Air Handwriting Detection and Recognition

Project report submitted for

IVth Semester Minor Project-I

Department of Data Science and Artificial Intelligence

By,

Abhishek Chaudhary (191020403)

Khushi Agrawal (191020429)

Prabhat Kumar (191020438)

Under the Guidance of Dr. Abhishek Sharma

Department of Data Science and Artificial Intelligence

Dr. Shyama Prasad Mukherjee

International Institute of Information Technology, Naya Raipur

(A Joint Initiative of Govt. of Chhattisgarh and NTPC)

Email: iiitnr@iiitnr.ac.in, Tel: (0771) 2474040, Web: www.iiitnr.ac.in

CERTIFICATE

This is to certify that the project titled “AIR HANDWRITING DETECTION AND RECOGNITION” by “ABHISHEK CHAUDHARY”, “KHUSHI AGRAWAL” and “PRABHAT KUMAR” has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree/diploma.

(Signature of Guide)

_____

Dr. Abhishek Sharma

Assistant Professor

Department of ECE

Dr. SPM IIIT-NR

Month, 2021

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature of Author)

_____

Author Name

(Roll Number)

Date : _____

Plagiarism Report

ORIGINALITY REPORT

17%	13%	9%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Dr. S. P. Mukherjee International Institute of Information Technology (IIIT-NR) Student Paper	5%
2	www.mdpi.com Internet Source	1%
3	Submitted to Intercollege Student Paper	1%
4	journals.plos.org Internet Source	1%
5	public.cranfield.ac.uk Internet Source	1%
6	Submitted to Sharda University Student Paper	1%
7	Mingyu Chen, Ghassan AlRegib, Biing-Hwang Juang. "Air-Writing Recognition—Part II: Detection and Recognition of Writing Activity in Continuous Stream of Motion Data", IEEE Transactions on Human-Machine Systems, 2016 Publication	1%

Approval Sheet

This project report “AIR HANDWRITING DETECTION AND RECOGNITION” by “ABHISHEK CHAUDHARY”, “KHUSHI AGRAWAL” and “PRABHAT KUMAR” is approved for IVth Semester Minor Project-I.

(Signature of Examiner - I)

_____

Dr. Abhishek Sharma

(Signature of Examiner - II)

_____

Dr. Shivani Sharma

(Signature of Chair)

_____

Dr. Muneendra Ojha

Date: _____ Place: _____

ABSTRACT

Air Handwriting refers to writing of characters or words in a free space by hand, finger or a specific color of a light. It's different from conventional pen-paper writing as we have a pen-down pen-up motion to start or end a character while in air handwriting detection we have to shift the color or turn the light on or off to start and end a characters or words.

The Air Handwriting Recognition project is a combination of computer vision object tracking and handwriting recognition using machine learning.

The air handwriting recognition system uses the webcam of a computer to track character and digits written in the air by the user, Computer then uses a mask to track a specific color selected by the user. This mask then tracks the color and it is then drawn onto a canvas (i.e. plain whiteboard).

We use this canvas image as an input for the recognition model to understand the air written words to the character digits into one of 62 different classes (i.e. 10 integers, 26 Capital letters, 26 small letters).

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

Air-writing is useful for users who don't want to type on a keyboard or write on a track pad/touch screen, as well as for text input for smart device control and a variety of other uses.

Air writing intrigue many medical and engineering felids such as neurological department, physiatrist department and many other medical felids and engineering felids such as IoT and use cases in smart home handsfree appliances.

But Why air writing is useful?

Imagine if we could see or visualize the unspoken and the unseen!!

In a world where scientific marvels have given access to land objects like rocket boosters vertically! We lack the means to communicate with the ones who are differently abled or those who lost something essential over their lifetime like losing the ability to hear. Just imagine, you are full of thoughts yet you can't deliver them to everyone.

Imagination is the language taught by none yet practiced by everyone. How convenient would it be if a parent could understand the patterns of their hyperlexic kid with minimum effort or to create a significant cognitive impression to help cement the word(s) in a child's or even an old person's memory!

We are making an endeavour to approach all these things using a visualization-based method which we call as "The Air Handwriting detection".

Writing in air is like writing on an invisible canvas in three-dimensional space. Since the camera only accounts for the two dimensions. We will consider the canvas in this project as it is two dimensional. When we move an object to draw on the assumed invisible canvas, computer need to draw whatever we draw on the invisible canvas onto a virtual canvas with the help of camera, to do this, camera is going to track and store its location from the actual space into an array to create a line on the canvas.

2. LITERATURE REVIEW

Many works are proposed related to Air Handwriting Detection and Recognition. We divided this project into two parts. They are “Detection of Air Writing” and “Recognising the Detected Writing”. In this part ‘Air Writing Detection’ we put forward our solution using Computer Vision concepts. It is mainly based on OpenCV library functions. Detecting objects live, makes our task more challenging due to face detected by webcam and presence of other objects of similar color in background. Many great works are presented throughout, some of them are listed below –

2.1 Air-Writing Recognition — Part I: Modelling and Recognition of Characters, Words, and Connecting Motions. The renowned authors of this project are ‘Mingyu Chen’, ‘Ghassan AlRegib’ and ‘Biing-Hwang Juang’. They presented a 6 degree of freedom hand motion of data to recognise the characters and addressed air-writing into two levels: motion words and motion characters [1].

2.2 Fingertip Detection and Tracking for Recognition of Air-Writing in Videos. This project is originally made by ‘Sohom Mukherjee’ [2]. It is proposed using three key methodologies starting with detection of fingertip followed by the recognising air written and then using hand datasets to recognise them. It is based on R-CNN framework, distance-weighted curvature entropy and some velocity-based termination criterion.

2.3 Trajectory-Based Air-Writing Recognition Using Deep Neural Network and Depth Sensor. This work is proposed by ‘Md. Shahinur Alam’, ‘Ki-Chul Kwon', ‘Md. Ashraful Alam’, ‘Mohammed Y. Abbass’, ‘Shariar Md Imtiaz’ and ‘Nam Kim’ [3]. Their project is divided into four major subclasses they are: Fingertip detection, data collection, normalization, and network design.

2.4 Writing in the air: Facilitative effects of finger writing in older adults. PLoS ONE 14(12): e0226832[4]. This study is proposed by Itaguchi Y, Yamada C, Fukuzawa K (2019). It confirms the benefits of finger movement for solving cognitive tasks involving visual processing of written language among older adults.

2.5 Air-Writing Recognition—Part II: Detection and Recognition of Writing Activity in Continuous Stream of Motion Data by Mingyu Chen; Ghassan AlRegib and Biing-Hwang Juang. This project entails detecting and recognising air-writing activities embedded in an unrestricted continuous motion trajectory. It employs a dataset with a mix of writing and nonwriting finger motions in each recording. Leap Motion's LEAP is used for marker-free and glove-free finger tracking. They propose a window-based approach that detects and extracts the air-writing event from a continuous stream of motion data that includes stray finger movements unrelated to writing.

3. PROPOSED SOLUTION

We propose a simple code which rely on some basic python libraries (like numpy and OpenCV) and the standard webcam of the laptop devices for tracking the hand/stylus movement in the air.

