**5. LITERATURE SURVEY**

**4] Visual Gesture Recognition for Text Writing in Air**

The authors in [4] solved problems faced by elderly people to type text on mobile phones by detecting gestures in air. In this system, a combination of computer vision and convolution neural networks is used for detecting drawn gesture and recognizing it. Few of the methods employed, use coloured fingertip for tracking the motion of the finger. Their proposed application supports drawing of gestures and writing of English text over air in front of the mobile camera by using bare fingertip. Convolution neural networks has been used for character recognition. The method presented can be applicable to other fields requiring hand gesture recognition for distant interaction with mobiles and computers. Their proposed System does not require sensors or any hardware other than the camera.

**2] Air-swipe gesture recognition using OpenCV in Android devices**

In [2], authors introduce Air-Swipe Gesture Recognition System which can be useful to enable user to make In-Air gestures in front of the camera and to do different operations. This System can give a user-friendly and a live-experience of interaction and visualization, enhancing the usability and making the android device more interactive. It does not require any hardware changes instead only uses the native camera of the device and a machine learning software such as Open Source Computer Vision (OpenCV) algorithms to detect the changes in environment and respond accordingly in varying conditions. They tested this classification and found out the result that considering the frames to be divided into quadrants along x-axis and y-axis and found that the value of the frame matrix changes. Their approach has the capability of recognizing gestures with precision of almost 96%.

**3] Text Recognition by Air Drawing**

In this paper [3], the text drawn by the user in the air is captured by the computer’s camera, followed by the identification of that text. So, the video camera will be turned on at the time of capturing the written text. Now the object is defined based on its colour to detect a movement done by the user. The colour is captured by the lower and upper bound of HSV (Hue Saturation Value), which finally leads to object detection at every instant. Lastly, the text will be recognized by the trained model. The model is trained by CNN (Convolution Neural Network) with an accuracy of 98.64% (training) and 98.24% (testing). For completion of the project, OpenCV, python programming language, and its libraries are used. This project requires only a camera and a defined object.

**6. REFERENCES**

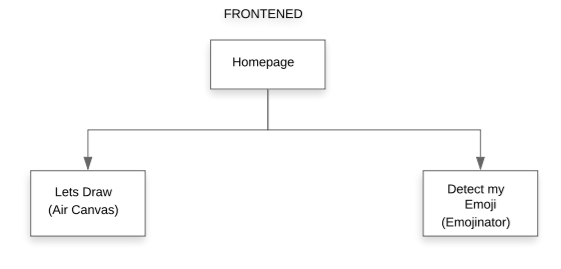
1] V. Joseph, A. Talpade, N. Suvarna and Z. Mendonca, "Visual Gesture Recognition for Text Writing in Air," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018, pp. 23-26, doi: 10.1109/ICCONS.2018.8663176

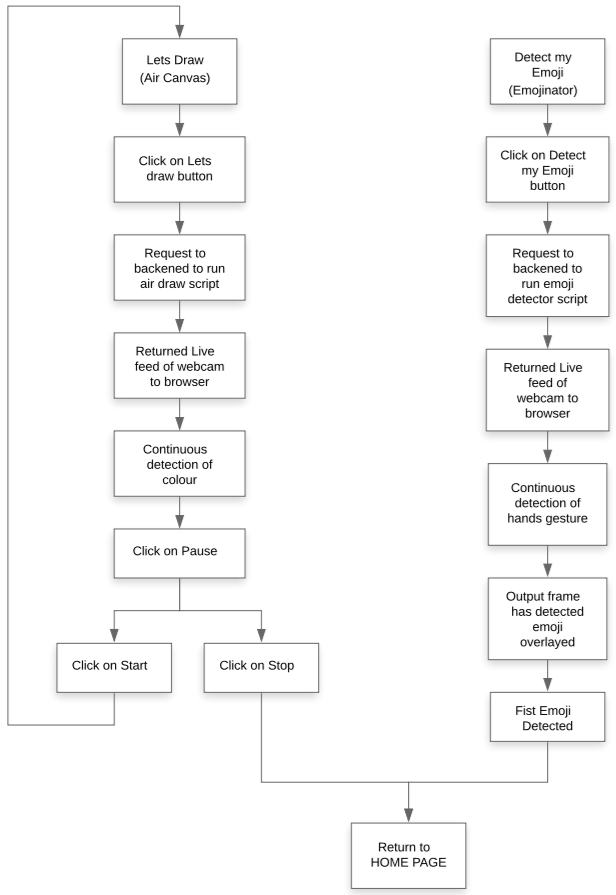
2] T. Sharma, S. Kumar, N. Yadav, K. Sharma and P. Bhardwaj, "Air-swipe gesture recognition using OpenCV in Android devices," 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), 2017, pp. 1-6, doi: 10.1109/ICAMMAET.2017.8186632.

