Gesture detection and classification to detect confusion using MediaPipe, Locally Linear Embedding and KMeans clustering.

Meet Pragnesh Shah, Chen Song, Yang Xu  
San Diego State University: Email: {mshah3375, csong, yang.xu} @sdsu.edu

*Abstract*: Detect the current pose of a person from a video by processing it frame by frame and then classifying it if the state of a person is idle or confused. We use MediaPipe for detecting the coordinates of the skeleton of a person from an image.

Keywords— MediaPipe, Locally Linear Embedding (LLE), KMeans Clustering.

# Introduction

The main goal of this project is for making the Virtual Classroom experience of a professor as easy as possible. Normally when a professor is teaching in class, he/she is able to visually identify if a particular student is able to cope with what is being taught in the lecture. Students who are not understanding or have difficulties generally have the facial expression and gestures that highlight confusion or dilemma which is easily noticeable by the professor. Students may sometimes may not outspeak in the class but since the professor is able to see that the student is reluctant and hesitating then he/she can simply solve the doubt without the student explicitly asking him/her to do so. In the case of a virtual classroom a professor is not able to focus on all the faces of the students with their webcam setup while he/she is sharing the screen or explaining some topic. Here our project will help the professor flag the students who have a confused state of mind by the gesture she/she is portraying. The project will analyze the videos of students attending the lecture with their webcam on with their upper body being captured in the view, process the videos frame by frame and determine whether the current gesture is portraying a state of confusion.



Fig 1: Example of a Virtual Classroom where a Professor is teaching a batch of students.

Our project will help solve the issue here by processing each student’s webcam video and highlighting the students who belong to the “confused gesture” classification.

# Pose Detection

We use MediaPipe for detecting the pose of a person from an image or a video. MediaPipe is a framework for building pipelines to perform inference over arbitrary sensory data. With MediaPipe, a perception pipeline can be built as a graph of modular components, including model inference, media processing algorithms and data transformations, etc. Sensory data such as audio and video streams enter the graph, and perceived descriptions such as object-localization and face landmark streams exit the graph. [1]

MediaPipe Pose is a ML solution for high-fidelity body pose tracking, inferring 33 3D landmarks on the whole body (or 25 upper-body landmarks) from RGB video frames utilizing our BlazePose research that also powers the ML Kit Pose Detection API. Pose Landmark Model (BlazePose GHUM 3D). The landmark model in MediaPipe Pose comes in two versions: a full-body model that predicts the location of 33 pose landmarks (see figure below), and an upper-body version that only predicts the first 25. The latter may be more accurate than the former in our case as we use webcam-based input from the students attending the lectures where in most cases only their upper body is in frame.[2]

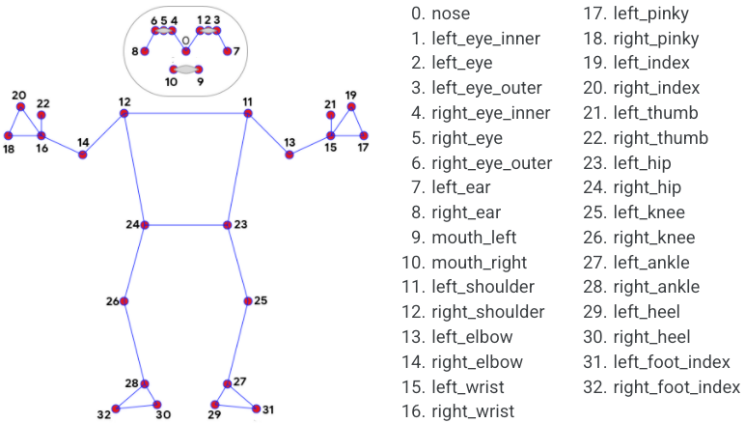


Fig 2: The 33 pose landmarks detected by MediaPipe from a video or image.[2]

