Alexandria University,

Faculty of Engineering,

Computer and Systems Engineering Department

B. Eng. Final Year Project

Automated Video Interviews Analysis

By:

Arsany Atef Abdo
Toka Alaa
Jamal El-Din Ahmed
Hazem Morsy
Nada Salama
Yomna Gamal El-Din

Supervised By:

Dr. Marwan Torki

Dr. Hicham Elmongui

July – 2021

ACKNOWLEDGMENT

First, we would like to thank ALLAH for giving us the insistence and energy to do and complete this project successfully. Second, we are sincerely grateful to our supervisors Dr. Marwan Torki and Dr. Hicham Elmongui for their advice and supporting us and thank them for guiding us in our software's aspects and never sparing any effort or time to guide us for the better. Third, we would like to thank Ejada Company for its financial support for our project. We finally would like to express our deepest gratitude and appreciation to our families for their great support, where without whom we would not have been successful.

ABSTRACT

Nowadays, sites for interviews have become popular with companies all over the world. Most companies are assigned to follow-up expressions during interviews, whether registered or online, and they ask a question to know the personal characteristics, and for expressions of opinion, dislike, and opinions on various topics.

So, we built a web application for analyzing interviews by feeding it with five questions and asking users to upload five video answers. Then analyze the answer video by three modalities (Facial expressions, body movement, and Speech). Then showing to the user a set of numbers that indicate the extent of his performance and a set of advice or comments on his performance.

We integrated the fields of computer vision and Machine Learning. where our problem in the field of computer vision is tracking of body movements and estimation of agitation and facial movements. Also, the field of machine learning is analyzing personality characteristics based on the tone of voice.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

ACRONYM	Definition of Acronym
MFCC	Mel-Scale Frequency Cepstral Coefficients
RMPE	Regional Multi-person Pose Estimation
YOLOv3-SPP	You Only Look Once version 3 - Spatial Pyramid Pooling
HOG	Histogram of Oriented Gradient
3D CRNN	3-D Convolutional Recurrent Neural Networks with Attention Model for Speech Emotion Recognition.
PCA	Principal Component Analysis
SVC	Support Vector Classifier
SVR	Support Vector Regression
mAP	Mean Average Precision
AP	Average Precision
SVM	Support Vector Machine
MIT	Massachusetts Institute of Technology

Chapter One

1 INTRODUCTION

1.1 GENERAL

In this chapter, we present a general overview of the report, and we discuss the motivation and the scope of our work. Section 1.2 presents the motivation of our work and the currently unsolved problems. Section 1.3 shows the goals and the scope of our work. Section 1.3 describes the organization of the report.

1.2 MOTIVATION

Nowadays, the COVID-19 pandemic caused many businesses to lay off and put hiring on pause. The number one challenge that companies will face as they look to restart hiring is that their people and financial resources may be dramatically limited. A typical human recruiter can handle up to 20-50 candidates at a given time, but a machine can perform the same task much better and faster. Digital interviews are an absolute win in this situation. Not only do you protect yourself, your employees, and the applicant, but also you save a lot of time and effort in whether you recruit this applicant or not and provide your prospects with a measure of flexibility.

Analysis of non-verbal behavior to predict the outcome of social interaction has been studied for many years in different domains, with predictions ranging from marriage stability based on interactions between newlywed couples to patient satisfaction based on doctor-patient interaction, to teacher evaluation by analyzing classroom interactions between a teacher and the students, most of these automated methods are focused on a single modality of interaction but in job interview domain the challenge is the automated understanding of multimodal human interactions, including facial expression, prosody, and body language which is a less explored domain. This increases the motivation of researchers to propose new solutions for automated understanding of multimodal human interactions and use them in different domains.

Since job interviews are ubiquitous and play inevitable and important roles in our life and career. Over many years, social psychologists and career coaches have accumulated knowledge and guidelines for success in job interviews. Studies in social psychology have shown that smiling, using a louder voice, and maintaining eye contact contribute positively to our interpersonal communications.

For the previous reasons, we were motivated to present an Automated Video Analysis system, which tends to evaluate your performance in a job interview.

1.3 GOAL AND SCOPE

The goal of this project is to present a complete automated video interviews analysis system to recognize individual personalities and improve some of a person's soft skills by introducing himself and answering different questions.

1.4 ORGANIZATION OF REPORT

The rest of the report is organized as follows: Chapter Two gives background and the literature of the previous work. In Chapter Three, we discuss an overview of the proposed system and its architecture. Chapter four describes the implementation of each module of the system in detail. In Chapter Five, we discuss each model's evaluation and its comparison with the paper. Chapter six describes the application design in detail. Finally, we conclude the report and give directions to future work in Chapter seven.

Chapter Two

2 BACKGROUND AND RELATED WORK

2.1 OVERVIEW

Since the recognition and analysis of human activity have been actively studied, more than one research has been done to identify human thinking, analyze it, and try to make machines do the same tasks and with the same precision.

Related works range from simple transfer learning approaches for classifying input videos in different ways like (text analysis, facial expression analysis, and tone analysis) to seven basic emotions. Then came deep learning to combine all the methods to be a multi-model. Nowadays researchers want to build a multi-model that can auto analyze a person's personality from all sides (ex. Friendliness, Calm, focused, etc.…) as if he is interviewing with him, and not just identifying the seven basic emotions. Throughout the class, a range of methods, papers, surveys, and data sets are presented.

2.2 RELATED APPLICATION

HireVue[1] is a digital recruiting company dedicated to finding the best talent for their clients using video intelligence for interviews.

HireVue interviews are video-based and allow a company's recruiter to see non-verbal cues – such as facial expressions, eye movements, body movements, details of clothes, and nuances of voice. These non-verbal cues are collected as data points and processed by HireVue technology to perform meaningful assessments.

The process of the interview goes as follows[2]:
First, you find the invitation for the interview in your mail. Then you can choose to start a practice session or go straight to the interview. Then a description of the interview will appear. Next, you will be presented with the Terms & Conditions. You must accept them to begin. After that, you will be presented with an intro video from the company you applied for and test your webcam and microphone. Once you have tested your camera and microphone, you will see the option to try a

[1] https://www.hirevue.com/
[2] interview walkthrough

practice question or continue without practicing. The next step tells you how many attempts you will have to answer each question. Then you follow up the instructions to answer every question.

The recorded interview is evaluated by HR of the company, but they recently added AI to video in HireVue assessments[3] to let recruiters and hiring managers accurately and objectively measure a variety of candidate competencies.

2.3 IMPRESSIONS IN THE INTERVIEW

2.3.1 Facial Expressions

It is well known that first impressions reflected from the interviewee are facial expressions. They play an important role in determining the impressions of people in any interview, and on this basis, it is very important to focus on them and practice doing them.

1. Calming and relaxing during the interview.
2. Smiling.
3. Maintain eye contact.

2.3.2 Speech Emotion

One important factor that can make the interviewee have a good impression and be eligible to be hired is the sound of his voice. Job applicants should make sure that their voices are heard through the interview as the interviewer can discover the way that the interviewee thinks according to the changes in his voice's tone, cadence, and pitch as well as the power of his voice.

2.3.3 Body Language

During a job interview, the way you present yourself is just as important as what you say. Body language is an important indicator of your comfort, confidence, and interest for interviewers but why is it such an important thing to care about?!

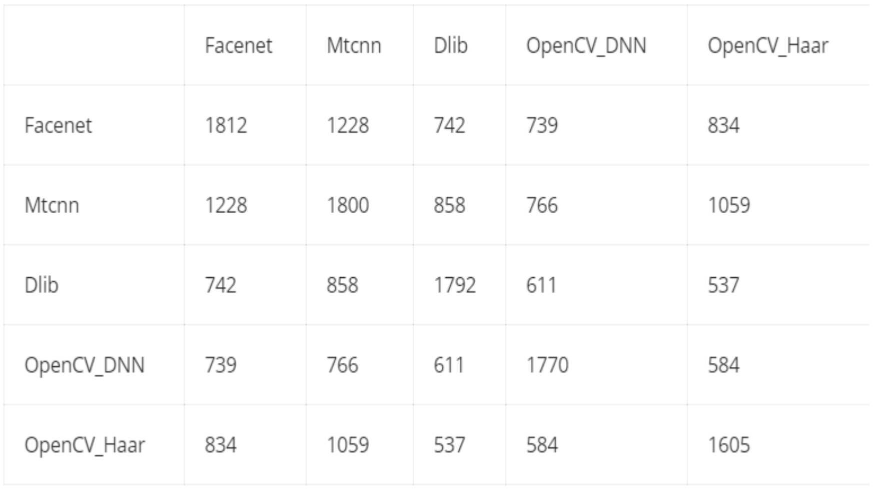
Your body language during a job interview reflects your emotions to the interviewer. There are certain movements and body positions that indicate negative emotions like nervousness or disdain, while others reflect interest and ease, and here are some of the tips collected from several articles[4-5-6-7].

[3] Hiring with video and AI
[4] body language tips 1
[5] body language tips 2
[6] body language tips 3
[7] body language tips 4

- Keep your body open.
- Lean forward from the waist.
- Slow your breathing down.
- Use a steady hand gesture to emphasize a point.
- Use hand movement or positive gestures.
- Palm upwards signifies honest and trustworthiness.
- Avoid clasping hands and clenching fists as they reflect anxiety.
- Avoid touching your face “signal of lying”.
- Wild hand or arm movements are not good, it reflects that you are nervous and unpredictable.
- Do not cross your arms, as it shows disrespect and could add negative impressions to the interviewer.

2.4 FACE DETECTION

At first, we try the five most popular algorithms which were developed for face detection algorithms[8]. Where their detection accuracy rate is at the following figure Fig [2.1]. Also, we will see details about five algorithms (OpenCV_Haar, OpenCV_DNN, Dlib, Mtcnn, and Facenet) in section (2.4.1, 2.4.2, 2.6.1, 2.4.4, 2.4.5).



	Facenet	Mtcnn	Dlib	OpenCV_DNN	OpenCV_Haar
Facenet	1812	1228	742	739	834
Mtcnn	1228	1800	858	766	1059
Dlib	742	858	1792	611	537
OpenCV_DNN	739	766	611	1770	584
OpenCV_Haar	834	1059	537	584	1605

Figure 2-1

[8] face detection algorithms

The time needed to process 699 frames.

Algorithms were run on Google Colab Using its GPU hardware accelerator.

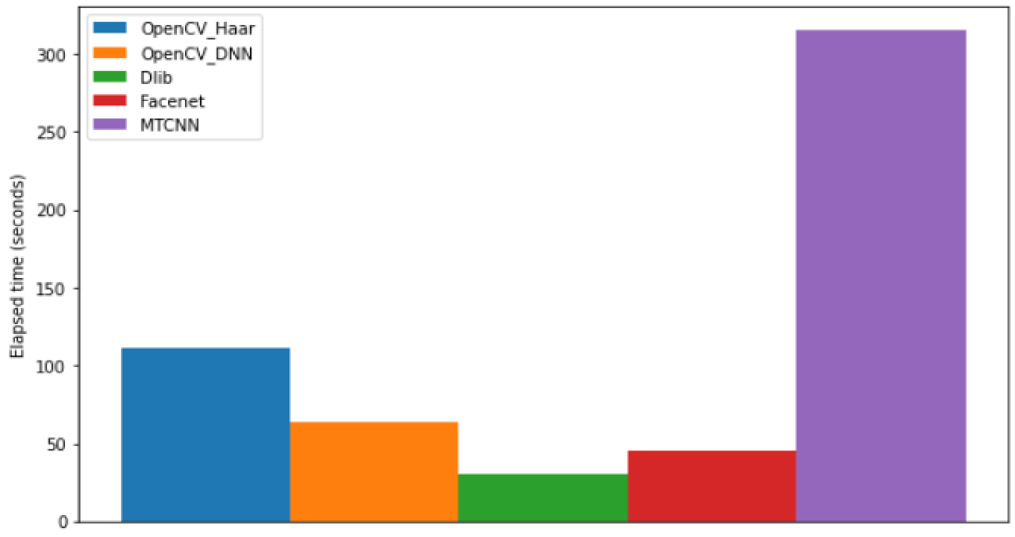


Figure 2-2

The previous graph, Showing the total time that the algorithms needed to process the video. The Dlib algorithm needed the shortest time to process the video. So, we decided to use the Dlib algorithm to detect faces in videos.

2.4.1 OpenCV Haar Cascade

It is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images. Then, it can be used on any image we want to detect faces in. It can be trained to detect a vast majority of objects. The following figure is an example of using it.

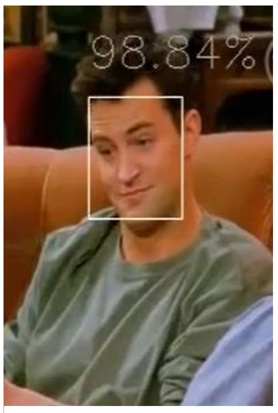


Figure 2-3

2.4.2 OpenCV DNN (Deep Neural Network)

In addition to OpenCV's Haar Cascade filter-based detection algorithm, OpenCV has released a DNN module, which stands for a deep neural network. To use it we need some files where these files can be downloaded from the Internet, or created and trained manually:

1. For Caffe:
   - res10_300x300_ssd_iter_140000_fp16.caffemodel.
   - deploy. prototxt.
2. For Tensorflow:
   - opencv_face_detector_uint8.pb.
   - opencv_face_detector.pbtxt.

2.4.3 Detecting a Face Using Dlib

It is a CNN-based detector, and it is generally capable of detecting faces from almost all angles. To use it, first, download the weights[9]. We will talk about it in detail in section 2.6.1.

2.4.4 MTCNN for Face Detection

MTCNN or Multi-Task Cascaded Convolutional Neural Network is unquestionably one of the most popular and most accurate face detection tools today. As such, it is based on a deep learning architecture, it specifically consists of three neural networks (P-Net, R-Net, and O-Net) connected in a cascade.

2.4.5 Detecting Faces with Facenet

It can be described as a unified embedding for Face detection and Clustering. It is a system that, when getting an image of a face, extracts feature from the face. This 128-element vector is used for future prediction and detection of faces, and it is generally known as face-embedding.
This model is a deep convolutional neural network that uses a triplet loss function for training. It encourages vectors of the same identity to become more similar, whereas vectors of different identities are expected to become less similar. Where The training of models focuses on creating embeddings directly rather than extracting them from intermediate layers of a model.

