Real-time Hand Gesture Recognition for Computer Interaction

Aditi(180001002), Yuvnish Malhotra(180002065), Harshil Bhavsar(180001019)

Introduction

Hand gestures are an aspect of body language that can be conveyed through the centre of the palm, the finger position and the shape constructed by the hand. Hand gestures can be classified into static and dynamic. Gesture recognition is a subtopic in computer vision with the goal of interpreting human gestures via mathematical algorithms. The project aims at Hand Gesture Recognition. Hand gestures can be classified into static and dynamic. For the former, the gesture of the hand denotes a sign while for the latter, the movement of the hand conveys some messages. We will be aiming at recognizing dynamic gestures to control certain functions of media players. The motive is to help users control or interact with devices without physically touching them. The main aim of this real-time hand gesture recognition application is to classify and recognize the gestures.

Tech Stack Used

- Python==3.8.5
- tensorflow_cpu==2.4.1
- numpy==1.19.5
- pycaw==20181226
- screen_brightness_control==0.11.3
- comtypes==1.1.10
- opencv_python_headless==4.5.1.48
- tensorflow==2.7.0

Files and Folders in the directory -

- dataset_collection.py - for dataset creation
- train_model_CV_project.ipynb - for training the model

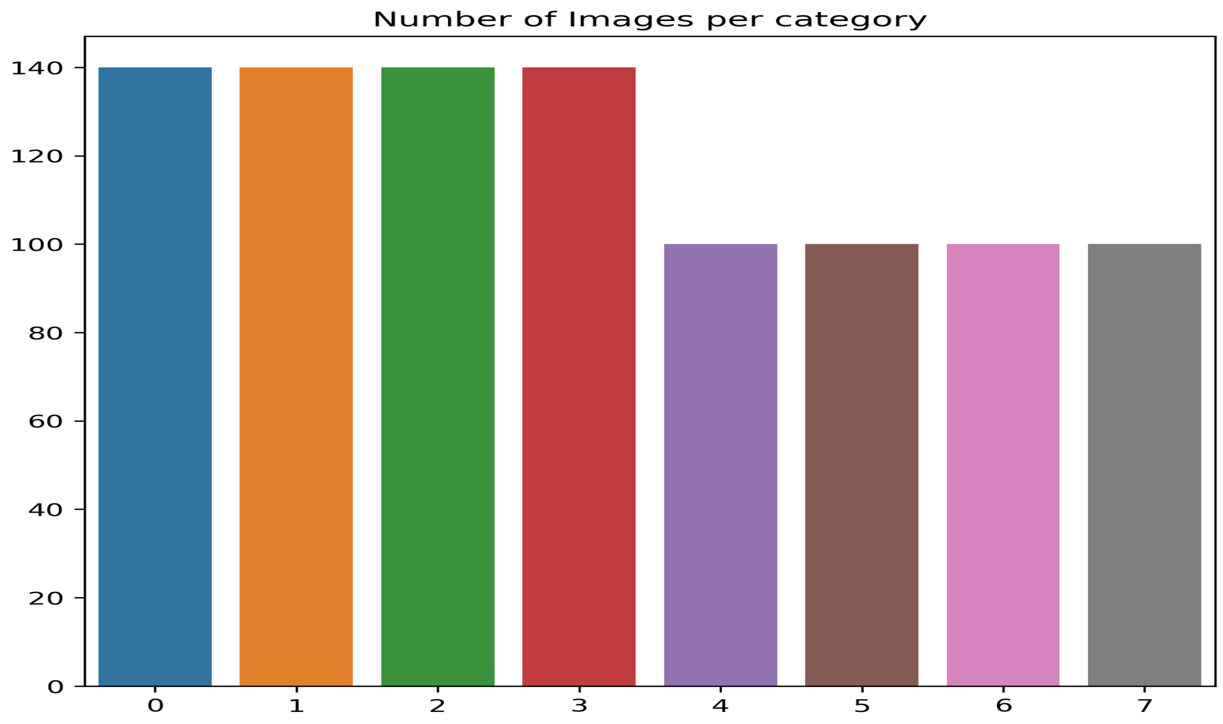
- testing.py - for testing the trained model
- dataset folder - contains the dataset
- dynamic_hand_gesture_recognition.h5 - the trained model