We reckon that this method/approach to communicate with some unfortunates like those who have lost the ability to hear would be much easier if they can simply scribble their thoughts in the air (because they do know how to use language and just can't receive it) rather than jotting them down on a paper. Air writing is not only fun to practice but also advantageous for improving the kinesthetic and tactile feedback which comes from hand and arm movement in children at kindergarten and nursery levels [4]. Not only this, a study conducted by Yoshihiro Itaguchi, Chiharu Yamada and Kazuyoshi Fukuzawa [5] in 2019 confirm the benefit of finger movement in the resolution of kūsho effects in education, which include visual processing of written language among the elderly. [5]. Kūsho is a technique for recalling the shape of a written character or the spelling of a word that involves moving the index finger in the air or on a surface as a replacement for a pen. It is most often used when attempting to remember the shape of a written character or the spelling of a word. Which has been shown to help users of the kanji writing system perform better on cognitive tasks.

We are trying to add an extra feature of virtual visualization of these unspoken and unseen words and thoughts with the most basic methods and resources available.

The basic knowledge of array, numpy and OpenCV is required to understand the working of the code.

We achieve "Writing detection" by having the project divided into five different parts –

1. Color Selector
2. Morphological Operations for Masking
3. Contour
4. Paint Sheet
5. Live Tracker

3.1 Color Selector –

The color selector consists of 6 different sliders to accurately chose the color of the object to be tracked. These 6 sliders namely "Upper Hue", "Upper Saturation", "Upper Value", "Lower Hue", "Lower Saturation" and "Lower Value".

Upper Hue and Lower Hue helps us to pick the color of the object to be tracked from 32,400 shades of color.

Upper Saturation and Lower Saturation let us decide the saturation for each 32,400 shades of color making the total color count approximately 2.1 Billion theoretically.

Since camera present on the laptop can capture approximately 16.78 million colors only so we are only limited to 16.78 million colors.

The Upper Value and the lower value help us to limit the luminance or brightness of the color.

We created this color selector using the functions from cv2 library. The syntax for creating the color selector is as follows –

cv2.namedWindow(): This function creates a window of any desired name

cv2.createTrackbar("a", "b", c, d, e): This function creates a trackbar (slider) and have parameters "name of slider", "window name", "lower limit" (integer value), "higher limit" (integer value) and "setValues" to set the values as "a", "b", "c", "d" and "e" respectively.

cv2.getTrackbarPos(): This function gets the location of the slider.

3.2 Morphological Operations for Masking –

When simple basic operations performed on the image shape, such operations are known as morphological operations. These mask operations replace the value of each pixel in the image or video from the given mark or matrix also known as kernel. Morphological operations make extensive use of arrays for noise removal, isolation of individual elements in an image, and joining different elements in an image. Array in morphological transformations is also used for finding intensity smash and holes.

Fig: 3.2.1. Original Image [5]

There are some major categories or can say basic operations of Morphological operations:

3.2.1 Erosion: Erosion Operation in Morphological Transformation calculates the local minima over the area of the given kernel. And then replace the minimum value calculated with the provided image or video frame. The erosion operation is –

$$\mathbf{dst(x, y) = min_{(x', y'):element(x', y')=0}\ src(x + x', y + y')}$$

The function for erosion operation is:

```
erosion = cv.erode(img, kernel, iterations = 1)
```

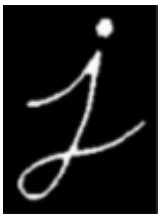

Fig: 3.2.1.1. Eroded Image [5]

3.2.2 Dilation –

Similar to erosion operation dilation operation is used to convolve pixels of original image to maximum value in the given kernel. The dilation operation is-

$$\mathbf{dst(x, y) = max_{(x', y') : element(x', y') \neq 0} src(x + x', y + y')}$$

The function for dilation operation is:

dilation = cv.dilate(img, kernel, iterations = 1)

Fig: 3.2.2.1. Dilated Image [5]

3.2.3 Opening –

Opening is a single term for erosion followed by dilation. This method is useful for removing noise. The opening operation is:

dst(x, y) = open(src, element) = dilate (erode(src, element)

The function used to preform Opening is:

opening = cv2.morphologyEx(img, cv.MORPH_OPEN, kernel)

Fig: 3.2.3.1. Opening [5]

In this project we used this method to detect object in the video frame.

3.3 Contour –

The curve joining all continuous points having same intensity and pixel are known as contours. They are useful mainly for detection and recognition purpose. To find contours object to be detected should be white and the background should be black. So, we have to perform some binary operations before finding contours. In this project we used –

cv2.findContours(): This function finds contours from a binary image using an algorithm.

cv2.drawContours(): This function is used to draw shape from the given boundary points.

Simple result of the detected contours:

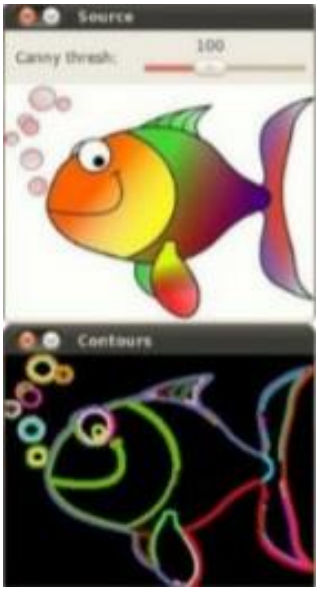

Fig: 3.3.1. Finding Contour [5]

3.4 Paint Sheet –

After creating the contour and object tracking using the mask, we need to create a sheet which we will later in future will use this paint sheet for handwriting recognition.

Paint sheet is made by using OpenCV.

The centre of the contour is calculated by formulating the cv2.moments(), an inbuilt function of library cv2:

> **M = cv2.moments():** This function calculates moments, up to the third order, of a vector or rasterized shape (essentially the locus of the outline). The output is returned in the parameter structure param array Raster image (single-channel, 8-bit or floating-point 2D array) or 2D point array (1 × N or N × 1)
>
> The centre is calculated as follows –
>
> **centre = (**int(M['m10'] / M['m00']), int(M['m01'] / M['m00'])**)**

Through the centre of the contour, we can now create lines on the plain sheet. These lines have three different components –

1. Color
2. Length
3. Location on the paint screen as well as on the Live tracker Screen.

i. Color –

We can get color of the lines by first initializing an array with respected color information (0-255 as for 8 bit per pixel, $2^8 = 256$ combinations are possible)

Then we specify the color as per our liking, some standard color combinations are given in the Table 3.4.1.1.

Table: 3.4.1.1. Coordinate and their respective color

Coordinate	Color
(0,0,0)	Black
(255,255,255)	White
(255,0,0)	Blue
(0,255,0)	Green
(0,0,255)	Red
(0,255,255)	Yellow
(150,150,150)	Grey

ii. Length –

The length of the line plays a major role in the user experience. We have selected a size of 512 units of pixel per color as neither lengthy or short for this project and it does not consume a lot of resources.

Each location of the pixel is stored onto a dequeue which helps to keep the location of the line even if we move the window.

iii. Location –

The location of the contour centre has been stored as an element to the double ended queue. We used dequeue because if we want to remove an element from the queue, we can do that easily from either front or the end. When this queue gets full, system will automatically delete the previous elements from the end.

3.5 Live Tracker –

Live tracker is a window which uses everything from contour, mask, paint sheet and color selector to produce a virtual screen having whatever we draw, live camera output and colored lines.

