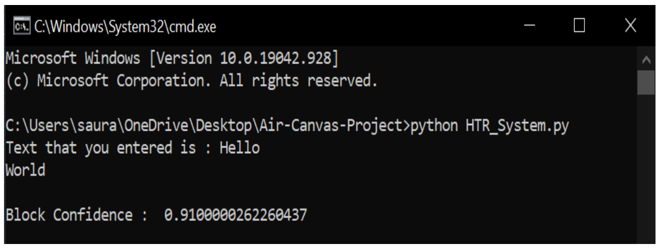




```
C:\Windows\System32\cmd.exe
[INFO] loading handwriting OCR model...
2021-05-18 16:23:44.722351: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not creating XLA
 devices, tf_xla_enable_xla_devices not set
2021-05-18 16:23:44.723452: W tensorflow/stream_executor/platform/default/dso_loader.cc:60]
Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found
2021-05-18 16:23:44.723550: W tensorflow/stream_executor/cuda/cuda_driver.cc:326] failed cal
l to cuInit: UNKNOWN ERROR (303)
2021-05-18 16:23:44.728651: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retri
eving CUDA diagnostic information for host: MSI
2021-05-18 16:23:44.728886: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostn
ame: MSI
2021-05-18 16:23:44.729264: I tensorflow/core/platform/cpu_feature_guard.cc:142] This Tensor
Flow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the followi
ng CPU instructions in performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-05-18 16:23:44.729820: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA
 devices, tf_xla_enable_xla_devices not set
2021-05-18 16:23:45.491725: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116]
None of the MLIR optimization passes are enabled (registered 2)
[INFO] 1 - 56.94%
[INFO] D - 80.97%
[INFO] Y - 39.82%
[INFO] 7 - 39.91%
[INFO] Y - 44.69%
[INFO] 4 - 69.42%
[INFO] 7 - 61.91%
[INFO] D - 15.42%
[INFO] 8 - 35.63%
[INFO] 1 - 13.74%
```

3) Now finally when it comes to the **Google-Cloud's Vision API**, it gives surprisingly accurate results. The same image when we gave as an input to the Vision API module, the results we got are as follows:





```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saura\OneDrive\Desktop\Air-Canvas-Project>python HTR_System.py
Text that you entered is : Hello
World

Block Confidence :  0.9100000262260437
```

As we can see that this Vision API correctly predicted the text written in the image file i.e., **Hello World**, with more than **91%** accuracy.
Hence, we selected this Google-cloud Vision API for the HTR System module of our project.

V. CONCLUSIONS

In this paper we have proposed a system which makes an imagination come true. The best use of AI is to operate devices without physically touching them. Writing in air, and the system detecting the text written by you by simply waving your fingers in the air is one of the best applications of Artificial Intelligence (AI). So finally, this system will use your webcam to detect the movement of your fingers and predict the text written by you. This application can be further modified to take commands using your fingertips. For example, if you write 'P' using your fingers you may be instructing your system to open "Paint Application", writing 'M' may instruct the system to "Play Music" and so on.

REFERENCES

[1] GeeksForGeeks: Create Air Canvas using Python-OpenCV - GeeksforGeeks

[2] Write in Air (Medium): Draw Using a Virtual Pen on a Computer Screen using OpenCV in Python | by Praveen | programming fever | Medium

[3] Google-Cloud Vision API: Detect text in images | Cloud Vision API | Google Cloud

[4] You Tube Links: (221) Google Vision API in Python (Part 2): Configuration and Setup - YouTube

(221) Google Vision API in Python (Part 4): Detect and Extract Text (Handwriting) - YouTube

[5] Vision API (Medium): Printed and handwritten text extraction from images using Tesseract and Google Cloud Vision API | by Derrick Wang | Medium

[6] IAM Dataset Model: Build a Handwritten Text Recognition System using TensorFlow | by Harald Scheidl | Towards Data Science

[7] MNIST Dataset Model (PyImageSearch): OCR: Handwriting recognition with OpenCV, Keras, and TensorFlow - PyImageSearch

[8] Flask Documentation: Flask - (Creating first simple application) - GeeksforGeeks

Welcome to Flask — Flask Documentation (2.0.x) (palletsprojects.com)