[9] face detection weights

2.5 FACIAL LANDMARKS DETECTION

2.5.1 DLIB and OpenCV[10]

Localizing the key points that describe the unique location of a facial component in an image (eyes, nose eyebrows, mouth, jawline, etc...). To do this, we need to develop a shape prediction method that identifies important facial structures. In our code we are going to implement a method developed by two Swedish Computer Vision researchers Kazemi and Sullivan in 2014, called One Millisecond Face Alignment with an Ensemble of Regression Trees. This detector is built in the Dlib library.

The following figure shows a set of 68 labeled facial points with specific coordinates that surround certain parts of the face.

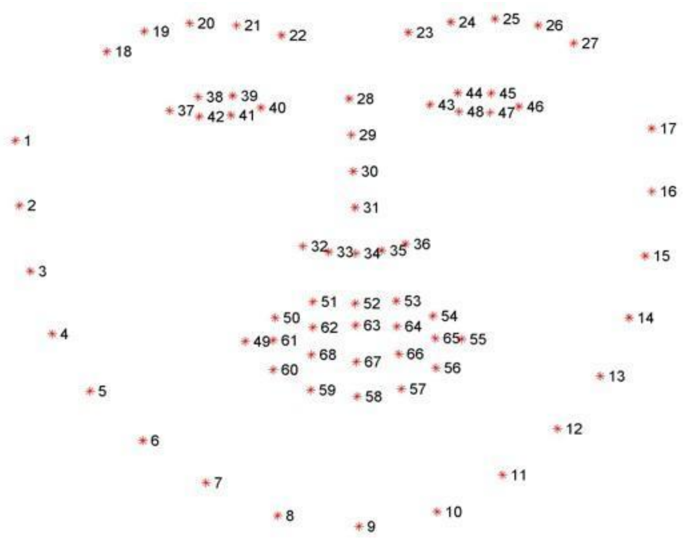


Figure 2-4

So, we calculate features of the face as follows:

- JAWLINE POINTS: 1 – 17.
- RIGHT EYEBROW POINTS: 17 – 22.
- LEFT EYEBROW POINTS: 22 – 27.
- NOSE BRIDGE POINTS: 27 – 31.
- LOWER NOSE POINTS: 31 – 36.
- RIGHT EYE POINTS: 36 – 42.
- LEFT EYE POINTS: 42 – 48.
- LEFT EYE POINTS: 42 – 48.
- MOUTH INNER POINTS: 61 – 68.

This process consists of two steps:

1. To localize the face on the image or video.
2. To detect the facial landmarks.

[10] dlib & opencv guide

2.5.1.1 Localize The Face on The Video Frame.

To detect the face in our image, we call a frontal face detector from the Dlib library. This is a pre-trained detector based on Histogram of Oriented Gradients (HOG) features, and a linear classifier in a sliding window detection approach. Then we create the object called faces in which we store all detected faces. In the object named faces, we have a list of coordinates of the detected faces (ex: Output: rectangles [[(128, 199) (770, 841)]] here we obtained coordinates for only two points, which indicates that we have only one face in the image).

2.5.1.2 Detect The Facial Landmarks.

To detect the facial landmarks, we load the facial landmark predictor shape predictor from the Dlib library. Additionally, for this shape prediction method, we download the file called "shape_predictor_68_face_landmarks.dat". Then, we use this predictor to extract the key facial features from the input image.

So, at first, we create a predictor object called landmarks in which two arguments are passed. As a first argument, we pass a gray image from which we want to detect available faces. As the second argument, we will specify the area where we are going to predict the facial landmarks. In our case, this area is represented by coordinates of the face that we already detected in the previous step.

2.5.2 FaceOSC[11]

We tried this tool to get files the same as MIT data, but the attempt did not work because the files came out differently. It is a tool[12] for prototyping face-based interaction. It is built on non-commercial open-source Face Tracker code from Jason Saragih. where FaceOSC comes as an example app with the ofxFaceTracker addon for open Frameworks. It will track a face and send its pose and gesture data over OSC, as well as the raw, tracked points (when selected in the GUI). Also, FaceOSC is developed with open Frameworks[13] and ofxFaceTracker[14], built on top of Jason Saragih's Face Tracker[15]. The windows build of FaceOSC was prepared by Dan Moore[16] on the CreativeInquiry fork of ofxFaceTracker[17].

Note: FaceOSC is used in paper [3] to extract facial features which are the same as OSC data. but we return to using Dlib and OpenCV as in section 2.5.1 with videos of the MIT dataset.

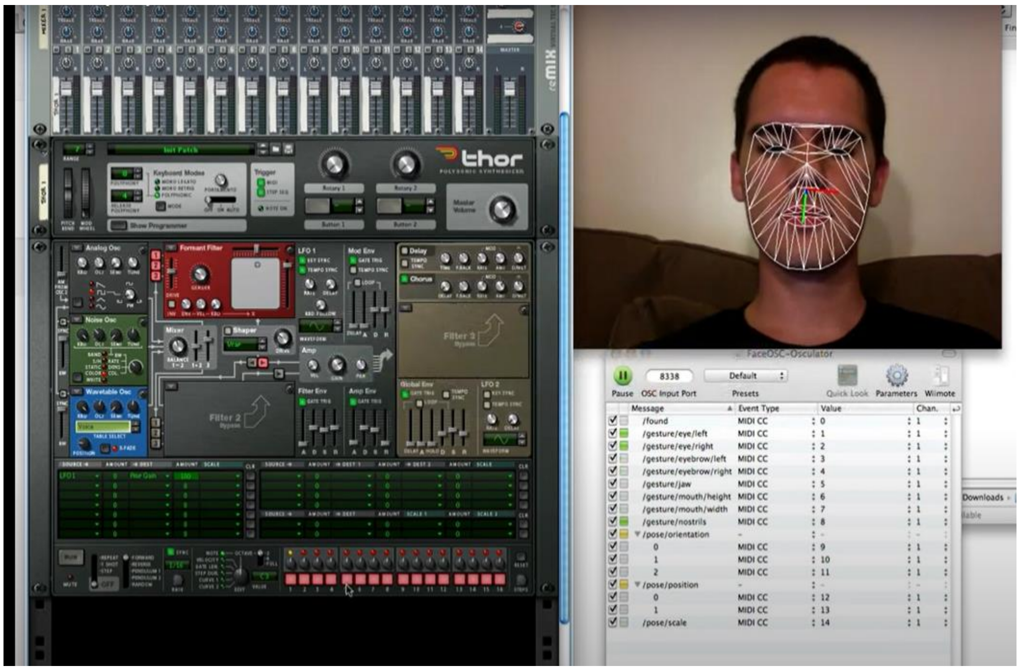
[11] FaceOSC
[12] FaceOSC tool download
[13] openFramWorks
[14] ofxFaceTracker
[15] Face tracker
[16] Dan Moore
[17] FaceOSC templates

2.5.2.1 OSC Data

By default, it sends to localhost, port 8338.

- Pose
  - center position: /pose/position.
  - scale: /pose/scale.
  - orientation (which direction you are facing): /pose/orientation.
- Gestures
  - mouth width: /gesture/mouth/width
  - mouth height: /gesture/mouth/height.
  - left eyebrow height: /gesture/eyebrow/left.
  - right eyebrow height: /gesture/eyebrow/right.
  - left eye openness: /gesture/eye/left.
  - right eye openness: /gesture/eye/right.
  - jaw openness: /gesture/jaw.
  - nostril flate: /gesture/nostrils.
- Raw
  - raw points (66 xy-pairs): /raw.

The following figures show samples when using it at MAC OS.



Message	Event Type	Value	Chan.
/found	MIDI CC	0	1
/gesture/eye/left	MIDI CC	1	1
/gesture/eye/right	MIDI CC	2	1
/gesture/eyebrow/left	MIDI CC	3	1
/gesture/eyebrow/right	MIDI CC	4	1
/gesture/jaw	MIDI CC	5	1
/gesture/mouth/height	MIDI CC	6	1
/gesture/mouth/width	MIDI CC	7	1
/gesture/nostrils	MIDI CC	8	1
/pose/orientation	-	-	-
0	MIDI CC	9	1
1	MIDI CC	10	1
2	MIDI CC	11	1
/pose/position	-	-	-
0	MIDI CC	12	1
1	MIDI CC	13	1
/pose/scale	MIDI CC	14	1

Figure 2-5

2.5.2.2 The Calculation to Extract Facial Features.

As explained in paper [3], The face tracker detects 66 interest points (as shown in the figure) on a face image. It works by fitting the following parametric shape model $x_i = sR(x_i + \psi_i q) + t$ where x_i is the coordinate of ith interest point and x_i denotes its mean location pre-trained from a large collection of hand-labeled training images. ψ_i denotes the bases of local variation for the ith interest point. Each element of the vector q represents a coefficient corresponding to a basis of local variation.

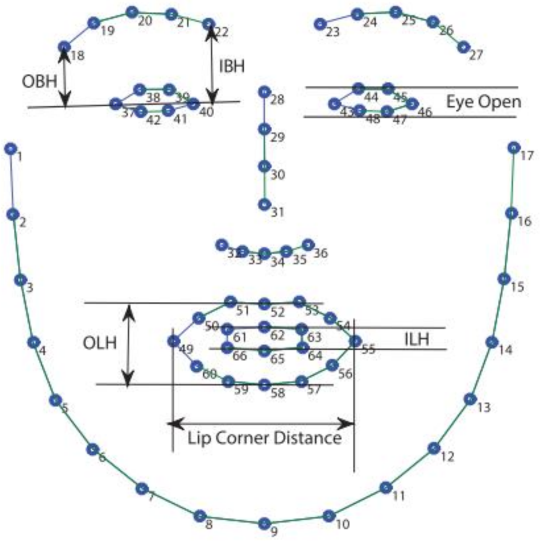


Figure 2-6

The parameters $s, R,$ and t correspond to the global transformation, respectively. The face tracker adjusts the model parameters $p = \{s, R, q, t\}$so that each of the mean interest points (x_i) on the test face. While extracting features from these tracked interest points, we want to disregard the global transformations (translation, rotation, and scaling), and consider only the local transformations which provide useful information regarding our facial expressions. After the face tracker converges to an optimal estimate of the parameters, we recalculate each of the interest points x_i by applying the local transformations only while disregarding the global transformations (s, R, and t). Mathematically, we calculate the following shape model from the optimal parameters obtained from the face tracker: $\widehat{x_i} = (\underline{x_i} + \psi_i q)$. Once we find $\widehat{x_i}$, we calculate the distances between the corresponding interest points to find out the features OBH (outer eye-brow height), IBH (inner eyebrow height), OLH (outer lip height), and ILH (inner lip height), eye-opening, and LipCDT (lip corner distance), as illustrated in Figure 2.6.

2.6 SELFIE CAPTURE WHEN WE SMILE USING OPENCV

We tried to use it instead of smile data features in the original MIT dataset, but the attempt did not improve the performance but rather disappointed it, so we decided not to add it as input features to our model. Selfie Capture is considered a separate application that is used in the camera of phones. Where taking an auto picture when detecting any smiling[18].

Process Overview:

[18] Smilefie

1. Use the facial landmark detector in Dlib to get the mouth coordinates.
2. Set up a smile threshold, using a mouth aspect ratio (MAR).
3. Take the image and calculate its MAR.

Fig. [2-7] shows only the twenty mouth coordinates:

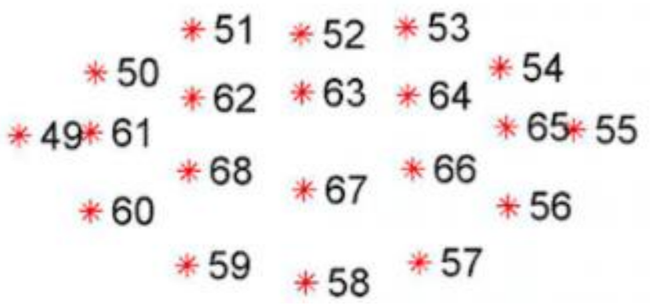


Figure 2-7

To find the ratio (MAR) of the region we can use the Euclidean distance formula as follows:

$$\frac{|P51 - P59| + |P52 - P58| + |P53 - P57|}{3\,|P49 - P55|}$$

Figure [2.8] shows how the MAR changes when we change mouth shapes.

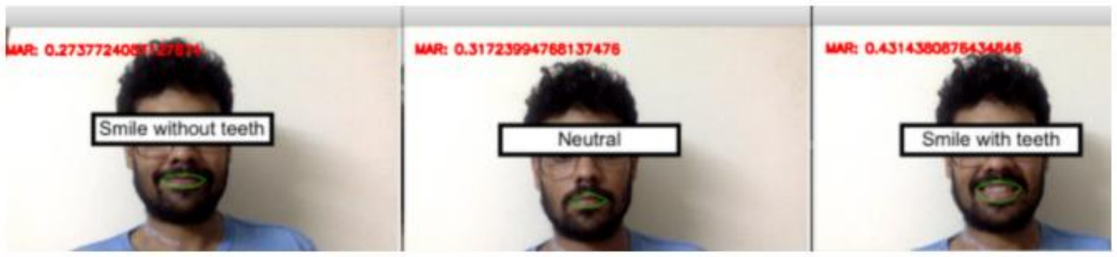


Figure 2-8

So, based on this, we set a smile to be a MAR (mouth aspect ratio) of <= 0.3 or > 0.38.

2.7 FACIAL POSE ESTIMATION

We use it to extract the Yaw, Roll, and Pitch features of detected faces in videos, where they are represented as following figures.

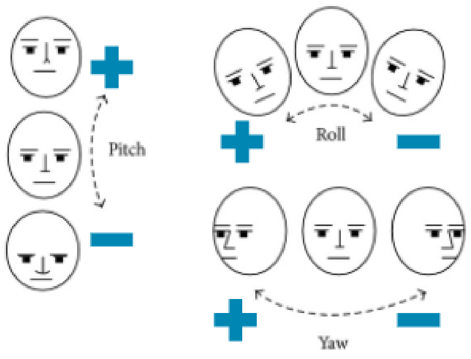


Figure 2-9

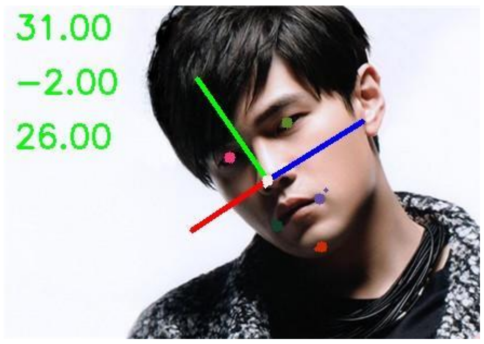


Figure 2-10

So, we use this algorithm [19] which extracts them by using OpenCV and Dlib.

