Toronto Science Fair Log Book

2025-2026, Joshua Shao

Project Creation & Initial Research

2025-11-09

- Established the project goal: To design, build, and train a machine learning classifier capable of distinguishing between various human actions and a true fall event using real-time video input. The final result is a reliable, easily implementable, low-false-alarm fall detection system.

2025-11-13

- Decided on using the OpenCV library for video handling and display, the MediaPipe Pose model for efficient human pose landmark extraction, and Matplotlib/Tkinter for a secondary, real-time 3D visualization feature.

2025-11-15

- Researched and analyzed optimal camera input and placement. A key requirement is placement flexibility (need a bird's-eye view) and 24-hour monitoring capability.
- Decided on a camera with an automatic IR/Colour switch (night vision) to gather diverse data and be accurate for low-light/nighttime conditions.
- Brainstormed several possible fall-detection algorithms
    - Bounding boxes? Not really, as camera angles can result in drastically varied bounding boxes.
    - Velocity? Trails? Possible, but how?
    - Pose detection models. Possible, but several regular poses can trigger the model output. Needs to be more accurate.

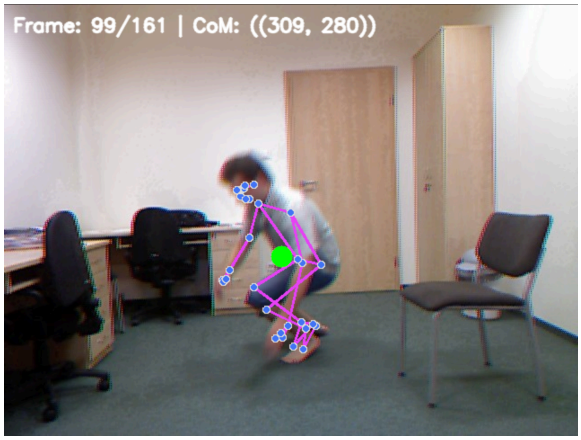
2025-11-16

- Choose a Raspberry Pi or similar Single-Board Computer (SBC) for remote camera placement and video streaming (RTSP protocol). The Pi handles network encoding; the powerful Mac/PC handles the MediaPipe Pose detection and ML classification to ensure low-latency analysis.

Code Implementation & Debugging

2025-11-17

- Created a demo pose detection program with Mediapipe's built-in pose detection model, with inputs using camera feed and also photos.

- 

2025-11-19

- Experienced several dependency issues, with each other and especially with macOS. For example, the latest Mediapipe binaries are not properly supported on the latest versions of MacBooks.
    - pip install numpy==1.26.4
    - pip install opencv-python==4.9.0.80
    - pip install mediapipe==0.10.21
- Experienced a critical crash on the macOS development machine: 'NSWindow should only be instantiated on the main thread!' (Tkinter/AppKit conflict), preventing the 3D graph from launching.
- *Resolution: Implemented a **thread synchronization** fix using a temporary **tk.Tk()** initialization on the main thread and **threading.Event()** to manage the launch of the 3D viewer on a separate thread, stabilizing the GUI environment.*

2025-11-20

- Questioned whether to use the more robust 2D pixel coordinates (x, y) or the less stable but spatially accurate 3D world coordinates (x, y, z) for the final ML model.
- Decision: Prioritized project stability by choosing 2D features for the initial model training. Marked 3D world coordinates (especially z-axis data) as a future performance enhancement if time allows.

2025-11-21

- Confused about how to calculate and return the 3D Center of Mass (CoM). Specifically, what unit to use for the z-coordinates?
- Resolution: Confirmed that MediaPipe's world meters are already in a scale equivalent to meters. Finalized the calculate_center_of_mass function to return the raw, unscaled z value for 3D analysis, while scaling x and y for 2D visualization

- Identified that averaging only shoulders and hips for CoM might not be accurate during complex motions, which could lead to false negatives.
    - Result: Expanded the CoM calculation landmarks to include the Knees (LEFT_KNEE, RIGHT_KNEE) to better represent the lower body mass distribution. Concluded that the CoM Vertical Velocity (Z) is the most important feature derived from the point.
- Organized the past log ideas & grammar.

Model Design Plan

2025-11-22

- Further brainstormed several possible algorithms for both accurate and fast fall detection.
    - Bounding box changes?
        - A fall can be reliably detected when the CoM Z-Velocity shows a sharp, uncontrolled negative spike, quickly followed by a large change in the Pose Bounding Box Aspect Ratio (H/W).
    - Maybe find the average change of the different frames or poses?

2025-11-23

- Defined the necessary output classes for the classifier. Initial thoughts: Fall, Standing, Lying Down.
- Decision: Crucially added Sitting Down and Bending Down classes. These "negative samples" are essential to prevent the model from triggering a false alarm when the CoM naturally lowers, significantly improving real-world performance.

2025-11-25

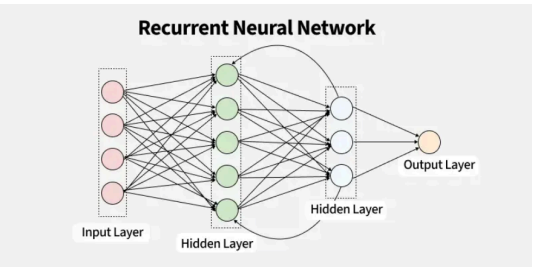
- As a fall is a sequence of events, a static image classifier is unsuitable. The model must analyze time series data.
- *Decision: Will use a **Recurrent Neural Network (RNN),** specifically an **LSTM (Long Short-Term Memory) model**, to classify sequential feature data extracted from 15-30 frame time windows.*

