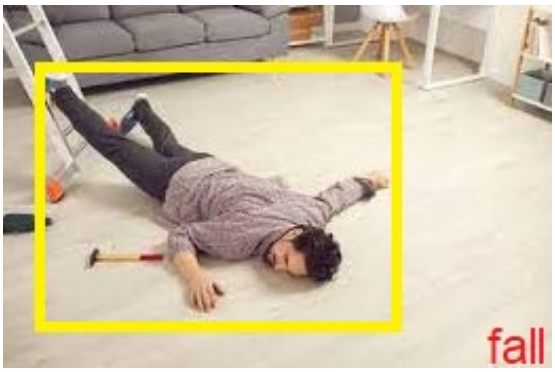
Automated Fall Detection using AI Vision

- **Due:** *11:59 pm 24/5/2023 (Wednesday of Week 12)*
- **Contributes** 50% of your final result
- **Group Assignment – Group of 3-4 students**

Summary

The number of elderly people is increasing day by day all over the world. One of the biggest health problems for older people is fall injuries, and this problem is exacerbated for older people living alone. In fact, falls are one of the leading causes of death among the elderly. For this reason, automatic fall detection is receiving considerable attention. Although wearable sensor-based approaches have high detection rates, some potential users are reluctant to wear them. The recent impact of deep learning has changed the computer vision landscape, improving performance in many relevant tasks, such as object recognition, image segmentation, captioning and etc. In this project, we seek an alternative approach, focusing on vision-based approaches for fall detection. It consists of object detection and classification based on deep learning.



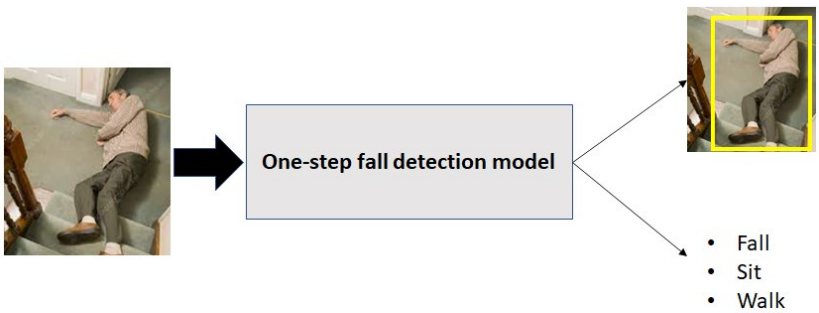
Constraints:

Fall detection Dataset

You need to download a fall detection dataset from this link. The dataset can also be found on Kaggle. There are two directories with *Images* and *Labels*. The Images directory consists of two subdirectories: *train* (374 images) and *val* (111 images). **Note that, *the validation set here will be included in the training set.*** The Labels directory also consists of two subdirectories *train* and *Val*. In this directory there is a text file for each image. The text file contains a set of values representing a *class label* and *4 bounding box values*. The class labels are "fall detected" (0), "walk" (1), "sit" (2). The values in the bounding box are used to locate people in the image. Depending on the number of people on the images, the set of values varies according to the number of people. **Note that you must collect your own testing set.** In order to avoid overlapping training and test images, the testing set must be collected by yourselves without online crawling. **Please collect at least 50 test images for each class**. As for annotation, you can use LabelMe or any other annotation tool that allows you to make bounding box annotations.

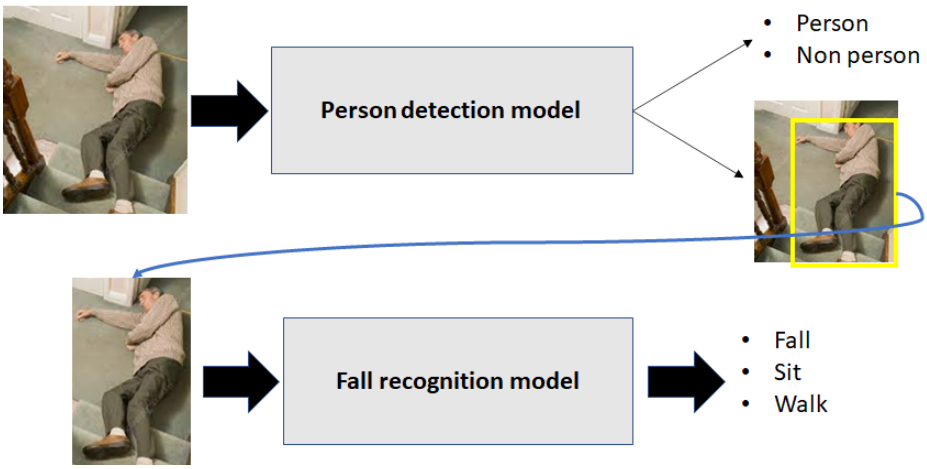
Fall detection

Two architectures are to be implemented and compared: (1) one-step fall detection and (2) two-step fall detection. For one-step fall detection, the architecture will be trained with a single object detection model and the target label will be human actions and bounding box values.



