Air writing Recognition of Tamil letters
Project report

NAME	REG.NO	AADHAR NO	E-MAIL	MOBILE NO
Visshal. A	2018103625	9061 8089 4058	visshal72@gmail.com	9500097775
Syed Adnan Moin	2018103074	7073 3906 8001	adnanofirm@gmail.com	9840800561

Contents

1. Abstract

Air-writing is the writing of linguistic characters or words in air by using hand or finger movements. Motion gestures play a very important role in general human– computer interaction. Motion gestures are meant to be simple so that users can easily memorize and perform them. Air-writing will be useful in user interfaces that do not allow the user to type on a keyboard or write on a track pad/touch screen, or for text input for smart system control, among many applications.

Air-writing can be realized in several ways from the perspective of a user. Writing individual isolated letters in an imaginary box in space, one at a time, is the first and most essential thing. The second is to write multiple letters from left to right across the room in a style similar to writing on paper. It is accomplished based on six degrees of freedom and motion data.

We are planning to use a web-cam to capture stylus movements and use CNN(Convolutional Neural Networks) to recognise letters from those gestures.

2. Introduction

Motion gestures play a very important role in general human– computer interaction. Motion gestures are meant to be simple so thatauser can easily memorize and perform them. Air-writing will be useful in user interfaces that do notallow the user to type on a keyboard or write on a track pad/touch screen, or for text input for smart system control, among many applications.

This model is very useful in the establishment of a user-interface. The input is provided by the user in the form of a gesture. This method may be classified as a new method of recognising sign language. The movement of the target tip is traced and it is compared to the model which is prepared after training through the dataset.

When a Deaf or Dumb person wishes to exchange their view using a sign language/, Other people might not always understand them. They can use our software to communicate with other people or for interacting with a computer.

This can also be used by normal people for contact less interaction with computers. Computers in public places like ticketing systems, virtual maps etc. can use our method for contact-less interaction which will prevent the spread of germs.

Compared to other non-traditional input methods such as typing with a virtual keyboard or similar schemes, airwriting offers the advantage of “eye-free” execution, requiring minimum attention focus. Air-writing provides a viable alternative interface for text input, particularly when conventional input devices, such as a keyboard or a mouse, are not available or suitable.

The scope of this project can be widened by using it for applications like Ticketing Systems, Information Booths in public places, etc. It can also be utilized by deaf or dumb people as an alternative for sign language and converse in conventional language.

3. Literature Survey

Authors and year	Title	Concept	Methodology	Dataset analysis	Relevant Finding	Limitations
Mingyu Chen, Ghassan AlRegib, BiingHwang Juang(2016)	Air-writing Recognition, Part 1: Modeling and Recognition of Characters, Words and Connecting Motions	6-DOF air writing method	6-DOF method air writing using character recognition	UCI dataset	Air writing motion tracking	It is not very cost effective
Mingyu Chen, Ghassan AlRegib(2015)	Air-writing Recognition, Part 2: Detection and Recognition of Writing Activity in Continuous Stream of Motion Data	LEAP from leap motion for finger tracking	Air writing using character recognition	UCI dataset	Controller free air writing	No real time application
Deepa D,R.Dharmalingam(2017)	Feature and processing of recognition characters,words and connecting motion	HMM models, 6 DOF motion tracking	Isolation of air writing objects to easily detect it in air, 2D motion is detected and tested	-	Air writing with motion tracking system	Average speed of writing is slower than that of keyboard and difficult to implement
Nithya.S, Chandru.S,Krishnakanth.G, Pavithran.P(2018)	An Optimized approach foro deaf and dumb people using air writing	OpenCV and OCR method	Character recognition using optical character recognition	-	-	Not cost effective

Xin Zhang; Zhichao Ye; Lianwen Jin; Ziyong Feng; Shaojie Xu(2013)	A New Writing Experience: Finger Writing in the Air Using a Kinect Sensor	DSB-MM, MQDF for finger writing detection	Hand segmentation using depth and colour sequences uses ANN based DSB-MM for hand segmentation and dual mode switching algorithm for writing	-	Handwriting character recognition method	Non synchronization. Contains large falsely segmented area for a moving hand because of colour depth non synchronization issue
Kai Xing, Zhen Ding, Chifu Yang, Jing-Hao San (2019)	Research on Hand gesture recognition based on deep learning	Accuracy of EMG-based hand gesture recognition	A parallel architecture with five convolution layers is adopted	-	Handwriting character recognition method	Negative effect on some of our control algorithms and does not give good results and this can also affect the convergence of the system due to the parallel option.
Prasun Roy, Subhankar Ghosh, and Umapada Pal(2018)	A CNN Based Framework for Unistroke Numeral Recognition in Air-Writing	6-DOF air writing method, color based segmentation ,transfer learning	6-DOF method air writing using character recognition, color based segmentation to identify the marker and track the trajectory of marker tip	-	Marker Segmentation and character recognition	The proposed framework is not directly comparable with most of the previous works because of difference in evaluation methods and types of datasets

4. Problem Statement

The main objective of this project is to identify the Tamil alphabets captured from airwriting by the computer using convolutional neural network and computer vision. Here, the computer needs to identify the Tamil alphabets which were written by the human in the air. The computer needs to observe the pattern captured with our stylus movements that were drawn by us using a camera, then parse the letters and recognize the letters drawn and display it back in the python terminal.