This window is made using OpenCV, Addition to this we have 5 rectangular On-Screen Display buttons from which when we get to the button, it performs some specific tasks like changing the ink color of the tracker or clearing the screen.

These buttons are created using OpenCV and they perform operations according to our algorithm.

To access the camera the OpenCV inbuilt function “cv2.VideoCapture()” to play the video as a background for the Live Tracker window. We also use “cv2.VideoCapture().read()” to access the basic information about the camera such as frame size.

This Frame size is used as the size for the Paint, Mask and the Live Tracker window so that we do not get out of the frame while writing or have blank areas as borders while writing.

In addition to this We have done Asynchronous coding to reduce time taken by the whole code to execute.

3.6 Recognition

We used **CNN** method for training our model for predicting the input image of the character drawn on the paint sheet.

3.6.1 Why CNN?

Deep learning is actively helping man power to create new kinds of technologies. Computer Vision is one of its applications. Facial Recognition, self-driving cars are some of the most valuable developments of computer vision. One of the major challenges of CNN is that real world images are too large to be formulated using simple networks. Here comes the role of Convolutional Neural Network (CNN). Classification and recognition through CNN increase the accuracy of the model. Hence, we are using CNN architecture to deploy our model.

3.6.2 What is CNN?

A Convolutional neural network or CNN for short is an algorithm which is most commonly used in the purpose of image classification. If one has to give a high-level description of what CNN is. He may say that it takes an image as an input and then assign some weights to various characteristic of the object(s) in the image then differentiates them from one another.

Hardly any image recognition we come through which doesn't employ the use of this deep learning method and we just followed the tradition as being beginners.

The CNN contains 4 main components: -

1. Convolution layer
2. Padding and strides
3. Activation function – ReLU, SoftMax
4. Pooling

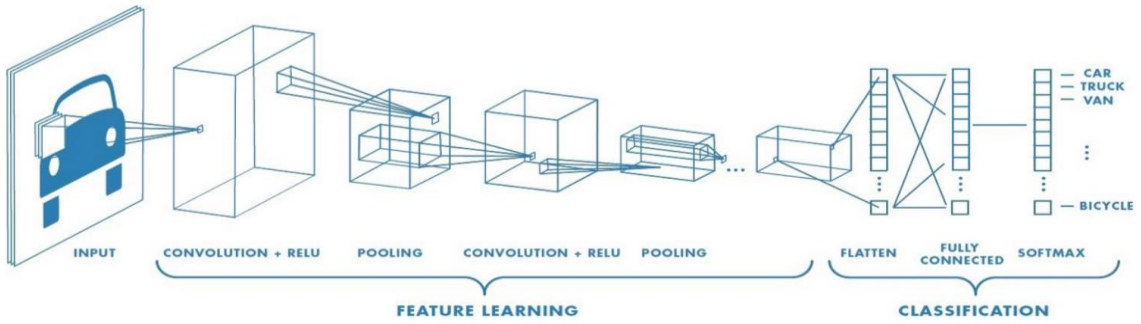

Fig: 3.6.1.1 Complete process of CNN

3.6.2.1 Convolution layer

CNN compares the image piece by piece. This means that an image of the object is broken into smaller pieces which are called features. It is these features which it looks for in the image that is given to it as an input.

By finding the rough feature matches in almost or approximately the same position in the two images, CNN improves a lot in finding similarities between two images and thus us able to predict whether they are same or not.

So, these features which are parts of a bigger image are put on the input image and then if it matches accurately, it is classified as a correct classification. This piece or feature is also known as filter or kernel.

Each element of the filter matrix is multiplied to its corresponding positional element in the input image and then all of them are summed up to result a final value.

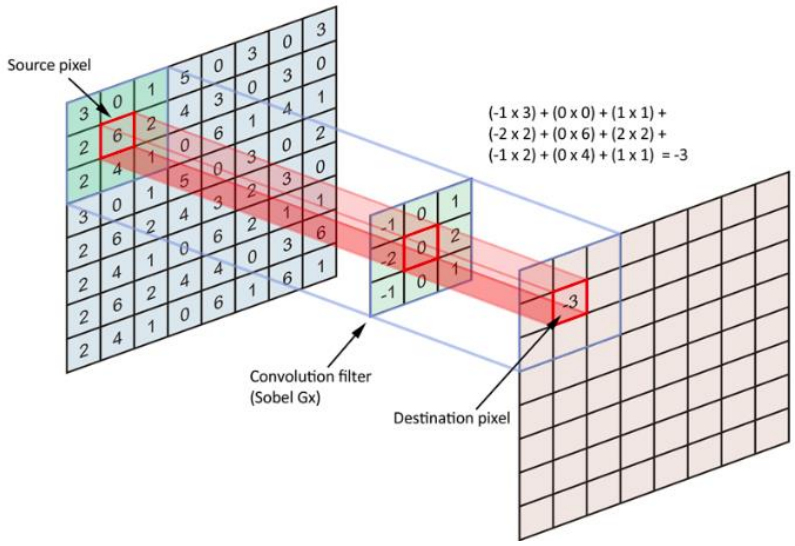

Fig: 3.6.2.1.1 A visualization of the convolution filter

3.6.2.2 Padding and strides

- **Padding:** Padding refers to adding some number of rows and columns to the given input images. In most of the case the padding value is zero. It is important to use, as after convolution most of the data or important information are lost after using convolution and the output shrinks. It is used before applying convolution so that data or edges remains conserved. Suppose the matrix n x n is convolved with f x f filter/kernel then the padding p give us n+2p-f+1, n+2p-f+1 matrix. There are mainly two types of padding:

  - **Valid Padding:** Valid padding means no padding. It assumes that all the dimensions are valid, hence the input images get fully covered by the filter and the stride specified by the user.

- **Same Padding:** Same padding is also known as zero padding. This padding is used when we want shape of the input image is equal to the shape of the output image. Generally, digit zero is padded to the input image.

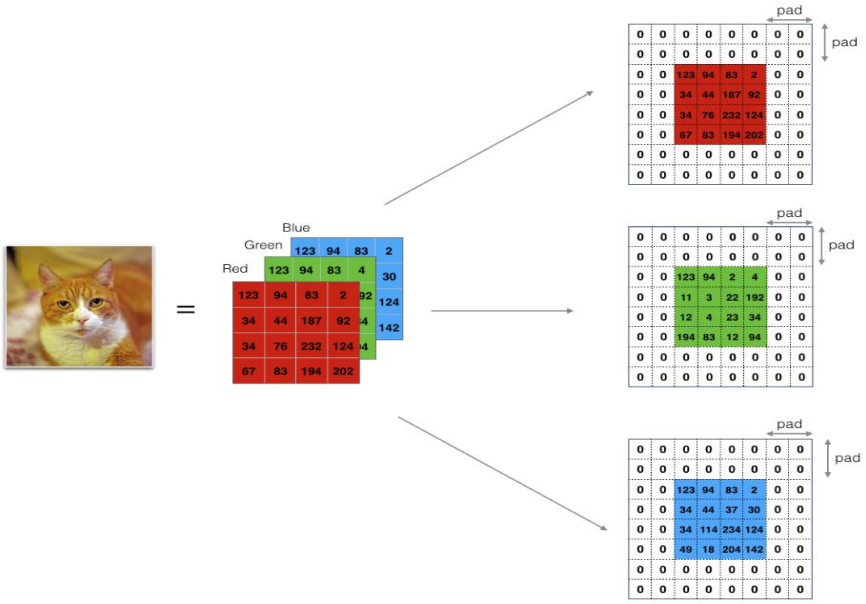

Fig: 3.6.2.2.1 The padding of each feature map of each channel

The above figure shows that the cat image and its RGB pixels. Two rows and two columns are padded to each of the matrix. It is an example of same padding.