[19] Face-Yaw-Roll-Pitch-from-Pose-Estimation-using-OpenCV

2.8 HAND LANDMARKS DETECTION

There are many ways to get the hand gesture as shown in the figure, but the most one suitable for our problem is using skeleton recognition[20].

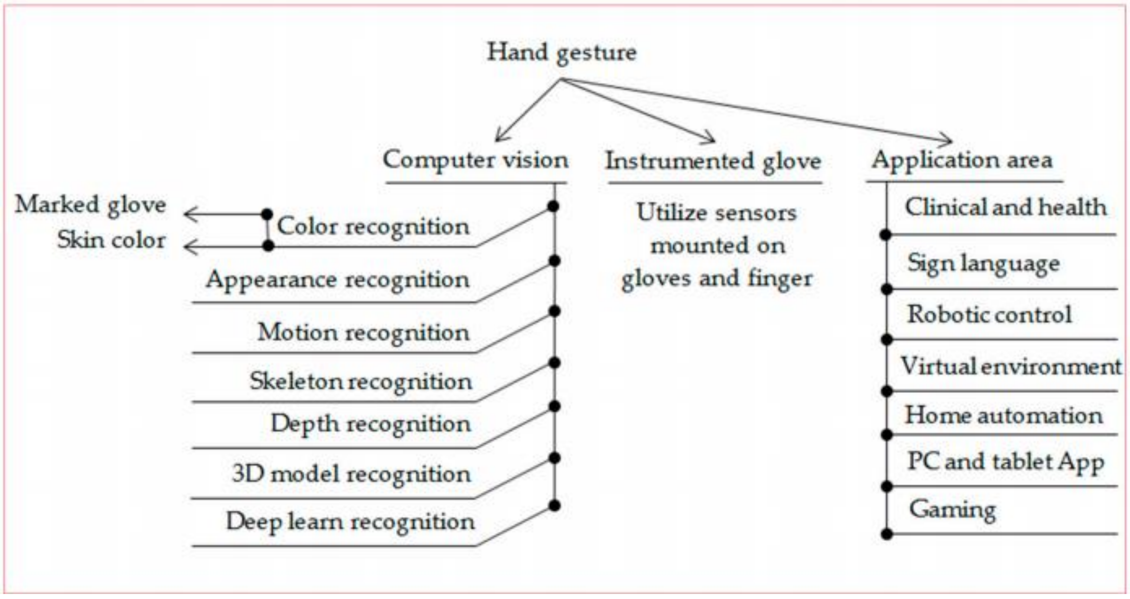


Figure 2-11

The skeleton-based recognition specifies model parameters that can improve the detection of complex features. Where the various representations of skeleton data for the hand model can be used for classification, it describes geometric attributes and constraints and easily translates features and correlations of data, to focus on geometric and statistical features. Hand segmentation techniques are using the depth sensor of the Kinect camera, followed by the location of the fingertips using 3D connections, Euclidean distance, and Geodesic distance over hand skeleton pixels to provide more accuracy. We cannot use the Kinect camera as it is not suitable for a web application, but some models could get the 3D location of points from normal RGB images. Gesture unit segmentation using support vector machines and skeletal dataset for segmenting gestures from rest positions[21], where supervised learning (SVM) is used for classification with an RBF kernel to classify each frame in a video if the person in it is [Rest -- Preparation -- Stroke -- Hold -- Retraction] using gesture phase segmentation data set. To be able to use such a method to classify between classes of hand gestures we want, we need to get skeletal data. There are many ways to get skeletal data, we need it to be fast and accurate.

[20] Hand Gesture Recognition using skeleton recognition
[21] presentation of gesture unit

This comparison was in Jan 2020[22]. it compares runtime with the number of people per image. According to our problem, there will be only one person in the image so AlphaPose will be the fastest.

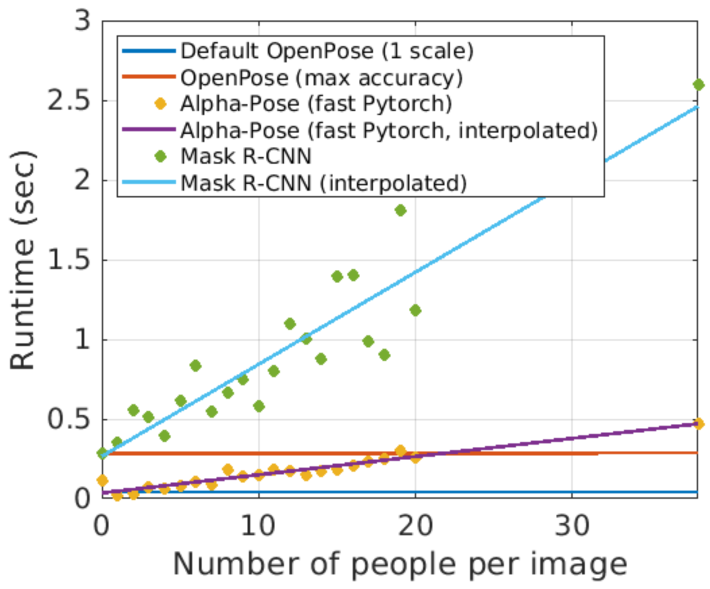


Figure 2-12

2.8.1 Openpose - CMU

In this method[23] Pure OpenCV has embedded DNN codes [cv2.dnn.blobFromImage] from v3.4 onwards and download pose_iter_102000.caffemodel weights file, one can do a net. forward to extract out the hand points. As always, the use of a single image is limited in application. Using a webcam will allow a stream of hands and the speed of prediction will be more important and techniques can be added to increase the accuracy such as green screen the background before detection and python multithreading/processing. It is a fast method that can be used in real-time. The problem is that it is a heavy model to use on some machines and it is not easy to work with like other models.

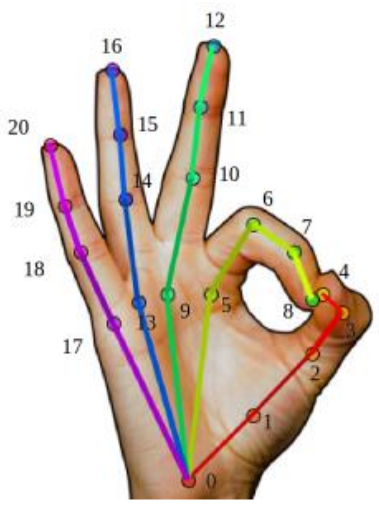


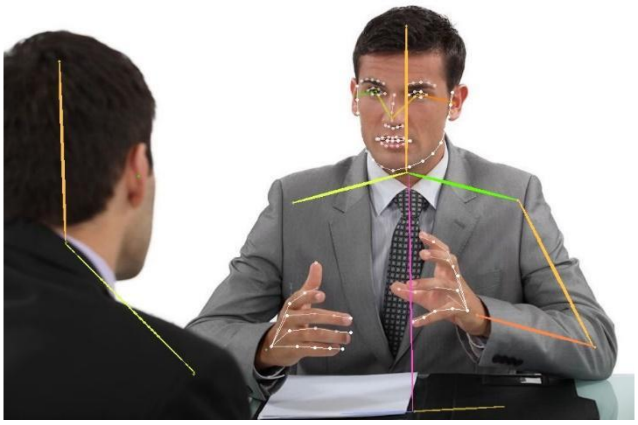
Figure 2-13

[22] comparison Jan 2020
[23] openpose hand detection github

2.8.2 Alpha Pose “RMPE”

Hand detection was added to Alpha-Pose in v0.4.0 on Aug 2020 by Halpe 136 model[24].

It is not as fast as we want since the Halpe model takes more time due to getting all data points in the hand, not only the arm and shoulder.



It takes 48 sec on GPU for ten-sec videos with dim 640 x 480.

Figure 2-14

If we use it, we will not take all 136 points but only shoulders, bows, wrists, and all palm key points.

2.8.3 Openpose - PyTorch

Pytorch implementation of Openpose including Body and Hand Pose Estimation, it is directly converted from openpose caffemodel by caffemodel 2 PyTorch[25-26].

It is also not as fast as we want.

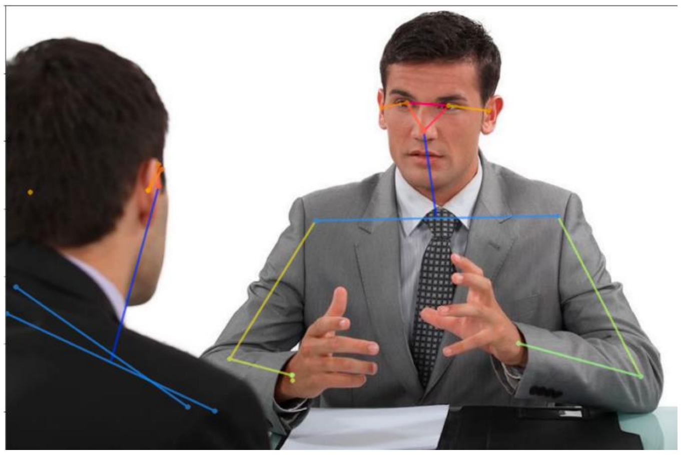


Figure 2-15

2.8.4 RCNN Keypoint - PyTorch

The PyTorch pre-trained models for human pose and keypoint detection. It is the key point RCNN deep learning model with a ResNet-50 base architecture[27].

This model has been pre-trained on the COCO key point dataset. It outputs the key points for 17 human parts and body joints.

It is a fast model, gets points of arms, shoulders, ...etc. but not hand and fingers key points.

[24] RMPE github
[25] Pytorch openpose 1
[26] Pytorch open pose 2
[27] RCNN key-point pytorch

2.8.5 PoseNet - TensorFlow

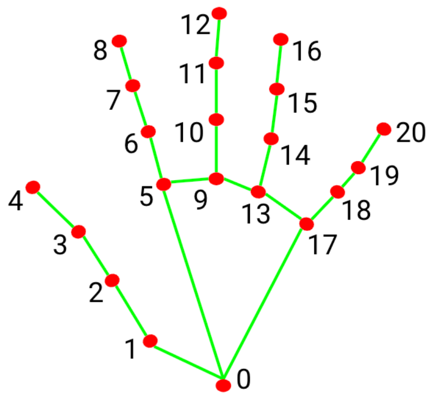
The PoseNet model[28] takes a processed camera image as the input and output information about key points. The key points detected are indexed by a part ID, with a confidence score between 0.0 and 1.0. The confidence score indicates the probability that a key point exists in that position. It outputs the key points for 17 human parts and body joints.

PoseNet is much faster than any other model “Execute approximately 65 FBS”, but it had a lot of missed poses throughout the video, and it does not get fingers key points.

The only needed points will be five left Shoulder - six right Shoulder - seven left Elbow - eight right Elbow - nine left Wrist - ten right Wrist.

2.8.6 MediaPipe Hands Landmark

MediaPipe hand landmark model[29-30] performs precise key point localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 2-16

Its output is:

- X, Y coordinates: 21 hand landmarks coordinates normalized to [0.0, 1.0] by the image width and height, respectively.
- Z: Represents the landmark depth with the depth at the wrist being the origin, and the smaller the value, the closer the landmark is to the camera.
- Label: "Left" or "Right" Hand.

[28] Pose estimation tensorflow
[29] mediapipe Hands
[30] mediapipe github

- Score: Estimated probability of the predicted handedness >= 0.5

It is very fast, as an example from MIT data:
2:15 is the time needed to get data from a video of length = 3:17.

2.9 HAND OVER FACE

2.9.1 Face Alignment and HOG

The methodology according to [1] was first to get the facial area from the videos then normalize it using a similarity transformation, then do the face alignment step to get a 160 x 120 normalized pixel image. The second step was to extract the HOG features from a similarity normalized 160 x 120 pixel image of a face by using 8 x 8 pixel cells with 18 gradient orientations and a block size of 2 x 2 cells. Then, slide a window of size ten frames on every video and take the average of it. Finally, train a linear SVM classifier to detect face touch.

2.9.2 YOLOv3 Object Detection

The problem with the first method is that it takes too much time for the preprocessing step, so using Keras creator Francois Chollet's idea[31].
Using a pre-trained YOLOv3 then training the model on the facetouch dataset we will get the model that we want with good performance[32].

2.10 SPEECH EMOTION RECOGNITION

In this part, we need to extract features from the speech of people. The most popular effective way for feature extraction techniques is the MFCC[4]. Figure [2.17] shows the MFCC block diagram.

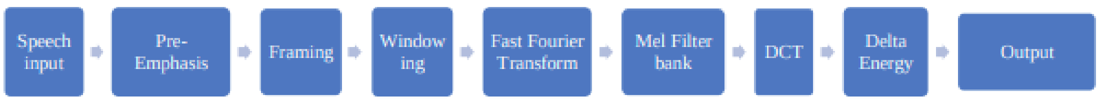


Figure 2-17: MFCC block diagram

"An Approach to Extract Feature using MFCC," by Parwinder Pal Singh, Pushpa Rani. August. 2014, *IOSRJEN,* Vol. 04, Issue 08, 1. ISSN (e): 2250-3021, ISSN (p): 2278-8719.

[31] keras creator tweet
[32] face touch github

After that, we need to extract Delta features, Delta of Deltas features, and the logbank filters that extract 40 features from each frame which helps in training with the models.

2.10.1 Using PyAudioAnalysis Library

We tried to use the PyAudioAnalysis library[33] in extracting features of MFCC. However, we calculate by hand the Delta and Delta of Deltas. It helps us also in reading the audio with the output of sample rate and amplitude of the audio. Also, in amplifying the audio to make the voice clear. We found then that this amplification did not affect very much.

The problem of this library is that we cannot get the Delta, and Delta of Deltas features, as well as the logbank filters which we found, are important in training the Speech Emotion Recognition model. So, we searched and found librosa which helps us with the missing factors we need.

2.10.2 Using Librosa Library

We tried to use the librosa library[34] in extracting features of MFCC, Delta, and Delta of Deltas and get the logbank filters which help us in training the data of audios from MIT interviews.