5. Problem Solution

a. Dataset

i. This dataset contains approximately 500 isolated samples each of 156 Tamil “characters” (details) written by native Tamil writers including school children, university graduates, and adults from the cities of Bangalore, Karnataka, India and Salem, Tamil Nadu, India. The data was collected using HP TabletPCs and is in standard UNIPEN format.

ii. **Link:** http://lipitk.sourceforge.net/datasets/tamilchardata.htm

b. Input

i. The user traces the Tamil letter using a stylus tip in front of the webcam

c. Approach

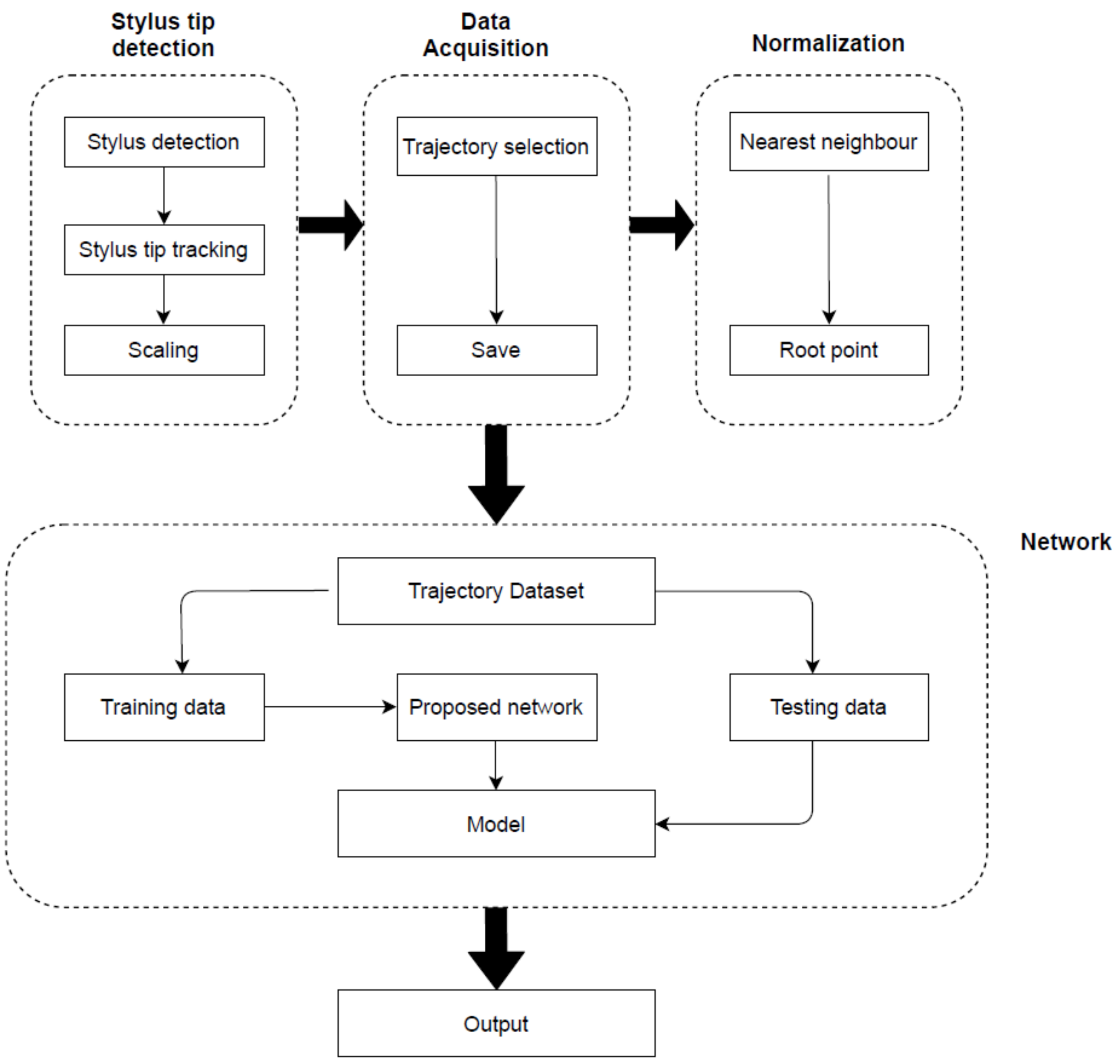
i. The user first isolates stylus tip using openCV and obtains the hue,saturation and value
ii. Then, the isolated hsv coordinates are fed to another python program to trace the stylus tip.
iii. The user traces the Tamil letter using a stylus tip in front of the webcam
iv. The CNN predicts the Tamil letter

6. Novelty

We have built a Tamil letter recognizer based off of an english letter recognizer. We have built a simple parser to separate the letters in a word. Trained the model with limited number of images and prevented overfitting by a combination of regularization , dropouts and early stopping.

7. Architecture

7.1 Design



7.1 Architecture Description:

➔ Stylus tip detection,

- ◆ can be performed using a webcam from your laptop/desktop.
- ◆ is used to detect a trajectory which is traced by the tip.
- ◆ to fit and display the trajectory within the virtual window, scaling is required.

➔ Data acquisition,
- ◆ we follow a method known as the ‘push-to-write’ approach to collect the data.
- ◆ letters must be traced in front of the web camera to be collected as a spatial trajectory sequence.

➔ Normalization,
- ◆ Nearest neighbor point normalization technique is used to detect the user's trajectory shape using displaced points as it is not a pen up/down system.
- ◆ To transform the displaced points into a smooth straight line,nearest neighbor point normalization is used.
- ◆ User’s usually write the letters in an imaginary virtual box which doesn’t have fixed boundaries.So the users have the tendency to overwrite the letters.
- ◆ In order to deal with this situation Root point normalization method to detect the initial point for each letter.

➔ Neural Network design,
- ◆ To train the model, we split the dataset as 80/20.
- ◆ 80% as a training set and 20% as a testing set.
- ◆ CNN network consists of input and output layers composed of multiple hidden layers.
- ◆ Hidden layers include convolution, pooling, and activation layers.