- **Strides:** Stride is the component of convolutional neural network which tells by how much column or row the component have to jump. It is denoted by S. Suppose if a matrix of size n x n is convolved with a filter/kernel size of f x f and padding p and stride s then it gives us a matrix of size (n+2p-f)/s + 1, (n+2p-f)/s + 1.

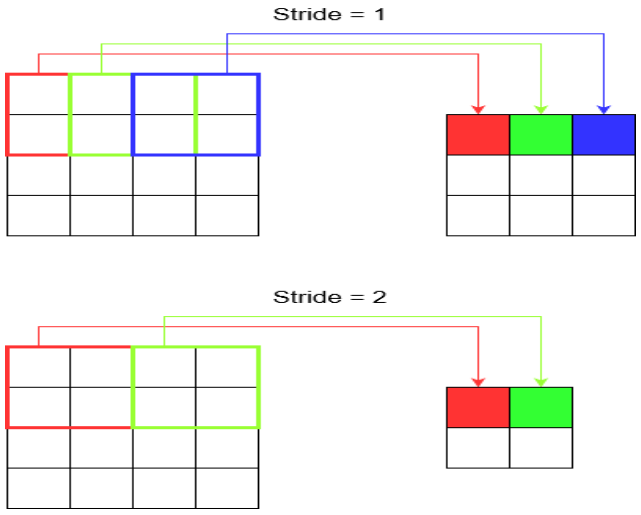

Fig: 3.6.2.2.2 The strides visualization

In the above image the size of input matrix is 4 x 4. The 1st image in it represent stride as 1, showing that it jumps one box each time and the 2nd image represent the stride as 2, shows the jump of two boxes each time.

3.6.2.3 Activation function – ReLU, SoftMax

- **ReLU:** ReLU stands for rectified activation function, this activation function will output input directly if positive, if not then it gives zero. This function is mainly used when the hidden layers are completely unknown. This activation function is a default activation function for many neural networks. It gives computationally faster and better results than other activation functions. It is simple to use because its Justs a linear function. Its equation is given as:

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \Rightarrow 0 \end{cases}$$

The graph of this function is:

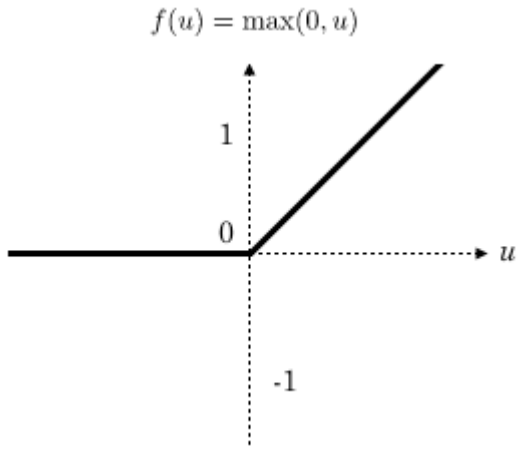

Fig: 3.6.2.3.1 The graph of ReLU activation function

This image shows that for negative values, the function outputs 0 but for positive values it gives the result same as the original function value.

- **SoftMax:** This activation function is used in the output layer to predict a multinomial probability distribution. This activation function is mainly used for multi-class classification problems. The equation of SoftMax function is represented as:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ = softmax
$\vec{z}$ = input vector
e^{z_i} = standard exponential function for input vector
K = number of classes in the multi-class classifier
e^{z_j} = standard exponential function for output vector
e^{z_j} = standard exponential function for output vector

The graph of SoftMax function is:

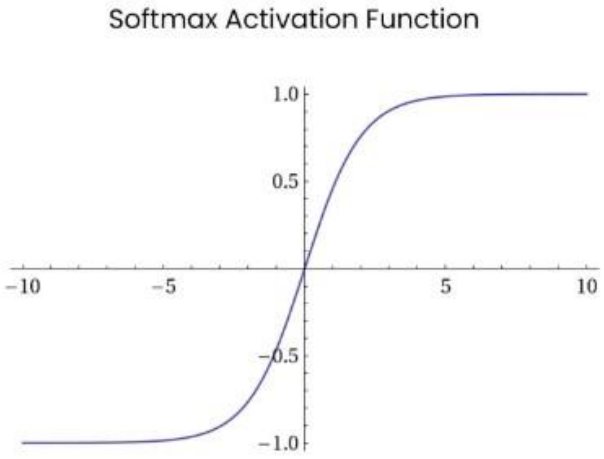

Fig: 3.6.2.3.2 The graph of the softmax function

3.6.2.4 Pooling

The pooling layer is where the entire image mapping by the kernels gets summarised. In other words, this layer reduces the dimensions of the feature map. It would be a complex and a computationally expensive task to learn from all the parameters or the values of the feature map. Thus, by summarising the multi features in a given region the total no. of parameters to be studied gets reduced significantly which in turn reduces the computational cost and time taken for the training by a great extent.

The pooling is of three types –

1. Max pooling: The max element from a region of the feature map gets selected. The output has the most prominent features from the feature map.
2. Average pooling: The average of elements present in the region of a feature map is calculated. The output has the average value of the features from the feature map.
3. Global pooling: Each and every channel or region gets reduced to a single value. It can be either global max pooling or global average pooling. The output of a n*n*n feature map is reduced to a feature map of 1*1*n dimensions.

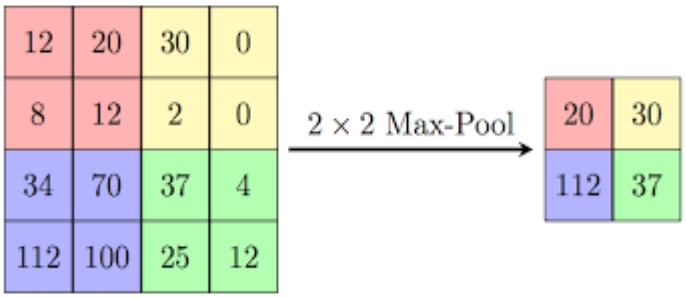

Fig: 3.6.2.4.1 The max pooling

3.6.3 How we used CNN

- Sequential model is used for implementing our model. This method is used to build model layer by layer. We used add() function to add layers in our model. We implemented this using keras library.
- The 1st layer is Conv2D() layer with ReLU activation function. The size of input image is (28, 28, 1) and kernel size is (3, 3). Here, we used 32 filters
- Then added maximum pooling with pool size as (2, 2) and stride as 2.
- Again Conv2D() layer is used with same ReLU activation function and applied same padding then maximum pooling is applied to our model. Here, we used 64 filters
- Similarly new layer is made with 128 filters, ReLU activation function and valid padding and max pooling is aplied after it.
- Then we flatten our model and used two fully connected, Dense layer with ReLU activation function.
- In the output layer SoftMax activation function is applied to get the final result.
- Moreover, we used Adams Optimization function to optimize our model and get better and more accurate results.

