Documentation for body language decoder

June 2024

1 Documentation: Body Language Detection and Analysis

1.1 Overview

This script processes a video file to detect body language using the Mediapipe Holistic model and a pre-trained RandomForestClassifier. It visualizes the detections on the video feed and calculates the probabilities of different body language classes.

1.2 Components

Libraries and Model Loading:

- `cv2`: OpenCV for video processing.
- `mediapipe`: For holistic body detection.
- `numpy` and `pandas`: For data manipulation.
- `pickle`: For loading the pre-trained RandomForestClassifier model.

Class Names:

- `class_names_set1`: Contains general body language classes (e.g., Open, Neutral, Closed).
- `class_names_set2`: Contains specific expressions (e.g., Smile, Eye contact, Space occupation).

Video Processing:

- Opens the video file specified by `video_path`.
- Initializes Mediapipe Holistic model.

1.3 Main Loop

- Reads frames from the video.
- Processes each frame to detect body landmarks (pose, face, hands).
- Draws landmarks on the frame.
- Extracts landmark coordinates.
- Predicts body language class using the pre-trained model.
- Calculates and displays probabilities of detected classes.
- Visualizes the detected class and its probability on the video frame.
- Displays probabilities for both sets of classes.

1.4 Probability Calculation

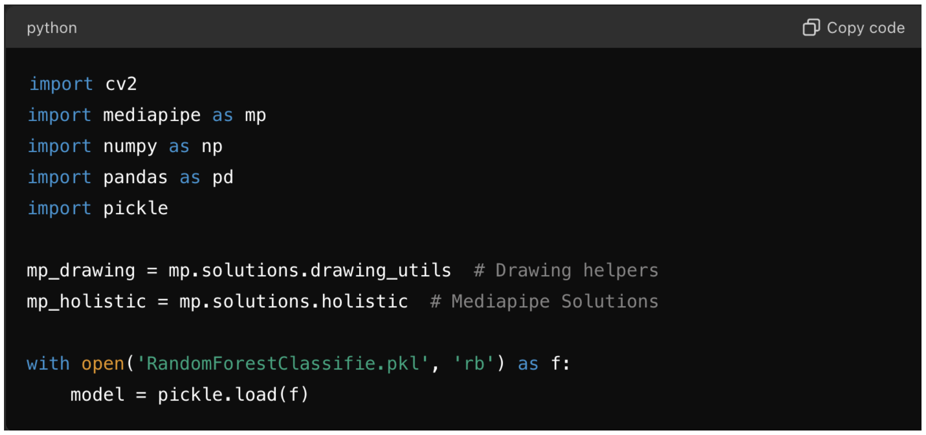
- Counts occurrences of each class in `detected_classes`.
- Calculates probabilities by dividing counts by the total number of detections.

1.5 Output

- Displays the video feed with visualizations.
- Prints probabilities of detected classes.

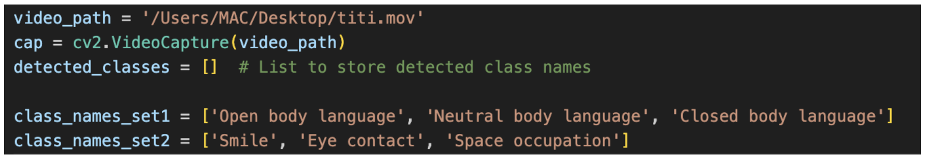
2 Code Walkthrough

2.1 Import Libraries and Load Model



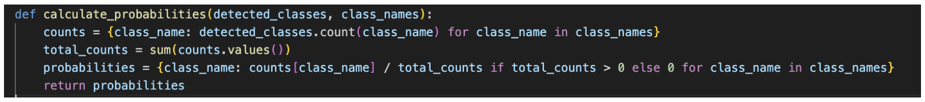
```python
import cv2
import mediapipe as mp
import numpy as np
import pandas as pd
import pickle

mp_drawing = mp.solutions.drawing_utils  # Drawing helpers
mp_holistic = mp.solutions.holistic  # Mediapipe Solutions

with open('RandomForestClassifie.pkl', 'rb') as f:
    model = pickle.load(f)
```