8. Detailed Module Design

Module 1

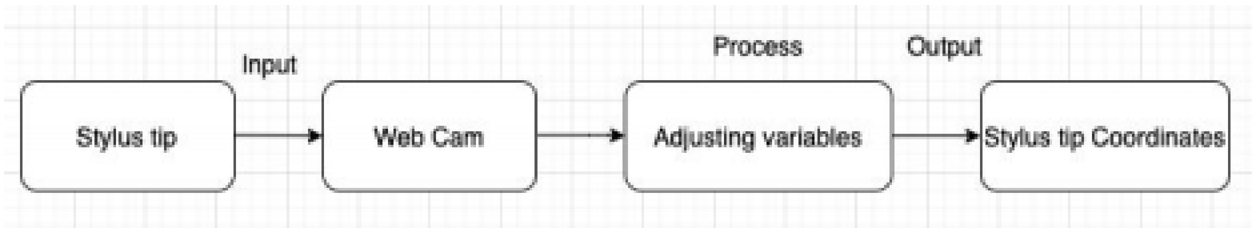
- Stylus Tip Detection
    - **Actor:** User
    - **Description:**

        - The user presents the stylus tip in front of the webcam for its coordinates to be detected by the program.

        - The HSV program isolates the coordinates for the stylus tip which was presented. Which is in inturn used as the starting point for the trajectory which is to be traced.

        - The webcam captures the video feed of the stylus tip being traced using python's openCV library which can be processed into the desired image and can be fed into the CNN.

- **Block Diagram**

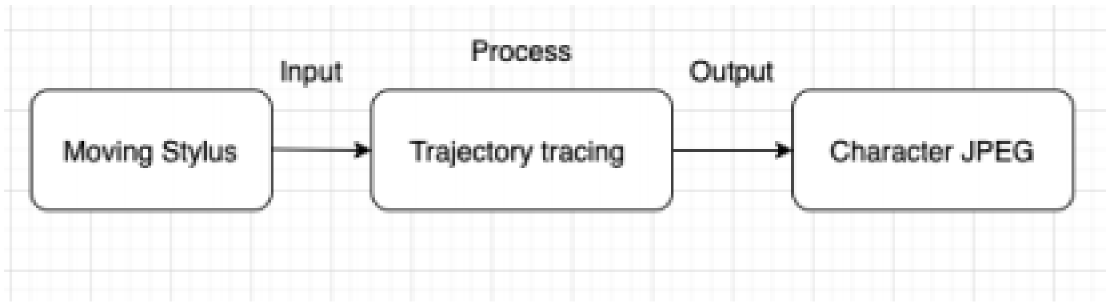


Module 2

- Stylus Tip Tracking
    - **Actor:** User
    - **Description**

        - The user maneuvers the stylus, tracing the character. And hsv values are imported from module 1 to track the stylus tip.
        - The HSV program isolates the coordinates for the stylus tip which was presented. Which is in inturn used as the starting point for the trajectory which is to be traced.
        - The trajectory is traced.
        - The webcam captures the video feed of the stylus tip being traced using python's openCV library.
        - The traced character is saved as a jpg.

- **Block Diagram**

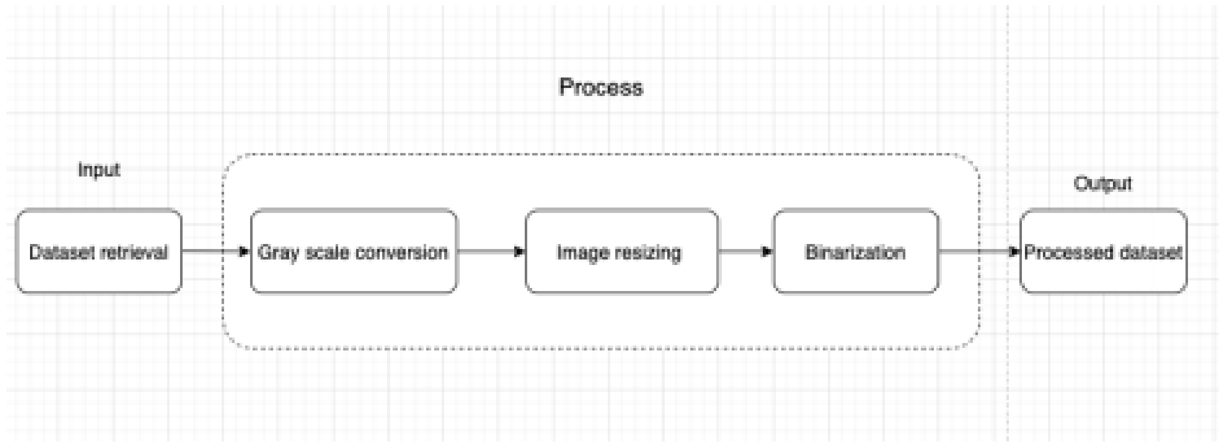


Module 3

- Pre Processing Dataset
    - **Input:** The data set is retrieved from:

      http://lipitk.sourceforge.net/datasets/tamilchardata.htm
    - **Output:** Processed Dataset
    - **Actor:** User
    - **Description:**
        - Convert images in the dataset to grayscale using cvtColour function to be usable by threshold function.
        - Image Resizing(Normalising all pictures to a standard set of pixels)
        - Binarization,threshold function does a binary assignment of the pixels to produce the desired threshold image
        - The processed data set is produced as a result.