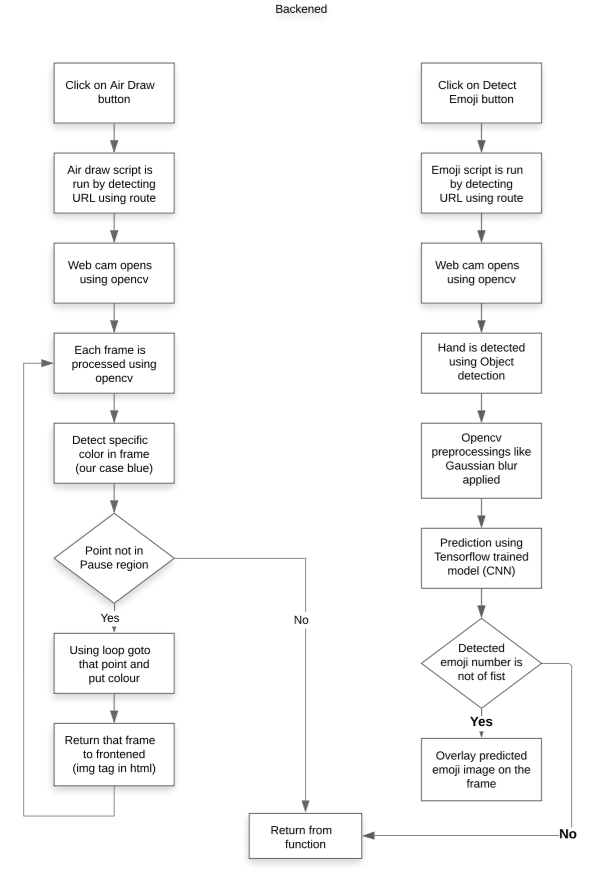
3] J. Patel, U. Mehta, K. Panchal, D. Tailor and D. Zanzmera, "Text Recognition by Air Drawing," 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT), 2021, pp. 292-295, doi: 10.1109/CCICT53244.2021.00061.

**CHAPTER-3**

**5. PROPOSED SYSTEM DESIGN ARCHITECTURE**







**5. ARCHITECTURE EXPLANATION**

**Frontend-**

**Homepage-** In homepage, two buttons are provided which can be selected by mouse click. “Let’s draw” button runs the air canvas script which further opens webcam feed into the browser enabling the user to draw in the air. Second button is “Detect my emoji” which enables users to run emoji detector script. After clicking on this button, webcam feed is shown in browser detecting real time emoji gestures made by hand.

**Air Draw page-** When Let’s draw button is clicked, a request to blackened is made to display live feed of webcam. After clicking, on next page, live feed is shown with different options to select different colours and start or stop the feed. When detecting colour is shown (in our case as blue), it is detected and successfully draws on the screen shown. On clicking stop, feed is stopped the given point and clicking on start, resumes it again. Back button takes back to homepage.

**Emoji Detector page-** When Detect my emoji button is clicked, a request to backend is made to run the model and show the detected emoji as well the live webcam feed on to the browser. On clicking, it successfully shows the feed on the page with the detected emoji over laying on the same feed.

**Backend-**

**Main page script-**

**Air Draw script-** When URL route matches with video URL, video.html file gets rendered and webpage having live webcam feed is sent to frontend using flask. The code function which produces webcam feed uses open cv and performs many other pre-processing to detect the colour of the pen.

