Aim

To create a real-time push-up counter using computer vision techniques. The system will utilize a front view and side view of the user to accurately count push-ups and provide feedback on form.

Objective

1. **Front View:**
    - Count the number of push-ups based on the angle between the shoulder, elbow, and wrist.
    - Display the count and the current stage (UP or DOWN) on the screen.
2. **Side View:**
    - Assess the correctness of push-up form by evaluating angles between shoulder, elbow, and hip.
    - Provide feedback and visualize progress using a bar indicating the percentage of the push-up range.

Summary

The project involves two main scripts, one for counting push-ups from a front view and another for evaluating form from a side view. Both scripts use Mediapipe and OpenCV libraries to process video feeds from a webcam. The front view script calculates angles between key joints to count push-ups, while the side view script assesses form and provides feedback.

Tools and Libraries Used

- **Libraries:**
    - mediapipe: For pose detection and landmark extraction.
    - opencv-python: For image processing and real-time video display.
    - numpy: For mathematical operations, especially angle calculations and interpolations.
- **Environment:**
    - Jupyter Lab on a local machine.

Procedure

Code 1 (Front View)

1. **Installation and Imports:**

CODE:

```
!pip install mediapipe opencv-python

import cv2
import mediapipe as mp

import numpy as np
```

2. **Define Functions:**
    - calculate_angle(a, b, c): Computes the angle between three points.
3. **Setup Video Capture:**

CODE:

```
cap = cv2.VideoCapture(0)
```

4. **Initialize Mediapipe Pose Instance:**

CODE:

```
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
```

5. **Processing Loop:**
    - Convert frames to RGB and process for pose landmarks.
    - Calculate angles for push-up counting.
    - Update counter and stage based on angle thresholds.
    - Display count and stage on the video feed.
6. **Display Results:**
    - Use OpenCV to draw landmarks and overlay text for rep count and stage.
7. **Cleanup:**

CODE:

```
cap.release()

cv2.destroyAllWindows()
```

Code 2 (Side View)

1. **Imports:**

CODE:

```
import cv2

import mediapipe as mp

import numpy as np

import PoseModule as pm
```

2. **Initialize Video Capture and Pose Detector:**

CODE:

```
cap = cv2.VideoCapture(0)

detector = pm.poseDetector()
```

3. **Processing Loop:**
    - Capture frame and find pose landmarks.
    - Calculate angles for elbow, shoulder, and hip.
    - Determine the percentage of push-up completion and update progress bar.
    - Provide feedback on form and count push-ups based on form correctness.
    - Display count, progress bar, and feedback on the video feed.
4. **Display Results:**
    - Use OpenCV to draw progress bar, count, and feedback.
5. **Cleanup:**

CODE:

```
cap.release()

cv2.destroyAllWindows()
```

Highlights

- **Front View Code:**
    - Utilizes angle calculations to determine push-up completion.
    - Provides real-time feedback and counting with visual indicators.
- **Side View Code:**
    - Includes a progress bar to visualize push-up progress.
    - Evaluates and provides feedback on push-up form.
- **Integration of Mediapipe:**
    - Efficiently detects and tracks body landmarks for accurate form analysis and counting.

Conclusion

The push-up counter project effectively demonstrates the use of computer vision for exercise tracking. The front view code provides a straightforward count of push-ups, while the side view code adds depth by evaluating form and offering feedback. This approach allows for real-time monitoring and improvement of exercise performance using accessible tools and libraries.