**Posture Detection**

* Algorithm based
* YOLOv8-pose model
* Static camera based

**Theory:**

1. Keypoints: The YOLOv8-pose model is designed to detect and identify 17 specific keypoints on the human body. These keypoints correspond to distinct anatomical locations and are represented by their 2D coordinates. During the training phase, the model learns to accurately identify these keypoints by analyzing labelled data. This labelled data is stored in a CSV file, which contains the coordinates for each keypoint, enabling the model to learn the spatial relationships and positions of these points. The model is trained on the csv file obtained containing the coordinates of keypoints and the labels associated with each set of keypoints in every frame. Then model when provided with a video, reads the keypoints and then superimposes them on the video to give the output video with specified keypoints and the label of the posture (standing, sitting, squatting, leaning in our case).

Another use case of sitting idle can be achieved by adding a time limit when the model detects sitting posture. If this limit is crossed the model gives out an additional ‘idle’ label on the side of ‘sitting’ label.

**Files required:**

* (yolov8x-pose-p6.pt) or any other pt file you want to use
* Video file in mp4 format
* python file (actual code)

Go to <https://docs.ultralytics.com/tasks/pose/> for downloading pt files. The site contains pt files of multiple YOLOv8-pose models. We chose YOLOv8x-pose-p6. Other models can also be tried for different use cases.

For testing and training data, use any online datasets or videos of persons in different postures available. Locally shot video can also be used.

**Order of running:**

1. *Posturedetection\_somekeypoints.py* -> (extracts csv file of keypoints from video)
2. *Plot\_somekeypoints.py* -> (plot keypoints from csv file on a video)
3. *MLP.py* or *leaning.py* -> [train a ANN model on csv file, MLP.py has 3 classes(sitting,standing and squatting) but leaning.py has 4 classes(sitting, standing, squatting and leaning)]
4. *testingMLP.py* or *idle.py* or *leaning\_plus\_idle.py* ->[test trained model on csv file of test video and overlays the prediction on test video( idle.py and leaning\_plus\_idle.py also have a timer inbuilt for idle usecase))

Have to run these four files according to the use case we are working on.

**Folder structure:**

posture\_detections:

-csv (sample csv files for training)

-FRONTVIEW (csv file for each position)

-neural network

*-MLP.py* (script to train model on csv file) (3rd)

*-testingMLP.py*  (testing script) (4rth)

-SIDEVIEW (csv file for each position)

-TESTING (test video with its csv)

-use\_case\_idle:

*-idle.py*  (testing script with timer) +output video (4rth for idle usecase)

-usecase\_leaning

*-leaning.py* (code to train with one extra class “leaning”) (3rd for leaning usecase)

*-leaning\_plus\_idle.py* (testing script with timer) (4rth for leaning usecase)

-output videos

-trained .pth model

-video\_parts (cropped videos per posture)

*-posturedetection\_somekeypoints.py (1st)*

*-plot\_somekeypoints.py (2nd)*

**Requirements:** ultralytics, cv2, PyTorch

**How to run?**

Run the python script using: python filename.py

Make sure to change the input path and output paths and also specify the paths of classes and pt file in the main source code file.

**Algorithms used:**

1)Decision tree

2)RandomForest Classifier

3)Artificial Neural Network

Maximum efficiency was achieved by the use of Artificial Neural Network algorithm. Decision tree algorithm will work only on very specific camera angles due to the logic it uses in its code. RandomForest algorithm also produces good efficiency. Model trained on epochs between 85 and 100 gave the best result for available videos.