- **Block Diagram**



Module 4

- Training Neural Network
    - **Input:** The processed training data
    - **Output:** Trained Neural network
    - **Actor:** User
    - **Process**
        - Here the convolutional neural network learns based on the inputs provided by the preprocessed dataset from the previous module which is fed to the neural network in order to obtain the desired result.
    - **Description**
        - **Conv2D layer**
            - This layer creates Convolution Kernel convolved with the layer input to produce a tensor of outputs
        - **Max pooling layer**
            - It is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.
        - **Dropout Layer**

    - It randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting

- **ReLU layer**

    - It is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero and ‘softmax’ activation function is used to determine

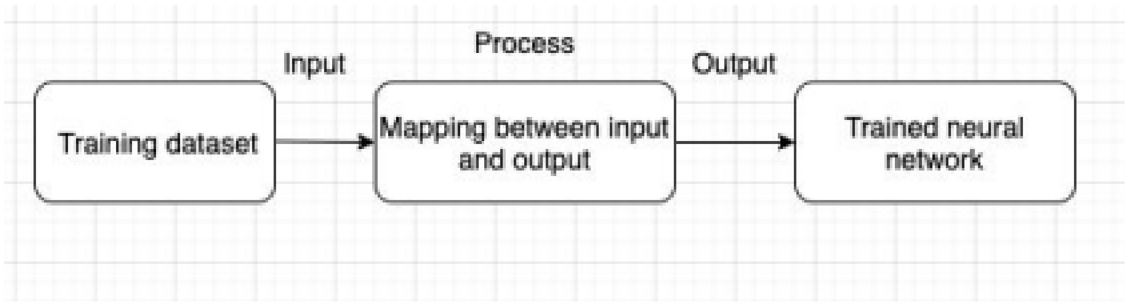
- **Softmax layer**

    - It assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0

- **Training (Epochs)**

    - One Epoch is when an entire dataset is passed forward and backward through the neural network only once. Since one epoch is too big to feed to the computer at once we divide it in several smaller batches.

- **Block Diagram**



9. Implementation

a. Initial setup

i. Download anaconda in your system.
ii. Within the anaconda navigator, download Jupyter notebook and Spyder.
iii. Libraries such as tensorflow, numpy and openCV are downloaded onto the environment.
iv. The Jupyter notebook is used to code the neural network whereas Spyder is used to obtain hsv coordinates and user input.

b. Tools

i. Python 3.3.7
ii. Tensorflow 2.2.0 3)
iii. opencv-python 4.2.0.32 4)

iv. numpy 1.18.2 5)

v. Tifffile 2020.5.30

vi. Pickleshare(0.7.5 7)

vii. Web-cam

viii. CUDA

ix. CuDNN

x. Nvidia GPU

xi. Google colab

c. Code Snippets

i. Module 1

```
# importing libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
import keyboard
import sys
import tensorflow as tf
import os
from tensorflow.keras.models import load_model



#Creating track bar for Hue Saturation Value
cv2.createTrackbar('h', 'result',0,179,nothing)
cv2.createTrackbar('s', 'result',0,255,nothing)
cv2.createTrackbar('v', 'result',0,255,nothing)

#converting to HSV
hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
```

# get info from track bar and apply to result

```
h = cv2.getTrackbarPos('h','result')
s = cv2.getTrackbarPos('s','result')
v = cv2.getTrackbarPos('v','result')
```

#Detect the object based on HSV Range Values

```
mask = cv2.inRange(hsv,lower_blue, upper_blue)
```

#Extracting desired image using two inputs

```
result = cv2.bitwise_and(frame,frame,mask = mask
```

ii. Module 2

#Detecting colours within the range specified as lb and ub

```
mask = cv2.inRange(frame2, lb, ub)
```

#Reducing noise

```
mas = cv2.morphologyEx(mask, cv2.MORPH_OPEN, Kernal)
```

#Connects the plotted points.

```
contours, hierarchy = cv2.findContours(mas,
cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

#detects the smallest possible circle enclosing the character

```
(x, y), radius = cv2.minEnclosingCircle(cnt)
            x = int(x)
            y = int(y)
            x1.append(x)
            y1.append(y)
```

iii. Module 3

#This stores the data from array X and Y onto the file

```
import pickle
p1=open('tam.pickle','wb')
pickle.dump([X,Y],p1)
```

#colour space conversion

```
x = cv2.cvtColor(x, cv2.COLOR_BGR2GRAY)
```

#resizing the dataset images to 100x100

```
x=cv2.resize(x,(100,100))
(thresh, x) = cv2.threshold(x, 128, 255,
cv2.THRESH_BINARY|cv2.THRESH_OTSU)
```

#rescaling

```
x=x/255
x=x.astype('uint8')
```

iv. Module 4

#groups a linear stack of layers into a tf.keras.Model.

```
model=tf.keras.models.Sequential()
```

#creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs

```
model.add(Conv2D(64,(3,3),input_shape=X.shape[1:]))
```

#Applies the rectified linear unit activation function

```
model.add(Activation("relu"))
```

#returns a tensor of rank 4 representing the maximum pooled values

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

#prevents overfitting

```
model.add(Dropout(0.2))
```

#converts a real vector to a vector of categorical probabilities

```
model.add(Activation('softmax'))
```

#converting the data into a 1-dimensional array

```
model.add(Flatten())
```

#every input is connected to every output by a weight

