Facial Recognition and Pose Recognition for a Socially Assistive Robot

Project Report

Table of Contents

List of figures 3

Executive Summary 4

Introduction 4

Methodology 5

Database Generation 5

Facial Recognition 5

Approach 1: FaceNET 5

Approach 2: Principal Component Analysis 6

Step 1 - Pre-processing 7

Step 2 - Save preprocessed dataset 8

Step 3 - Applying PCA without n_components 8

Step 4 - Extracting eigenfaces from PCA 9

Step 5 - Perform PCA 10

Pose Recognition 10

Results 12

Facial Recognition 12

Pose Recognition: 14

Conclusions 14

Acknowledgments 15

References 15

List of figures

Figure Number	Label
1	Internal mechanism of PABI
2	Database generated
3	Block Diagram of Face Recognition Model using FaceNET
4	Steps for PCA approach
5	Image set after pre-processing for PCA
6	Snapshot of CSV file generated
7	Commands to filter components based on threshold value
8	Result showing the components passing the threshold value
9	Commands to plot filtered components vs variance
10	Variance vs Number of components
11	Eigenfaces extracted
12	Pose keypoints
13	Pose Recognition Block Diagram
14	Commands to test PCA approach with random image
15	Classification report for PCA
16	Result obtained for Facial Recognition
17	Result obtained with labelling in live video feed
18	Pose classification result

Executive Summary

Face and pose recognition can help determine behavioral aspects of patients during ABA therapy. Face recognition at the beginning of the therapy session is necessary to store the patient's data to their respective folders. For this, two approaches were used. First was using the FaceNET model which resulted in 100% accuracy for the collected data. However, for larger dataset, computational time can be adversely affected. Thus, a second approach for face recognition using Principal Component Analysis was implemented. Using this an accuracy of 97% was achieved. For pose recognition, the MediaPipe API was used which resulted in an accuracy of 100% for determining whether the person in focus was 'studying' or 'looking here and there'. These results can help derive conclusions regarding the attention span of the subject. This project is a foundation for a system which can be used to infer behavioural changes and help decide the course of future therapy sessions.

Introduction

Socially assistive robots have played an important role in applications like elder care, research, and therapy. The use of socially assistive robots in applied behavior analysis (ABA) therapy has proven to help the therapists focus more on the session rather than keeping track of progress. In this project, we focus on concepts related to Applied Behavior Analysis Therapy for Autism Spectrum Disorder. Applied Behavior Analysis involves observing behavior and developing methods that determine the variables that affect the behavior change. It is the most notable and widely accepted form of therapy for Autism Spectrum Disorder (ASD). ASD is generally associated with problems involving a lack of social skills and communication. While there is no cure for Autism, many intervention therapies like ABA are known to help improve the patient's life skills.

The scenario considered for this project involves the use of PABI (Penguin for Autism Behavioral Intervention), a socially assistive robot created specifically to assist in ABA therapies. As given in [1], PABI acts as a mediator for ABA therapy to help perform data logging and analysis. It is equipped with the Intel RealSense camera for recording the data and analyzing it.

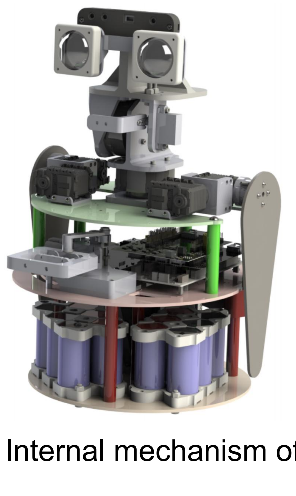


Fig 1 : Internal mechanism of PABI

The main objective of this project is to track the head and limbs of the person of interest in the video and live feed to infer the behavior portrayed. This data can be used to identify the person of interest (PoI) and detect the pose during the entire therapy session. The pose data can be used to infer whether the PoI was focused on the task at hand or distracted. As of now, only two inferences have been derived but this can be extended later on to include more poses. Also, the data collected while detecting the pose can be used to track the limbs and head of the PoI.

Methodology

The following section will outline the steps followed in the implementation of the project.