It also helps us in preprocessing the data, which is to get the sample rate, MFCC, logbank filter for each audio.

We decided not to continue with this library as the code of it is big to extract what we need as we find another library to use which is PythonSpeechFeatures.

2.10.3 Using PythonSpeechFeatures Library

PythonSpeechFeatures library[35] helps us to get the MFCC, Delta, Delta of Deltas features most simply and easily as it just needs a few lines of codes without much calculation to get the features we need to be used in training the model of the Speech Emotion Recognition.

2.10.4 Training Using the SVR Model

The model implemented to be used in training the dataset of MIT is the SVR model with some preprocessing like scaling the data and with some parameter tuning using GridSearchCV.

[33] pyAudioAnalysis github
[34] librosa library
[35] python speech features library github

2.10.5 Train With a Pre-Trained Model

We use the RNN model with trained weights[36] of the RAVDESS dataset. We load the weights trained and load the labels of the MIT dataset then split the data into train and test sets to make the transfer learning on the pre-trained model with SVR model to get the correlation result from the five traits' results found in the paper of MIT to be compared with.

We think about it that way because the data from MIT is small and we thought that we could get better results. Unfortunately, the results we got from the transfer learning were not much better than what we got from training data from MIT using the SVR model to get the regression results of the traits. So, we decided to not rely on this model and continue with the SVR model.

2.11 DATASETS

2.11.1 MIT Interview Dataset

They release the MIT Interview Dataset containing the audio-visual recordings of 138 mock job interviews, conducted by professional career counselors with 69 undergraduate MIT students[37]. In addition to the videos, they release the Amazon Mechanical Turk ratings for each of the videos, the final ground truth ratings, and the processed feature values.
The data is divided as follows:

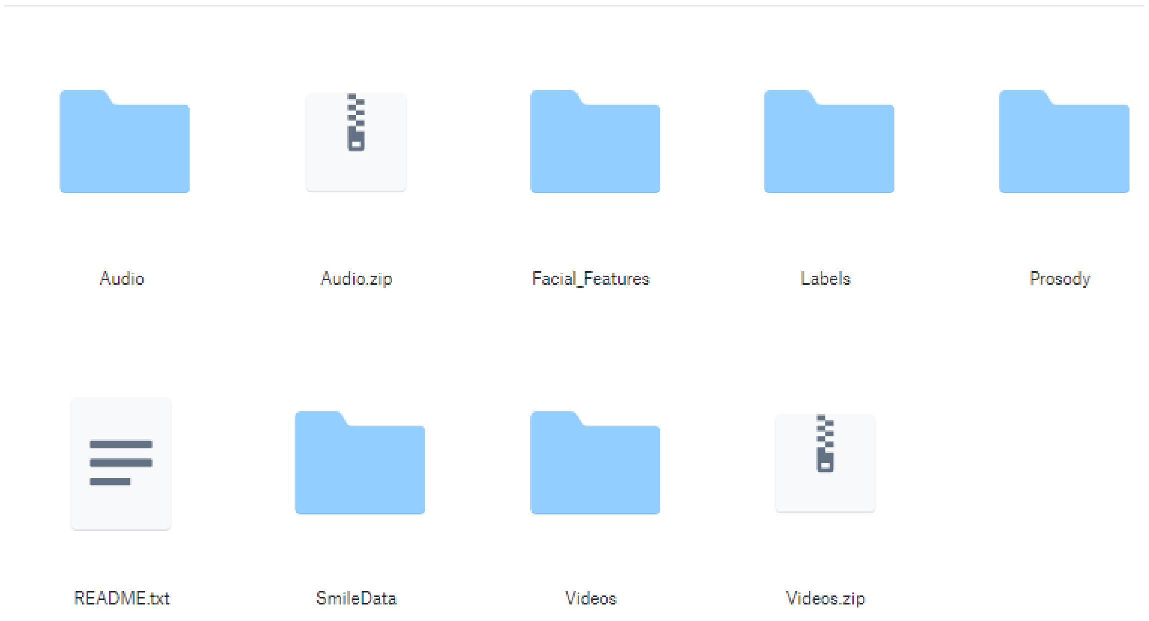


Figure 2-18

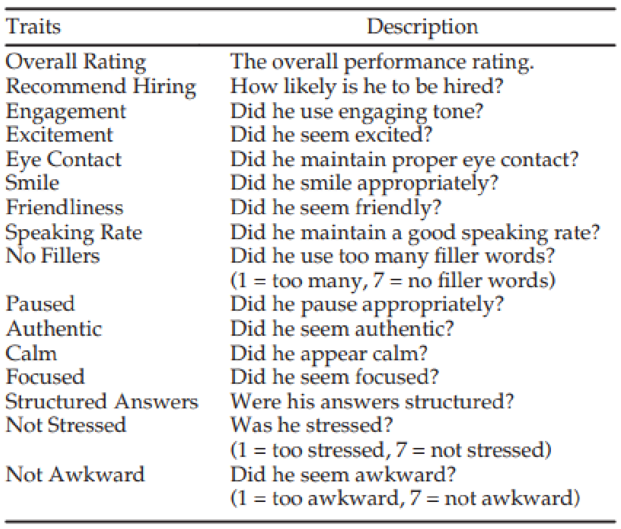
[36] Pretrained Model github
[37] MIT dataset

1. Videos' folder:
   - This directory contains 138 videos of 69 participants.
   - Each participant was interviewed twice, once before and once after counselor intervention. Each video file name looks like either "P#.avi" or "PP#.avi"; Where # is the participant id.
   - Video files with names like "P#.avi" indicate pre-intervention interviews.
   - Video files with names like "PP#.avi" indicate post-intervention interviews.
2. Audios:
   - This directory contains audio recordings for 138 interviews, in a similar format as the video files.
3. Facial Features:
   - This directory contains one CSV file for each interview video.
   - Each CSV file contains face tracking features extracted from each frame in the video.
   - The most crucial features are the first 12 features: Pitch, Yaw, Roll, Left Inner Eyebrow (inBrL), Left Outer Eyebrow (otBrL), Right Inner Eyebrow (inBrR), Right Outer Eyebrow (otBrR), Left Outer Eye (EyeOL), Right Outer Eye (EyeOR), Outer Lip Height (oLipH), Inner Lip Height (iLipH), and Lip Corner Distance (LipCDt).
4. SmileData:
   - This directory contains facial features extracted via SHORE.
   - This directory contains two sub-directories: (1) pre and (2) post.
   - The directory "pre" contains features extracted from the pre-intervention videos.
   - The directory "post" contains features extracted from post-intervention videos.
   - The first three columns in each text file were used in our experiments:
     (1) smile intensity, (2) head nod, and (3) head shake.
5. Prosody:
   - This directory contains one CSV file, containing the prosodic features. Each line in the CSV file stores the aggregated PRAAT features for one of the questions in one of the interviews.
6. Labels:
   - This directory contains the annotations from Amazon Mechanical Turk workers.
   - It contains two CSV files:

     (1) interview_transcripts_by_turkers.csv:
     - contains the speech transcripts for each of the 138 video files.

(2) turker_scores_full_interview.csv:

- contains the Turkers' ratings for full interviews.
- For each video, we collect ratings from multiple Turkers.
- The CSV file contains both the participant/video id and worker id.
- We aggregate the ratings from multiple Turkers and compute one aggregated rating using Expectation Maximization.
- The aggregated rating is marked as worker id "AGGR".



Traits	Description
Overall Rating	The overall performance rating.
Recommend Hiring	How likely is he to be hired?
Engagement	Did he use engaging tone?
Excitement	Did he seem excited?
Eye Contact	Did he maintain proper eye contact?
Smile	Did he smile appropriately?
Friendliness	Did he seem friendly?
Speaking Rate	Did he maintain a good speaking rate?
No Fillers	Did he use too many filler words? (1 = too many, 7 = no filler words)
Paused	Did he pause appropriately?
Authentic	Did he seem authentic?
Calm	Did he appear calm?
Focused	Did he seem focused?
Structured Answers	Were his answers structured?
Not Stressed	Was he stressed? (1 = too stressed, 7 = not stressed)
Not Awkward	Did he seem awkward? (1 = too awkward, 7 = not awkward)

Figure 2-19

2.11.2 The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)[38]

Description:

- It contains 7356 files (total size: 24.8 GB).
- The actors of the database are 24 professional actors (12 female, 12 male)
- Vocalizing two lexically matched statements in a neutral North American accent.
- Speech includes calm, happy, sad, angry, fearful, surprise, and disgusted expressions.
- The song contains calm, happy, sad, angry, and fearful emotions.
- Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.
- All conditions are available in three modality formats:

[38] RAVDESS dataset

    - Audio-only (16bit, 48kHz .wav)
    - Audio-Video (720p H.264, AAC 48kHz, .mp4)
    - Video-only (no sound).

File naming convention:

- Each of the 7356 RAVDESS files has a unique filename.
- The filename consists of a 7-part numerical identifier (e.g., 02-01-06-01-02-01-12.mp4). These identifiers define the stimulus characteristics.

Filename identifiers:

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

2.11.3 20BN[39]

Total number of videos	148,092
Total number of frames	5,331,312
Number of classes	27
Avg. duration of videos	3 sec
Avg. number of videos per class	4391

Table 1

The problem with this dataset is that it is too large, and we cannot extract it either locally or on google drive as some files are corrupted. One solution is that Part of the data is uploaded on Kaggle which

[39] 20BN dataset

we get easily without any problems, but the distribution of classes changed. Here the total number of frames is approximately half, from 5,331,312 to 2,380,000.

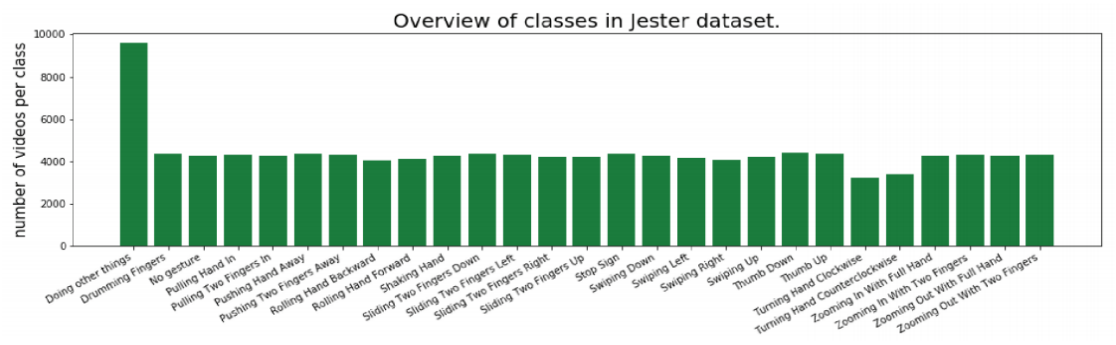


Figure 2-20

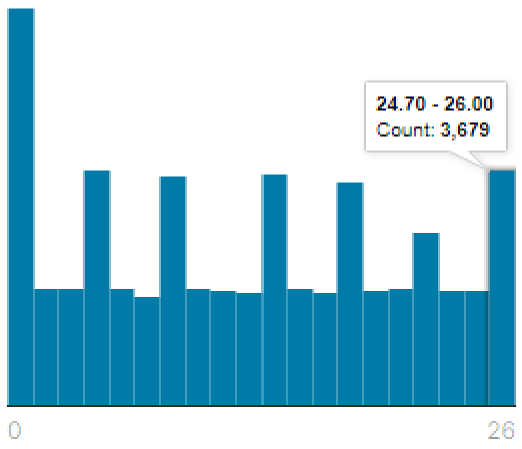


Figure 2-21

2.11.4 Gesture Phase Segmentation Dataset

The dataset[40] is composed of features extracted from seven videos with people gesticulating, aiming at studying gesture phase segmentation.

Frames = 9593 frames, Features = 32 features "for four points position, velocity and acceleration"

Each video is represented by two files:

A raw file, which contains the position of hands, wrists, head, and spine of the user in each frame as 18 numeric attributes (double), a timestamp, and a class attribute (nominal).

[40] Gesture Phase Segmentation Data Set

A processed file, which contains velocity and acceleration of hands and wrists as 32 numeric attributes (double) and a class attribute (nominal).

Classes: Rest -- Preparation -- Stroke -- Hold -- Retraction

A feature vector with up to 50 numeric attributes can be generated with the two files mentioned above.

Positions are as in this figure, In the first line of frames, there is a rest position sequence from the fourth to the ninth frame. It is difficult to determine the first frame of the sequence, and there is a hand displacement into the sequence. In the second line, there is a hold sequence from the fourth to the seventh frame. It is a period with an almost entire absence of movements into a gesture unit[41].

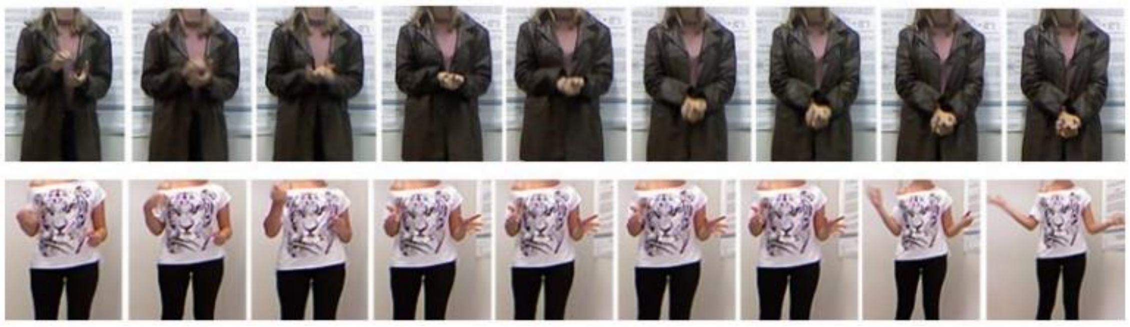


Figure 2-22

2.11.5 Face Touch Dataset

The data[42] contains 1040 images of the hand touching different facial areas it is then split into 75% train, 15% validation, and 10% test the dataset directory contains.