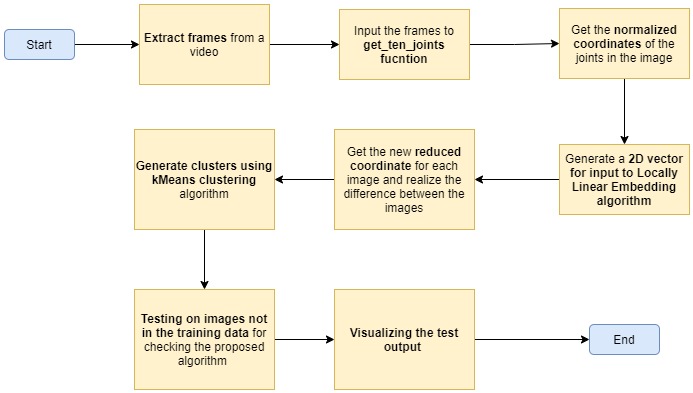


Fig 3: A diagram of the process flow. Text in bold represent the module.

### We create a function named as “get\_ten\_joints” that takes an image as an input and will return the coordinates of the required joints from an image. As we can see that for the upper body MediaPipe is able to detect 25 landmarks but from these we do not require all of the landmarks so we narrow down to 10 necessary joints.

### Head: First step will be to replace the points from 0 to 10 (the face coordinates) with just one simple point that will be the midpoint between two ears of a person. This will help represent the entire head by a single point.

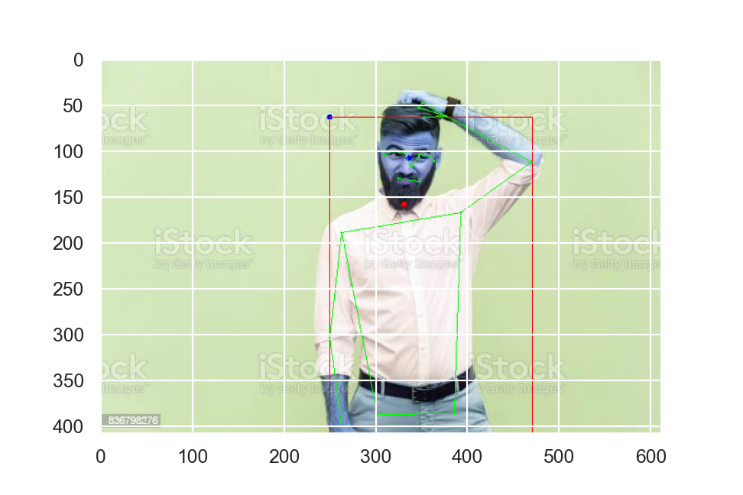
### Centroid: Later we proceed to calculate the geometric centroid formed by the nose and the two shoulders in the image and label that as the origin (0, 0). We calculate the origin point because the body may be located anywhere in the image (extreme left, right, center, etc.) so in order the coordinates obtianed to us do not rely on the loation of the subject in the image we proceed to create this point as the centre of the body of a person detected in the frame.

### Torso and hands: From the torso we only require the coordinates of the two shoulders, two hips, two elbows and two wrsits. The fingers are not so important to us as of now.

This way we obtain the coordinated of the following:

1. Head
2. Origin (centroid)
3. Left shoulder
4. Right shoulder
5. Left elbow
6. Right elbow
7. Left wrist
8. Right wrist
9. Left hip
10. Right hip

Now we apply the geometric change of origin formula and we recalculate all the coordinates of the points with the new Origin as the centre (0, 0).