4. RESULTS

4.1. Detection

Fig: 4.1.1. The live tracking window

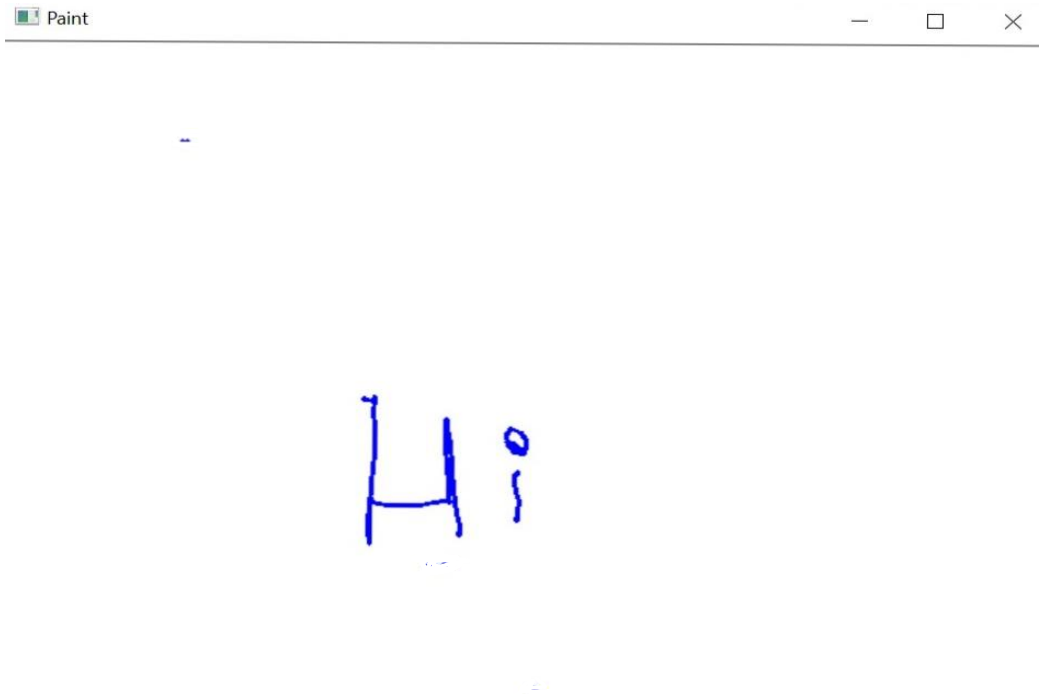

Fig: 4.1.2. The Paint window

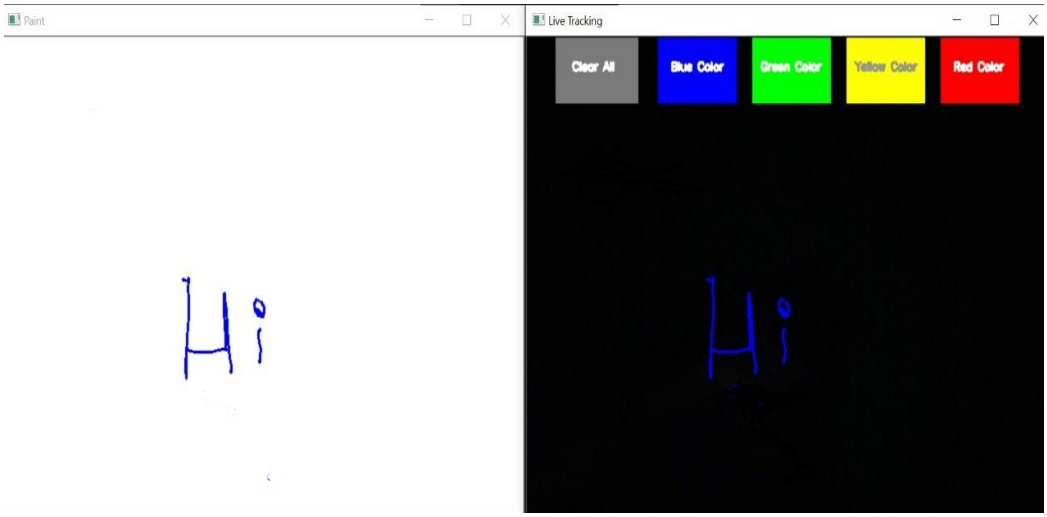

Fig: 4.1.3. Paint and tracking window side by side

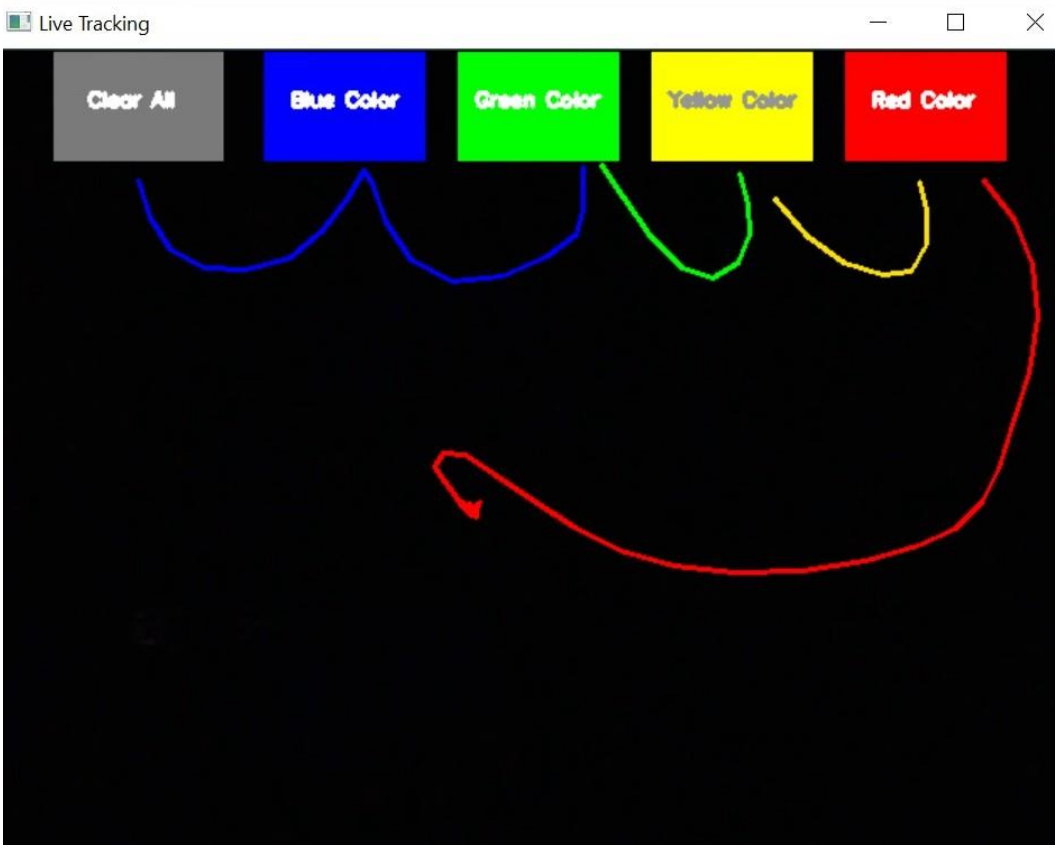

Fig: 4.1.4. The drawing ink color selection

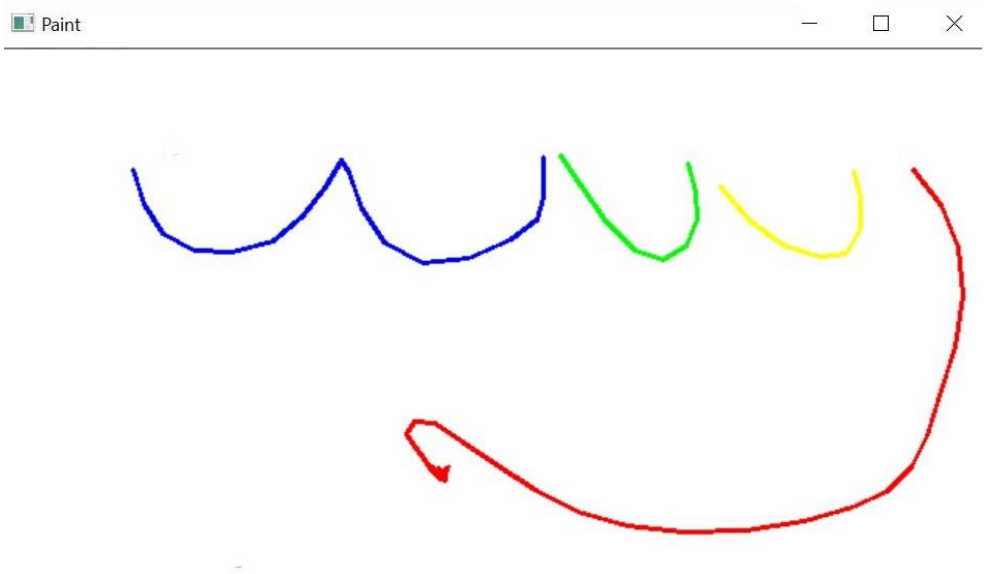

Fig: 4.1.5. The Ink variation on the Paint window

Fig: 4.1.6. The mask window

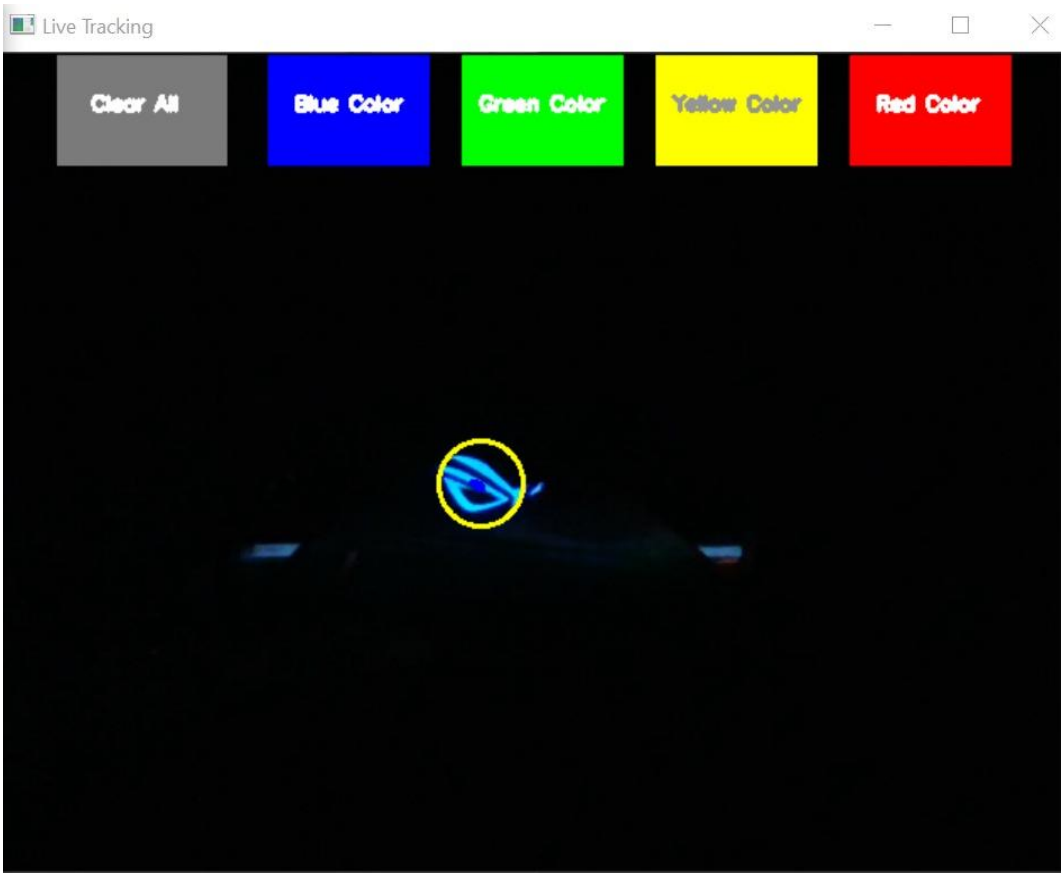

Fig: 4.1.7. The live colour tracking contour formation

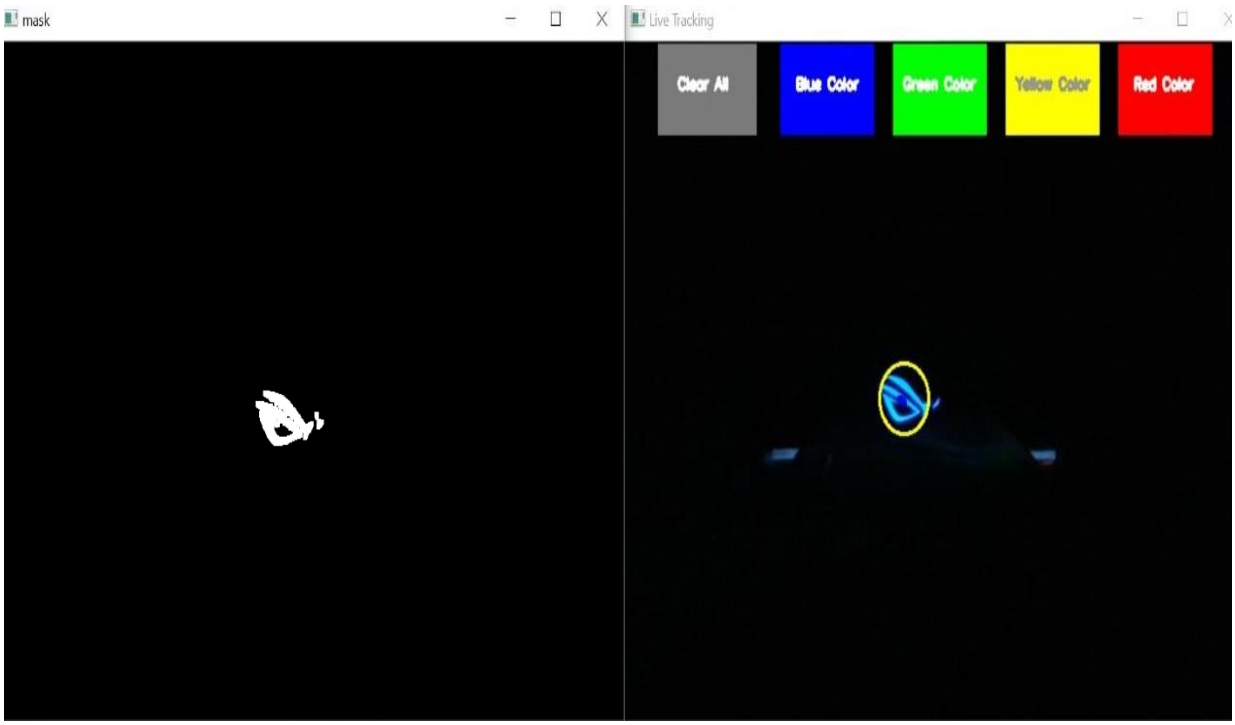

Fig: 4.1.8. The mask and contour tracking window side by side

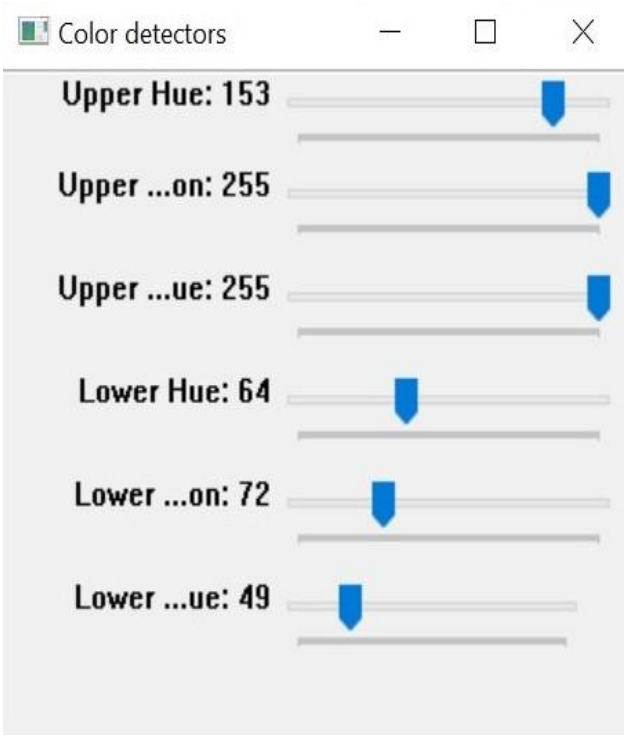

Fig: 4.1.9. The tracking color slider window

4.2. Recognition

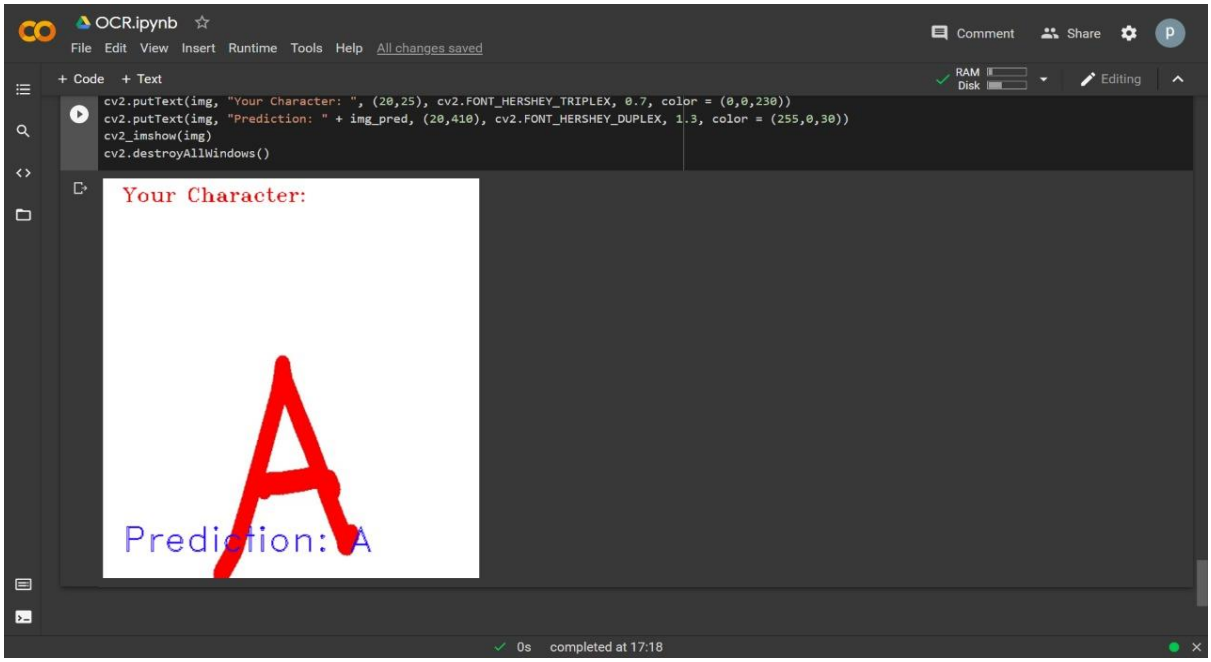

Fig: 4.2.1. Letter ‘A’ predicted correctly

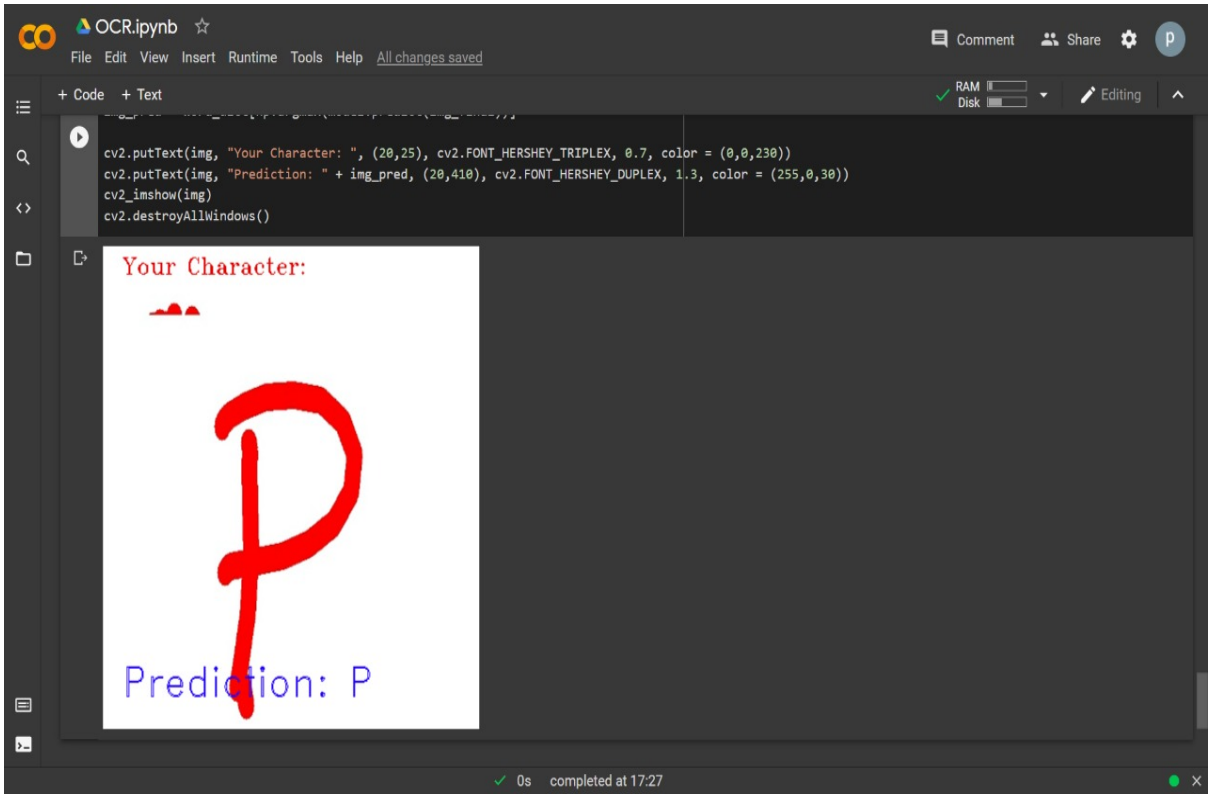

Fig: 4.2.2 Letter ‘P’ predicted correctly

Fig: 4.2.3 Letter ‘K’ predicted correctly

4.3. Limitation

- The model failed to identify and predict multiple characters when given together**.**

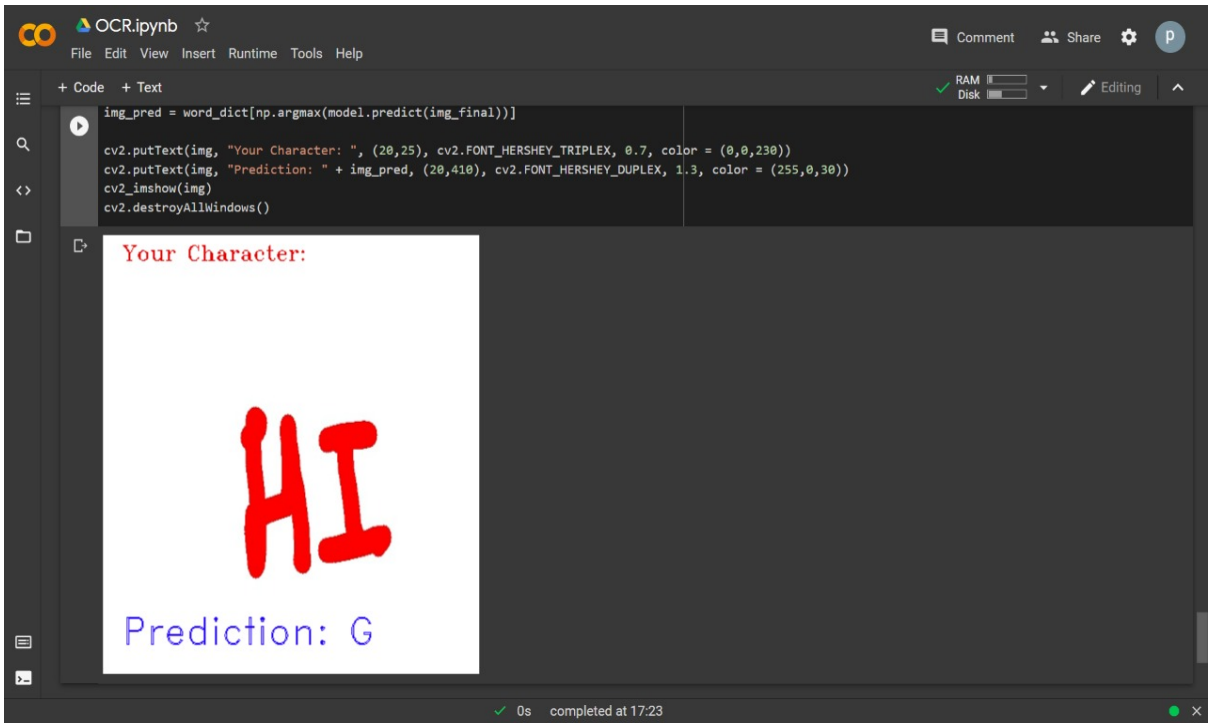

Fig: 4.3.1 The word “Hi” is predicted wrong i.e, as ‘G’

