Model Development Phase Template

Date	21 July 2024
Team ID	739717
Project Title	Unlocking Silent Signals :Decoding Body Language with mediapipe
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

Paste the screenshot of the model training code

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics

Model 1	The code reads a CSV file named 'coords.csv' into a DataFrame using Pandas and displays the first few rows of the DataFrame. This helps to quickly inspect the structure and contents of the data for further processing.	Make Some Detections `cap = cv2.VideoCapture(0)` `# Initiate holistic model` `with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:` `while cap.isOpened():` `ret, frame = cap.read()` `# Recolor Feed` `image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)` `image.flags.writeable = False` `# Make Detections` `results = holistic.process(image)` `# print(results.face_landmarks)`
Model 2	Video capture wcapturing real-time video and applying holistic model detection to identify and track human poses, hands, and faces. It's useful for applications in interactive systems, gesture recognition, and augmented reality.	`# 1. Draw face Landmarks` `mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION,` `mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),` `mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)` `)` `# 2. Right hand` `mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,` `mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),` `mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)` `)` `# 3. Left Hand` `mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,` `mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),` `mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)` `)` `# 4. Pose Detections` `mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,` `mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),` `mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)` `)` `cv2.imshow('Raw Webcam Feed', image)` `if cv2.waitKey(10) & 0xFF == ord('q'):` `break` `cap.release()` `cv2.destroyAllWindows()`
Model 3	capture landmarks from an image, potentially using a library like opencv, and then export these landmarks to a CSV file.	Capture Landmarks and Export to CSV `import csv` `import os` `import numpy as np` `num_cords = len(results.pose_landmarks.landmark)+len(results.face_landmarks.landmark)` `num_cords` `501` `landmarks = ['class']` `for val in range(1,num_cords+1):` `landmarks += ['x{}'.format(val),'y{}'.format(val),'z{}'.format(val),'v{}'.format(val)]` `landmarks` `['class', 'x1', 'y1', 'z1', 'v1', 'x2', 'y2', 'z2', 'v2', 'x3', 'y3', 'z3', 'v3', 'x4', 'y4', 'z4',`

Model 4	OpenCv webcam when it get activated it collects data for happy mood and also fight,victorious,sad, collects the data	Collecting data for Happy mood 16]: `class_name = "Happy"` 17]: ``` cap = cv2.VideoCapture(0) # Initiate holistic model with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic: while cap.isOpened(): ret, frame = cap.read() # Recolor Feed image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) image.flags.writeable = False # Make Detections results = holistic.process(image) # print(results.face_landmarks) # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks # Recolor image back to BGR for rendering image.flags.writeable = True image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) ```