```
model.add(Dense(18))
```

#defining the loss function, optimizers, and metrics for prediction

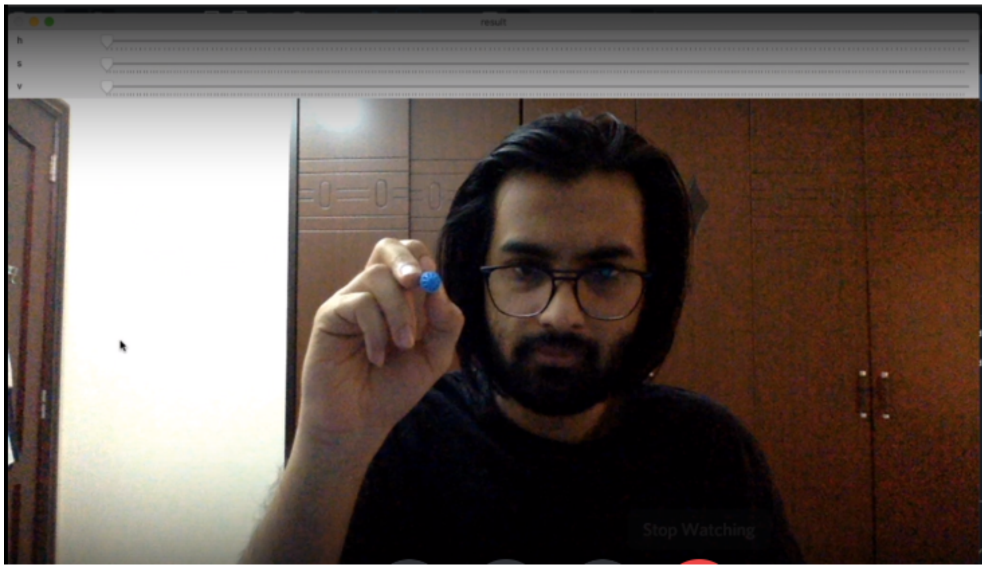
```
model.compile(loss="sparse_categorical_crossentropy",optimize
r="adam",metrics=['accuracy'])
```

# training the model with the given inputs

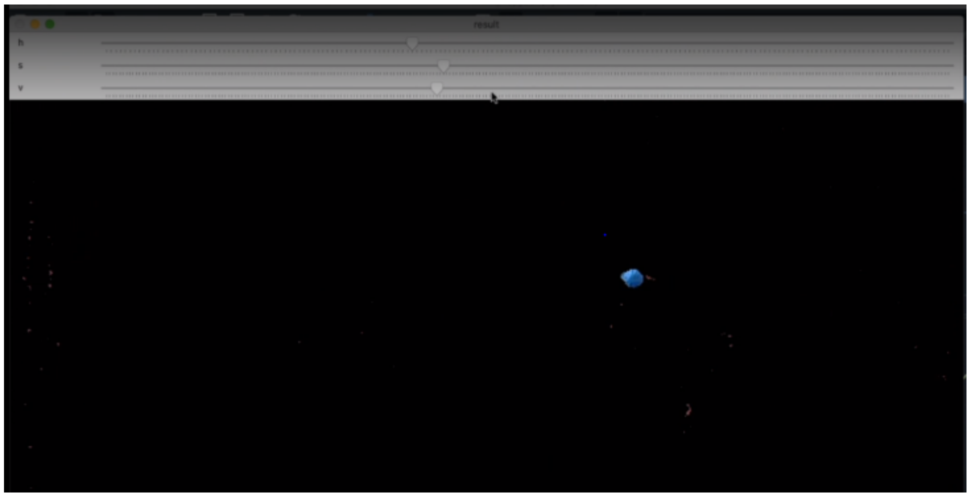
```
model.fit(X,Y,batch_size=16,validation_split=0.1,epochs=25,ve
rbose=2)
```

d. Procedure

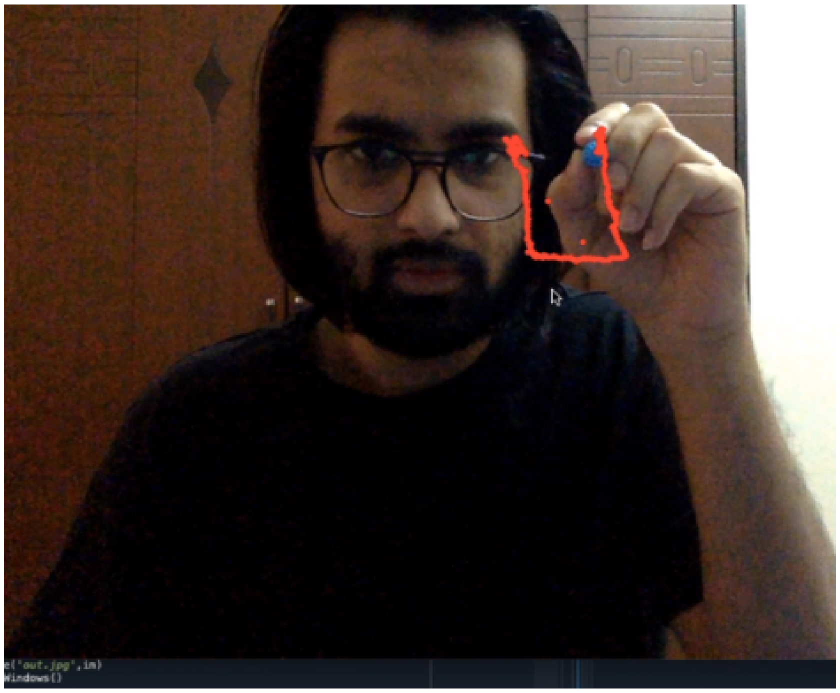
- Presenting Stylus tip



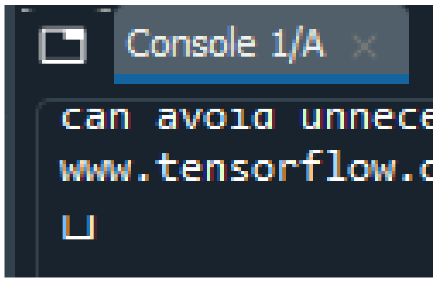
- Isolating Stylus tip



- Tracing Tamil letter



- Feeding it to the neural network