2.2 Initialize Variables



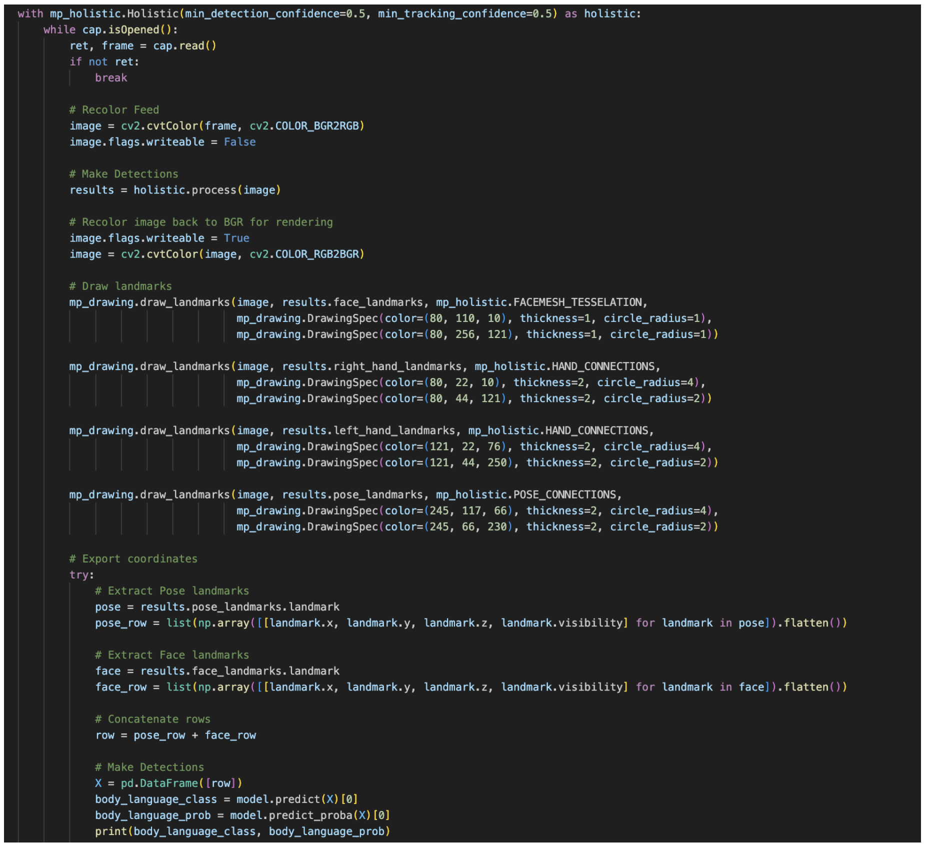
```python
video_path = '/Users/MAC/Desktop/titi.mov'
cap = cv2.VideoCapture(video_path)
detected_classes = []  # List to store detected class names

class_names_set1 = ['Open body language', 'Neutral body language', 'Closed body language']
class_names_set2 = ['Smile', 'Eye contact', 'Space occupation']
```

2.3 Define Probability Calculation Function

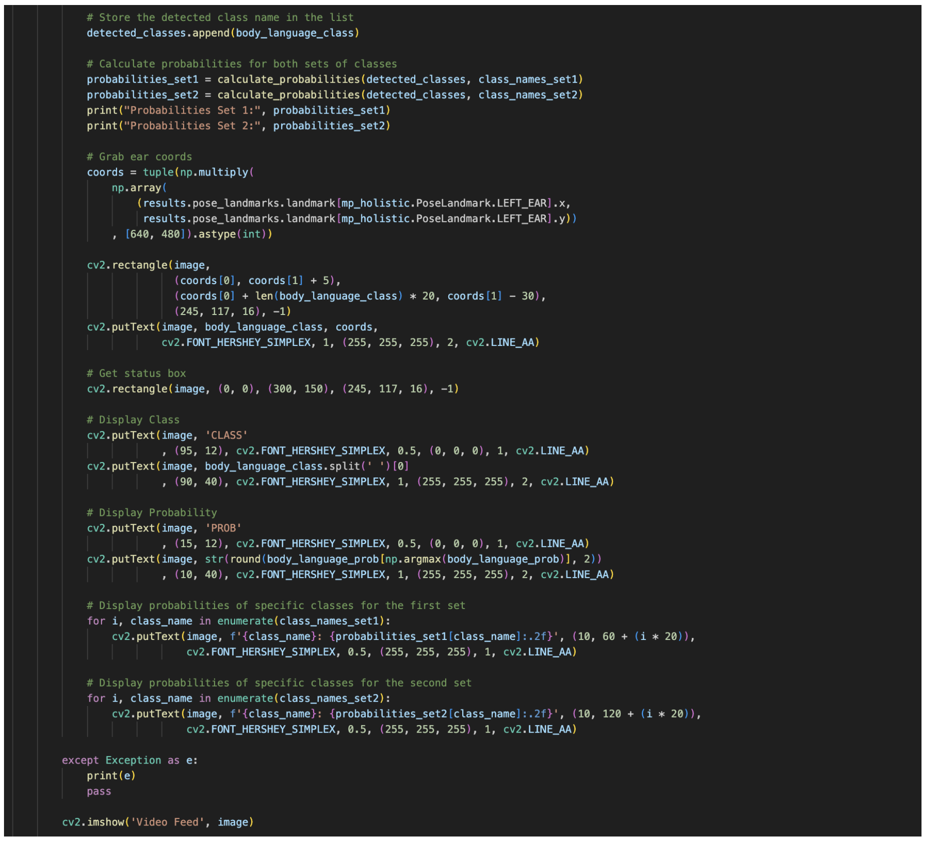


```python
def calculate_probabilities(detected_classes, class_names):
    counts = {class_name: detected_classes.count(class_name) for class_name in class_names}
    total_counts = sum(counts.values())
    probabilities = {class_name: counts[class_name] / total_counts if total_counts > 0 else 0 for class_name in class_names}
    return probabilities
```

