Tech Neck Project: Detect and Correct "Tech-Neck" Using IMU Data or Webcam Video

Your Name

June 5, 2025

Abstract

Tech-neck (excessive forward tilting of the head while looking at screens) has become a widespread ergonomic issue. Prolonged neck flexion can lead to muscle strain, headaches, and long-term spinal misalignment. This project explores two complementary approaches—(1) a video-based Pose Estimation pipeline and (2) a smartphone IMU-based app—to detect and potentially alert users when they adopt harmful neck posture. By providing real-time feedback, users can be encouraged to maintain healthier posture and prevent strain before it becomes a problem.

1 Overall Goal

Automatically detect when a person's head/neck is tilted downward (i.e. "tech-neck" posture) using only a smartphone's built-in sensors or a standard webcam, so that end users can be alerted before strain occurs. Tech-neck, or excessive forward tilting of the head while looking at screens (smartphones, laptops, etc.), has become a widespread ergonomic issue. Prolonged neck flexion can lead to muscle strain, headaches, and long-term spinal misalignment. This project explores two complementary approaches to detect and potentially alert the user when they adopt a harmful neck posture, with the aim of encouraging healthier posture habits and helping to prevent strain before it becomes a problem.

2 Two Approaches

2.1 Pose Estimation – Tech-Neck Detection Using Videos

The goal of this approach is to automatically detect "tech-neck" posture in prerecorded (or live) video. Potential applications include:

- Posture monitoring and real-time feedback when people are video recorded.
- Dataset creation: generating labeled data for training downstream machine learning models that might combine both vision and inertial cues.

2.1.1 Methodology

We use MediaPipe Pose (by Google), which offers a real-time TensorFlow Lite model that returns 33 anatomical landmarks (nose, shoulders, hips, knees, etc.) in each frame. It is robust to varying lighting and moderate occlusions, making it suitable for webcam-only posture analysis.

Processing Pipeline: For a video in which someone is sitting, each frame undergoes:

1. Conversion from BGR to RGB.
2. Inference through `mp_pose.Pose()`, producing a `results` object containing `pose_landmarks`.

We have implemented two scripts to detect tech-neck:

1. `pose_estimate_neck.py` – Neck & Torso Angle & Duration Warning
This script calculates both neck and torso angles, tracks how long "bad posture" persists, and triggers an alert after 180 seconds. From the 33 pose landmarks, we extract the 2D normalized coordinates (x, y) of:

- Left shoulder (keypoint 11)
- Right shoulder (keypoint 12)
- Left ear (keypoint 7)
- Left hip (keypoint 23)

Using these four points, we compute three key metrics:

1. **Shoulder alignment (offset):** the Euclidean distance between the left and right shoulders to check if they remain roughly level:
$$\text{offset} \;=\; \sqrt{\left(x_{L_{\text{shoulder}}} - x_{R_{\text{shoulder}}}\right)^2 + \left(y_{L_{\text{shoulder}}} - y_{R_{\text{shoulder}}}\right)^2}\,.$$
2. **Neck inclination:** form a 2D vector from the left shoulder to the left ear (in pixel space) and compute its angle relative to the horizontal axis:
$$\theta_{\text{neck}} \;=\; \frac{\pi}{180}\,2\Big(y_{L_{\text{ear}}} - y_{L_{\text{shoulder}}},\; x_{L_{\text{ear}}} - x_{L_{\text{shoulder}}}\Big)\,,$$
where the result is given in degrees.
3. **Torso inclination:** form a 2D vector from the left hip to the left shoulder and compute its angle:
$$\theta_{\text{torso}} \;=\; \frac{\pi}{180}\,2\Big(y_{L_{\text{shoulder}}} - y_{L_{\text{hip}}},\; x_{L_{\text{shoulder}}} - x_{L_{\text{hip}}}\Big)\,.$$

Each frame is classified as "good posture" if
$$\theta_{\text{neck}} < 40^\circ \quad \text{and} \quad \theta_{\text{torso}} < 10^\circ,$$
otherwise it is "bad posture." We count consecutive good/bad frames:
$$N_{\text{good}} \leftarrow \begin{cases} N_{\text{good}} + 1, & \text{if good posture,} \\ 0, & \text{otherwise,} \end{cases} \qquad N_{\text{bad}} \leftarrow \begin{cases} N_{\text{bad}} + 1, & \text{if bad posture,} \\ 0, & \text{otherwise.} \end{cases}$$
Since the video runs at fps frames per second, each frame is $\frac{1}{\text{fps}}$ seconds. We compute
$$t_{\text{good}} = \frac{N_{\text{good}}}{\text{fps}}, \quad t_{\text{bad}} = \frac{N_{\text{bad}}}{\text{fps}}.$$
If $t_{\text{bad}} > 180$ seconds, a warning is triggered:
$$\texttt{sendWarning()} \quad \big(\text{e.g. print "Bad posture ¿3 min"}\big).$$

Each annotated frame is written to `output.avi` with:

- Yellow circles marking left shoulder, left ear, and left hip.
- Colored lines:
  - Shoulder → ear (neck)
  - Hip → shoulder (torso)

  in green for good posture and red for bad posture.
- Overlaid text showing θ_{neck} (degrees).