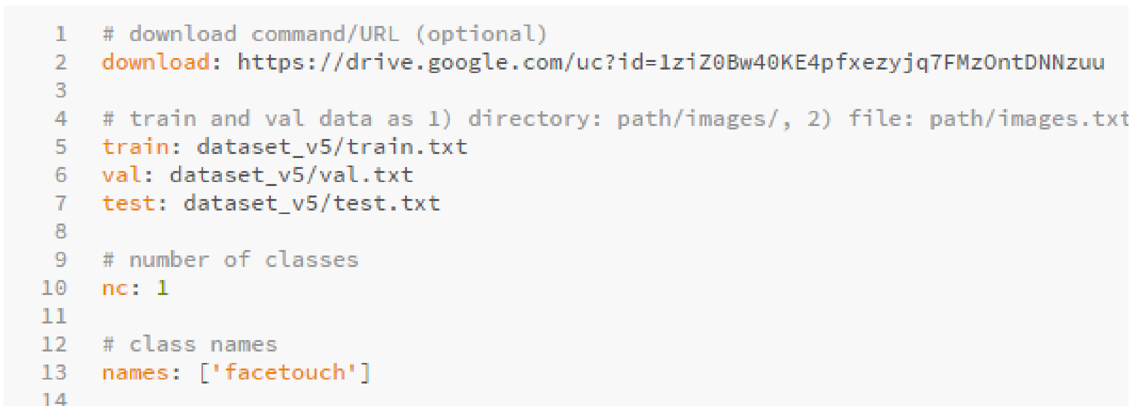
- dataset_v5
  - ➢ Images' directory:
    - ○ train
    - ○ validation
    - ○ test
  - ➢ Labels' directory
    - ○ train
    - ○ validation
    - ○ test
  - ➢ train.txt

[41] Gesture Unit Segmentation Using Spatial-Temporal Information and Machine Learning
[42] facetouch dataset_v5

- ➢ val.txt
- ➢ test.txt.
- ➢ coco1.names
- ➢ coco1.yaml

The images directory contains images touching the different facial areas and the labels directory contains text files every row on this format <object-class> <x> <y> <width> <height> where object-class is the class number, x and y are the centers of the bounding box that contains the object, width is the width of the bounding box and height is the height of the bounding box. x,y, width, and height are normalized between zero and one.

train.txt contains the path of the training samples, val.txt contains the path of the validation samples and test.txt contains the path of the validation samples. coco1.names contain the name of the classes, but it is not needed as it can be merged with coco1.yaml. coco1.yaml is needed for the yolov3 model and it will be rewritten into this form.



```
# download command/URL (optional)
download: https://drive.google.com/uc?id=1ziZ0Bw40KE4pfxezyjq7FMzOntDNNzuu

# train and val data as 1) directory: path/images/, 2) file: path/images.txt
train: dataset_v5/train.txt
val: dataset_v5/val.txt
test: dataset_v5/test.txt

# number of classes
nc: 1

# class names
names: ['facetouch']

```

Figure 2-23

2.11.6 Cam3d

The dataset was taken from Cambridge it contains 108 videos each video is 30FPS and seven seconds long. The data is split manually into two directories “occlusion” and “no occlusion”. There were 24 videos for occlusion and 84 videos for no occlusion. It is used with face alignment and the HOG method and can be used with the YOLOv3 model if we extract the frames that have a facetouch.

There were some problems in the dataset:

- A person covering the entire camera with his hand.
- The light in some frames goes off so dark frames exist.
- extreme head rotation will cause a problem while using the first method in hand over face.

Chapter Three

3 PROPOSED SYSTEM OVERVIEW

3.1 OVERVIEW

The system consists of four modules (Facial Expression - Speech Emotion - Hand Gesture - Hand Over Face) each module is independent of the other. After finishing all of them, combine them to get the best results for analyzing interview videos.

3.2 FACIAL EXPRESSIONS MODULE

We trained the model in this part using videos and labels of MIT data. Also using SVR as a model. We tried many combinations of features, but in the end, we settled on the following: twelve features which extracted from videos using Dlib (where nine features are extracted using an algorithm in section 2.5.1 and three features of pose estimation extracted using an algorithm in section 2.7) by following steps:

Firstly, we create files for each video of the MIT dataset which contains twelve features. Second, we create a split file that contains the time of the start and end of five question's answers. Third, we take the average of each collection of rows of each question's answer of each video and save it into a new file. Where for each video we have (12 × 5) average numbers so finally we have one file have all feature of all videos which contain 138 rows (correspond to each video) and 60 columns (correspond to 12 feature × 5 answers = 60).

Then we use this file that contains all these features and use grid-search with the SVR model to find the best parameters for each label in the data

3.3 SPEECH EMOTION MODULE

Firstly, we access all paths of the audio and get the labels of AGGR from the turker scores full interview file. After that, we get the sampling rate and signal from each audio and calculate the mean of the signal. Then, we extract MFCC, Delta, and Delta of Deltas using log bank filters from the PythonSpeechFeatures library with window size 0.025 and window step 0.01 and with a filter of 40 and with the parameter of fast Fourier transform which helps in converting the audios from continues signals to discrete signals to get the fast Fourier transform points to be computed in one frame. The parameter of fast Fourier transform is computed by multiplying the window length to a sampling rate of the signal. we did this preprocessing to all audios of the P# and PP# audios.

After that, we trained the model in this part using audios and labels of MIT data. Also using SVR as a model. We tried many combinations of features, but in the end, we settled on the following: 110 × 40 × 3 features which extracted from audios using PythonSpeechFeatures library (110 is the number of the training dataset, 40 is the number of features extracted from each interview, and 3 is the number of the MFCC (log_mel), Delta, and Delta of Deltas). So, there are 120 features for each interview audio.

So, the model has input 110 × 40 × 3 features and outputs the correlation value for each trait of the labels from the MIT dataset by regression.

3.4 HAND GESTURE MODULE

Hand Gesture model with MIT dataset after labeling videos in it (each frame is either rest - move - clasped hand - crossed arms), the result was 57 videos had labels and the rest of 138 videos hands did not appear in them.
Using the same idea used in this paper [2] but with some changes. Since the analysis of results in the paper proves that using hand "wrist" points only has results better than using shoulders and using only past time displacement gives better results as well, we decide to work on all hand-points we could get and use only past time displacement.

So, the model will take data of points on the hand and velocity/acceleration of wrist point and pass it to the SVC model to result in a label from (rest - move - clasped hand - crossed arms)
Steps to work with data was:

1. Using MediaPipe Hands to get points of both hands (21points × X, Y, Z) ×2 hands = 126.
2. Make the same calculations as above to get velocity and accuracy "for wrist only" = 8

$$v_{i,i-d} = \frac{\Delta r_{i,i-d}}{t_i - t_{i-d}} \qquad a_{i,i-1} = \frac{v_i - v_{i-1}}{t_i - (t_{i-1})}$$

3. Using only past time displacement with d = 10

After processing data, Applying Grid Search on SVC to get the best parameters, Train with it.
The result of a video is an array containing labels for each frame. By counting the number of frames for each label and dividing it with the number of total labels we get the ratio of this label to the video and can also get the ratio of undetected hand frames by subtracting all ratios from one.

3.5 HAND OVER FACE MODULE

The model used is the YOLOv3-SPP model from ultralytics[43] with facetouch data[44] the input image, video, path images, path videos, or video URL and the output frames whether they have facetouch or not.

YOLOv3-SPP uses the same architecture of YOLOv3 which uses a variant of Darknet, which originally has a 53-layer network trained on ImageNet. For the task of the detection, 53 more layers are stacked into it giving us a 106 layer fully convolutional underlying architecture for YOLOv3, but the difference is that YOLOv3-SPP add an SPP structure to the architecture as shown in the figure which helps to solve two problems in the YOLOv3:

1) Effectively avoid the problems of image distortion caused by cropping and zooming the image area.
2) The problem of repeated feature extraction of the image by the convolutional neural network is solved, the speed of generating candidate frames is greatly improved, and the calculation cost is saved.

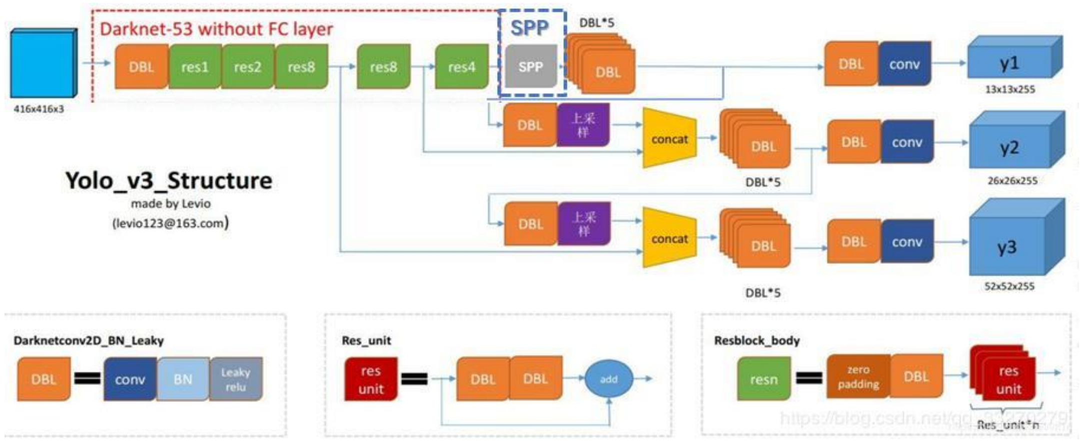


Figure 3-1: YOLOv3-SPP architecture

[43] ultralytics yolov3
[44] facetouch dataset

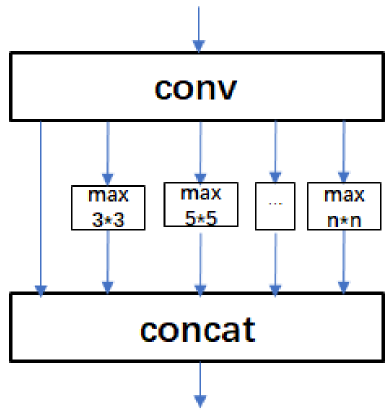


Figure 3-2: SPP Structure

3.6 COMBINATION OF MODULES

We have tried to combine models in many ways:

1- Combining hand movement and the proportion of hands touching the face

2- Combining hand movement, facial and sound

3- Combining facial and sound only

And in each case, either we use a feature and insert it on the SVM model, or we do a linear regression on the results of each model separately, or we do simple averaging or weighted averaging.

Chapter Four

4 IMPLEMENTATION OF SYSTEM

4.1 FACIAL EXPRESSIONS MODULE

4.1.1 Data Pre-Processing

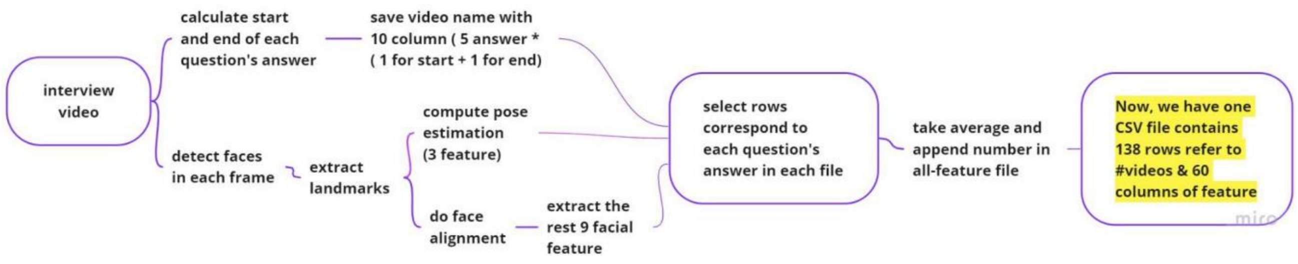

interview video
calculate start and end of each question's answer
save video name with 10 column (5 answer * (1 for start + 1 for end)
detect faces in each frame
extract landmarks
compute pose estimation (3 feature)
do face alignment
extract the rest 9 facial feature
select rows correspond to each question's answer in each file
take average and append number in all-feature file
Now, we have one CSV file contains 138 rows refer to #videos & 60 columns of feature
miro

Figure 4-1

As shown in the figure steps for doing preprocessing to data to extract input features. So let us explain the generated data. It is a CSV file consisting of 60 columns and 138 rows, where the original data are 138 videos, and each person in the video answers the same five questions, so we multiply the number of features by five. Also, we extract the following twelve features from detected faces in video's frames:

- Pitch, Yaw, Roll
- inBrL & otBrL: Left Inner Eyebrow & Left Outer Eyebrow
- inBrR & otBrR: Right Inner Eyebrow & Right Outer Eyebrow
- EyeOL & EyeOR: Right & left Outer Eye.
- oLipH, iLipH: Outer & Inner Lip Height
- LipCDt': Lip Corner Distance

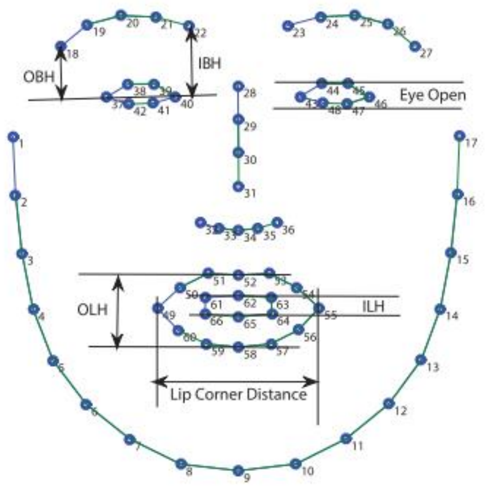


Figure 4-2

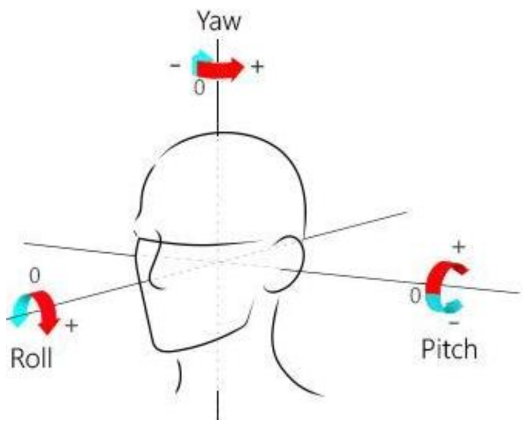


Figure 4-3

4.1.2 Training Model

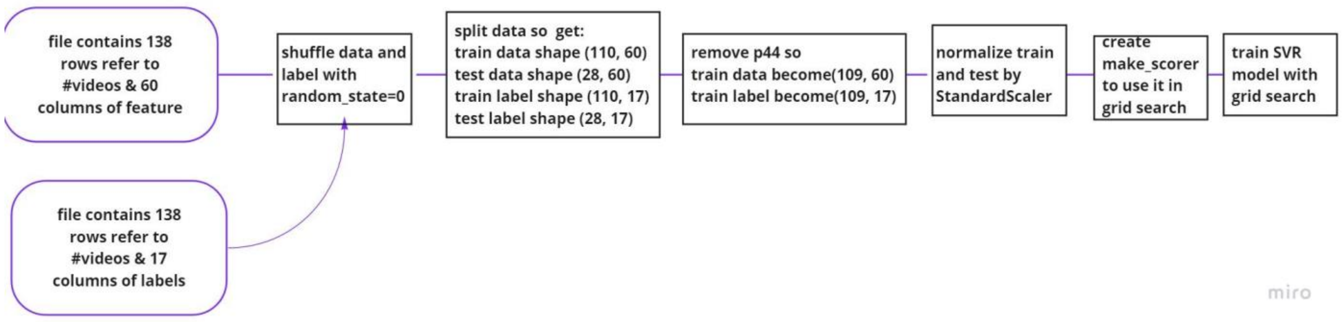

file contains 138 rows refer to #videos & 60 columns of feature
file contains 138 rows refer to #videos & 17 columns of labels
shuffle data and label with random_state=0
split data so get: train data shape (110, 60) test data shape (28, 60) train label shape (110, 17) test label shape (28, 17)
remove p44 so train data become(109, 60) train label become(109, 17)
normalize train and test by StandardScaler
create make_scorer to use it in grid search
train SVR model with grid search
miro

