Fall Detection System

Justin Haryanto

Nhan Nguyen

**SUBSYSTEM REPORTS**

REVISION – Draft

04 December 2022

Subsystem Reports

for

Fall Detection System

Prepared by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Author Date

Approved by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Leader Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

John Lusher II, P.E. Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T/A Date

**Change Record**

| **Rev.** | **Date** | **Originator** | **Approvals** | **Description** |
| --- | --- | --- | --- | --- |
| **-** | 04 Dec 22 | Nhan Nguyen |  | Draft Release |

**Table of Contents**

[**No list of figure entries found.**](#_heading=h.usqmm6d3jvty) **5**

[**1. Introduction**](#_heading=h.m2cv2ram3r6l) **6**

[**2. References and Definitions**](#_heading=h.gkylv93my0te) **7**

[2.1. References](#_heading=h.iri3kxbqbkb) 7

[2.2. Definitions](#_heading=h.clfxckdyn7op) 7

[**3. Pose Estimation Subsystem Report**](#_heading=h.tyjcwt) **8**

[3.1. Subsystem Introduction](#_heading=h.jb6akegerg9a) 8

[3.2. Subsystem Details](#_heading=h.9v47apis0mxv) 8

[3.3. Subsystem Validation](#_heading=h.6liw5tt9m6i9) 9

[3.4. Subsystem Diagnostic](#_heading=h.xsmjhu2h09kb) 11

[3.5. Subsystem Mitigation](#_heading=h.coy2mebcpzzx) 11

[**4. Fall Detection Subsystem**](#_heading=h.6klac97ey89p) **12**

[4.1. Subsystem Introduction](#_heading=h.psfvz2wmql22) 12

[4.2. Subsystem Details](#_heading=h.h0vzhfz3v3sc) 12

[4.3. Subsystem Validation](#_heading=h.369eje5b2nnv) 14

[4.4. Subsystem Diagnostic](#_heading=h.jn9eye9jegkp) 15

[4.5. Subsystem Mitigation](#_heading=h.32u9s181hhjt) 15

**List of Tables**

No list of table entries was found.

**List of Figures**

# No list of figure entries was found.

# 

# Introduction

The fall detection system will detect when a person has fallen in a video. There are two main subsystems, the pose estimation system and the fall detection system, which have both been evaluated on their performance.

It does this by breaking down a video into its individual images and feeding it into the pose estimation subsystem. During this part, the subsystem extracts relevant key points and bounding box information, then sends this information to the fall detection system to determine if a fall has occurred.

While these systems are currently not working at their best performance and more validation is needed, it is clear that they could be eventually integrated into a whole system seen in the ConOps, ICD, and FSR.

# References and Definitions

## References

Fall Detection System (GitHub)

[https://github.com/Nhannguyen993/TAMU-GitHub](https://urldefense.com/v3/__https://github.com/Nhannguyen993/TAMU-GitHub__;!!KwNVnqRv!GqhHrXEEvFW1nE-B1zXPex1ZIgvYBdPPrgFjJ8sAyx6plz6v-Jtc1UqE3bzU1cLvkWOucI7YarFElLxtJ9sKnsky0VQDWKE$)

## Definitions

| CNN | Convolutional Neural Network |
| --- | --- |
|  |  |
|  |  |

# Pose Estimation Subsystem Report

## Subsystem Introduction

The pose estimation system uses AlphaPose to extract key points and bounding box information from individual video frames for use in fall detection. The system is currently run on google colab and uses the GPU resource there. The system has been evaluated on its ability to work on videos and images of the training data set collected from other universities. This is done to verify that key points and bounding box information could be extracted.

## Subsystem Details

The pose estimation system is able to process videos of a variety of qualities with the use of AlphaPose, which is capable of whole-body pose estimation and tracking. It was found that AlphaPose performed well when ran on COCO, a renowned data set for large-scale object detection, segmentation, and captioning, achieving a mean average precision of 75. The COCO dataset has hundreds of thousands of regular scenes containing common objects placed in natural context to allow programs to detect and segment these objects.

The goal of this system is to be able to take in a variety of videos and extract data from them. For training and testing, the UR Fall data set was used, which has labeled images as well as their associated video of various falling motions. The system should be able to rapidly process video at various frame rates and differing quality. Since AlphaPose is already capable of running at 23 frames per second (fps), a challenge is getting it to work at 30 fps. Another challenge is to get the system to work with videos of poor quality and with live video.

There are various methods used to examine the subsystem’s performance. To check its ability to provide key points and bounding box data, several hundred labeled images from a fall data set. To check the current limits of the system, the pose estimation subsystem is given videos of varying quality and duration.

Efforts to improve the frame rate that the system runs at, operation with low-quality images and live video were put on hold to focus on other systems.