Database Generation

Every new person who is admitted into ABA therapy will get their face profile recorded by taking a series of pictures of them with different face orientations. These pictures are then labelled with the person's name and stored in the database for creation of a face recognition model. To get a dataset for pose recognition, the subjects were asked to mimic writing while sitting on a chair in front of the camera. This pose was considered to be a basis for recognition. Actual ABA therapy sessions can be used for better performance. Five minute recordings are used for further analysis. A snapshot of the database generated is shown in Fig 2.

Fig 2 : Database generated

Facial Recognition

To recognize the faces, two different approaches were tested in this project. The first, using a pre-trained FaceNET model, and the second using Principal Component Analysis.

Approach 1: FaceNET

FaceNET model is an open-source deep learning model available on GitHub. It takes 60x60 pixel images as input and generates 1x168 feature embeddings. These feature embeddings are generated forevery person of interest using the database of their face profile created in the introduction phase. Then these embeddings are used to train a Support Vector Machine Classifier(SVM classifier).

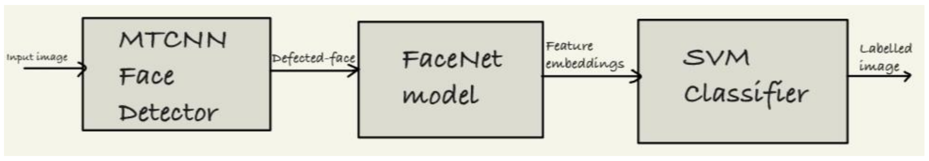


Fig 3 : Block Diagram of Face Recognition Model using FaceNET

Summary of steps taken for FaceNET approach:

1. Pre-process image dataset: image frame is fed into an MTCNN face detector module to get cropped out face images of the person of interest.
2. Convert these face images in a 60x60 pixel array with a normalized value.
3. Apply them to the FaceNET model and generate feature embeddings.
4. Use these feature embeddings to train a SVM classifier.
5. Use this SVM model to predict the right person of interest.

Once the SVM classifier is trained and ready it can be used to recognise faces in the therapy session. It helps socially assistive robots to keep track of the person of interest in his field of view.

Currently this project built a face recognition system to recognise 5 distinct people. This model got a training and testing accuracy of 100%. This model is able to perform real-time face-recognition on live feed of a camera.

Approach 2: Principal Component Analysis

Principal Component Analysis is chosen since it is a preferred method for reducing computation. It uses the image data projected on a small feature space thus providing dimensional reduction.

Summary of steps taken for PCA approach

1. Pre-process image dataset : Crop around face and convert to grayscale
2. Store image dataset as csv file : Using PIL Library getdata() function
3. Apply PCA without n_components
4. Extract eigenfaces from PCA
5. Perform PCA

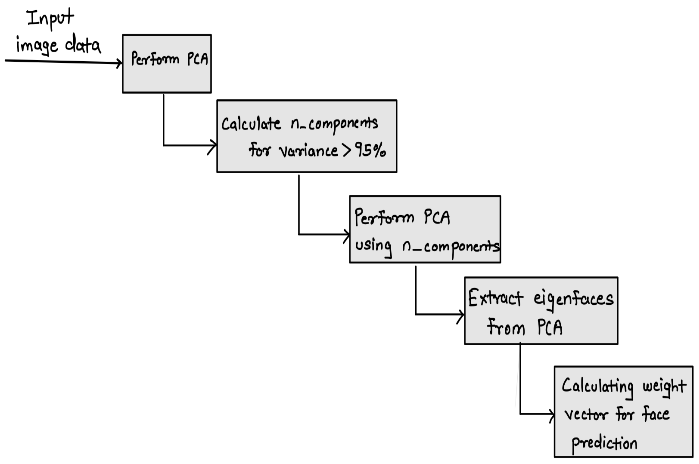

Input image data
Perform PCA
Calculate n_components for variance > 95%
Perform PCA using n_components
Extract eigenfaces from PCA
Calculating weight vector for face prediction

Fig 4: Steps for PCA approach

Step 1 - Pre-processing

To use PCA, the images need to be cropped to show just the face and converted to grayscale. The preferred size for each image is 64 x 64. Thus, the dataset generated needs to be pre-processed to create an appropriate image data set for PCA. The frontal face haar cascade is used for cropping out the faces in the dataset. These new images are stored in a new directory. Fig 5 shows the dataset after pre-processing for PCA.