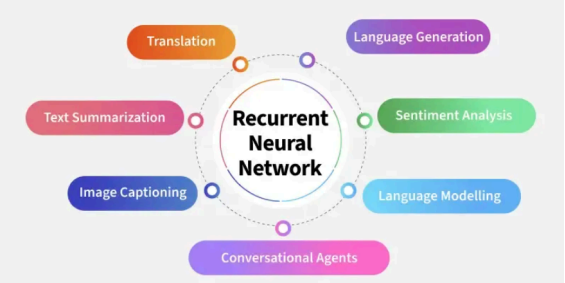
2025-11-26

- Established general architecture/algorithm

Part	Component	Input	Output	Purpose
Stage 1 (Extractor)	MediaPipe Pose	Raw Video Frame	Key Features (e.g., 2D/3D Coords, Bounding Box	Feature Extraction: Reduces the complex image (millions of pixels)

			Ratios, Velocity, Angles)	into a simple, abstract data vector (dozens of numbers) that describes the pose.
Stage 2 (Sequencer)	RNN (LSTM/GRU)	Sequence of Feature Vectors (e.g., 30 frames of features)	A Single Classification (e.g., 98% Fall)	Temporal Analysis: Analyzes the change in the features over time, capturing the dynamic motion of the fall.
Stage 3 (Classifier)	Dense Layer (Softmax)	Final Output of RNN	Probability distribution over all classes (Fall, Standing, Sitting, Bending)	Prediction: Converts the RNN's internal state into the final answer.

- Researched more into RNNs and LSTMs.
    - **The Core Idea: Memory**
        - Normal Network (FFNN): This person has no short-term memory. If you show them a fall sequence frame-by-frame, by the time they see the person hit the floor (the current frame), they've forgotten the person was standing (the previous frames). They see the landing, but not the action of falling.
        - RNN: This person has a tiny notepad (the Hidden State). After seeing Frame 1 (standing up), they write a note: "The person is upright." They pass that note to the process for Frame 2. By the end of the sequence, the note contains a summary of the whole motion, allowing the network to say, "The person was upright, then rapidly accelerated downward, and now they are prone. Conclusion: Fall."
- 



- Thought of other good indicators

    - Near-Fall/Trip Recovery: This is a movement where the person loses balance (rapid, uncontrolled limb flailing or staggering) but successfully recovers without hitting the floor. Detecting this is essential for identifying areas that need safety adjustments (e.g., adding grab bars).
    - Wandering/Pacing: Repetitive movement, especially during nighttime hours, can indicate confusion, anxiety, or early stages of cognitive impairment.
    - Immobility/Prolonged Stillness: Detecting that a person has been completely motionless outside of a normal sleeping area (e.g., sitting on the floor, slumped in a chair) for an extended period (e.g., over 5 minutes) is a strong indicator of an acute health problem (stroke, cardiac event, or injury).
- How would we consider gaps in data?
    - Imputation/interpolation?
    - For a missing data point, linear interpolation draws a straight line between the last known good coordinate and the next known good coordinate and estimates the values in between.
    - For any coordinate X at a missing frame t_m:

$$X_{t_m} = X_{\text{start}} + \frac{X_{\text{end}} - X_{\text{start}}}{N_{\text{gap}} + 1} \times (t_m - t_{\text{start}})$$

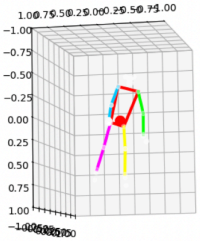
Where:

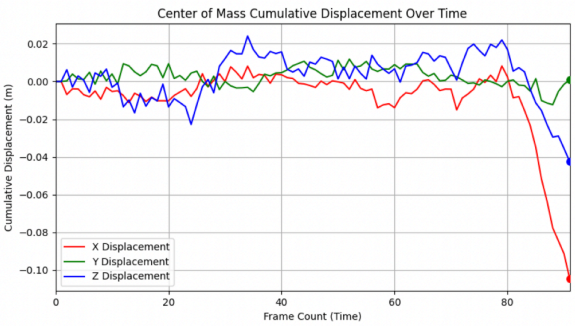
- X_{start}: The last visible X coordinate.
- X_{end}: The first visible X coordinate after the gap.
- N_{gap}: The number of missing frames in the gap (e.g., 4 frames in a 10-to-15 gap).
- t_m: The index of the missing frame.

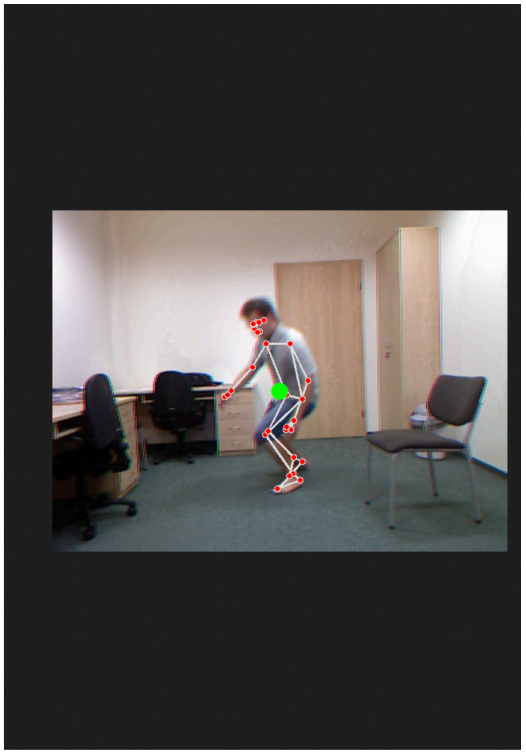
    - Use the Pandas library

2025-11-27

- Implemented a strategy of calculating the velocity/change of the CoM (Center of Mass) of people.
    - It works, but there are several possible actions (e.g. jumping) that would be flagged by the algorithm as “falling.”