- We were unable to integrate the environment for prediction of the character and detection of the character drawn on the paint sheet into a single environment (i.e. Model for the prediction is used in the Google Colaboratory due to some problem regarding the environment setup and Air Handwriting detection is done through the Visual Studio Code).

- For initiation recognition, certain keys have to be pressed thus not making it a fully gesture controlled solution.

- This model does not achieve good accuracy for the Air Handwriting Detection part when used in a utterly colorful room and in high luminous rooms which depend on the dynamic range of the camera used. This Model achieves great accuracy when used in a darker room.

- We were not able to recognize the integers with the current database.

5. CONCLUSION

In this project we proposed a simple algorithm for Air-Writing Detection mainly by using OpenCV library. This is real time detection model which captures video via webcam. We then detect object in the video through which we are able to write and can make different design. This project uses core concepts like Morphological Operations, Detecting Contours, Drawing in Canvas, Computing Moments and many more. This project is beneficial for differently abled peoples, hyperlexic kids, elderly aged people who has difficultly to write or kids who just can't talk or write. Detecting objects via webcam detects various other objects present in the background of similar colour as well as faces of similar colour tone detected through webcam decreases the accuracy of our model. This is major drawback of our model.

To recognize the detected characters or writings from the canvas we used CNN model to the detect the alphabets written by the user on the canvas. We used keras and TensorFlow libraries to accomplish it. The convolutional model is itself very deep model and then the use of ReLU activation in it improved the accuracy of the model. Another advantage of using this model is that we have to perform very less pre-processing for the CNN model. We achieved the training accuracy as 95.73% which is one of the major advantages of our model. Furthermore, we can improve its accuracy by tunning our model or by using class weights or changing the optimizer, learning rate, loss function. Major drawback of our proposed model is, it is two-step process i.e. we first have to capture video then click picture from it and finally give it as a input to our CNN model. We can further renovate this project by detecting objects using hand gesture movements.