Dataset

As a satisfactory public dataset was not available, we created a new dataset for the project specifically. The dataset was created for Eight Dynamic Hand Gestures, namely - Opening Fist, Closing Fist, Swipe Up, Swipe Down, Swipe Left, Swipe Right, Thumbs Up, Thumbs Down. The ‘dataset_collection.py’ file contains the main code for creating the dataset. The flow of the code is as follows: -

1.) Firstly, the necessary directories and folders are created automatically if not created already. These folders store the required images of gestures in a systematic format.
2.) The second step must manually input the gesture name for which the images need to be saved in the second step.
3.) The code will then access the webcam and will display the frames.
4.) Each frame contains a Region of Interest (RoI) marked with a green box on the frame’s top right. The gestures need to be shown in the RoI only.
5.) For the first 200 frames, no gesture should be displayed. Let the code gather the background information and generate an average background used in the preprocessing step, details of which will be discussed in the next section.
6.) After 200 frames, the user can now display hand gestures in the desired region. Firstly, the user needs to adjust the hand in front of the camera. Once done, the user needs to press the ‘q’ button to instruct the code to start saving the images.
7.) The code will look at the subsequent 150 frames (roughly 4-5 seconds) and save every 5th frame in the appropriate folder.
8.) Press the Esc key to close the webcam.

The dataset created contains images corresponding to 480 videos of hand gestures.



Methodology

- ### Preprocessing

The images saved in the dataset collection step are first processed to segment the hand from the background before assigning them to the appropriate folder. The current implementation uses the Background Subtraction technique for the segmentation of hands.

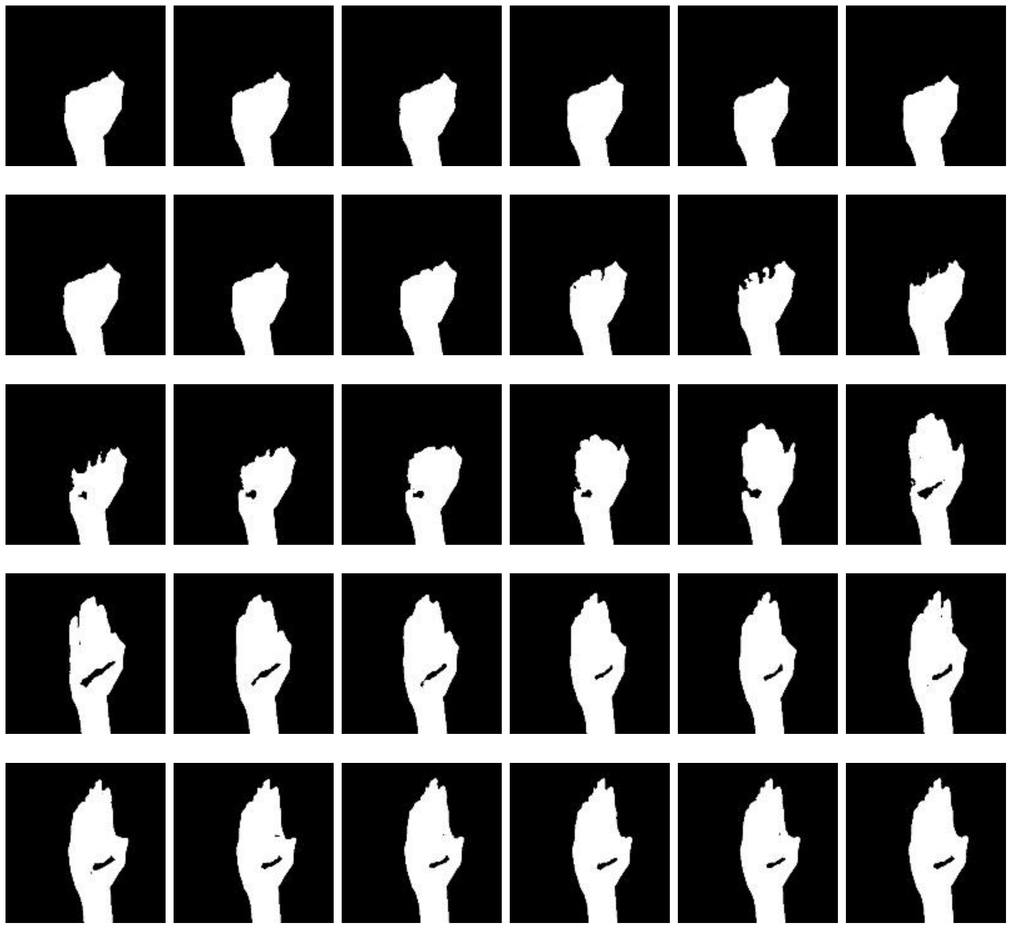
Background Subtraction Technique

Background subtraction is a mainstream algorithm for moving object detection in video surveillance systems. It segments moving objects by using the difference between the background and input images. The key to background subtraction is to establish a reliable initial background. As mentioned in the previous section, a reliable initial background is extracted as an average background generated from the first 200 frames.

