Basic things to understand before running a code for landmarks extraction

Keypoints:

- **Holistic:** Use for applications requiring comprehensive body tracking, including face, hands, and body pose.
- **Drawing Utils**: Use for visualizing detected landmarks on images or video frames. Drawing Styles: Use for applying consistent visual styles to the drawn landmarks and connections.
- **Face Mesh**: Use for applications focused on facial features and expressions. Hands: Use for hand gesture recognition and sign language translation.
- **Pose**: Use for full-body motion tracking and analysis.
- **min_detection_confidence** is applied during the initial detection phase, where the goal is to identify objects or landmarks in the input data.
- **min_tracking_confidence** is applied during the subsequent tracking phase, where the goal is to maintain continuity and reliability in tracking objects or landmarks over time.
- **Concatenate:** The np.concatenate function in NumPy is used to join two or more arrays along an existing axis. This is particularly useful when you have multiple arrays or sequences of data that you want to merge into a single array
- **isOpened :** When working with video files or camera streams, you often need to check if the video source has been successfully opened before attempting to read frames from it. The isOpened() method serves this purpose by returning True if the video capture object has been initialized and is ready for use, and False otherwise.
- **np.**array : np.array is a function in the NumPy library that is used to create an array object. NumPy arrays are more efficient and provide more functionality compared to Python lists. They are widely used in scientific computing, data analysis, and machine learning for handling large datasets and performing various numerical operations.
- **np.savetxt :** The np.savetxt function in NumPy is used to save an array to a text file. This function is useful for exporting data in a human-readable format, which can be easily shared or used for later processing.
- **Delimiter:** The `delimiter` parameter in the `np.savetxt` function specifies the string or character that separates columns in the text file. By default, the delimiter is a space (`' '`), but you can customize it to any character or string, such as a comma, tab, or semicolon.

Indices Explaination for this video(“FAIL.mp4):

Left_hand_indices: 21*3 (x,y,z axis) = 63

Right_hand_indices: 21*3 (x,y,z axis) = 63

mouth_indices = [61, 146, 91, 181, 84, 17, 314, 405, 321, 375, 291] = 11*3=33

left_eye_indices = [263, 249, 390, 373, 374, 380, 381, 382, 362] = 9*3=27

right_eye_indices = [33, 7, 163, 144, 145, 153, 154, 155, 133] =9*3=27

nose_indices = [1, 2, 98, 327, 168] = 5*3 =15

optional:

pose_indices =33 *4(x,y,z,visibility(optional)=132 (we can take only axis also)

Total number of landmarks are : 360

Frames are 76

Video specific points("FAIL.mp4"):

- We got 76 frames in the video as the basic one second video contains a minimum of 24 fps(Frames per second) . but in our video we tested with the 3seconds video so, we got number of frames of 74.
- We will maintain the frames count in the video which is known as fps.
- Every frames landmarks were appended after each frame landmark extraction
- To the variable to whom we appended the landmarks eventually, converted into the specific file preferable is csv (also we can go with excel,notepad,etc).
- A specific delimiter needs to be maintained for the separation of the landmarks which is utilized while retrieving those landmarks.
- The extracted landmarks are arranged in such a way that
    1. Rows : frames landmarks
    2. Columns : number of frames

Procedure :

- Import Libraries

The script begins by importing necessary libraries:

- OpenCV (`cv2`) : For capturing and processing video frames.

- NumPy (`np`) : For numerical operations, such as creating and manipulating arrays.

- MediaPipe (`mp`): For using the holistic model which detects facial landmarks, hand landmarks, and pose landmarks.

- Initialize MediaPipe Holistic Model

The holistic model from MediaPipe is initialized. This model integrates face, hand, and pose detection into a single pipeline. The drawing utilities from MediaPipe are also initialized to assist with visualizing the detected landmarks.

- Define a Function for MediaPipe Detection

A function named `mediapipe_detection` is defined to process each video frame:

- The function converts the frame from BGR color space (used by OpenCV) to RGB (required by MediaPipe).

- It sets the frame to non-writable mode to speed up processing.

- The frame is then passed through the holistic model to detect landmarks.

- The frame is set back to writable mode.

- Finally, the function converts the frame back to BGR.

- Define a Function to Extract Keypoints

A function named `extract_keypoints` is defined to extract the landmarks from the detection results:

- Pose Landmarks: Extracts pose landmarks if detected, otherwise returns an array of zeros.

- Hand Landmarks: Extracts left and right hand landmarks if detected, otherwise returns arrays of zeros.

- Face Landmarks: Extracts specific facial landmarks (mouth, eyes, nose) if detected, otherwise returns an array of zeros. Specific indices for these landmarks are predefined.

- All extracted landmarks are concatenated into a single array representing all keypoints for a frame.

- Capture Video

A video file is opened using OpenCV's `VideoCapture` function, allowing frames to be read sequentially.

- Process Video Frames

Using the holistic model:

- A loop reads each frame from the video.

- Each frame is processed using the `mediapipe_detection` function to detect landmarks.

- Key landmarks are extracted using the `extract_keypoints` function.

- The extracted keypoints for each frame are stored in a list.

- Release Video Capture

After all frames have been processed, the video capture is released, freeing up resources.

- Save Landmarks to CSV

The list of all extracted landmarks is converted into a NumPy array. This array is then saved to a CSV file using `np.savetxt`, with each row in the CSV representing the keypoints for a single frame.