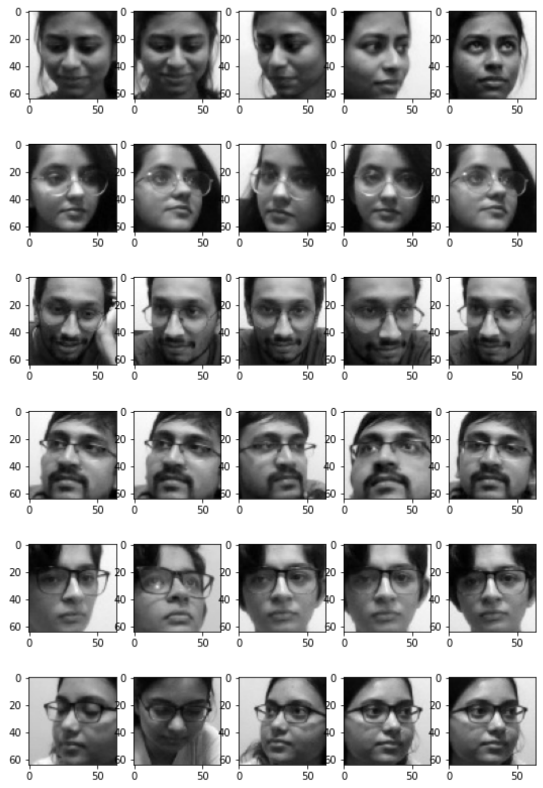
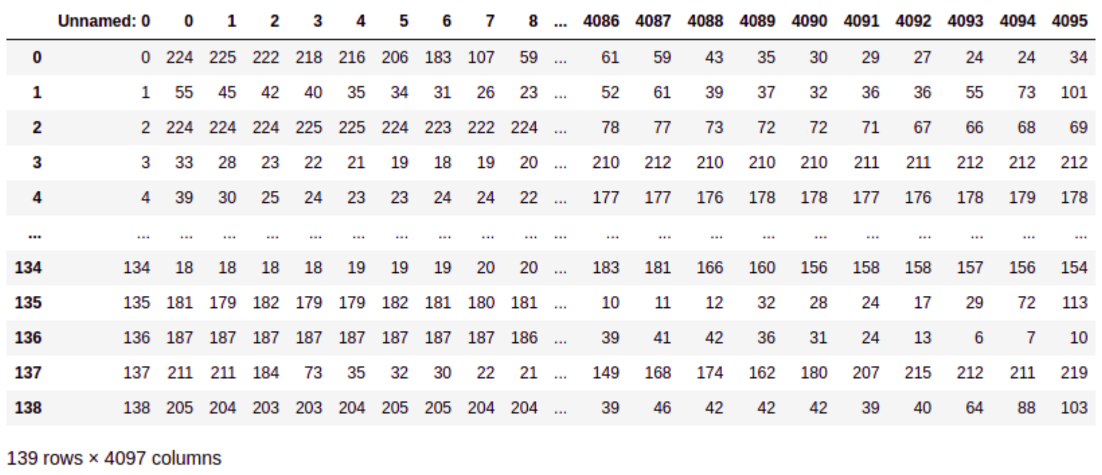


Fig 5 : Image set after pre-processing for PCA

Step 2 - Save preprocessed dataset

A total of 139 images are stored where each image is 64 x 64 in size. The *getdata* function from PIL library is used to convert the (139, 64, 64) dataset to (139, 4096). These converted images are stored in a csv file using the *to_csv* function from pandas library. The resulting csv file now has 139 rows, each of which is an array of flattened pixels for each image. Fig 6 shows a snapshot of the csv file generated in this step.