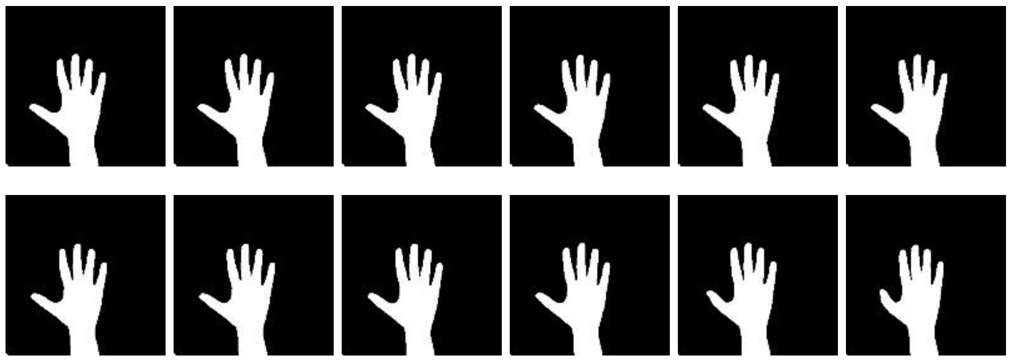
This initial knowledge of the background is subtracted from the subsequent input images to segment foreground and background.

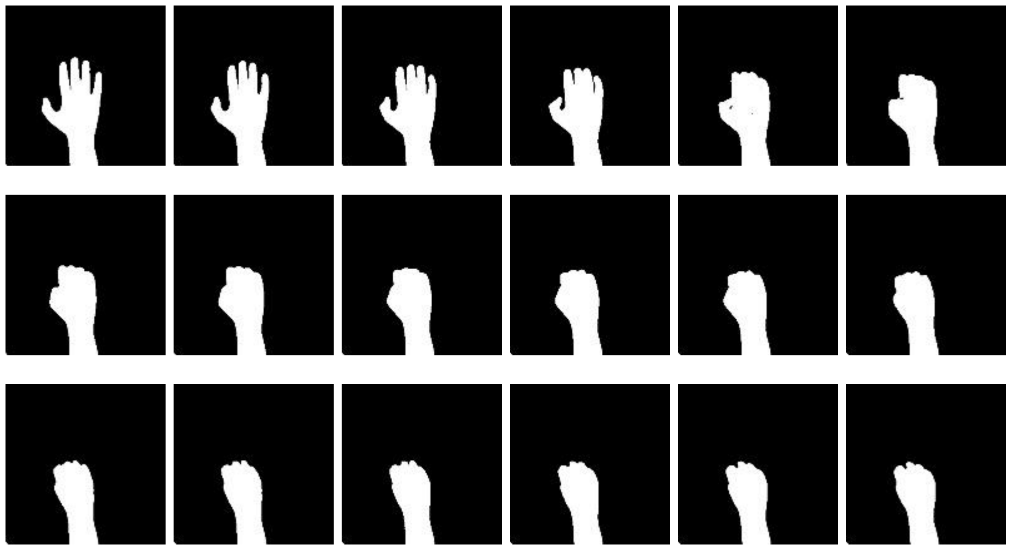
The saved images after background subtraction will look as follows -