○

2025-11-28

- Researched more on LTSM models and watched tutorials explaining the algorithm in detail.

2025-12-02

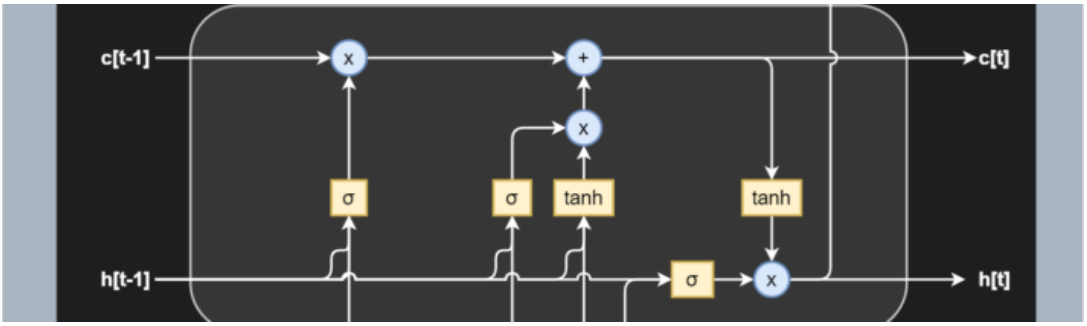
- Found several new datasets regarding fall detection
- Organized & cleaned up a room for the collection of my own dataset videos
- Continued to research and watch videos investigating LTSM models.

2025-12-07

- Cleaned up the living room to collect fall, standing, walking, and other ADL data.
- Turned several video data into processable frame folders (UR fall dataset, Fall_Simulation_Data folders)

2025-12-09

- Continued following the LTSM and RNN tutorials to better understand the field

- 

    - Source: MachineCurve
        - From https://www.analyticsvidhya.com/blog/2022/01/the-complete-lstm-tutorial-with-implementation/

2025-12-13

- Encountered several problems with data being skewed when people enter
    - Part of their body will be covered, leading to data inaccuracies
    - Possible solution: use the average data of several frames.
- Decided to reduce input parameters
    - Several facial features are redundant and will result in more inaccuracies in training.
    - This makes the RNN:
        - Faster to train.
        - More accurate (it won't get distracted by blinking or head turns).
        - Better at handling "jerks" because the torso is more stable than the face.
- Found that mediapipe's pose detection model only works with one person.
    - Solution: Decided to first develop an algorithm that works with one person (simpler).
- Occlusion handling
    - When a person goes behind an object, their confidence drops, and the tracker often "forgets" them or assigns them a new ID when they reappear

2025-12-14

- Problem with COM jerking outwards when multiple people overlap
- Thought of a possible solution to breaks in data by comparing the colour of clothes and seeing if they are very similar (so they are the same person)

- Possible problem: hospitals and nursing home patients may wear similar clothes to one another, and the colours of their clothes may be similar to the room environments.

2025-12-18

- Learned about normalizing data
  - You want the model to learn not where in the frame/image the person is, but their pose and their movement; **how** the person is moving, not **where** in the video frame they are standing.
- Decided to switch from the mediapipe pose detection model to the YOLOv11n pose detection model. Why?
  - The mediapipe pose detection model is much slower, with approximately 33 landmarks detected every frame compared to the 17 landmarks of their YOLOv11n pose model.
    - A lot of the landmarks of the mediapipe model are redundant; it has several face, chin, ear, and eye landmarks. The YOLOv11n is a perfect balance between accuracy and speed.
    - The YOLOv11n model is very fast, with 52.4
  - The YOLOv11n pose model also has ID tracking, which is key in scenarios with multiple people.
  - ## Models

    Ultralytics YOLO11 pretrained Pose models are shown here. Detect, Segment and Pose models are pretrained on the COCO dataset, while Classify models are pretrained on the ImageNet dataset.

    Models download automatically from the latest Ultralytics release on first use.