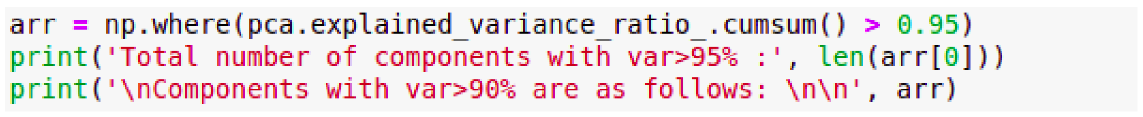
	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095
0	0	224	225	222	218	216	206	183	107	59	...	61	59	43	35	30	29	27	24	24	34
1	1	55	45	42	40	35	34	31	26	23	...	52	61	39	37	32	36	36	55	73	101
2	2	224	224	224	225	225	224	223	222	224	...	78	77	73	72	72	71	67	66	68	69
3	3	33	28	23	22	21	19	18	19	20	...	210	212	210	210	210	211	211	212	212	212
4	4	39	30	25	24	23	23	24	24	22	...	177	177	176	178	178	177	176	178	179	178
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
134	134	18	18	18	18	19	19	19	20	20	...	183	181	166	160	156	158	158	157	156	154
135	135	181	179	182	179	179	182	181	180	181	...	10	11	12	32	28	24	17	29	72	113
136	136	187	187	187	187	187	187	187	187	186	...	39	41	42	36	31	24	13	6	7	10
137	137	211	211	184	73	35	32	30	22	21	...	149	168	174	162	180	207	215	212	211	219
138	138	205	204	203	203	204	205	205	204	204	...	39	46	42	42	42	39	40	64	88	103

139 rows × 4097 columns

Fig 6 : Snapshot of CSV file generated

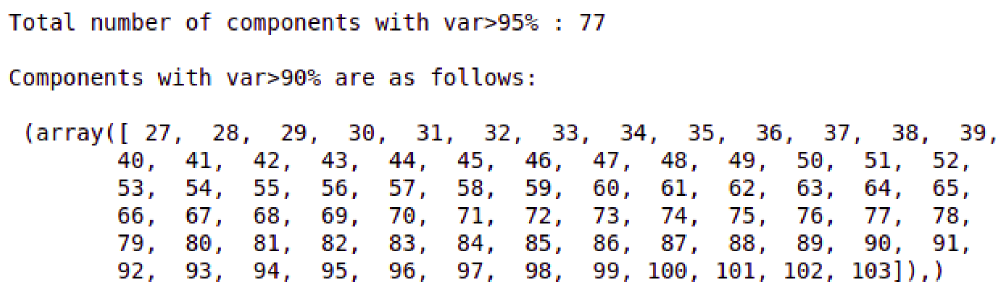
Step 3 - Applying PCA without n_components

The scikit learn library is used to train our dataset for PCA. Using this training, a suitable value for ‘n_components’ is deduced [9]. To understand the significance of every projected component, *pca.explained_variance_ratio_* is used to display the variance ratio values for all components. The aim is to determine the number of components that achieve high variance values. In this use case, a threshold of 95% is chosen for the variance arbitrarily. This threshold value is used to filter out the components as shown in Fig 7. A total of 77 such components are obtained as represented in Fig 8.



```
arr = np.where(pca.explained_variance_ratio_.cumsum() > 0.95)
print('Total number of components with var>95% :', len(arr[0]))
print('\nComponents with var>90% are as follows: \n\n', arr)
```

Fig 7 : Commands to filter components based on threshold value



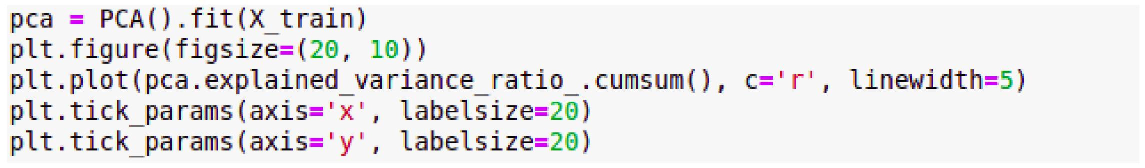
```
Total number of components with var>95% : 77

Components with var>90% are as follows:

 (array([ 27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103]),)
```

Fig 8: Result showing the components passing the threshold value

In order to verify the number of components obtained after filtering, the number of components are plotted against their corresponding variance values using the commands given in Fig 9. This plot, as shown in Fig 10, is analysed to observe that for the number of components ~(30, 100) variance greater than 95% is achieved.



```
pca = PCA().fit(X_train)
plt.figure(figsize=(20, 10))
plt.plot(pca.explained_variance_ratio_.cumsum(), c='r', linewidth=5)
plt.tick_params(axis='x', labelsize=20)
plt.tick_params(axis='y', labelsize=20)
```

Fig 9: Commands to plot filtered components against corresponding variance.