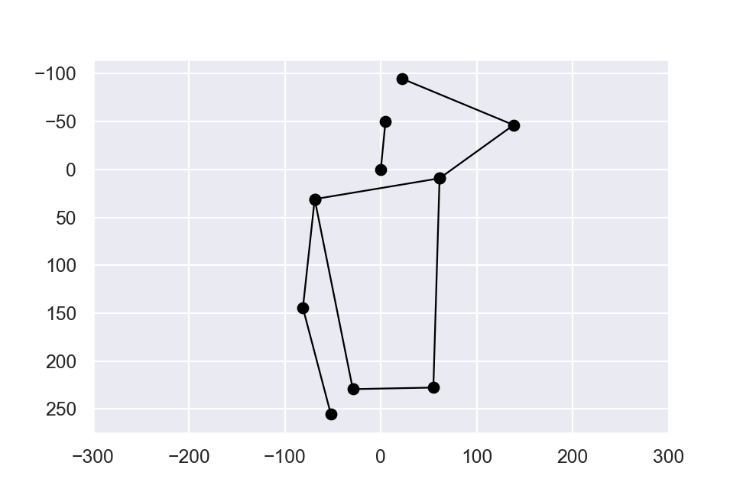


Fig 4: The graph obtained after applying the change of origin formula for each coordinate and later joining the ten coordinates/ points to represent a human-like skeleton from the image above it.

### Final step: The coordinates we obtained are still not normalized as the size of the image and the subject in the image may vary hecne the coorinates are varied. We set a fixed ratio for the width and height of the rectanlge bounded by the coordinates of the two shoulders and hips. We rescale all the coordinates based on the fixed vlaue of the bounding rectangle. We can clearly see the difference before and after doing the normalization.

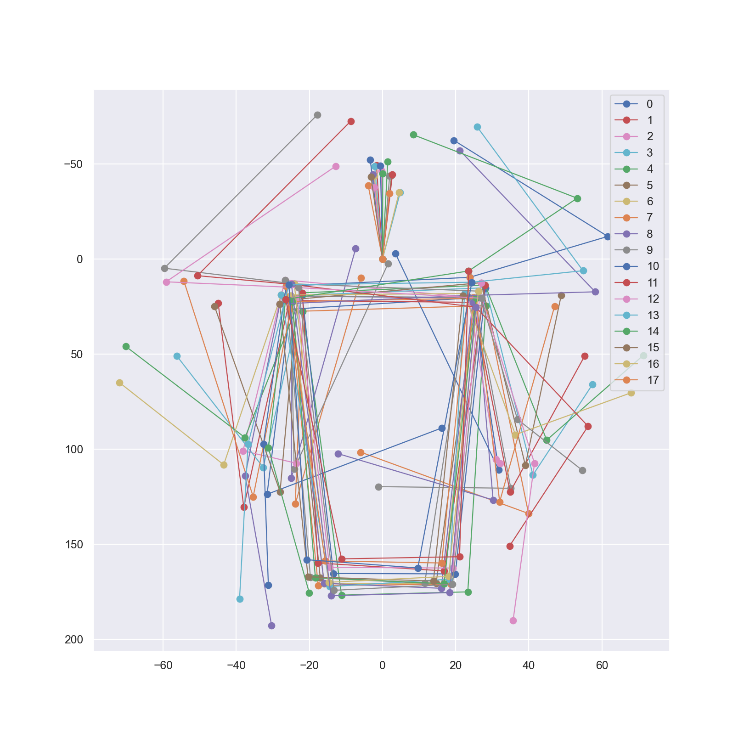
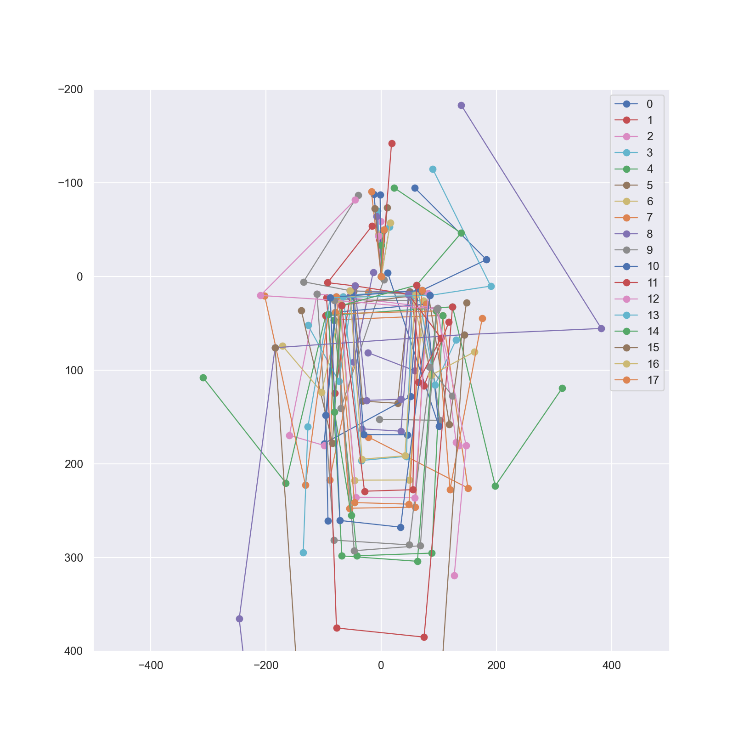