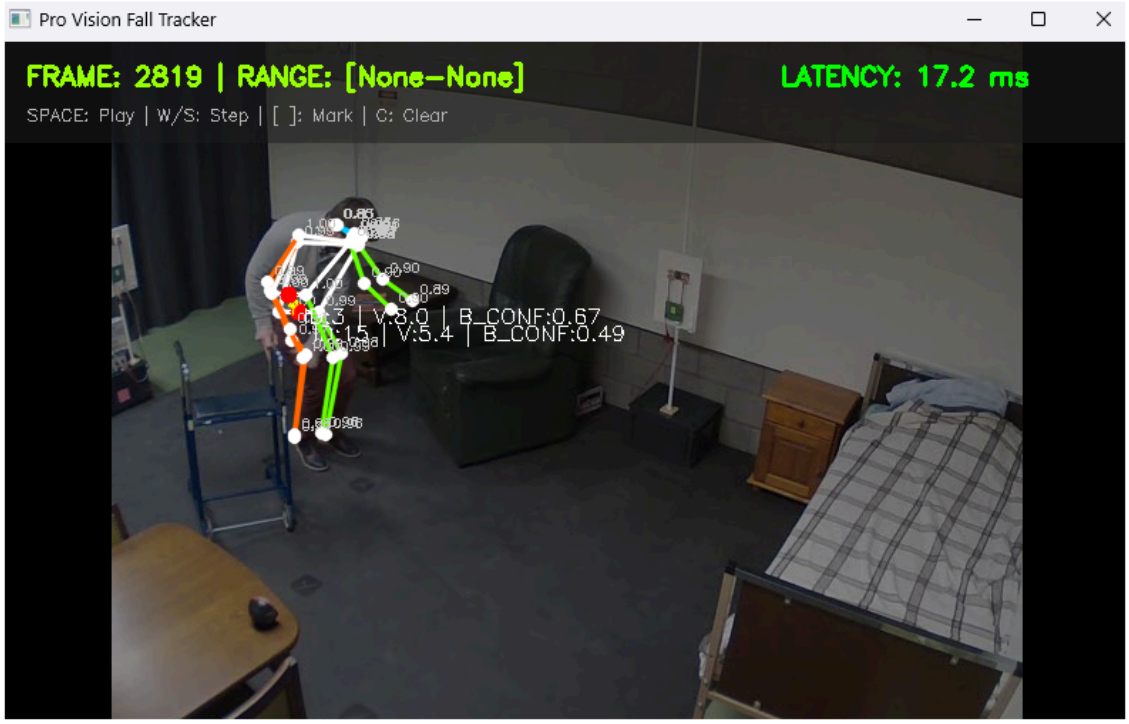
Model	size (pixels)	mAP^{pose} 50-95	mAP^{pose} 50	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n-pose	640	50.0	81.0	52.4 ± 0.5	1.7 ± 0.0	2.9	7.4
YOLO11s-pose	640	58.9	86.3	90.5 ± 0.6	2.6 ± 0.0	9.9	23.1
YOLO11m-pose	640	64.9	89.4	187.3 ± 0.8	4.9 ± 0.1	20.9	71.4
YOLO11l-pose	640	66.1	89.9	247.7 ± 1.1	6.4 ± 0.1	26.1	90.3
YOLO11x-pose	640	69.5	91.1	488.0 ± 13.9	12.1 ± 0.2	58.8	202.8
  - https://docs.ultralytics.com/tasks/pose/
- Brainstormed an idea of grouping the landmark limbs into sections

- Core, arms, etc…

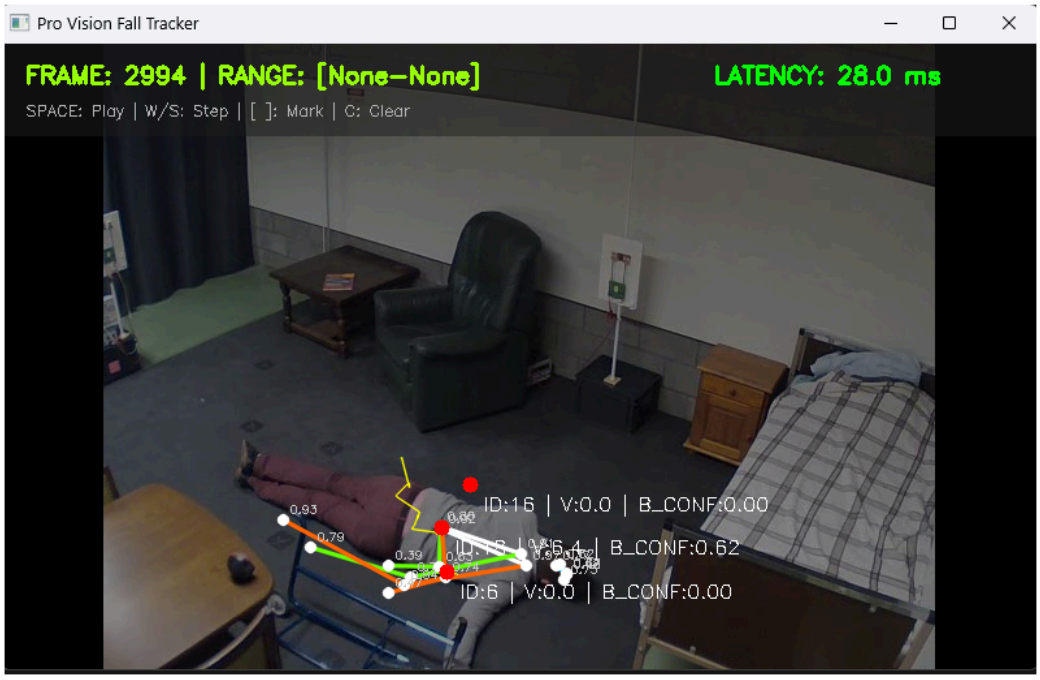
2025-12-20

- Decided on the different landmarks/inputs for the LSTM model inputs.
- Grouped certain body landmarks into limbs, so that the model will learn connections faster.
  - The “temporal state” (velocities, bounding boxes)
  - Torso core
  - Left arm
  - Right arm
  - legs/stance

2025-12-21

- Encountered problems with ID tracking after a few data gaps.
  - happens when the object tracker (bytetrack) loses its lock on the person for a moment (due to occlusion or low confidence), and when the person reappears, the tracker mistakenly assigns them a new, fresh ID.
  - 

  - **The Fix: Max Distance Association Check**
    - Adding a mechanism to check if a "new" ID has appeared too close to the last known location of an "old" ID that was recently lost. If they are very close, it should assume they are the same person and force the tracker to reuse the old, stable ID's state.

- Problems with YOLOv11n-pose's model "pulling" limb landmarks to furniture (keypoint bias/occlusion)
    - When a limb is too close to a large, textured object (like furniture), the model gets confused and snaps the arms to the object's edge (in this case, the sofa).
    - The fix: The One Euro Filter is good, but it can be improved upon by dynamically adjusting the smoothing level based on the keypoint's confidence. I.e.:
        - If the wrist confidence is low (e.g.,< 0.4), we trust the previous smoothed position more than the current detection to resist the "pull."
- Still struggling with two major, related problems: data gaps due to ID switching, and inaccurate pose estimation during occlusion (face to the ground).
    - 

- Problem: Short data gaps greatly impact the accuracy of model training and implementation
    - Solution: Filling in keypoint data based on the previous frames, if there is a data gap.