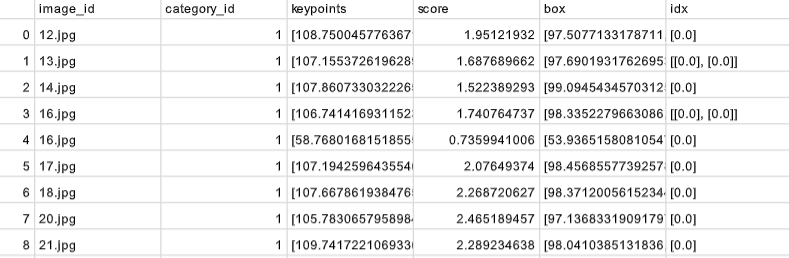
## Subsystem Validation

The system should be able to extract bounding box and key point data regardless of the quality of the video so that the fall detection system could operate properly. The capability of the AlphaPose system to work with lower-quality video was demonstrated, but not thoroughly tested to test other systems. Figure 1 shows a video clip, adjusted for quality, that was processed.

**Figure 1:** Frame from Video at Low Quality after Processing by AlphaPose



**Figure 2:** Data Collected from Video at Low Quality

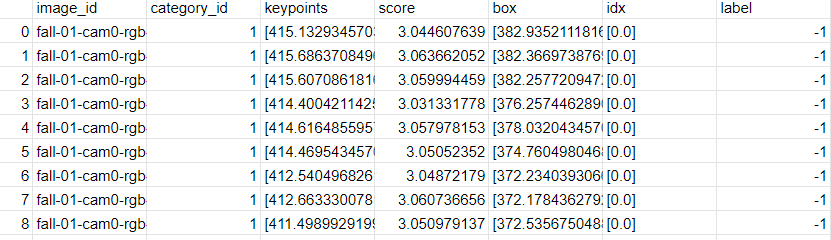


In addition to this, the pose estimation system was tested with images from a labeled fall data set to see if bounding boxes and key points information could be obtained. Figure 4 confirms the system’s ability to supply the fall detection subsystem with bounding boxes and key point data.

**Figure 3:** Image from Fall Data Set after Processing by AlphaPose



**Figure 4:** Data Collected from Fall Data Set Images



## Subsystem Diagnostic

The system has not faced rigorous testing with videos of different frame rates or quality. It also does not possess the interface for live video feed to be tested in its current configuration.

## Subsystem Mitigation

An expanded data set with videos at various frame rates and quality could be applied to the system to examine its performance. Any improvements from there would be made by adjusting the underlying model of AlphaPose. For live video feed, an interface could be created to operate with the code either on Google colab or directly on a machine.

* 1. ***Subsystem Conclusion***

The subsystem's ability to provide bounding boxes and key point data is working correctly. Its ability to handle videos at differing quality, frame rate, or as live feed has not been thoroughly tested or improved upon. When interfaced with the rest of the subsystems, it will allow a steady flow of data collected from inputted videos for the fall detection system to work with.

# Fall Detection Subsystem

## Subsystem Introduction

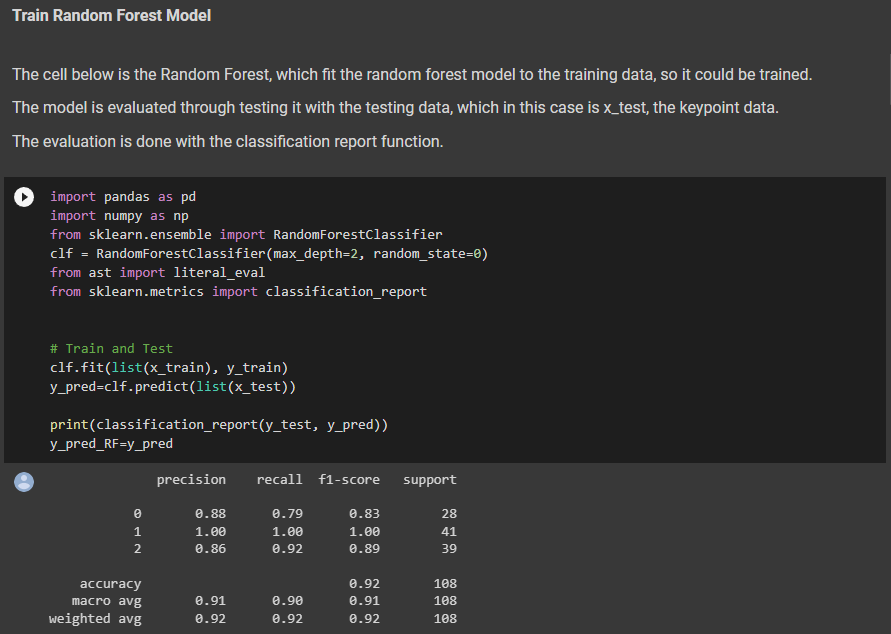
The fall detection subsystem takes the data processed by the pose estimation subsystem and runs it through two machine learning models to determine if a fall has occurred. The results of the fall detection programs are then combined to give the final verdict on whether a fall has occurred or not. The result is then sent out to the external system owned by a client for them to use.