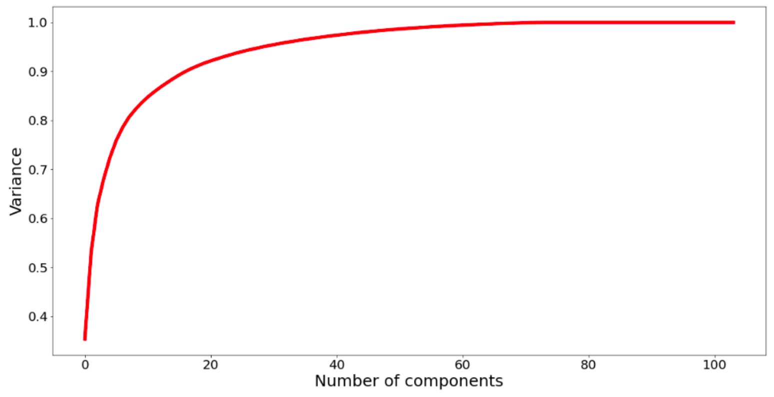


Fig 10 : Variance vs Number of components

Step 4 - Extracting eigenfaces from PCA

Features extracted from eigenfaces, a weight vector is calculated to assign value to each feature [10]. Once this vector is calculated, the algorithm can recognize any of the people in the dataset with an accuracy of 97%.

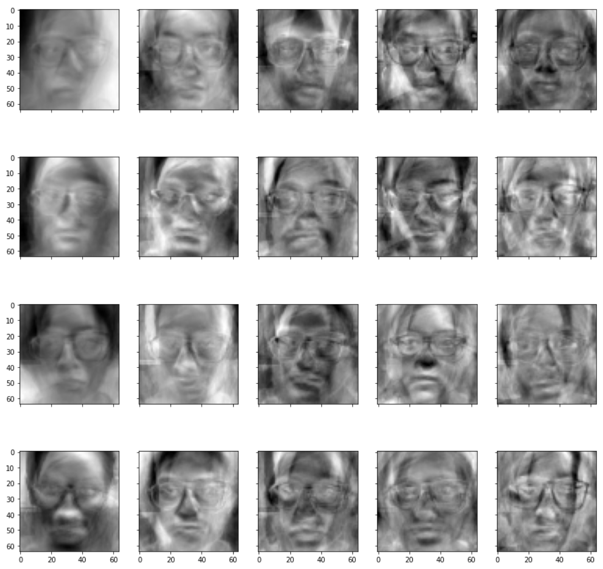


Fig 11: Eigenfaces extracted

Step 5 - Perform PCA

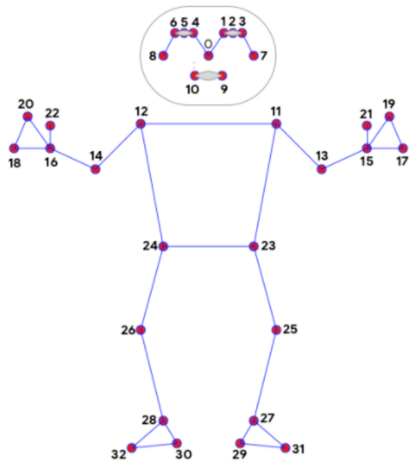
PCA is performed again with the number of components to be 80. A SVC classifier is used along with PCA to create a testing set for this model. Using this approach PCA can recognize faces with an accuracy of 97%. The commands and results are detailed in the Results section.

Pose Recognition

Pose Recognition and classification is done so as to analyze the pose of the autistic patient during the therapy session. The therapist can make conclusions regarding his/her concentration level and other parameters based on the pose data of the autistic children during the therapy session. The pose data is important and can be used effectively for consecutive therapy sessions, so as to plot or find improvements in the autistic patient's concentration and other parameters. This is the reason we found it imperative to incorporate pose recognition and classification in our project scope.

For Pose recognition, we used MediaPipe holistic pipeline, which integrates separate models for pose, face and hand components [8]. This pipeline is heavily used by Machine Learning enthusiasts for developing object detection, pose estimation, and face detection on edge devices as it is less processing intensive.

The MediaPipe Holistic pipeline estimates human pose and maps out all major pose keypoints. Once keypoints are found, they are connected by annotated lines so as to map the entire body parts. The advantage of using MediaPipe Holistic is it gives fast and accurate tracking of human pose,face landmarks and head tracking. It utilizes the pose, face and hand landmark models to generate a total of 543 landmarks(33 pose landmarks, 468 face landmarks and 21 hand landmarks per hand)[8].