Fig 5: The graphs before and after normalizing the coordinates to fit the fixed bounding rectangle.

# Dimentionality Reduction

We make the use of Locally Linear Embedding. It is an unsupervised learning algorithm that computes low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs. Unlike clustering methods for local dimensionality reduction, LLE maps its inputs into a single global coordinate system of lower dimensionality, and its optimizations do not involve local minima. By exploiting the local symmetries of linear reconstructions, LLE is able to learn the global structure of nonlinear manifolds. [3]

In order to use LLE algorithm we create a multidimensional array where each element is an array that comprises the coordinates of the joints in the format of x-coordinate followed by the y-coordinated. By this we create an array comprising of twenty elements. This is useful because we do not lose any kind of information as the sequence of each of the points (mentioned in Pose Detection (3)) is going to be the same for each image.

After applying the LLE Algorithm, we obtain a 2-dimentonal coordinate for each image which we plot on a graph to see how the points representing similar looking gestures are grouped together. We annotate each point with the corresponding image.

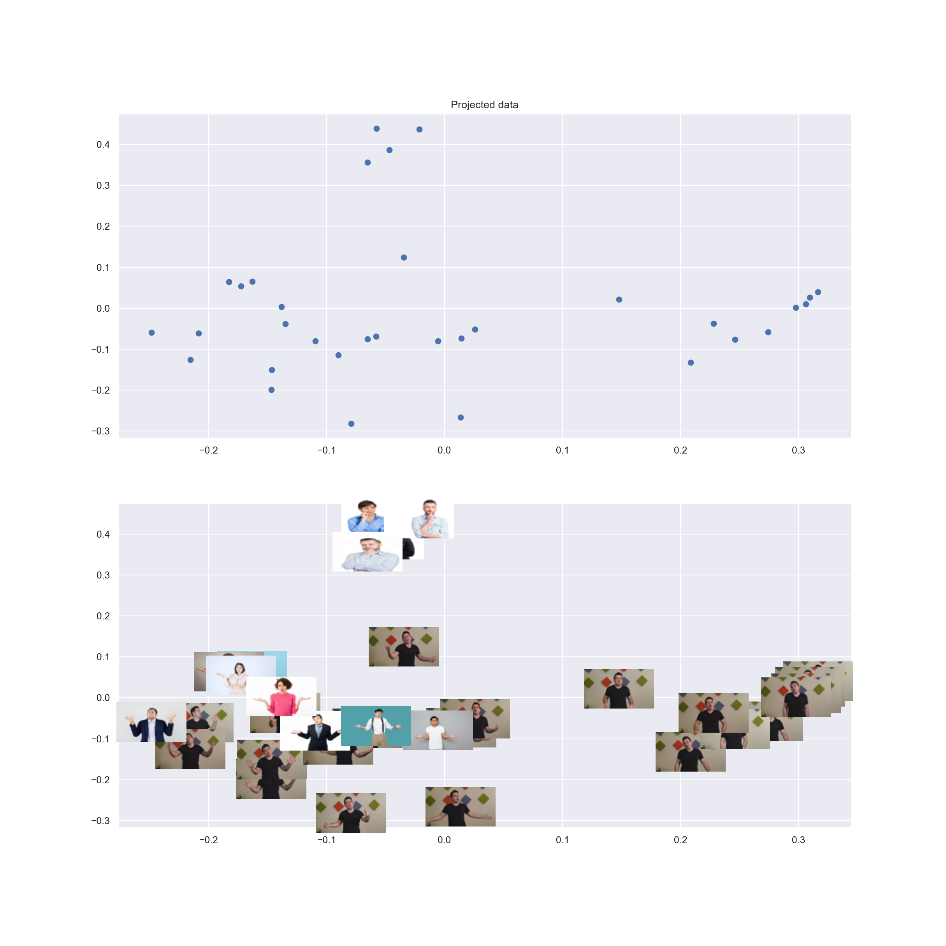
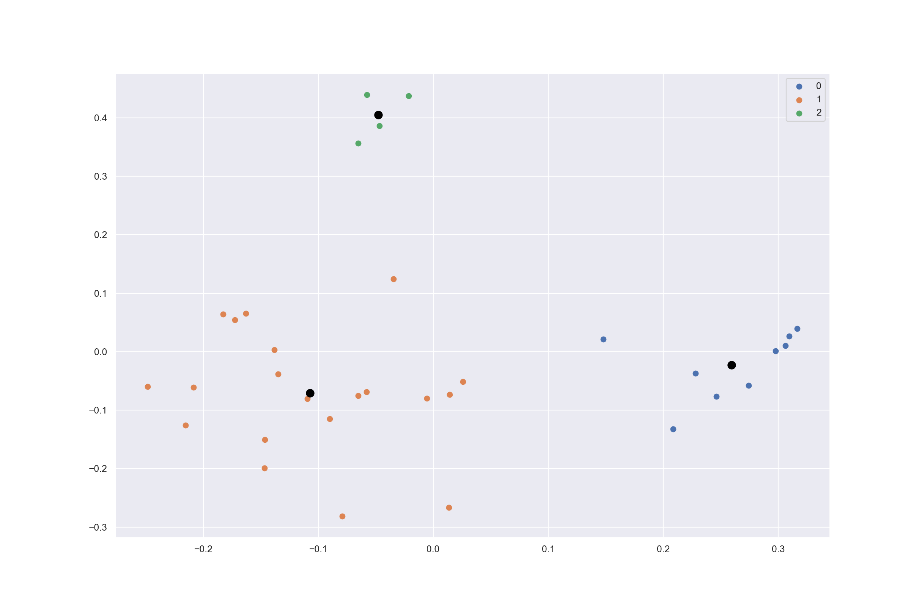


Fig 6: Locally Linear Embedding Algorithm reduced the twenty-feature coordinate vector to a two-feature coordinate conserving the image composition.

As we can see in the above graph that the images with similar gestures are grouped together by the LLE Algorithm. Here we can visually see three groups, there is one group for images where a person is scratching his/her chin, the other group is for the most common gesture for expressing confusion which is a shrugging gesture, the last one is the idle gesture where the subject is just sitting idle with is hands down. Once we are successfully able to create this, we call in the unsupervised KMeans clustering algorithm on this by which it will be able to determine the cluster centers.

# Clustering using KMeans clustering

K-Means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.[4]

  
Fig 7: Three clusters are identified with black dots as centre

From Fig 7 we can see that three clusters are identified for the three gestures we have with us. The centre of those clusters are marked with black colour.

# Testing

Once we are done with training the KMeans clustering algorithm, we can try testing to see if an image which is not in the dataset is allocated to the correct cluster. We have a testing set of 44 images where:

15 images are of chin scratching gesture,

10 images are of idle gesture,

19 images are of shrugging gesture.