Figure 4-4

As we see in the figure, we are doing some processing before training the SVR model. At first, we shuffle our data with random_state = ZERO then split our data to train set and test set at 80% and 20%, respectively. Second, we remove one video from train data because it has a null feature (when we watched this video, we noticed that he is a dark-skinned person, so Dlib could not determine the face from the frames). Then normalizing our train and test sets by using standard scalar. Finally, we create a grid search with different hyperparameters of SVR and create a new score function because our measures are calculated from correlation, so we used the Pearson correlation coefficient.

4.2 SPEECH EMOTION MODULE

4.2.1 Data Pre-processing

Split all audios into five parts for each audio interview as the interviewee answers will be split into five audios each one containing one answer for one question. Then save the audios in google drive to access the paths of the audios later. So, we have 138 × 5 = 690 audios.

In preprocessing the data, we load the labels of MIT corresponding to AGGR labels of all traits from the turker_scores_full_interview.csv file. Also, load the audios from the drive.

For each audio, we read the wave signal and get its sequence and sampling rate. Then get the mean of the signal.

After that, we extract the MFCC features using the logfbank() function from the PythonSpeechFeatures library which is the log bank filters with some parameters like sampling rate and mean of signal we extracted before, window length equals 0.025, window step equals 0.01, the number of filters equals 40, and the number of fast Fourier transform points to be computed in one frame equals window length (i.e. 0.025) multiplied by sampling rate (i.e. 4800 which is constant for audios) which equals 1200.

Then calculate the Delta by delta () function from the same library which is the derivative of MFCC features extracted. Calculate the Delta of Deltas by the same way which is the derivative of Delta features.

Then stack all the extracted features together into a frame and get the mean of that frame, then appended into a list to be used after that.

Repeat that step for all audios and do the same once for P# audios and another for PP# audios and save the preprocessed data.

Finally, we get the average of each five audios from the saved data and save the data from P# and PP# in one file to create one file containing all the preprocessed data that have all features extracted from each interview audio for the 138 interviews.

4.2.2 Training Model

Load the features saved from the preprocessing step, split the data into training and test sets as 80%: 20% respectively. Then do some normalization using a min-max scaler from python and do some tuning using GridSearchCV with C parameter (1, 10, 100, 1000) and kernel parameter (linear, RBF).

After that, train with the SVR model from SVM and get the correlation using Pearson for each trait for AGGR labels. Finally, save the weights created from each trait.

4.3 HAND GESTURE MODULE

4.3.1 Data Preparation

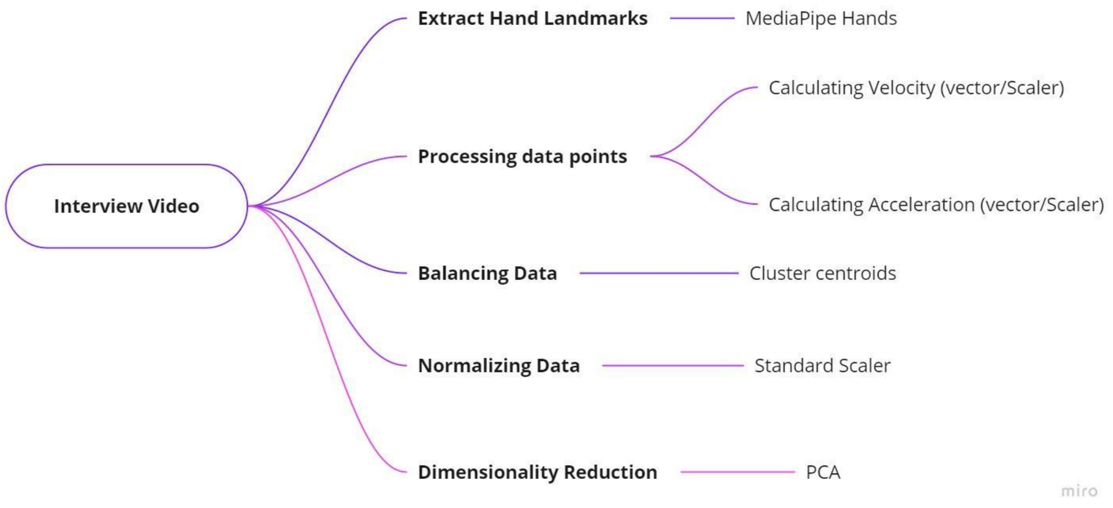


Figure 4-5

Get frames from the video then start getting hand landmarks in each frame using MediaPipe Hand. removing frames that cannot detect both hands of the interviewee.
Overall frames from videos of MIT after removing not detected were 277059 frames but as shown in plot data was unbalanced.

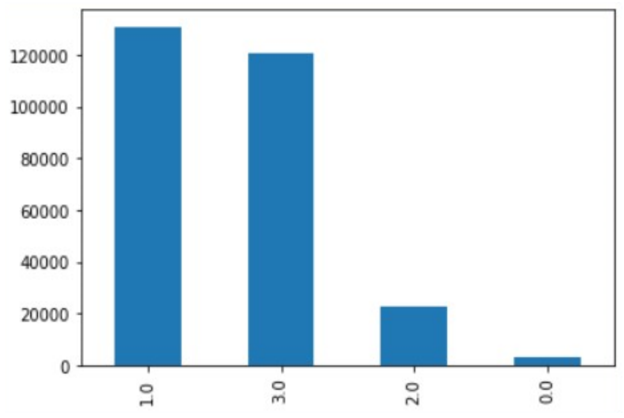


Figure 4-6

Figure 4 6: Crossed arms →0, Move→ 1, Rest→ 2, Clasped hands→ 3.

Using Cluster centroids to balance data, frames were reduced to 12420.

Normalizing with standard scaler using PCA which reduces features from 134 to 10.

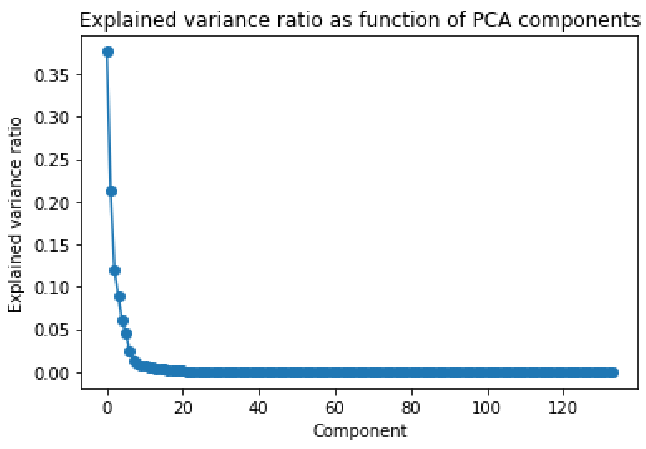


Figure 4-7

4.3.2 Training Model

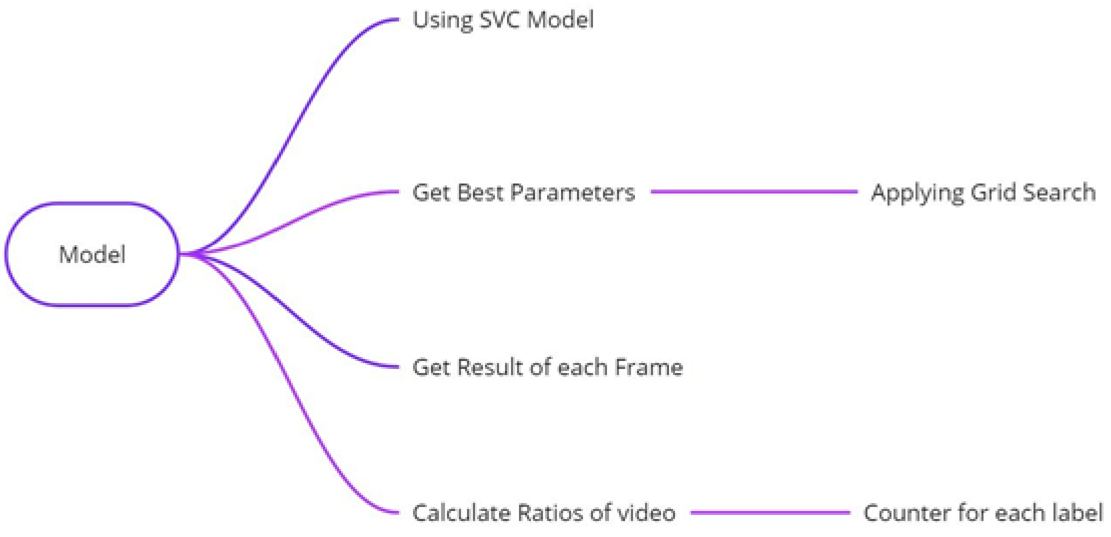


Figure 4-8

After processing data, Applying Grid Search on SVC to get the best parameters {'C': 10, 'gamma': 0.05, 'kernel': 'RBF'}, Then train with it.

The result of a video is an array containing labels for each frame. By counting the number of frames for each label and dividing it with the number of total labels we get the ratio of this label to the video and can also get the ratio of undetected hand frames by subtracting all ratios from one.

4.4 HAND OVER FACE MODULE

4.4.1 Data Preparation

Every object detection system requires annotation data for training, this annotation data consists of the information about the boundary box (ground truth) coordinates, height, width, and the class of object. YOLO requires annotation data in a specific format. YOLOv3 requires annotation information in the form of a text file.

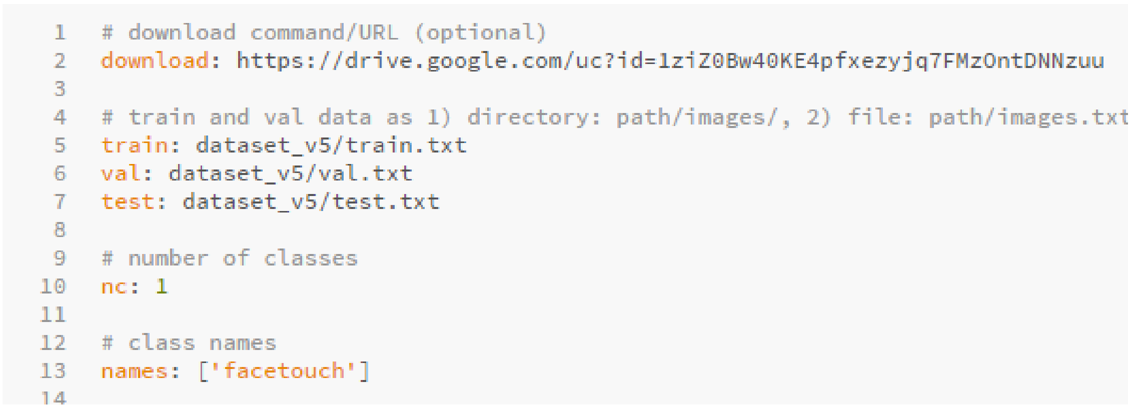
For training, we need to create a text file that has the same name as the image but in .txt extinction. Suppose the image name is sample.png then the text file name is sample.txt

The text file specifications are:

- One row per object
- Each row is class x_center y_center width height format.
- Box coordinates must be in normalized xywh format (from 0 - 1). If your boxes are in pixels, divide x_center and width by image width, and y_center and height by image height.
- Class numbers are zero-indexed (start from 0).

The annotations tools that can be used are the labelImg tool[45] or the Make Sense tool[46] they will generate the text files with the required specifications.

We need to create a dataset. YAML file that has the same format in this figure.



```yaml
# download command/URL (optional)
download: https://drive.google.com/uc?id=1ziZ0Bw40KE4pfxezyjq7FMzOntDNNzuu

# train and val data as 1) directory: path/images/, 2) file: path/images.txt
train: dataset_v5/train.txt
val: dataset_v5/val.txt
test: dataset_v5/test.txt

# number of classes
nc: 1

# class names
names: ['facetouch']

```

Figure 4-9

[45] labelImg tool
[46] Make Sense tool

4.4.2 Training Model

To train the model. First, clone the ultralytics/yolov3[47] repository then put the dataset.YAML inside the data folder. If your dataset is available to be downloaded and it has the same required format then put the URL of the dataset in the download field in the dataset.YAML file, if not then you need to put your dataset folder inside the downloaded repo folder.

Next, change the working directory to be inside the repo folder and run the following command: python train.py --img 416 --batch 64 --epochs 300 --data dataset.yaml --weights yolov3-spp.pt --nosave --cache
The --IMG argument is to define the image size after being normalized as the yolov3 performs better on images multiple of 32. The weights of the pre-trained model are yolov3-spp.pt. The --nosave is for saving the new weights we want after training and the --cache is just to start training from the last checkpoint we stopped in. We train the model for 300 epochs with 64 batch sizes.