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Fig 12 : Pose keypoints [8]

For our project, we only used the key points above the waist, as we always have to recognize the pose of the autistic subject who is always sitting on the chair and his/her lower body is never visible to the PABI robot.

Following is the block diagram of the entire pose recognition system we incorporated in our project:

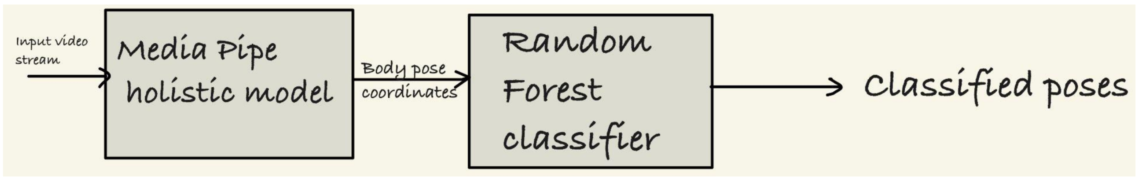


Fig 13: Pose Recognition Block Diagram

Following were the steps that were followed for recognizing the pose.

- **Feeding the data to the MediaPipe Holistic model**:
  OpenCV's *VideoCapture* API was used to stream the video data and input to the MediaPipe model

- **Detecting keypoint data of face, hand, torso using MediaPipe Holistic Model**:
  MediaPipe's "*mp.solutions.holistic*" API is used to process the frames from the video and capture key points of body parts. After capturing major key points of face, limbs and torso it was fed to "*mp.solutions.drawing_utils*" API for drawing landmarks and annotating the body parts by connecting the keypoints. Like connecting face to shoulder and drawing connecting lines between shoulder to elbow.

- **Data logging**:
  After getting the key points from the holistic model, we saved them into a csv file. Keypoint data of face, limbs, torso for every frame was saved in the csv file and the frame rate was kept as 30 FPS.

- **Pose Classification:**
  As we have got the location and key points of the body parts of the subject, the data that we have got is of no use as it only maps and annotates the pose of the body. But we are interested in a specific pose. In our case we wanted to monitor the "studying" pose and the "distracted/lookinghereandthere" pose. Once we are able to map this pose, it will help the therapist to analyze the behaviour and concentration pattern of the autistic patient.
  We thus fed the key point data from the csv file to a classifier model. Amongst many classifier model we decided to use **Random Forest algorithm** to classify the data points into two classes: **"Studying"** and **"lookinghereandthere".** While recording the data, studying, distracted poses and "lookinghereandthere" poses were recorded and fed to the classifier model. The entire dataset was split into 60 % and 40% for testing and training set. As said, training of the data was done using the Random Forest classifier model from scikit library.

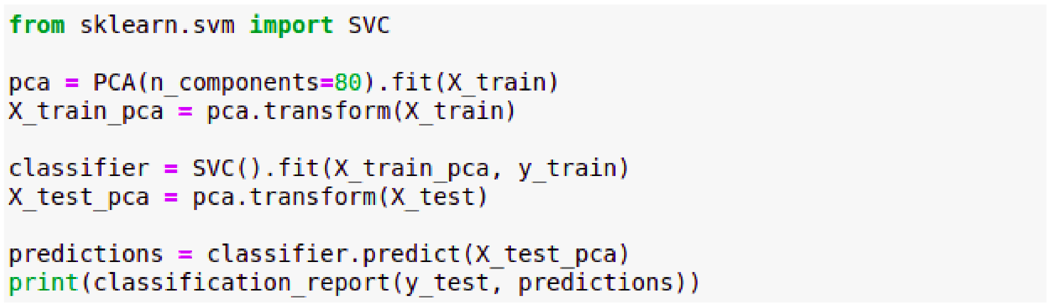
- **Testing the Classifier**:
  Once training is done, we downloaded the model based on the train data. We used scikit Pickle library to download and save the model. After saving the model, we uploaded it and loaded the model in our test classification code. In our test classification code we loaded the model and fed the test data set to it. After feeding it we got 100% accuracy. Once getting he accuracy, we fed live video data to the classifier model which classified the "**studying**" and "**lookinghereandthere**" pose successfully.