Fig 8: We use these images for testing if they are successfully classified to belong to the relevant cluster. We have

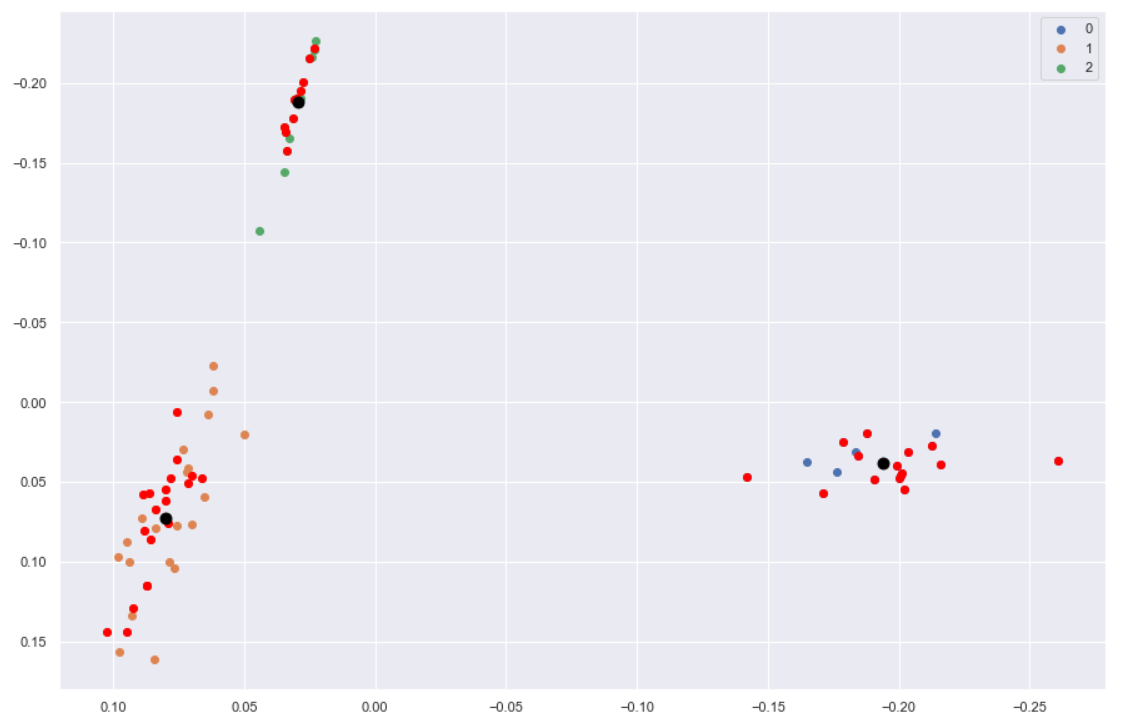


Fig 9: The red dots represent the test images. As we can see that they got assigned to the correct cluster.



Fig 10: the count of labels predicted for the test images, is same as the expected output.

# Conclusion

Upon multiple tests we determine that the location of the test image in the array for input to the LLE Algorithm does not matter. We got the same results for placing the image at the beginning, end and randomly in the middle in the array.

Finally, we create a function that is able to process a video frame by frame and extract every frame at a custom rate. By this process we can simply supply a video to the function and it will extract all the images from the video, later we supply the folder where the images are stored to get the gestures and check which cluster they belong to.

##### References

1. Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg and Matthias Grundmann “*MediaPipe: A Framework for Building Perception Pipelines*” Google Research arXiv:1906.08172v1 [cs.DC] 14 Jun 2019
2. https://google.github.io/mediapipe/solutions/pose.html
3. <https://cs.nyu.edu/~roweis/lle/papers/sciabs.txt>
4. https://en.wikipedia.org/wiki/K-means\_clustering#Description