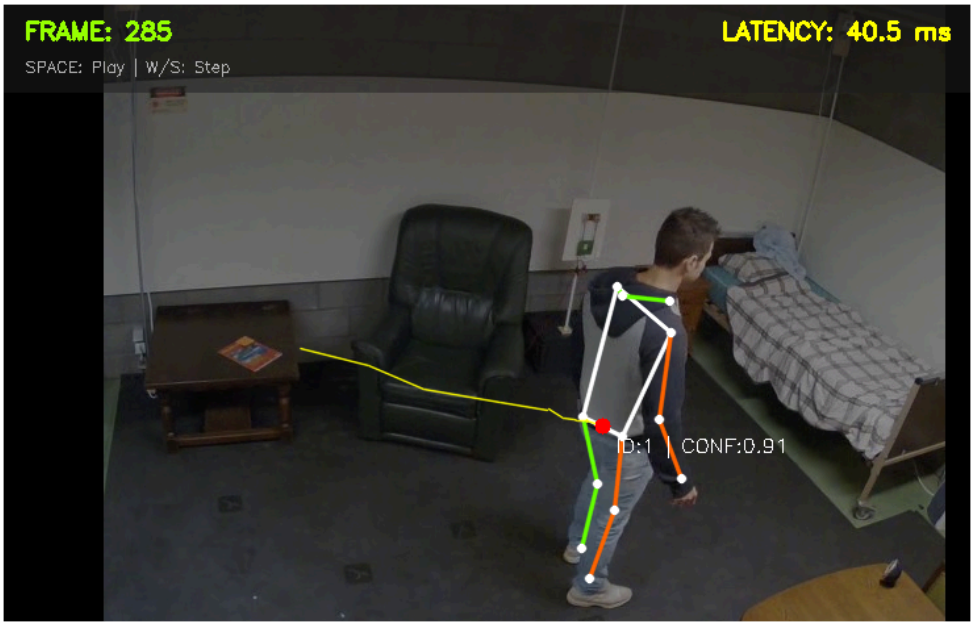
2025-12-22

- Ordered a better camera (1080p 60fps) for ~$22 on Temu. This will be great for training, though I will use a cheaper Raspberry Pi model camera for the final demonstration.

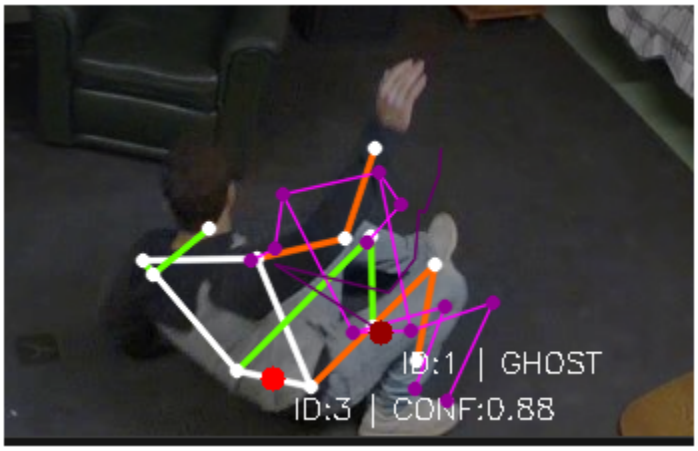
- Thought of building an app that will make using the models much easier and straightforward. Thought of other ways to make the project more modularized/easier to implement in real-world nursing homes or facilities.

2025-12-23

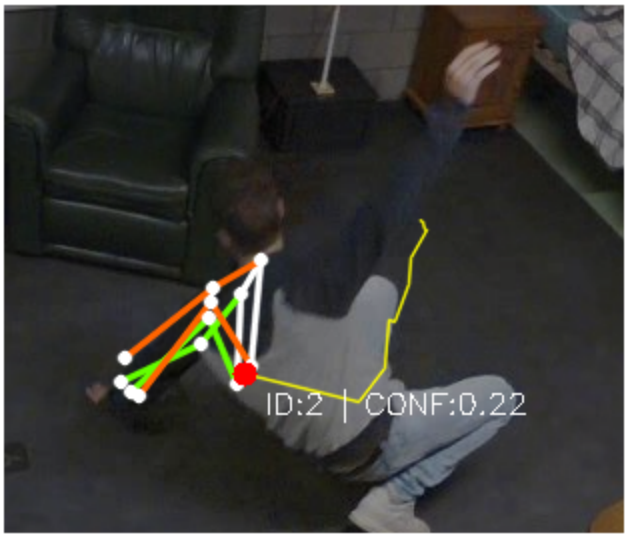
- Tested the pose model and saw that it stopped detection after falls.
    - Solution? Started using the Yolov11s-pose model instead of the YOLOv11n-pose model (it uses more computational power). Also created “ghost frames,” which were poses that continued to appear if no frames were detected during a fall.
        - Encountered another problem with ghost frames: when the person is detected again, they get a new ID, which results in model inaccuracies.
- Encountered problems with more data gaps
    - Especially in fall scenarios, one or two gaps in landmarks will result in model inaccuracies
    - Solution: Keypoint Persistence: In TrackState.update, if a joint confidence drops below a minimal threshold, it will now use the last smoothed position instead of the raw, jittery input. This stabilizes points that briefly disappear or jump.
- Summary of issues:
    - Occlusion-Induced Pose Errors (Walking Sideways): When the subject walks sideways toward the camera, the far arm/hand is occluded by the torso. The pose model sometimes collapses or aggregates all keypoints onto the visible arm, resulting in a physically impossible skeleton prediction (e.g., both elbows and wrists stacking on the visible arm). This is a model-level failure under high self-occlusion.