Results

The following section will explain and analyze the results obtained for the implementation as shown in the previous section.

Facial Recognition

The PCA approach is tested using the commands as shown in Fig 14. The resulting classification report is displayed as Fig 15 while the image prediction results are shown in Fig 16.



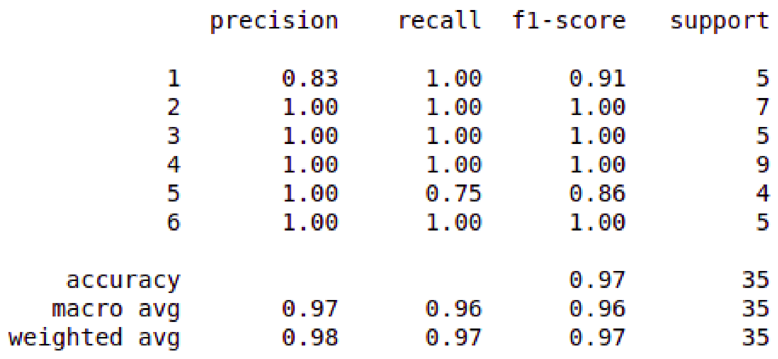
```
from sklearn.svm import SVC

pca = PCA(n_components=80).fit(X_train)
X_train_pca = pca.transform(X_train)

classifier = SVC().fit(X_train_pca, y_train)
X_test_pca = pca.transform(X_test)

predictions = classifier.predict(X_test_pca)
print(classification_report(y_test, predictions))
```

Fig 14 : Commands to test PCA approach with random image

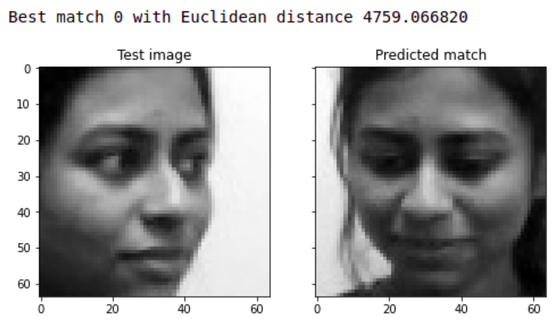


```
              precision    recall  f1-score   support

           1       0.83      1.00      0.91         5
           2       1.00      1.00      1.00         7
           3       1.00      1.00      1.00         5
           4       1.00      1.00      1.00         9
           5       1.00      0.75      0.86         4
           6       1.00      1.00      1.00         5

    accuracy                           0.97        35
   macro avg       0.97      0.96      0.96        35
weighted avg       0.98      0.97      0.97        35
```

Fig 15: Classification report for PCA



```
Best match 0 with Euclidean distance 4759.066820
```

Fig 16: Result obtained for Facial Recognition

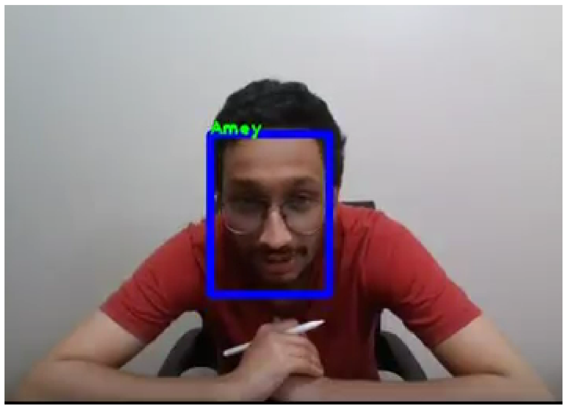


Fig :17: Result obtained with labelling in live video feed

Pose Recognition:

The upper torso keypoints are detected in live camera feeds and video inputs given for testing. The keypoints are taken together and classified to show basic behavior of the subject. As of now, this project implements only two basic behavior inferences: studying and looking_here_and_there.