```
can avoid unnece
www.tensorflow.c
ட
```

10. Support information

a. https://colab.research.google.com/drive/183mIWG8ntyJmjzwc84ZHUbHdhg728-hO?usp=sharing

b. https://docs.anaconda.com/anaconda/user-guide/tasks/tensorflow/

c. https://docs.opencv.org/master/dd/d6e/tutorial_windows_visual_studio_opencv.html

d. https://machinelearningmastery.com/save-load-keras-deep-learning-models/

e. https://docs.anaconda.com/anaconda/install/

11. Surveyed content

a. https://en.wikipedia.org/wiki/Handwriting_recognition

b. https://www.tensorflow.org/tutorials/images/cnn

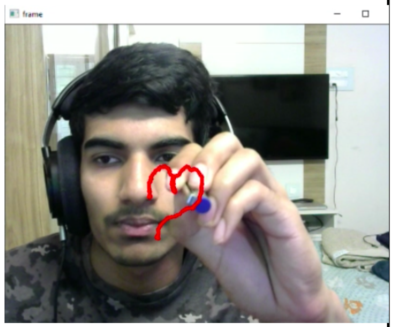
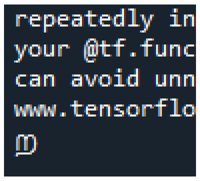
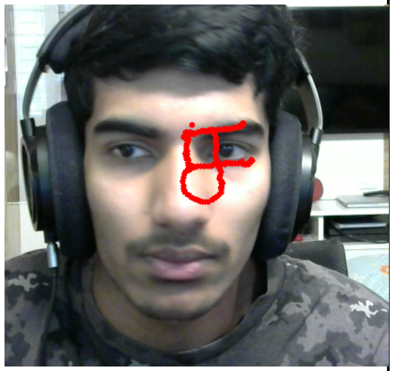
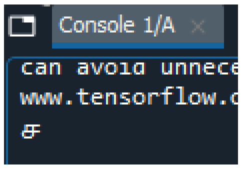
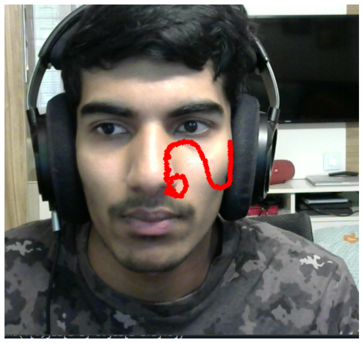
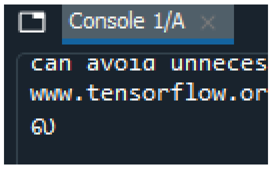
c. https://stackoverflow.com/questions/10948589/choosing-the-correct-upper-and-lower-hsv-boundaries-for-color-detection-withcv

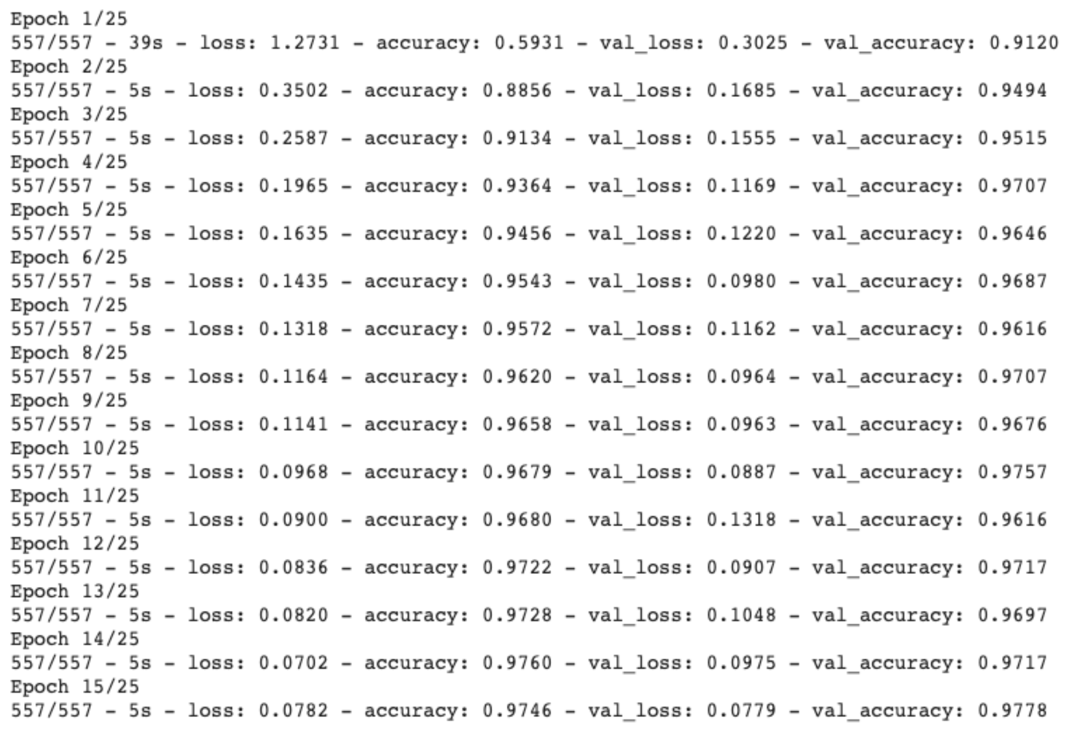
d. https://medium.datadriveninvestor.com/convolutional-neural-network-cnn-simplified-ecafd4ee52c5

e. https://www.geeksforgeeks.org/keyboard-module-in-python/

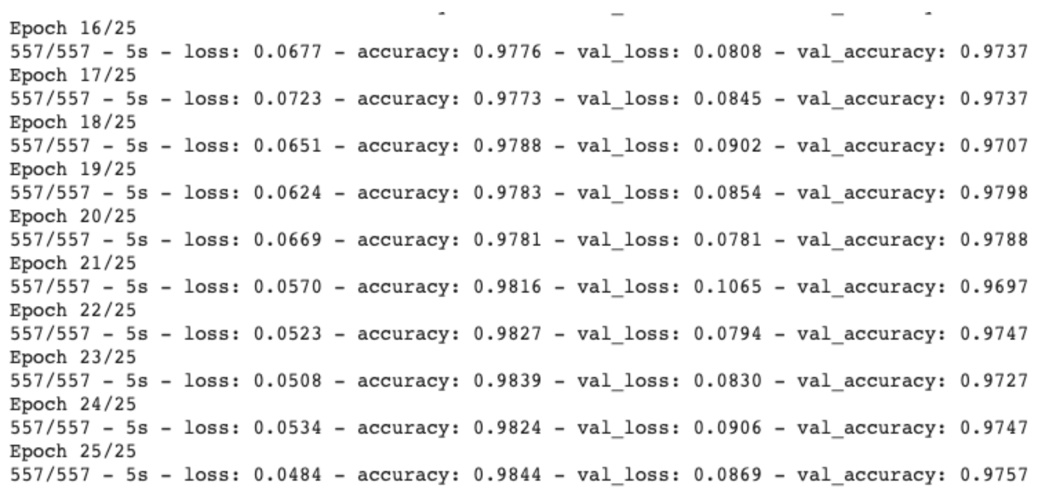
12. Results and Comparison

Test Case Table

S.No	Input	Expected output	Predicted output
1		ஐ	 repeatedly in your @tf.func can avoid unn www.tensorflo ஐ
2		ச	 can avoid unnece www.tensorflow.c ச
3		ல	 can avoid unneces www.tensorflow.or ல

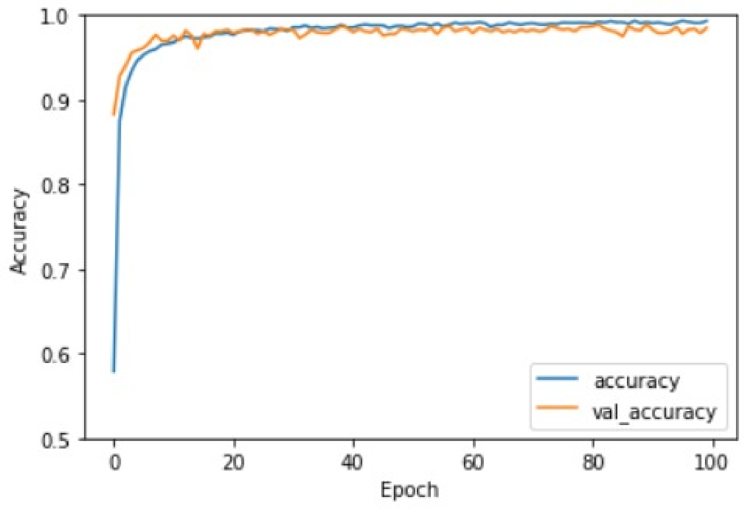