1. Opening Fist: -

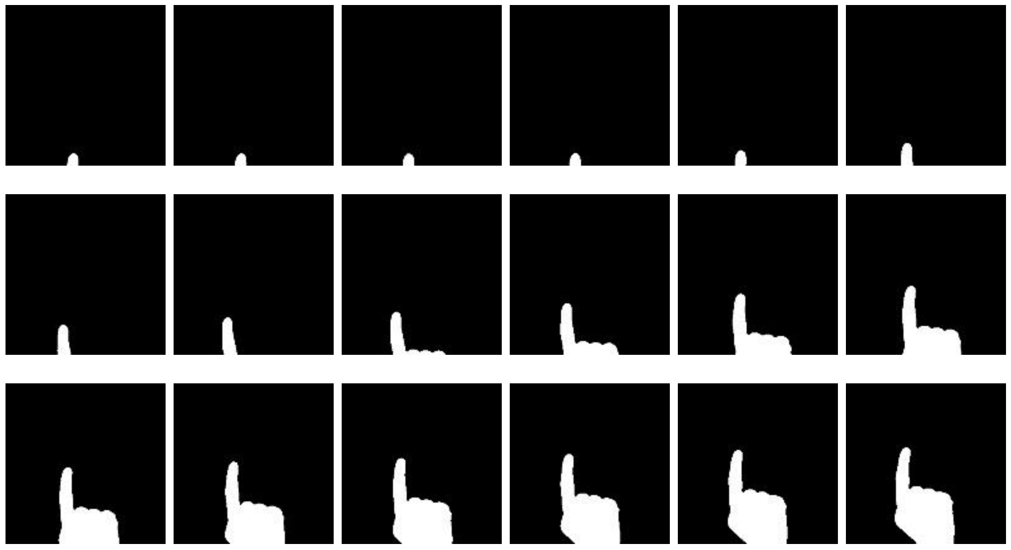


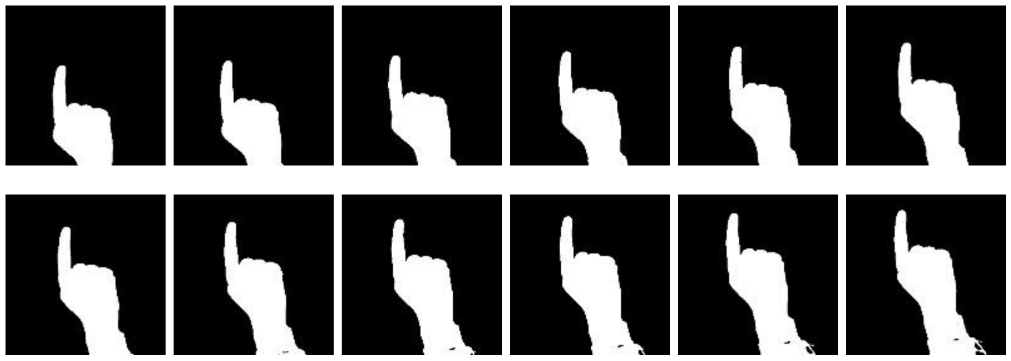
2. Closing Fist: -





3. Swipe Up: -





Apart from background subtraction, skin colour detection was also tested for hand segmentation. However, the background subtraction technique works much better than skin colour segmentation.

- **Training Process**

  A 3D Convolutional Neural Network (CNN) was used for training on the dataset created. Below is the model architecture used -

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv3D)              (None, 30, 128, 128, 16)  2016

 max_pooling3d (MaxPooling3D  (None, 15, 64, 64, 16)   0
 )

 batch_normalization (BatchN  (None, 15, 64, 64, 16)   64
 ormalization)

 conv2 (Conv3D)              (None, 15, 64, 64, 32)    13856

 max_pooling3d_1 (MaxPooling  (None, 15, 32, 32, 32)   0
 3D)

 batch_normalization_1 (Batc  (None, 15, 32, 32, 32)   128
 hNormalization)

 conv3 (Conv3D)              (None, 15, 32, 32, 64)    55360
 max_pooling3d_2 (MaxPooling  (None, 15, 16, 16, 64)   0
 3D)

 batch_normalization_2 (Batc  (None, 15, 16, 16, 64)   256
 hNormalization)

 flatten (Flatten)           (None, 245760)            0

 fc1 (Dense)                 (None, 128)               31457408

 batch_normalization_3 (Batc  (None, 128)              512
 hNormalization)

 dropout (Dropout)           (None, 128)               0

 fc2 (Dense)                 (None, 64)                8256

 batch_normalization_4 (Batc  (None, 64)               256
 hNormalization)