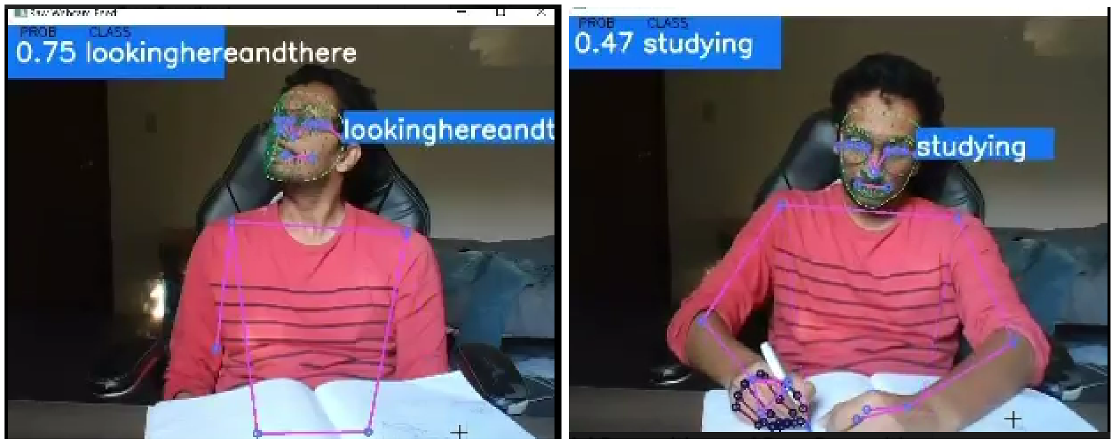


Fig 18: Pose classification result

The purple markers show the keypoints on the body of the person in frame. Face mesh is shown in green lines and black lines can show the exact pose of the fingers. The snapshots in Fig 18 are still frames from a video which showed the pose classified as studying or looking_here_and_there. An accuracy of 100% was achieved.

Conclusions

This project has allowed us to gain invaluable experience with FaceNET and MediaPipe. Implementation of Principal Component Analysis also provided a good understanding of the intricacies involved in the concept. The results obtained show excellent accuracy values but this value might decrease if the dataset used for training is increased. Use of actual ABA session videos might help build a system which can be used in the real world but obtaining subh data is very difficult. Nevertheless, this project can be a foundation for a system which can be used with PABI.

Acknowledgments

We would like to thank Prof Mike Gennert and Prof Greg Fischer for their support in this endeavour. Also, this project would not be possible without our friends who star in our datasets.

References

[1] Modular Social Assistive Robot Framework , Jacob Bader,Tyler Dubuke,Jonathan Sanchez,Raymond Schade, 2020
[2] de Belen, R.A.J., Bednarz, T., Sowmya, A. et al. Computer vision in autism spectrum disorder research: a systematic review of published studies from 2009 to 2019. Transl Psychiatry 10, 333 (2020). https://doi.org/10.1038/s41398-020-01015-w

[3]Lakkapragada, A., Kline, A., Mutlu, O.C., Paskov, K.M., Chrisman, B.S., Stockham, N.T., Washington, P., & Wall, D.P. (2021). Classification of Abnormal Hand Movement for Aiding in Autism Detection: Machine Learning Study.

[4]H. Zhang, H. Han, J. Cui, S. Shan, and X. Chen, "RGB-D Face Recognition via Deep Complementary and Common Feature Learning," 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), 2018, pp. 8-15, DOI: 10.1109/FG.2018.00012.

[5]G. Goswami, M. Vatsa and R. Singh, "RGB-D Face Recognition With Texture and Attribute Features," in IEEE Transactions on Information Forensics and Security, vol. 9, no. 10, pp. 1629-1640, Oct. 2014, DOI: 10.1109/TIFS.2014.2343913.

[6]RGB-D face detection database based on Microsoft Kinect camera: https://vap.aau.dk/rgb-d-face-database/

[7] Washington, P., Cezmi Mutlu, O., Leblanc, E., Kline, A., Hou, C., Chrisman, B., … Wall, D. (2021). Training Affective Computer Vision Models by Crowdsourcing Soft-Target Labels. arXiv e-prints, arXiv:2101.03477. Opgehaal van http://arxiv.org/abs/2101.03477

[8] MediaPipe Holistic AI Blog Google AI Blog: MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device (googleblog.com)

[9] Face Recognition using Principal Component Analysis
https://machinelearningmastery.com/face-recognition-using-principal-component-analysis/

[10] Eigenfaces- Face classification
https://towardsdatascience.com/eigenfaces-face-classification-in-python-7b8d2af3d3ea