- Tracking ID Fragmentation (During/After Fall): Despite aggressive re-association logic, the original tracking ID (e.g., ID: 1 | GHOST) persists as a ghost, while a new, separate, high-confidence track (e.g., ID: 3 | CONF: 0.85) is created for the person on the ground. This indicates the re-association criteria are still failing to bridge the gap between the lost/ghost CoM and the newly detected CoM.



- Keypoint Aggregation/Jitter (During Impact/Low Confidence): During the violent, low-confidence frames of the fall impact (as seen in Image 4), the pose detection model inaccurately shifts multiple landmark points (e.g., shoulder, elbow, wrist) toward a single, incorrect location, resulting in an unnatural and useless skeletal structure.

    - 

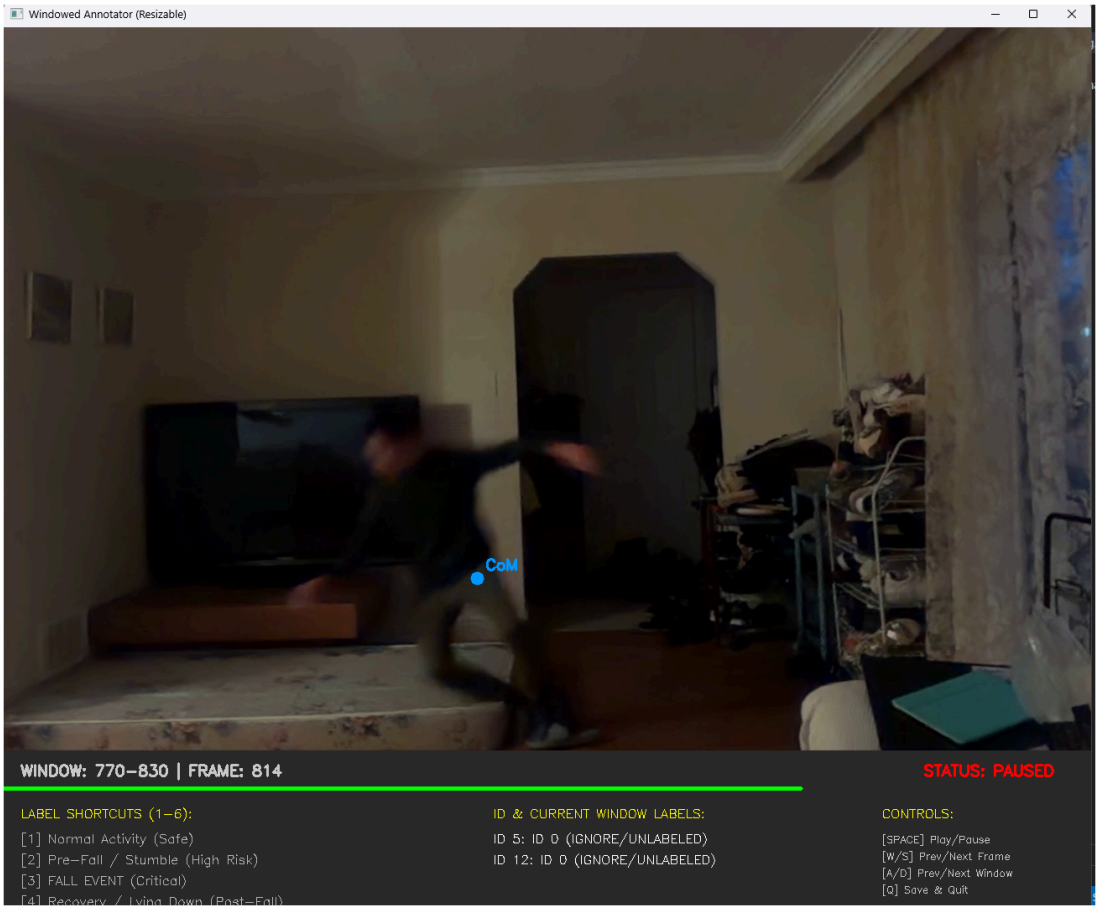
    - Solutions:
        - **Addressing Keypoint Aggregation/Collapse (The Arm Issue):** We will use **Velocity Clamping on KPT-to-KPT Movement**. If a keypoint moves more than a defined maximum distance (MAX_KPT_MOVE_PX) *relative to the adjacent keypoint* (e.g., the elbow relative to the shoulder), we assume it's a model jitter/collapse error and discard the input, using the smoothed previous position instead. This is a powerful structural fix.
        - **Aggressive Ghost ID Termination:** If a new, high-confidence track appears near the ghost's *last known location* (not its predicted ghost location), we aggressively terminate the ghost to force the re-association.
        - **Refined Persistence:** I've clarified the persistence logic in TrackState.update to ensure a point is held firmly in place only when confidence is low, and allowed to move when confidence is high.
- Looked closer at the data found online and realized that the frames per second are inaccurate (there are several frames; the image changes after 2-3 frames each time). Hence, I will need all of my data to be personally collected, with a 60fps or 30fps camera.
- Encountered even more errors; decided to fall back on a previous iteration of the pose-detection algorithm.
- Started to investigate the idea of “Probabilistic Keypoint Estimation (PKE)”, which predicts keypoints under occlusion.

2025-12-24

- Decided to first train a model on clear data where people are facing the camera, not occluded from view. Later models will work with semi-occluded data.
- Decided to train models on different camera angles for maximum accuracy and usability in different use cases.
    - **Frontal View (Primary):** Straight on, showing the person approaching the camera. (Good for bilateral symmetry and CoM change).
    - **Side View (Critical):** Straight side profile. (Best for capturing the height/vertical drop and changes in gait features like stride length).
    - **Corner View (≈45◦):** Diagonal view from a corner of the room. (Introduces perspective distortion, forcing the model to be robust).