In that function, first a trackbar is created to set hue and saturation to detect blue colour (or any colour of our choice using which we will write in air canvas). Frames are generated using open cv and each frame is pre-processed using different open cv techniques. Different rectangles of different colours are drawn on each of the captured frames using which different colours can be selected. Different open cv pre-processing like Eroding, Morphology, Dilation are used to improve detection of the colour. Then contour detection is done to detect specific point of the colour in the whole pixel range. Detected point is checked using if else that it lies in what region. According to region different colours are selected, similarly every frame of the video feed is processed and passed to frontend that combine and forms complete video.

**Emoji Detector page-** When URL route matches with emoji URL, emoji.html file gets rendered and webpage having live webcam feed is sent to frontend using flask. The code function which produces webcam feed uses open cv and performs many other pre-processing to detect the emoji made by hand gesture.

In that function, first a frame is extracted from the video using open cv, it is converted from BGR to RGB. Object detection algorithm is run on it to detect the required bounding box around the hand. In our case, TensorFlow Object Detection API is used to detect the object. Other open cv pre-processing techniques like Bitwise, Gaussian Blur, Morphology, and Dilation are used on the respective frame. After all the pre-processing, prediction using already trained Keras model is done using a predict function. The predicted emoji class number and the prediction probability are stored in the variables. Image of predicted emoji is extracted from one of the folders having emojis pictures and their class numbers. Emoji is then overlayed on the frame using overlay function. After all the steps, frame is sent to frontend to show it as video.

Model is trained using CNN (Convolutional Neural Network). CNN’s are the state-of-the-art algorithms for training on image data, their architecture learns faster and more accurately when the training data are images. Standard layers like convolution, max pool, dense are used to make the model of CNN for our project. The model is trained for 10 epochs, and it gets converged in such fewer number of epochs.

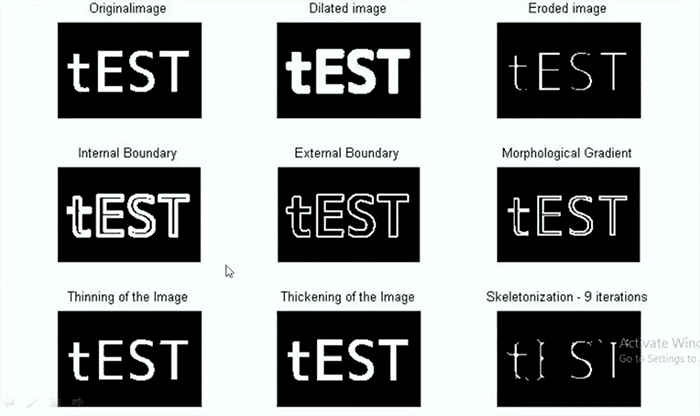
**5. ALGORTIHMS AND PSEUDOCODE**

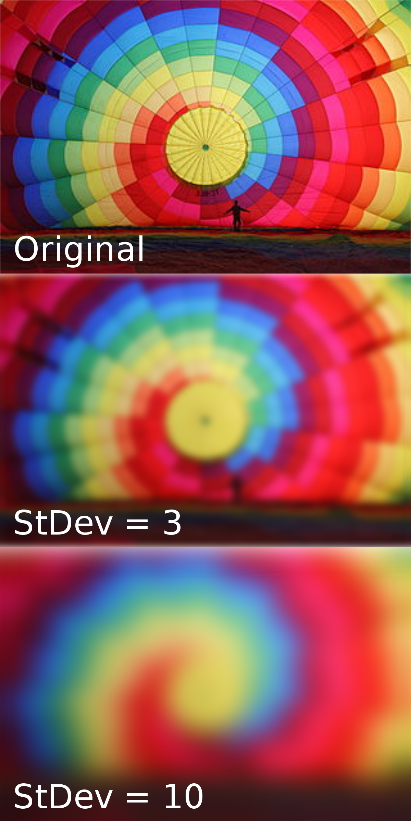
**Algorithms-**

Erosion and Dilation- The most basic morphological operations are dilation and erosion. Dilation **adds pixels to the boundaries of objects in an image**, while erosion removes pixels on object boundaries. The rule used to process the pixels defines the operation as a dilation or an erosion.