4.5 COMBINATION OF MODULES

We combine sound and facial modules in four ways where the sound module is 120 features and the facial module is 60 features for each video, also each video has 18 labels as mentioned in the explanation of the MIT data.

1- Combining the two groups of features (120 + 60 = 180) then entering them into the SVR model.

2- Combining the result of each SVR model of each module then entering them into the Linear Regression model.

3- Combining the result of each SVR model of each module then applying simple averaging.

4- Combining the result of each SVR model of each module then applying weighted averaging.

Then we compare our results each way and take the best test score.

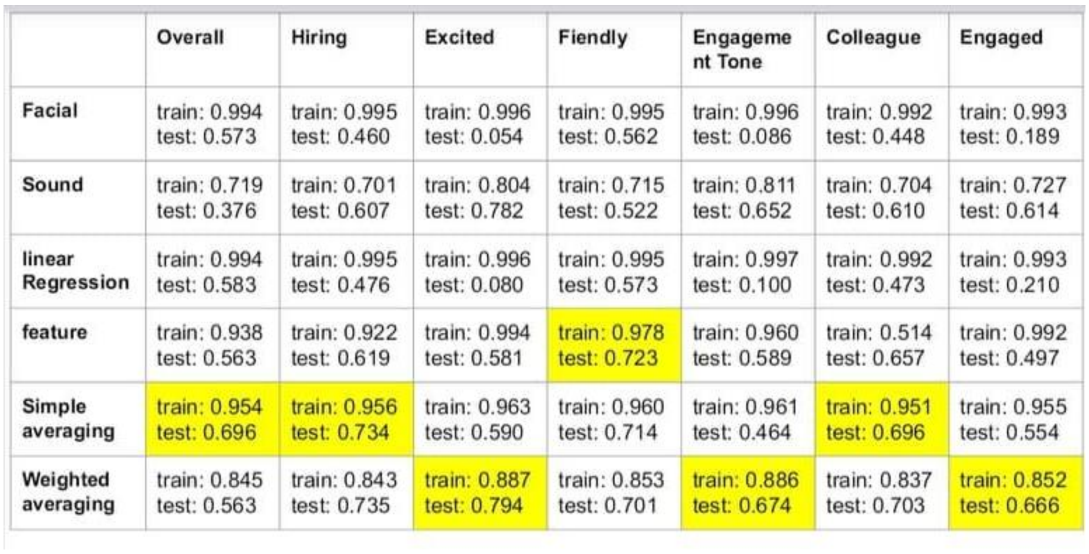
[47] yolov3 repository

Chapter Five

5 MODELS EVALUATION

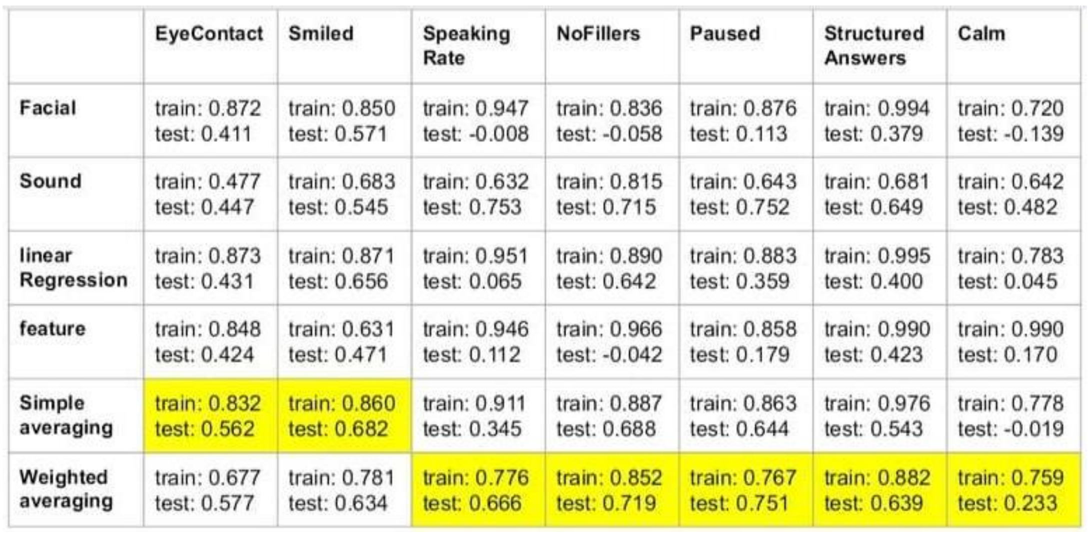
5.1 COMBINED EVALUATION

We use evaluation matrix correlation like what paper [3] used then we compare our result with the paper's result. Where there exist only five labels. and compare four ways for combining sound and facial modules and take the best. The following figure shows our result of all four ways and the yellow highlight is the best.



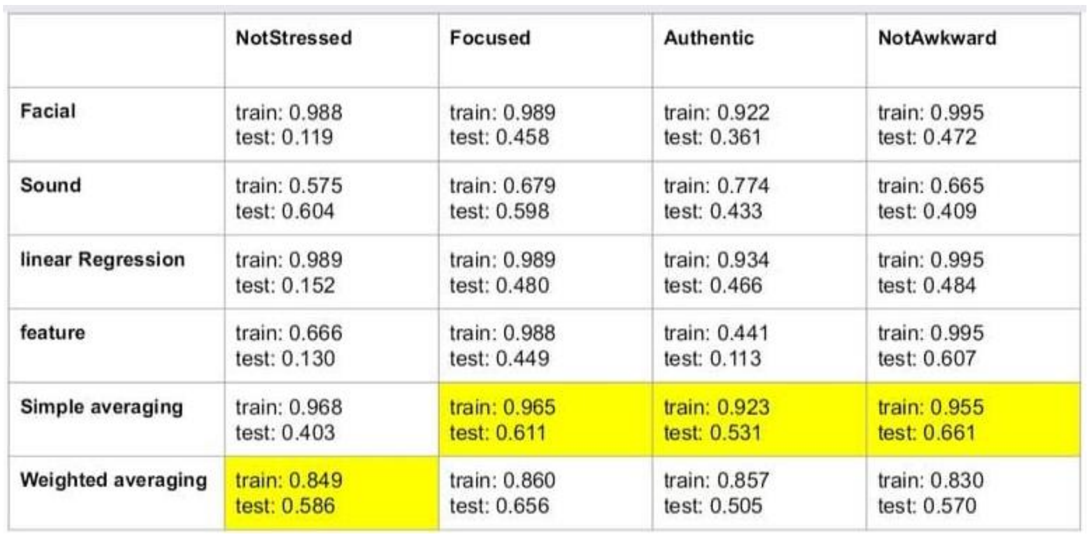
	Overall	Hiring	Excited	Fiendly	Engageme nt Tone	Colleague	Engaged
Facial	train: 0.994 test: 0.573	train: 0.995 test: 0.460	train: 0.996 test: 0.054	train: 0.995 test: 0.562	train: 0.996 test: 0.086	train: 0.992 test: 0.448	train: 0.993 test: 0.189
Sound	train: 0.719 test: 0.376	train: 0.701 test: 0.607	train: 0.804 test: 0.782	train: 0.715 test: 0.522	train: 0.811 test: 0.652	train: 0.704 test: 0.610	train: 0.727 test: 0.614
linear Regression	train: 0.994 test: 0.583	train: 0.995 test: 0.476	train: 0.996 test: 0.080	train: 0.995 test: 0.573	train: 0.997 test: 0.100	train: 0.992 test: 0.473	train: 0.993 test: 0.210
feature	train: 0.938 test: 0.563	train: 0.922 test: 0.619	train: 0.994 test: 0.581	train: 0.978 test: 0.723	train: 0.960 test: 0.589	train: 0.514 test: 0.657	train: 0.992 test: 0.497
Simple averaging	train: 0.954 test: 0.696	train: 0.956 test: 0.734	train: 0.963 test: 0.590	train: 0.960 test: 0.714	train: 0.961 test: 0.464	train: 0.951 test: 0.696	train: 0.955 test: 0.554
Weighted averaging	train: 0.845 test: 0.563	train: 0.843 test: 0.735	train: 0.887 test: 0.794	train: 0.853 test: 0.701	train: 0.886 test: 0.674	train: 0.837 test: 0.703	train: 0.852 test: 0.666

Figure 5-1



	EyeContact	Smiled	Speaking Rate	NoFillers	Paused	Structured Answers	Calm
Facial	train: 0.872 test: 0.411	train: 0.850 test: 0.571	train: 0.947 test: -0.008	train: 0.836 test: -0.058	train: 0.876 test: 0.113	train: 0.994 test: 0.379	train: 0.720 test: -0.139
Sound	train: 0.477 test: 0.447	train: 0.683 test: 0.545	train: 0.632 test: 0.753	train: 0.815 test: 0.715	train: 0.643 test: 0.752	train: 0.681 test: 0.649	train: 0.642 test: 0.482
linear Regression	train: 0.873 test: 0.431	train: 0.871 test: 0.656	train: 0.951 test: 0.065	train: 0.890 test: 0.642	train: 0.883 test: 0.359	train: 0.995 test: 0.400	train: 0.783 test: 0.045
feature	train: 0.848 test: 0.424	train: 0.631 test: 0.471	train: 0.946 test: 0.112	train: 0.966 test: -0.042	train: 0.858 test: 0.179	train: 0.990 test: 0.423	train: 0.990 test: 0.170
Simple averaging	train: 0.832 test: 0.562	train: 0.860 test: 0.682	train: 0.911 test: 0.345	train: 0.887 test: 0.688	train: 0.863 test: 0.644	train: 0.976 test: 0.543	train: 0.778 test: -0.019
Weighted averaging	train: 0.677 test: 0.577	train: 0.781 test: 0.634	train: 0.776 test: 0.666	train: 0.852 test: 0.719	train: 0.767 test: 0.751	train: 0.882 test: 0.639	train: 0.759 test: 0.233

Figure 5-2



	NotStressed	Focused	Authentic	NotAwkward
Facial	train: 0.988 test: 0.119	train: 0.989 test: 0.458	train: 0.922 test: 0.361	train: 0.995 test: 0.472
Sound	train: 0.575 test: 0.604	train: 0.679 test: 0.598	train: 0.774 test: 0.433	train: 0.665 test: 0.409
linear Regression	train: 0.989 test: 0.152	train: 0.989 test: 0.480	train: 0.934 test: 0.466	train: 0.995 test: 0.484
feature	train: 0.666 test: 0.130	train: 0.988 test: 0.449	train: 0.441 test: 0.113	train: 0.995 test: 0.607
Simple averaging	train: 0.968 test: 0.403	train: 0.965 test: 0.611	train: 0.923 test: 0.531	train: 0.955 test: 0.661
Weighted averaging	train: 0.849 test: 0.586	train: 0.860 test: 0.656	train: 0.857 test: 0.505	train: 0.830 test: 0.570

Figure 5-3

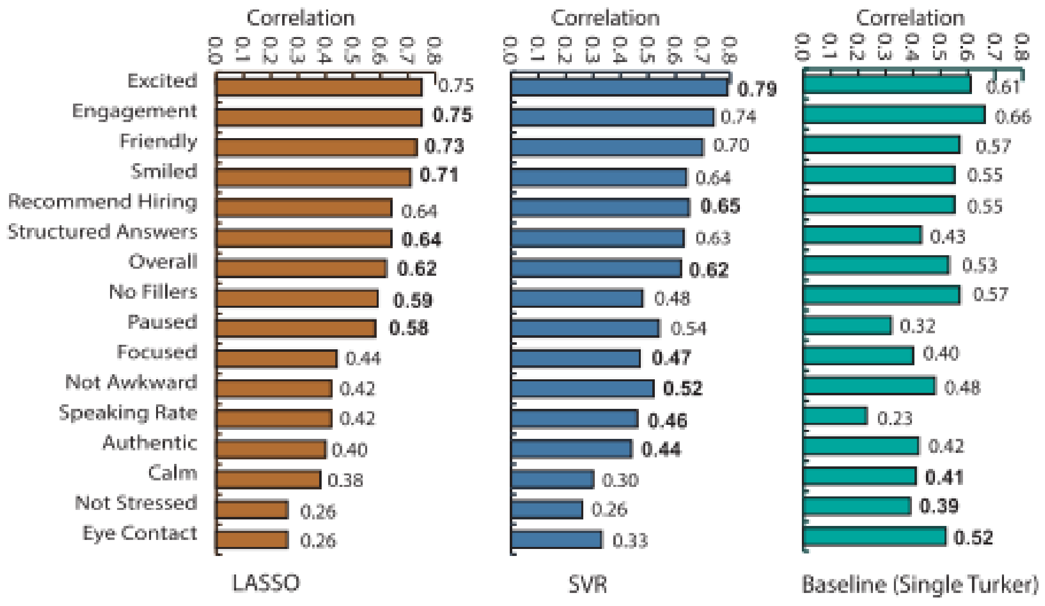


Figure 5-4

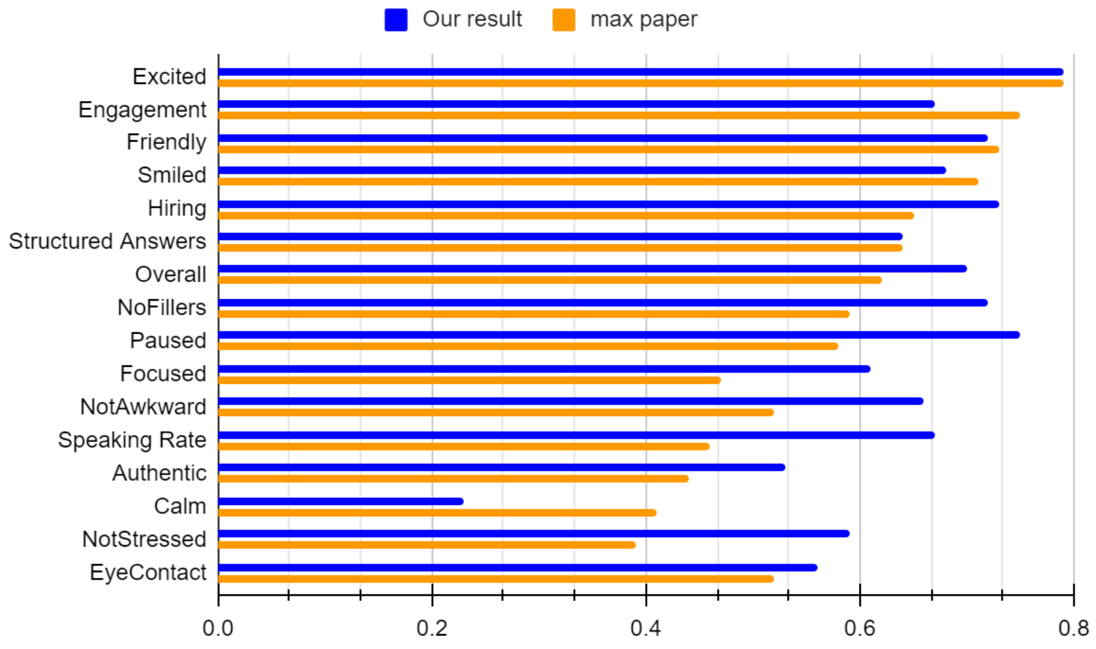


Figure 5-5

	Excited	Engagement	Friendly	Smiled	Hiring	Structured Answers	Overall	No Fillers
Our Result	0.79	0.67	0.72	0.68	**0.73**	**0.64**	**0.7**	**0.72**
max paper	**0.79**	**0.75**	**0.73**	**0.71**	0.65	0.64	0.62	0.59
	Paused	Focused	Not Awkward	Speaking Rate	Authentic	Calm	Not Stressed	Eye-Contact
Our Result	**0.75**	**0.61**	**0.66**	**0.67**	**0.53**	0.23	**0.59**	**0.56**
max paper	0.58	0.47	0.52	0.46	0.44	**0.41**	0.39	0.52

Table 2

5.2 FACIAL EXPRESSIONS EVALUATION

We use evaluation matrix correlation like what paper [3] used then we compare our result with the paper's result. Where the first finger is the paper's result and the second one is our result.