2205-12-25

- Thought of building an app to display the results, making the project more accessible and realistic
- Brainstormed a possible problem about different camera resolutions affecting the coordinate points.
- Experimented with socket communication between a MacBook and a Windows PC.
- Encountered problems with motion blur hindering pose-detection results



    - Possible fix: adjust the tracker's parameters to be much more tolerant of low confidence detections during motion, or to allow the "ghosting" period to last much longer, using the predicted velocity to bridge the gap.
- There are still many data gaps.
    - Solution: Gap filling.
        - Large gaps need complex buffers, and it will be difficult to incorporate; hence, gap filling should be limited to only a few frames.

2025-12-26

- Thought of ways to make the model more accessible to different cameras with different resolutions: using the torso as a “measuring stick”.
    - Introducing a calibration step to the system, setting up a scale factor to base the entire body’s coordinates on. That way, any resolution or camera width will work.
- The current pipeline execution order:
    - Video Processing/Tracking Loop (in landmarks_csv.py) → Calls modular.py for each frame.modular.py

    - (FallDataProcessor): Receives raw scaled coordinates, maintains a history buffer, and calculates raw velocities (change in position over time).
    - landmarks_csv.py: Collects the data from modular.py, then calls standardize_coordinates.

2025-12-27

- Encountered more problems with the normalization; decided to research several alternatives
    - Found Apple's "Core ML", which is built into Apple computers and can train several machine learning algorithms. It is also fast as it runs natively and is also smart. Decided to switch to this from the previous YOLO and mediapipe pose detection algorithms.
- Thought of extending the project to not only a camera-based feedback but also to a general emergency/distress indicator with a microphone.
    -

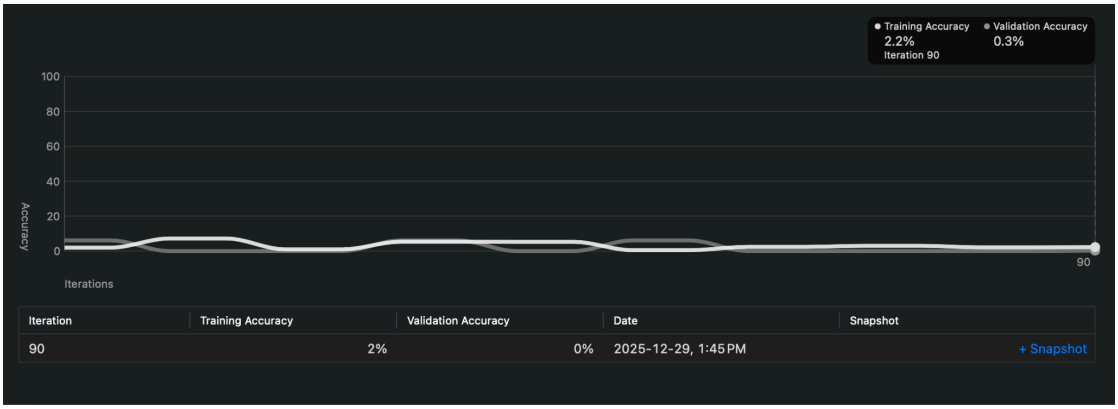
2025-12-28

- Began collecting data and finalized labels
    - Fall
    - Pathological_Motion (merged Dizzy_unstable with Seizure_like. Uncontrolled, highly abnormal, non-walking motions. Share a common feature of unstable, jerky movement)
    - Unresponsive_Floor
    - near_fall
    - Stand
    - Walk (normal stride, clean, rhythmic, continuous gait)
    - Shuffle (shuffle walking)
    - Resting_static
    - Bend_Pick_Up
    - Sit_Down_Up
    - Lay_Down_Up
    - Background
- Continued to label videos, got ~10 examples for each label so far.
- Began developing a swift app on Xcode to first display pose detection results. The app currently has three main sections: monitor (look at real-time model feedback), settings (tweak detection confidence, configure new cameras, etc), and history (history of falls).

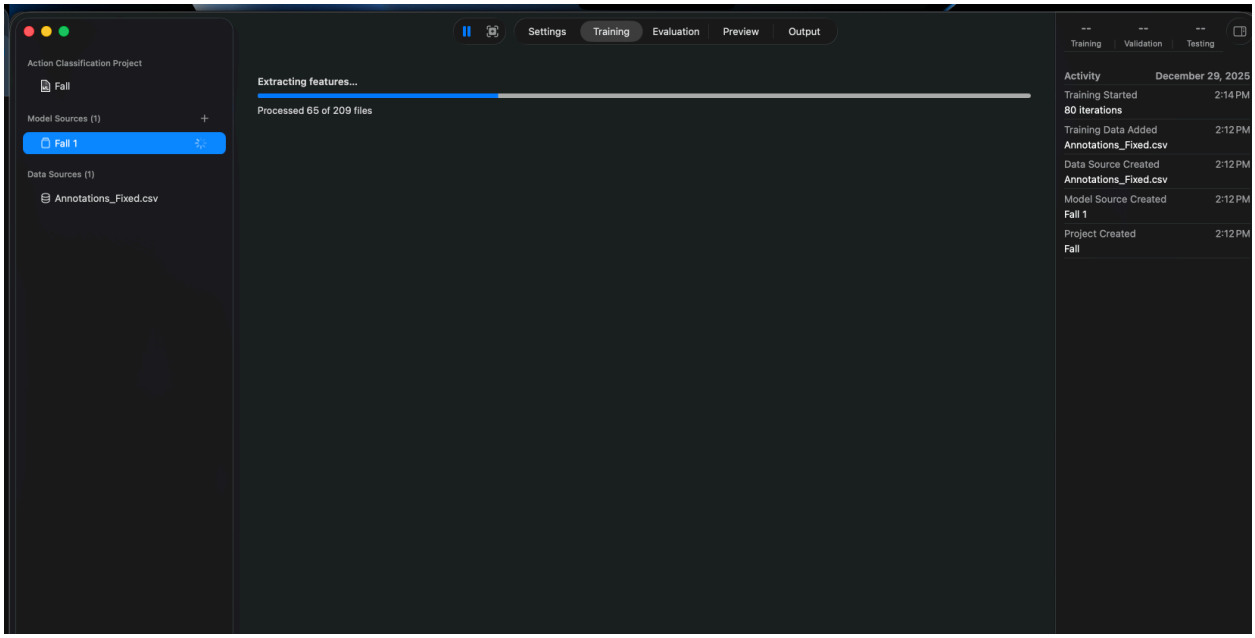
Environmental Cue	Emergency
Sounds of Struggle/Pain	Unassisted fall, injury, seizure.
Smoke/Fire Detection	Kitchen fire, electrical fault.
Water Leak/Flood Detection	Pipe burst, appliance failure.