```
Epoch 1/25
557/557 - 39s - loss: 1.2731 - accuracy: 0.5931 - val_loss: 0.3025 - val_accuracy: 0.9120
Epoch 2/25
557/557 - 5s - loss: 0.3502 - accuracy: 0.8856 - val_loss: 0.1685 - val_accuracy: 0.9494
Epoch 3/25
557/557 - 5s - loss: 0.2587 - accuracy: 0.9134 - val_loss: 0.1555 - val_accuracy: 0.9515
Epoch 4/25
557/557 - 5s - loss: 0.1965 - accuracy: 0.9364 - val_loss: 0.1169 - val_accuracy: 0.9707
Epoch 5/25
557/557 - 5s - loss: 0.1635 - accuracy: 0.9456 - val_loss: 0.1220 - val_accuracy: 0.9646
Epoch 6/25
557/557 - 5s - loss: 0.1435 - accuracy: 0.9543 - val_loss: 0.0980 - val_accuracy: 0.9687
Epoch 7/25
557/557 - 5s - loss: 0.1318 - accuracy: 0.9572 - val_loss: 0.1162 - val_accuracy: 0.9616
Epoch 8/25
557/557 - 5s - loss: 0.1164 - accuracy: 0.9620 - val_loss: 0.0964 - val_accuracy: 0.9707
Epoch 9/25
557/557 - 5s - loss: 0.1141 - accuracy: 0.9658 - val_loss: 0.0963 - val_accuracy: 0.9676
Epoch 10/25
557/557 - 5s - loss: 0.0968 - accuracy: 0.9679 - val_loss: 0.0887 - val_accuracy: 0.9757
Epoch 11/25
557/557 - 5s - loss: 0.0900 - accuracy: 0.9680 - val_loss: 0.1318 - val_accuracy: 0.9616
Epoch 12/25
557/557 - 5s - loss: 0.0836 - accuracy: 0.9722 - val_loss: 0.0907 - val_accuracy: 0.9717
Epoch 13/25
557/557 - 5s - loss: 0.0820 - accuracy: 0.9728 - val_loss: 0.1048 - val_accuracy: 0.9697
Epoch 14/25
557/557 - 5s - loss: 0.0702 - accuracy: 0.9760 - val_loss: 0.0975 - val_accuracy: 0.9717
Epoch 15/25
557/557 - 5s - loss: 0.0782 - accuracy: 0.9746 - val_loss: 0.0779 - val_accuracy: 0.9778
```

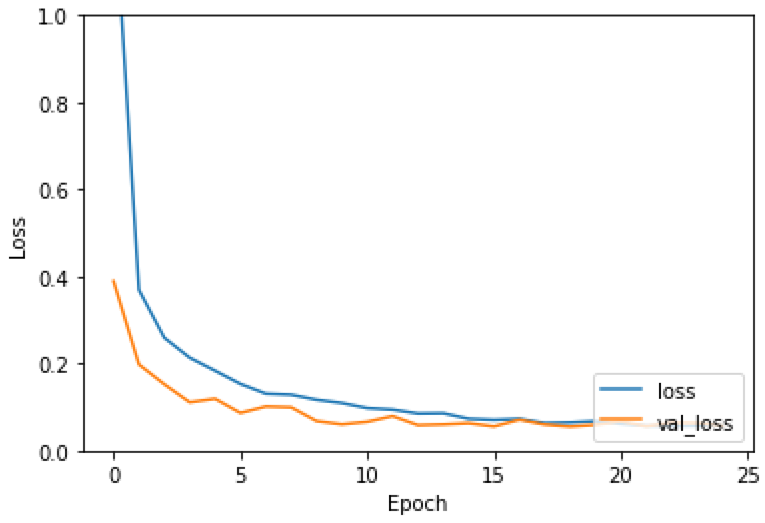