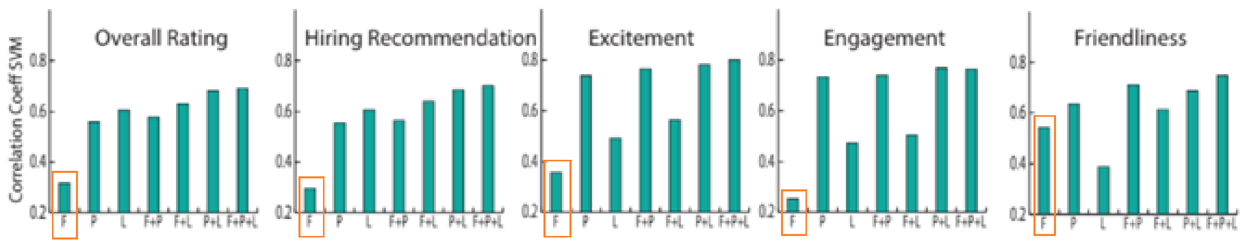


Figure 5-6

Overall	**Hiring**	**Excited**	**Engagement Tone**	**Friendly**
train: 0.994 test: 0.573	train: 0.995 test: 0.460	train: 0.996 test: 0.054	train: 0.996 test: 0.086	train: 0.995 test: 0.562

Table 3

5.3 HAND GESTURE EVALUATION

Using the same model explained before (SVC) and the same preprocessing for data with different numbers of points, trying it on two different datasets. Using accuracy metric, best was with MIT Dataset because it is higher accuracy than other dataset and because Gesture Phase Segmentation did not work well on other videos for interviews.

Dataset	*Input*	*Output*	*Train acc.*	*Test acc.*	*Notes*
Gesture Phase Segmentation	32 features for hand/shoulder points position, velocity, and acceleration reduced to 13	Rest Preparation Stroke Hold Retraction	90.35%	73.16%	The result on videos from the MIT dataset was bad (All hold class)
MIT Interview Videos	134 features for all hand points position, velocity, and acceleration reduced to 10	Rest Move Clasped Crossed	94.8%	69.69%	-

Table 4

5.4 HAND OVER FACE EVALUATION

Yolov3 metrics used are recall, precision, F1 score, and mAP.

- True positive is the number of detections with IoU > 0.5.

- False-positive is the number of detections with IoU <= 0.5 or detected more than once
- False negative is the number of objects that not detected or detected with IoU <= 0.5
- Precision = True positive / (True positive + False positive)
- Recall = True positive / (True positive + False negative)
- F1 score is HM (Harmonic Mean) of precision and recall which is $2 \times \frac{precision \times recall}{precision + recall}$
- The mAP for object detection is the average of the AP calculated for all the classes. mAP@0.5 means that it is the mAP calculated at IOU threshold 0.5. It's the area under the curve of PR.

Precision result

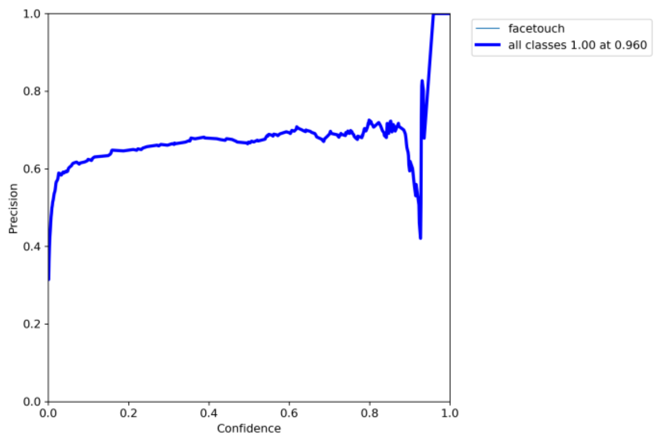


Figure 5-7

Recall result

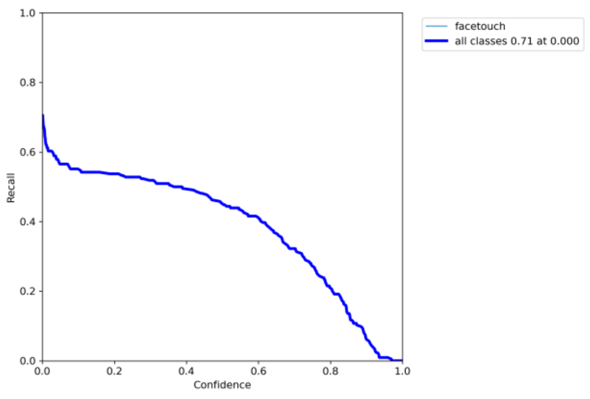


Figure 5-8

F1 score

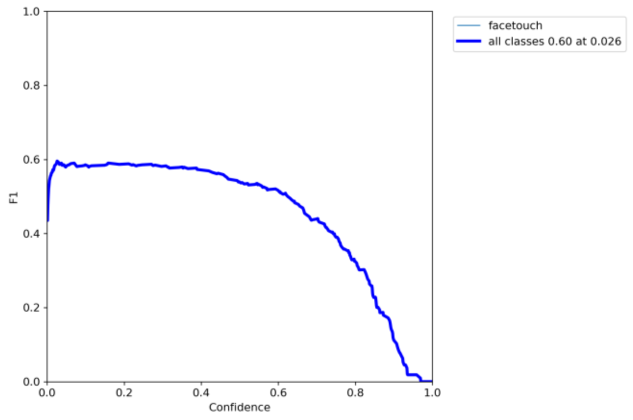


Figure 5-9

PR curve

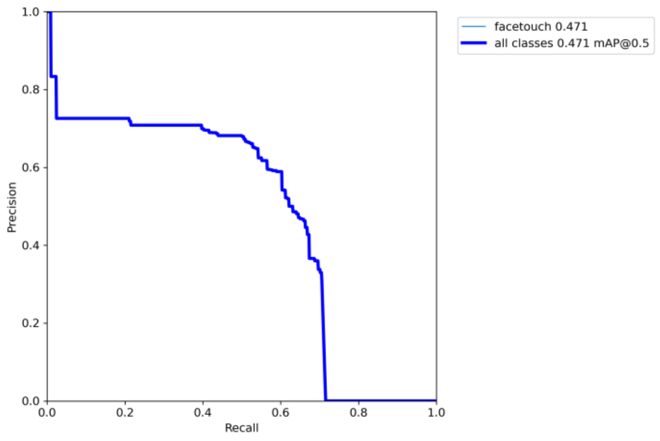


Figure 5-10

mAP@0.5 and mAP@0.5:0.95

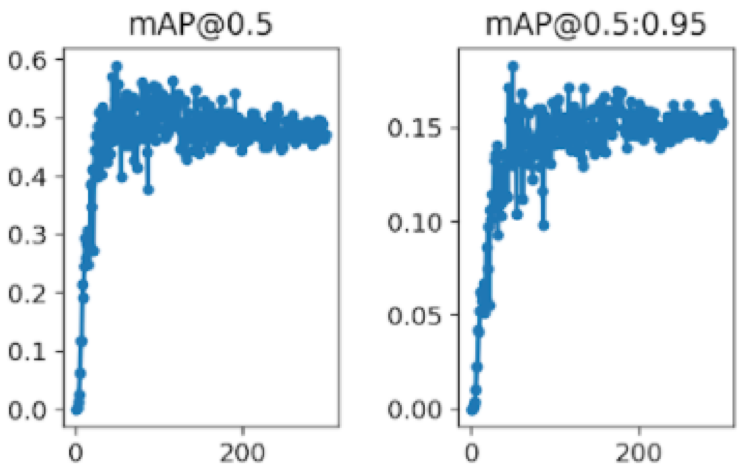


Figure 5-11

5.5 SPEECH EMOTION MODULE

5.5.1 Training with SVR Model

We use evaluation matrix correlation then we compare our result with the paper's results Fig [5.12]. Where there exist only five labels. So, the first finger is the paper’s result and the second one is our result. Our results are compared to the result from the paper in Fig [5.12] with the bar named ‘P’. (i.e., Prosody)

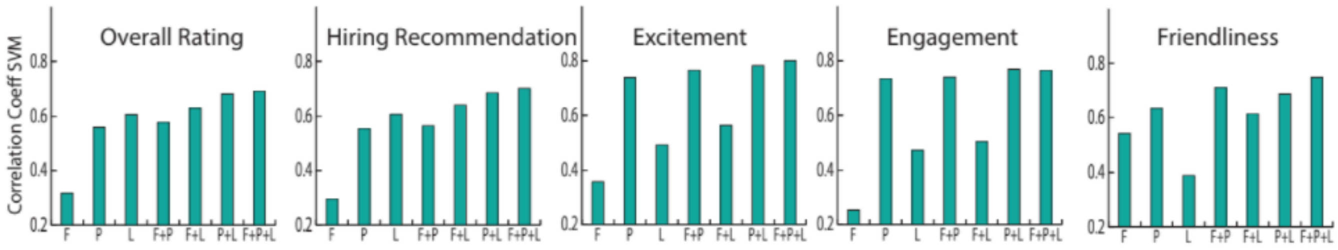


Figure 5-12 [3]

Overall	**Hiring**	**Excited**	**Friendly**	**Engagement Tone**
train: 0.719 test: 0.376	train: 0.701 test: 0.607	train: 0.804 test: 0.782	train: 0.715 test: 0.522	train: 0.763 test: 0.590

Table 5

5.5.2 Transfer Learning

The friendly train results from the SVR model were 0.522 while for that one form transfer learning was 0.4823. Similarly, the Overall trait from the SVR model was 0.376 while for that one form transfer learning was 0.2391.

Chapter Six

6 APPLICATION SOFTWARE

6.1 OVERVIEW

The website aims to help interviewees in preparing for a job interview. The system will analyze the facial expressions, audio, and body language of the interviewee while answering five different questions. In the end, it will give him rates for the 16 different traits.

6.2 HOW TO USE IT

The flow of the website is very clear. Before starting the questions, an overview of the aim of the website is shown. Then, it shows the instructions needed for the best benefit from the system. After that, the first question is shown, and the user is allowed to upload a video of his answer. After answering the five questions, the videos are analyzed and rates for the different traits are shown.

Chapter Seven

7 CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Our results with the combined model are better than the result in the paper on most labels.

The not-good ones are labels that depend on lexical features.

The results of labels that depend on facial expressions more have results better than speech emotion and vice versa.

Videos that contain dark-skinned people have problems detecting face landmarks. It is a common problem till now in most tools to get the landmarks.

The hand gesture model was not added to the combined model because of the leak of data. It only could get frames from 57 videos out of 138, and not all the videos as most of the time hands were under the table or the camera focused on just the face. Instead, we use ratios results for each video to give some instructions to the interviewer about his hand movement. So, it is a good model compared to the size of the data.

Bad brightness in the video will cause bad detection.

YOLO was the best and had good results in the hand-over-face part.

7.2 FUTURE WORK

Try to improve the application user interface, add input validation, and ensure the user that his data is not going to be used for blackmailing or cyberbullying.

We can speed up the system processing step by modifying some models, for example using the YOLO in detecting the facial features as it was good in the hand-over-face part.

For the hand gesture model, train with a bigger dataset and have a variety of classes as there is no good dataset available for this part.

Add another feature in the application by using GAN models to improve the user experience as the user not only can know his mistakes but also know how to fix them by seeing a video of himself without his mistakes.

We can improve the results by adding a lexical module that uses a speech recognition model to produce transcripts which helps in the analysis of the interview.

Try to make a new dataset like MIT videos that have the same concept, but the point of view will be better when it is in front of the interviewer and the upper half of the body appears clearly.

Chapter Eight

8 CITATION AND REFERENCING

[1] Mahmoud, Marwa & Baltrusaitis, Tadas & Robinson, Peter. (2014). Automatic Detection of Naturalistic Hand-over-Face Gesture Descriptors. 319-326. 10.1145/2663204.2663258.

[2] Madeo, Renata & Lima, Clodoaldo & Peres, Sarajane. (2013). Gesture unit segmentation using support vector machines: Segmenting gestures from rest positions. Proceedings of the ACM Symposium on Applied Computing. 46-52. 10.1145/2480362.2480373.

[3] Naim, Iftekhar & Tanveer, Md & Gildea, Daniel & Mohammed, & Hoque, Ehsan. (2015). Automated Analysis and Prediction of Job Interview Performance. IEEE Transactions on Affective Computing. PP. 10.1109/TAFFC.2016.2614299.

[4] Singh, Parwinder. (2014). An Approach to Extract Feature using MFCC. IOSR Journal of Engineering. 4. 21-25. 10.9790/3021-04812125.