## Subsystem Details

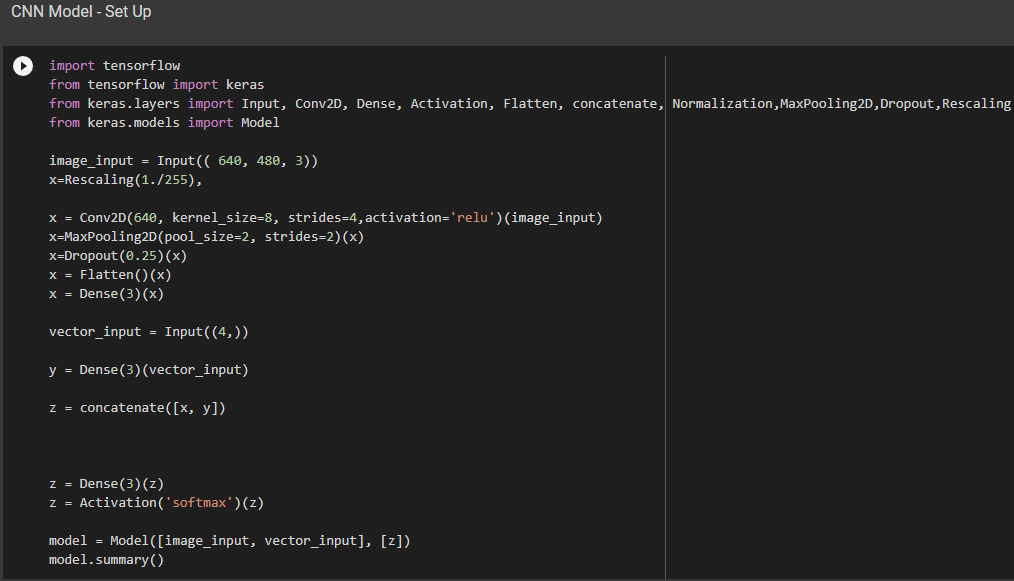
For determining falls, the random forests and convolutional neural networks (CNN) machine learning models were used, where they were chosen based on the data to be given. Random forest is a simple model that is sufficient for the analysis of key point information. In contrast, the CNN model’s image processing capability made it suitable for the analysis of video frames alongside bounding box information.

Training and testing were performed using the UR Fall Detection dataset, which contains a series of falling videos with their respective labeled image frames. After the pose estimation system generates the bounding box and key points information, the data is processed by the two machine learning models for analysis. Then the results of the models are compared, where a fall has occurred only if both models agree, and the verdict is sent off to an external system. For our demonstration, the external system is represented by a video overlaid with text indicating if a person has fallen or not.

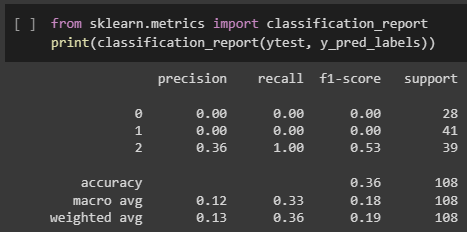
**Figure 5:** Random forest code snippet and accuracy



**Figure 6:** CNN code snippet



**Figure 7:** CNN accuracy



## Subsystem Validation

The following are the validation criteria for the fall detection subsystem:

| **Test** | **Success Criteria** | **Methodology** |
| --- | --- | --- |
| Fall Detection Accuracy | >90% Correct | Compare results to dataset |
| Random Forest Accuracy | >90% Correct | Analyze key points from the dataset, compare results to dataset |
| CNN Accuracy | >90% Correct | Analyze bounding boxes from the dataset, compare results to dataset |

The focus of the subsystem is to increase its categorization accuracy when determining if a fall has occurred or not. It should determine if a person has fallen regardless of fall-like movement. This was validated through training on a hundred images from the UR Fall data set and measuring the accuracy of the models on a test set of another couple hundred images. The models work correctly if they are both able to correctly identify over 90% of the images associated with falls. The random forest was able to meet this performance for this test. The CNN performance was poor though this will be corrected with future adjustments.

## Subsystem Diagnostic

False positives can occur when the subsystem determines a fall has occurred when there wasn’t. On the other hand, false negatives occur when a fall has occurred but the subsystem has not detected it. Based on the output, it was found that the subsystem could not decide between falling or not falling despite the person falling, resulting in a blinking of the text “fall”. However, the way the result is decided makes false negatives occur more often.

Another problem that was found was the processing time for the fall detection subsystem was slow, meaning that it will not keep up with a video.

## Subsystem Mitigation

* train the system with more datasets with diverse movements and fall-like movements
* Run the program on a GPU
* Optimize the code or write a custom version of models
* parallelize code

False results will be reduced when both models are given more training and diverse datasets with a variety of poses, especially ones with fall-like movements. The parameters of the machine learning models can also be modified to tune the system.

There are two ways where the speed of the fall detection subsystem could be improved. The first is to use a GPU to run the code, which allows for parallel processing. The second is to either make optimized random forests and CNN models that suit our purposes or decrease the size of the program.

* 1. ***Subsystem Conclusion***
* From example
  + Did it work correctly?
  + what else needs to be done
* our
  + what works or doesn't