 dropout_1 (Dropout)         (None, 64)                0

 fc3 (Dense)                 (None, 8)                 520

=================================================================
Total params: 31,538,632
Trainable params: 31,538,024
Non-trainable params: 608
_________________________________________________________________
```

*the model architecture was taken from the kaggle link mentioned in the references. All the other processes and codes are written independently without the use of any internet resources.

The model consists of three convolutional layers, each followed by a max-pooling and batch normalisation layer. Three fully connected layers follow the convolutional layers. The input size of the model is (30, 128, 128, 1). RELU activation function is used. Dropout layers are used for reducing overfitting.

The model was trained for 35 epochs with an initial learning rate of 2e-4 and a batch size of 32. Adam optimiser was used for training the model.

- **Testing Process**

  The testing process is similar to the dataset collection step with minute changes. For the first 200 frames, let the code generate initial information of the background. Adjust the hand before pressing the ‘q’ button. Once the button is pressed, the code will look at the subsequent 150 frames and save every 5th frame into a variable matrix. The code uses the trained model saved in the previous section for making the predictions. The predictions are shown on each frame. Press the ‘q’ button every time you want to make predictions.

  Each of the gestures is used to control the following video commands -

Thumbs Up	Increase the volume.	**Swipe Up**	Increase Brightness
Thumbs Down	Decrease the volume.	**Swipe Down**	Decrease Brightness
Left Swipe	'Jump' backwards 10 seconds.	**Opening Fist**	Play
Right Swipe	'Jump' forward 10 seconds.	**Closing Fist**	Pause

So, the code also displays a video alongside the live video stream output. Each time the code makes a prediction, the corresponding command for the hand gesture predicted is performed on this video.

Results

Training vs Testing Accuracy:

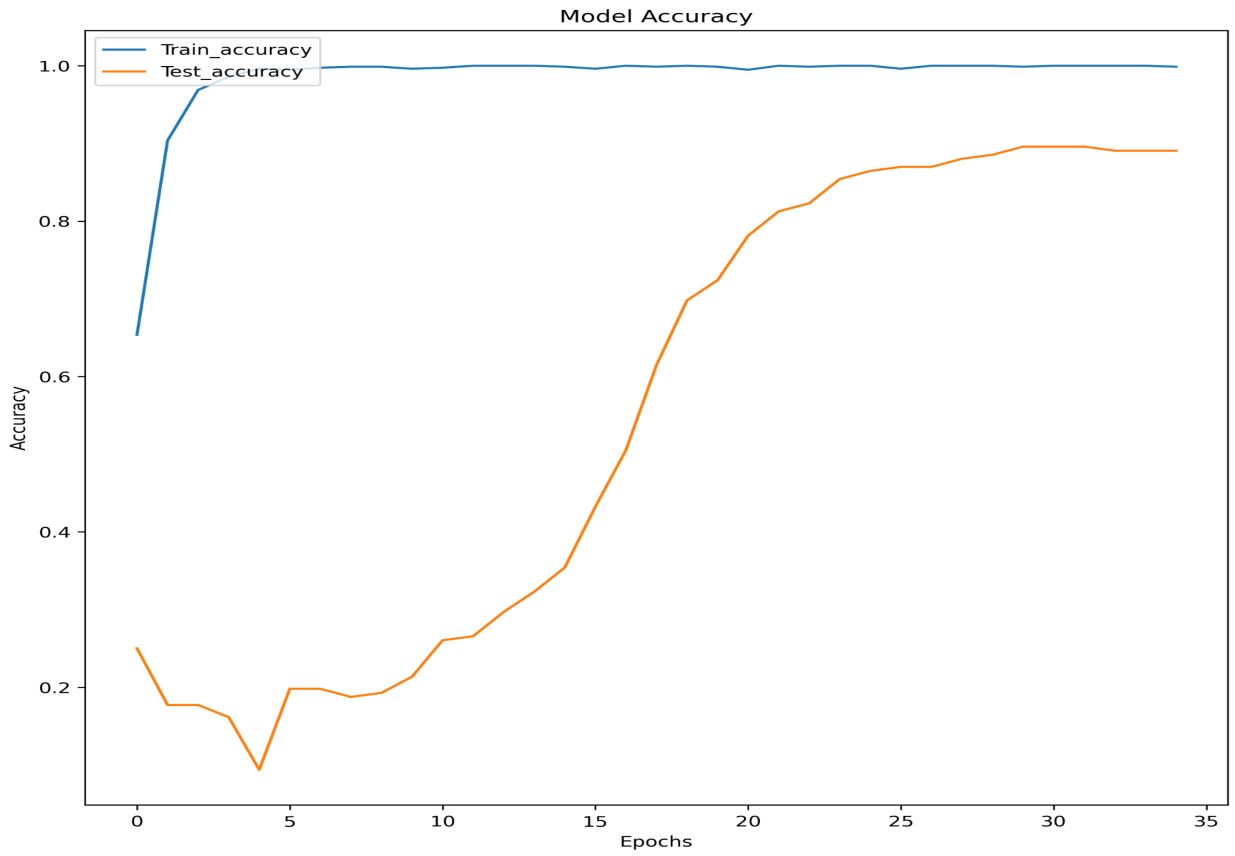
Training Set Accuracy - 99.87%

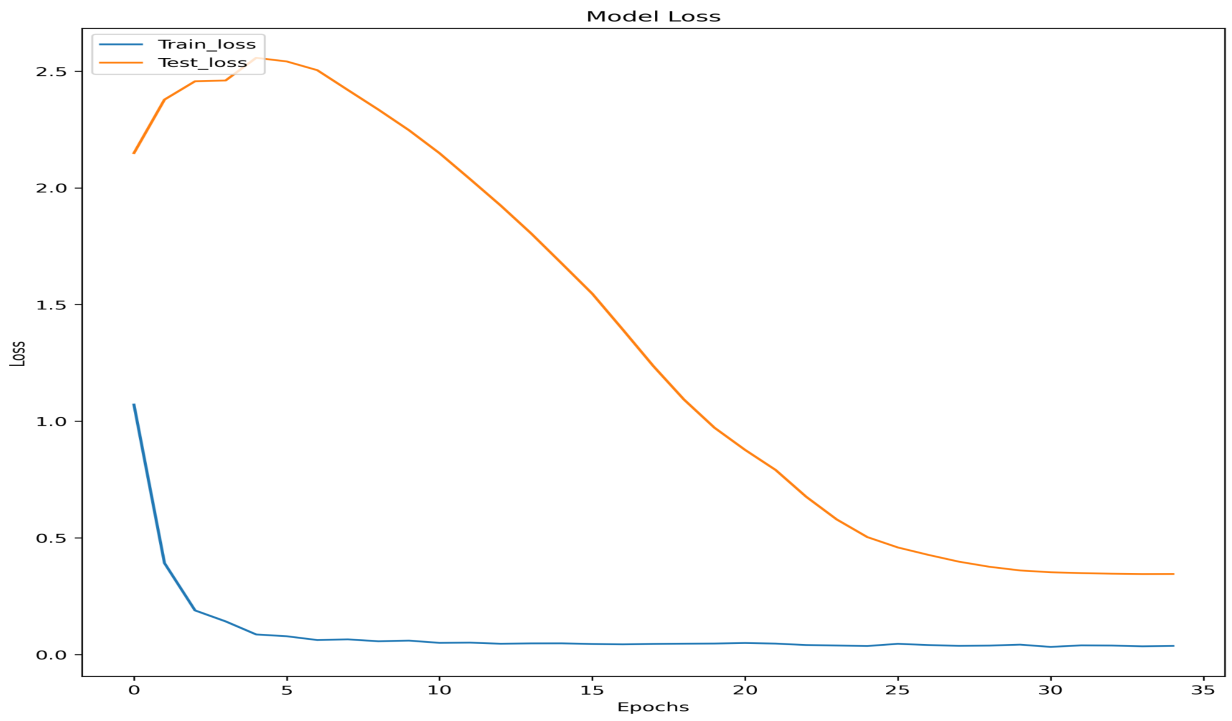
Test Set Accuracy - 89.06%

Classification Report:

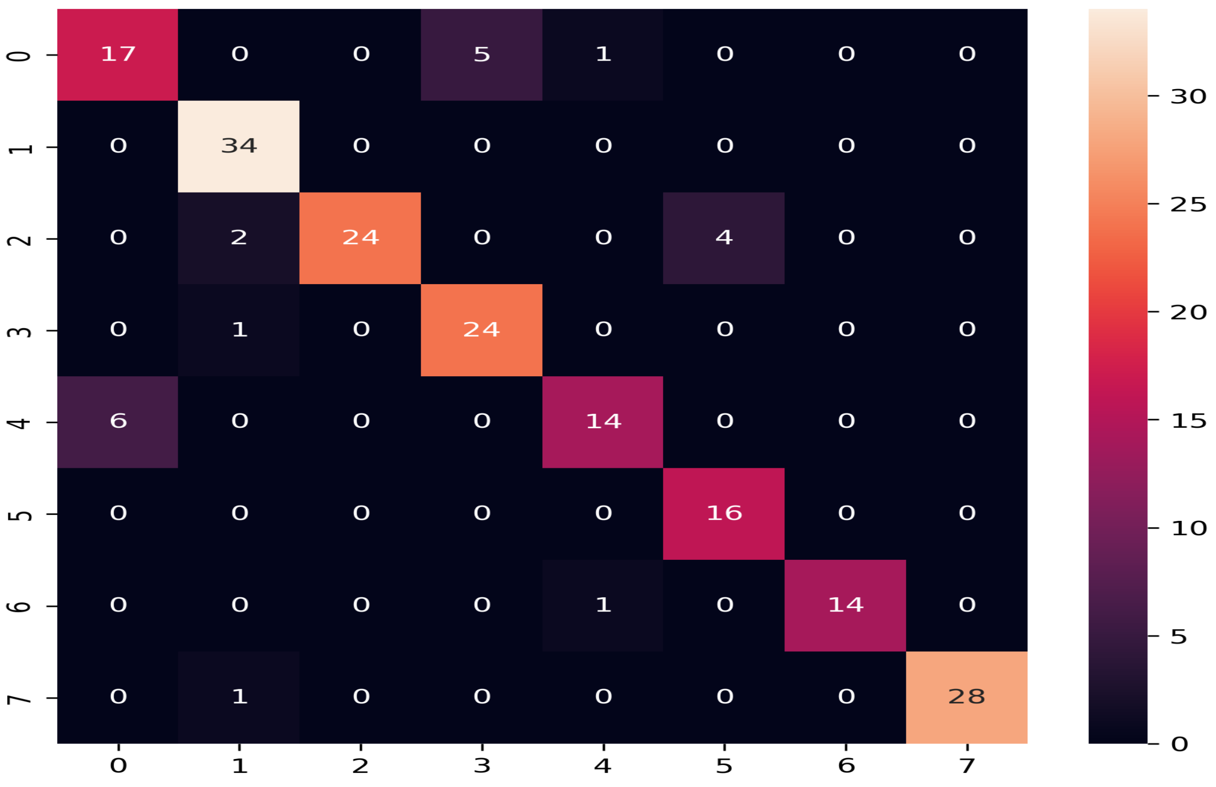
```
                precision    recall  f1-score   support
           0       0.74      0.74      0.74        23
           1       0.89      1.00      0.94        34
           2       1.00      0.80      0.89        30
           3       0.83      0.96      0.89        25
           4       0.88      0.70      0.78        20
           5       0.80      1.00      0.89        16
           6       1.00      0.93      0.97        15
           7       1.00      0.97      0.98        29