2. `angle_neck.py` – Neck Angle Only
This script calculates only the neck angle in each frame and displays it, producing `processed_NeckPosture.avi`. From the 33 landmarks, we extract:

- Left shoulder (keypoint 11)
- Right shoulder (keypoint 12)
- Nose (keypoint 0)

We approximate the "neck base" (midpoint at top of torso) by averaging left-shoulder and right-shoulder coordinates:

$$(x_{\text{neck\_base}},\, y_{\text{neck\_base}}) = \Big(\tfrac{x_{L_{\text{shoulder}}}+x_{R_{\text{shoulder}}}}{2},\ \tfrac{y_{L_{\text{shoulder}}}+y_{R_{\text{shoulder}}}}{2}\Big).$$

Define the 2D neck vector in normalized image coordinates:

$$\text{neck\_vector} \;=\; \big(x_{\text{nose}},\, y_{\text{nose}}\big) - \big(\tfrac{x_{L_{\text{shoulder}}}+x_{R_{\text{shoulder}}}}{2},\ \tfrac{y_{L_{\text{shoulder}}}+y_{R_{\text{shoulder}}}}{2}\big).$$

Compute its signed angle relative to the horizontal $\big[1, 0\big]$ via:

$$\theta_{\text{neck}} \;=\; \big|\,\text{degrees}\big(2(\,\Delta y,\ \Delta x)\big)\big|, \quad \text{where } (\Delta x, \Delta y) = \text{neck\_vector}.$$

Each frame is annotated with

$$\texttt{Neck Angle:}\quad \theta_{\text{neck}}\,\texttt{deg}$$

and the full pose skeleton is drawn. The result is saved to `processed_NeckPosture.avi`.

2.1.2 Results

This notebook reliably computes and displays neck angles for videos, and shows proof-of-concept "tech-neck" classification using manual thresholds. Note that this pipeline has only been run on two videos; more extensive testing across different sitting positions is needed to verify robustness.

2.2 Smartphone App – IMU-Based Tech-Neck Detection

The core idea of this approach is to use a smartphone's built-in inertial sensors (accelerometer + gyroscope) as a "neck-angle tracker." By streaming raw IMU data over UDP, a remote server (or on-device logic) can estimate neck angle and issue an alert if the user maintains a harmful forward tilt for too long.

We aim to interpret raw accelerometer (a_x, a_y, a_z) and gyroscope (g_x, g_y, g_z) readings as a reliable proxy for neck tilt while the phone is held naturally in a user's hand. To date, we have:

- Sampled both accelerometer and gyroscope at 10 Hz (100 ms intervals) using `react-native-sensors`.
- Streamed each new reading (x, y, z) via UDP to a listening server for real-time processing.
- Stored the latest (a_x, a_y, a_z) and (g_x, g_y, g_z) in React state for immediate UI debugging.
- Packaged each sensor tuple as a tiny ASCII string—"ACCEL,x, y, z" or "GYRO,x, y, z"—and sent to (`serverIP` : `6000`).
- Implemented a simple receiver (`receive_imu.py`) that listens on `0.0.0.0:<port>` and prints each incoming datagram.

2.2.1 How to Use the IMU App

1. **Set IP & Port**
   - In `App.tsx`, update `serverIP` and `serverPort` to match your desktop's IP and the UDP port your listener uses (e.g. 5005).
   - In `receive_imu.py`, set `UDP_PORT` = the same value.
2. **Run the receiver:**

   ```
   $ python receive_imu.py
   ```

3. **Start React Native Metro & install on device/emulator:**

   ```
   $ npx react-native start --reset-cache
   $ npm run android
   ```

4. **Hold the phone normally:** You will see live (a_x, a_y, a_z) and (g_x, g_y, g_z) displayed in the UI, and on the desktop console you will see lines such as:

   `ACCEL,0.12,-0.03,9.81` and `GYRO,-0.02,0.01,0.00`.

2.2.2 Results

So far, this pipeline has been tested using an Android simulator but not yet on a real phone. Further validation is needed to confirm that, when a user actually moves their head, the estimated neck posture correlates accurately with true neck tilt.