One-step detection module

As for the two-step fall detection architecture, two stages need to be implemented, that is the people detection, and then fall recognition. After detecting the people, you will need to extract it as an input to be fed into the fall recognition model. The fall recognition model should use a deep learning approach. You can also read this article to better understand the features generally used for human action recognition. You will get to know about pose-based and appearance-based features.



Two-steps detection module

Both architectures should be analyzed under various conditions: environment, viewpoint, distance, etc.

Object detection

Regarding the object detection module, you should take advantage of existing libraries such as PyTorch, Tensorflow, Keras, Theano, etc. At the very least, you should re-implement the object detection technique that will be taught in the lab session. After this attempt, you are expected to understand the methodology behind object detection and know how to form an object detection model using deep learning. Next, you should be able to adapt the code to work with fall detection using your own dataset. In addition to your own dataset, you are allowed to add additional training images from any existing people detection dataset to increase the robustness of your people detection model.

System requirements

- One of the key features of fall recognition system is to provide a service that can automatically trigger an alert signal when fall event is detected.
- **Basic version:**
  - The basic version is to implement the aforementioned two approaches: One-step and two-steps fall detection
  - Both fall detection architectures should be analyzed under various conditions: environment, viewpoint, distance
  - Increase the number and diversity of fall detection training set
  - GUI that should display:
    - Live prediction of fall detection, which means that the system will take the data from the video camera (can be a webcam) and provide a real-time analysis of the fall.
    - Display an alert signal in case of detected fall
- **Extension 1**: A robust fall detection system must be able to provide decent performance in low light conditions. Thus, the extension is to design an algorithm that could cope with this condition. This involves self-collection of datasets in various low-light conditions.
- **Extension 2**: The fall detection system is widely used as an assistive device whose main purpose is to alert when a fall has occurred. However, to enable patients to receive appropriate treatment in a shorter time frame, early fall detection is the main focus. Therefore, the extension of this work is to implement early fall detection using a computer vision-based approach. Note that, for the early prediction module, spatio-temporal features need to be taken into account, hence the composition of a time series dataset.

Project requirements

- Source code maintained on Git based VCS (Github/Bitbucket/GitLab/…). You must provide read-only access to the tutor/lecturer
- Running illustrative demo of a working prototype (please refer to **Marking Scheme** for details on functionality that needs to be implemented)
- Project report (8-10 pages) that includes the following sections
    - Cover Page (with team details) and a Table of Contents (TOC)
    - Introduction
    - Overall system architecture
    - Data collection and annotation,
    - Implemented machine learning techniques for people detection and fall recognition,
    - Scenarios/examples to demonstrate how the system works,
    - Some critical analysis of the implementation,
    - Practical application description and
    - Summary/Conclusion.
- Presentation (live) and oral Q&A afterwards. Questions will be regarding the project as well as the unit content of the lectures (fundamentals). All group members are required to participate and contribute to the presentation, and should be able to answer questions related to the project and detail own contributions

Marking Scheme

Requirements	***Mark***
Task 1: Data collection. Implement the one-step fall detection architecture with the new dataset. Perform an analysis under a variety of conditions: environment, viewpoint, distance and etc.	20
Task 2: Implement the two-steps fall detection architectures with the new dataset and compare its performance with the one-step architecture. Perform an analysis under a variety of conditions: environment, viewpoint, distance and etc.	30
GUI: Display the live prediction of human detection and fall recognition, which means that the system will take the data from the video camera (can be a webcam) and provide a real-time analysis of the fall. Also, display an alert signal in case of detected fall.	10
Project Report	10
Project Presentation (Live)	10
	80
Research Component (can be done by the whole team, a sub-team, or an individual) There are many potential extensions in this project. **Extension 1** and **Extension 2** are two examples but there are many more. Choose one and get your tutor's approval then complete it very well. You can get up to 40% for this component and your Assignment will get up to 120%	Up to 40
	120/100
You need to follow good programming practice (e.g., well-designed, well-structured codes with clear and helpful comments). Failure to do so get penalty.	**Up to -20**
You need to demonstrate the progress you make every week to your tutor. That is, if your tutor approach you and ask for the progress, you have to be able to show the tutor the progress you have made in comparison to the previous week. Failure to do so will get a penalty.	**Up to -50**

NOTE:

- Individual marks will be proportionally adjusted based on each team member's overall contribution to the project as indicated in the '*Who did what*' declaration.
- You must also provide to shlee@swinburne.edu.my read only access to your git repository within 1 week of forming teams.

Submission

- You must upload your work to Canvas by 11:59pm on 23/5/2023 (Wednesday). Create a single zip file with your code and a working version of your system. Standard late penalties apply - 10% for each day late, more than 5 days late is 0%.
- You will be asked to do a live demonstration and demo. The location, date and time of the presentation will be announced in week 11.