    accuracy                           0.89       192
   macro avg       0.89      0.89      0.88       192
weighted avg       0.90      0.89      0.89       192
```

Accuracy and Loss vs Epochs Plot





Confusion Matrix:



	0	1	2	3	4	5	6	7
0	17	0	0	5	1	0	0	0
1	0	34	0	0	0	0	0	0
2	0	2	24	0	0	4	0	0
3	0	1	0	24	0	0	0	0
4	6	0	0	0	14	0	0	0
5	0	0	0	0	0	16	0	0
6	0	0	0	0	1	0	14	0
7	0	1	0	0	0	0	0	28

Conclusion

Controlling things by hand is more natural, easier, more flexible and cheaper, and there is no need to fix problems caused by hardware devices since none is required. We were able to create robust hand gesture recognition with acceptable accuracy. We also achieved the purpose of interacting with devices without any physical contact. In future, we would like to add more gestures to implement more functions. We also aim to extend this mechanism to a range of users including disabled users.

References

- https://www.kaggle.com/imsparsh/gesture-recognition-conv3d-conv2d-rnn/data
- Farooq, Javeria & Ali, Muhaddisa. (2014). Real-Time Hand Gesture Recognition for Computer Interaction. 2014 International Conference on Robotics and Emerging Allied Technologies in Engineering
- Hand Gesture Recognition Based on Computer Vision: A Review of Techniques, by Munir Oudah 1, Ali Al-Naji 1,2,* and Javaan Chahl