2025-12-29

- Collected even more data and finished labelling for the first batch of training. Decided to test out model training and trained a pretty inaccurate model.
  - 

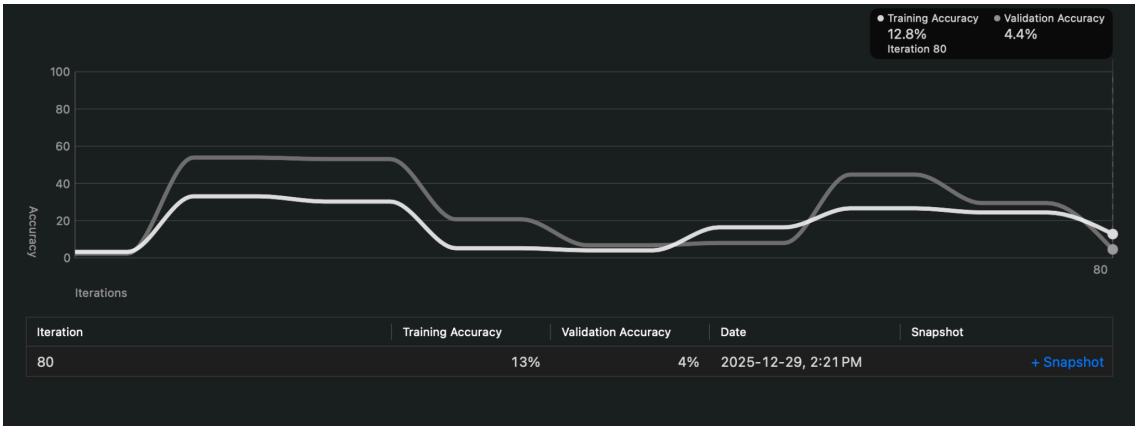
  - 

  - Experience problems with inconsistent labelling and inconsistent video window annotations.
- The model was extremely inaccurate, even more inaccurate than simply guessing (100/12 = ~8%). Decided to decrease the number of labels, as some labels were too similar to one another (e.g. walk & shuffle_walk). Additionally, the model needs much more training data, as the amount of training data is “skewed”, having many more examples of walking compared to lying down or falling.

- 

- 

- Reducing the number of classes significantly increased the model output significantly increased, but it still stayed relatively low:

- 

2025-12-30

- Decided on finalizing the labels:
  - Dividing the “rest” label into 2 separate labels, sit and lay (they are fundamentally different --- one is horizontal and the other is vertical)
  - Grouping the bend, sit down/up, and lay/down/up into one group called “transition”.
- The final set of labels:

Label	**Rationale for Choice / Why Chosen**	**When is the label found?**
walk	Core movement for ADL (Activities of Daily Living). Essential baseline for identifying gait issues.	The person is moving across the floor by stepping, regardless of speed.
stand	Defines a non-moving but non-sitting/non-lying static state.	The person is stationary in an upright position (e.g., waiting, holding position).
sitting_static	Defines a static state where the person is seated (e.g., on a chair, couch).	The person is stationary while seated, with no movement of limbs/torso.
resting_static_floor	Defines a static state on the floor, usually after a transition or a fall. (Renamed from lay for clarity).	The person is stationary on the floor, usually lying down.
transition	**CRITICAL:** This generic label covers all movements *between* static states (e.g., sit-to-stand, stand-to-sit, squatting). **It**	The person is actively moving from one static state to another, or performing a small functional movement.

	should be broken down into directional labels (e.g., sit_down, stand_up, bend_down) for TSF.	
fall	The primary event of interest. Captures the high-speed, uncontrolled descent.	The moment the body is losing stability and accelerating downwards towards the floor.
near_fall	Critical for proactive prediction. Captures recovery motions.	The person loses balance but successfully arrests the fall and recovers stability.
unresponsive_floor	Key post-fall state. This segment must be long enough to be meaningful. (Corrected typo from original data).	The person is on the floor and shows no significant movement (i.e., not attempting to get up or roll).
get_up_from_floor	Specific transition from the floor to standing (or sitting). (Renamed from get_up for clarity).	The person is actively engaging in the motion of rising from the floor back to an upright position.
background	Represents any activity outside the frame, or non-target events that clutter the video.	The segment where the activity is not visible, or the frame is empty/irrelevant to the target task.

- Started round 2 model training, with much more data and fewer labels.
  - 

  - The model accuracy was still extremely low.
    - Felt like giving up---spent hours collecting data
    - Possible reasons: the new camera was blurry ⇒ the pose detection model had many jitters of data