REFERENCES

URL

1. https://sightwords.com/sight-words/lessons/air-writing/
2. https://docs.opencv.org/master/d9/df8/tutorial_root.html
3. https://www.groundai.com/project/fingertip-detection-and-tracking-for-recognition-of-air-writing-in-videos/1

Journal

4. Modelling and Recognition of Characters, Words, and Connecting Motions. The renowned authors of this project are 'Mingyu Chen', 'Ghassan AlRegib' and 'Biing-Hwang Juang'

5. Trajectory-Based Air-Writing Recognition Using Deep Neural Network and Depth Sensor. This work is proposed by 'Md. Shahinur Alam', 'Ki-Chul Kwon', 'Md. Ashraful Alam', 'Mohammed Y. Abbass', 'Shariar Md Imtiaz' and 'Nam Kim'.

6. Itaguchi Y, Yamada C, Fukuzawa K (2019) Writing in the air: Facilitative effects of finger writing in older adults. PLoS ONE 14(12): e0226832.
7. Detection and Recognition of Writing Activity in Continuous Stream of Motion Data. This journal was proposed by 'Mingyu Chen', 'Ghassan AlRegib' and 'Biing-Hwang Juang'.

Thesis

8. An optimized approach for Deaf and Dumb People using Air Writing. By 'Nithya.S', 'Chandru.S', 'Krishnakanth.G' and 'Pavithran.P'.