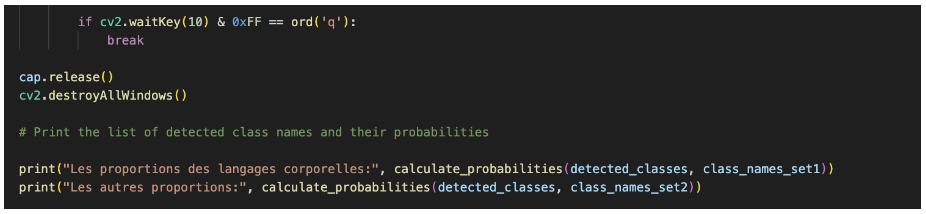
2.4 Process Video Frames



```python
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Recolor Feed
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make Detections
        results = holistic.process(image)

        # Recolor image back to BGR for rendering
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Draw 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))

        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))

        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))

        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))

        # Export coordinates
        try:
            # Extract Pose landmarks
            pose = results.pose_landmarks.landmark
            pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in pose]).flatten())

            # Extract Face landmarks
            face = results.face_landmarks.landmark
            face_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in face]).flatten())

            # Concatenate rows
            row = pose_row + face_row

            # Make Detections
            X = pd.DataFrame([row])
            body_language_class = model.predict(X)[0]
            body_language_prob = model.predict_proba(X)[0]
            print(body_language_class, body_language_prob)
```



```python
            # Store the detected class name in the list
            detected_classes.append(body_language_class)

            # Calculate probabilities for both sets of classes
            probabilities_set1 = calculate_probabilities(detected_classes, class_names_set1)
            probabilities_set2 = calculate_probabilities(detected_classes, class_names_set2)
            print("Probabilities Set 1:", probabilities_set1)
            print("Probabilities Set 2:", probabilities_set2)

            # Grab ear coords
            coords = tuple(np.multiply(
                np.array(
                    (results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].x,
                     results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].y))
                , [640, 480]).astype(int))

            cv2.rectangle(image,
                          (coords[0], coords[1] + 5),
                          (coords[0] + len(body_language_class) * 20, coords[1] - 30),
                          (245, 117, 16), -1)
            cv2.putText(image, body_language_class, coords,
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

            # Get status box
            cv2.rectangle(image, (0, 0), (300, 150), (245, 117, 16), -1)

            # Display Class
            cv2.putText(image, 'CLASS'
                        , (95, 12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
            cv2.putText(image, body_language_class.split(' ')[0]
                        , (90, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

            # Display Probability
            cv2.putText(image, 'PROB'
                        , (15, 12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
            cv2.putText(image, str(round(body_language_prob[np.argmax(body_language_prob)], 2))
                        , (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

            # Display probabilities of specific classes for the first set
            for i, class_name in enumerate(class_names_set1):
                cv2.putText(image, f'{class_name}: {probabilities_set1[class_name]:.2f}', (10, 60 + (i * 20)),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE_AA)

            # Display probabilities of specific classes for the second set
            for i, class_name in enumerate(class_names_set2):
                cv2.putText(image, f'{class_name}: {probabilities_set2[class_name]:.2f}', (10, 120 + (i * 20)),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE_AA)

        except Exception as e:
            print(e)
            pass

        cv2.imshow('Video Feed', image)
```



```python
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

cap.release()
cv2.destroyAllWindows()

# Print the list of detected class names and their probabilities

print("Les proportions des langages corporelles:", calculate_probabilities(detected_classes, class_names_set1))
print("Les autres proportions:", calculate_probabilities(detected_classes, class_names_set2))
```

3 Key Points

- **Video Feed:** Opens a video file and processes each frame to detect body landmarks.
- **Landmark Detection:** Uses Mediapipe to detect face, hand, and pose landmarks.
- **Classification:** Uses a pre-trained RandomForestClassifier to predict body language classes(RandomForestClassifie.pkl).

- **Probability Calculation:** Computes the probability of each detected class based on their occurrences.
- **Visualization:** Displays the detected class and its probability on the video feed.
- **Output:** Shows the probabilities of different body language classes and prints them at the end of the script.