```
Epoch 16/25
557/557 - 5s - loss: 0.0677 - accuracy: 0.9776 - val_loss: 0.0808 - val_accuracy: 0.9737
Epoch 17/25
557/557 - 5s - loss: 0.0723 - accuracy: 0.9773 - val_loss: 0.0845 - val_accuracy: 0.9737
Epoch 18/25
557/557 - 5s - loss: 0.0651 - accuracy: 0.9788 - val_loss: 0.0902 - val_accuracy: 0.9707
Epoch 19/25
557/557 - 5s - loss: 0.0624 - accuracy: 0.9783 - val_loss: 0.0854 - val_accuracy: 0.9798
Epoch 20/25
557/557 - 5s - loss: 0.0669 - accuracy: 0.9781 - val_loss: 0.0781 - val_accuracy: 0.9788
Epoch 21/25
557/557 - 5s - loss: 0.0570 - accuracy: 0.9816 - val_loss: 0.1065 - val_accuracy: 0.9697
Epoch 22/25
557/557 - 5s - loss: 0.0523 - accuracy: 0.9827 - val_loss: 0.0794 - val_accuracy: 0.9747
Epoch 23/25
557/557 - 5s - loss: 0.0508 - accuracy: 0.9839 - val_loss: 0.0830 - val_accuracy: 0.9727
Epoch 24/25
557/557 - 5s - loss: 0.0534 - accuracy: 0.9824 - val_loss: 0.0906 - val_accuracy: 0.9747
Epoch 25/25
557/557 - 5s - loss: 0.0484 - accuracy: 0.9844 - val_loss: 0.0869 - val_accuracy: 0.9757
```

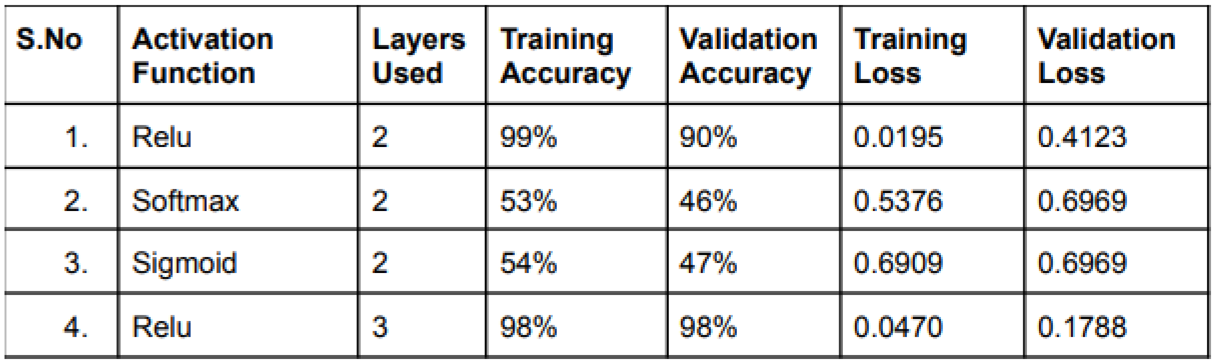
Graph (between accuracy and epoch)



Graph (between loss and epoch)



Tuning hyperparameters



S.No	Activation Function	Layers Used	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
1.	Relu	2	99%	90%	0.0195	0.4123
2.	Softmax	2	53%	46%	0.5376	0.6969
3.	Sigmoid	2	54%	47%	0.6909	0.6969
4.	Relu	3	98%	98%	0.0470	0.1788

13. Conclusion and future enhancements

We achieved a training accuracy of 99.38% and validation accuracy of 98.48% but the real world performance of the app doesn't match it as the dataset we used wasn't meant for airwriting. The dataset was originally captured in a hp writing tablet. The writing style of that and air writing is quite different. This will be good for recognising letters written in pen and paper but not our purpose. But after a lot of optimization we got a decent level of character recognition (Which you can see in the link that we have added in Screenshot and Demo). We could have got better performance by creating our own dataset for air-writing but we don't have enough manpower. This will have a better scope if we use a dataset that is more suitable for the purpose.

14. List of references

a. Research on Hand Gesture Recognition Based on Deep Learning. Jing-Hao San 07 February 2019
b. Visual Gesture Recognition for Text writing in Air. 11 March 2019
c. A CNN Based Framework for Unistroke Numeral Recognition in Air-Writing. 20 December 2018
d. An optimized approach for deaf and dumb people using air writing Nithya.S, 2Chandru.S, 3Krishnakanth.G, 4Pavithran.P. 04 April 2018
e. Real-Time Recognition of Sign Language Gestures and Air-Writing using Leap Motion. 20 July 2017
f. Air-writing Recognition, Part 1: Modeling and Recognition of Characters, Words and Connecting Motions Mingyu Chen, Ghassan AlRegib, Senior Member, IEEE, and BiingHwang Juang, Fellow, IEEE. 09 November 2015
g. Beyond Human Recognition: CNN-Based Framework for Handwritten Character Recognition. November 2015
h. Hand Gesture Recognition for Indian Sign Language. 01 March 2012
i. https://github.com/heycrystal/air-writing-recognition
j. https://github.com/akshaybahadur21/Alphabet-Recognition-EMNIST
k. https://data-flair.training/blogs/handwritten-character-recognition-neural-network/
l. https://github.com/kosmasK/air-writing-recognition