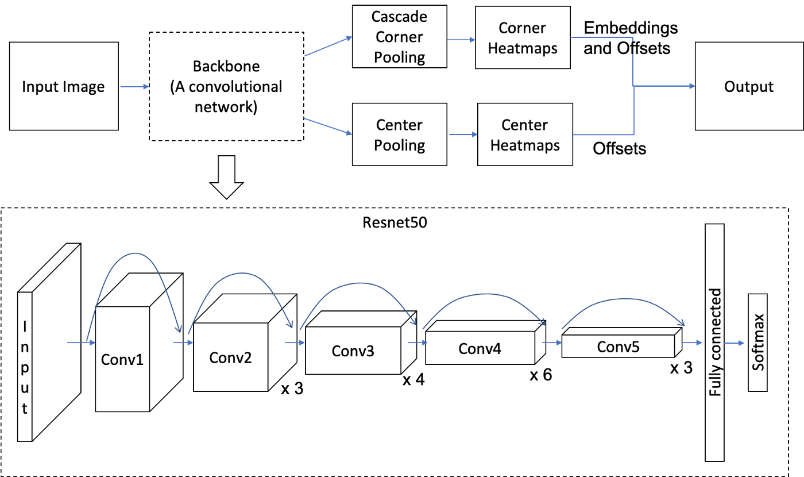


Morphology- Morphological operations are simple transformations applied to binary or grayscale images. We can use morphological operations to **increase the size of objects in images as well as decrease them**.

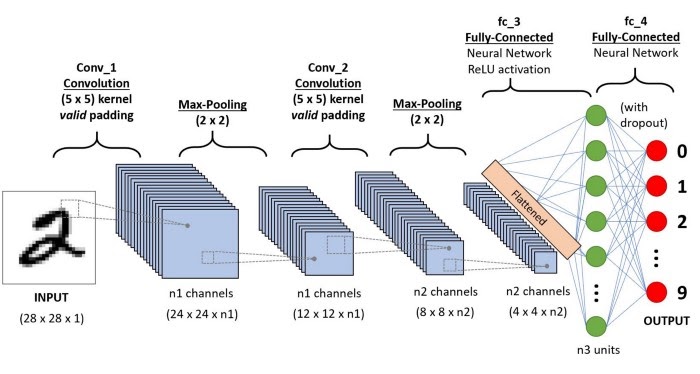


Gaussian Blur- In image processing, a Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function (named after mathematician and scientist Carl Friedrich Gauss). It is a widely used effect in graphics software, **typically to reduce image noise and reduce detail**.

Tensorflow Object Detection- Object Detection using Tensorflow is **a computer vision technique**. As the name suggests, it helps us in detecting, locating, and tracing an object from an image or a video.



Convolutional Neural Network (CNN)- A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vison that includes image and video recognition, along with recommender systems and natural language processing (NLP).



**Pseudocode-**

Frontened-

1. Homepage load
   1. If (Click on Let’s Draw button)
      1. Choose colour from shown
      2. Draw on screen
      3. If (click on pause)
         1. Click on Start
            1. Repeat same process again
         2. Click on Stop
            1. Return to Home Page
   2. Else (Click on Detect my emoji button)
      1. Make hand gesture
      2. Predicted emoji shown on webcam
      3. If (Show fist emoji)
         1. Feed stops
      4. If (Click on Back)
         1. Return to Home Page

Backened-

If (URL route == “video”)

1. Runs Air Draw script
2. Processing each frame
3. Applying all algorithms
4. Detect specific colour

If (point region == blue)

Colour=blue

Else if (point region == yellow)

Colour=yellow

Else if (point region == red)

Colour=red

Else if (point region == green)

Colour=green

Else if (point region == clear all)

Colour=no colour

Else if (point region == stop)

Break

1. If (button == back)

Return to main page

Else if (button == start)

Refresh the same page

1. Use loop to put colour up to that point region
2. Return the modified frame
3. All frame combines to make the video feed

Else if(URL route == “emoji”)

1. Runs Emoji detect script
2. Colour of detected frame is changed to RGB using open cv
3. Object detection algorithm is run (detect object function)
4. Draw box on image function is run to draw box
5. Gaussian Blur is applied using open cv
6. Similarly, morphology and dilation are applied
7. Thresholding is done using open cv
8. Image resized to 50 by 50
9. Prediction using keras.model.predict
10. If (pred class! = 10)
    1. Overlay emoji on the frame
    2. Convert it to jpg
    3. Yield it and send to frontend in img tag

Else

break