School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales

Deep learning approaches for managing falls in aged care facilities

by

Abanob Tawfik

Thesis submitted as a requirement for the degree of

Bachelor of Engineering in Software Engineering

Submitted: April 2021

Supervisor: Dr. Reza Argha

Student ID: z5075490

Abstract

Falls are one of the leading causes of death and unintentional injuries worldwide. Adults over the age of 65 and individuals in care facilities tend to suffer more falls. Activity recognition using camera data has been researched greatly, however they raise privacy concerns for subjects. Wearable sensors are another alternative including pendants and smart watches, however this approach will require responsibility of the individual.

In this thesis we set out to develop a deep learning anomaly-based fall detection approach that uses mmWave radar sensors to monitor subjects, and develop a back-end system to collect data from the sensors and detect falls in real-time.

Acknowledgements

The work from this topic has been heavily inspired by my wonderful supervisor Dr. Reza Argha. I would like to personally thank Reza for providing me with this topic, support and feedback along the way, and someone who would guide me down the right path especially for the deep-learning section. The mentorship and support received was invaluable.

I would also like to thank my assessor, Dr. Nigel Lovell, for providing critical feedback and guidance along the way. Furthermore i would also like to thank Dr. Michael Stevens for his constant support, he has provided alot of valuable insights onto the topic and created a fantastic atmosphere for weekly meetings. Lastly i would like to thank Ariyamehr Rezaei, he had helped me setup the code base and showed me how to access and use the data required for the model.

Abbreviations

ADL Activities of Daily Living

AE Autoencoder

VAE Variational Autoencoder

SAE Sparse Autoencoder

GAN Generative Adversarial Network

CNN Convolutional Neural Network

WHO World Health Orginization

OSVM One-Class Support Vector Machine

SVM Support Vector Machine

WHO World Health Orginization

URFD UR Fall Dataset

FDD Fall Detection Dataset

Multicam Multiple Cameras Fall Dataset

mmWave Millimeter Wave

OCC One Class Classifier

OCNN One Class Nearest Neighbour

DSCAE Deep Spatio-Convolutional Autoencoder

TFD Thermal Fall Dataset

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Falls are the second leading cause of accidental injury deaths worldwide, where each year an estimated 646,000 individuals die from falls globally [1]. This has led to the need for fall detection based systems in order to provide immediate response, however current methods in place all come with drawbacks.

Phone based and smartwatch based fall detection have risen in popularity however in research conducted by *Luque et al.* [2] it was shown that the rate of battery consumption whilst running complex algorithms was a major drawback. These methods also place responsibility on the user by making sure they always have their devices on them at all times, and that these devices are charged.

Camera based fall detection have been another popular choice for fall detection, especially with the prominence of CNN's. These methods however are supervised learning techniques that require a large amount of data. Real fall data is often unacquariable, as shown in research by *Khan et al.* [3] falls are rare events and simulated falls don't encapsulate all aspects of a fall.

Privacy concerns also arise when it comes to camera based fall detection. Data required for camera based fall detection tend to be a sequence of frames that is stored in video form. this effectively puts individuals under 24/7 surveillance where their activity data is being stored by some third-party. [5] has shown that one of the largest drawbacks of camera based recognition is the major public opinion that this technology causes a threat to individual privacy.

Currently the biggest issue in the field of fall detection technology using Machine Learning has been data gathering. Research performed by *Schwickert et al.* [4] has shown that 94% of data gathered for supervised learning techniques, and validating models for semi-supervised learning techniques has been done through the use of simulated falls. Amongst this, according to research by *Khan et al.* [3] there is still no standard for fall detection that has been shown to perform consistently well.

Due to these issues this thesis aims to develop a privacy-ensuring deep learning based model to detect human activities and identify falls as anomalies using mmWave sensors. The model developed during this thesis will also run in real-time to detect falls and alert an individual upon detection. If this model performs at a high specificty in real-time with a low count of false alarams, then this will be taken to facilities in VitalCare for use.

Chapter 2 sets out to explain in further detail; the motivation behind the topic choice, previous work conducted relating to fall detection, the reasoning behind mmWave sensors, and describing the problem statement. Chapter 3 sets out to describe, functional and non-functional requirements, use cases of the system, and the intended design and structure of the final model. Chapter 4 sets out to outline, work that was performed during this term so far and the project timeline and scheduling for Thesis part B & C.

Chapter 2

Background

37.3 million falls occur annually that are severe enough to require medical attention [1]. Elderly people in particular tend to suffer the largest number of falls that are fatal. According to [7] 1 in 4 people over the age of 60 experience a fall anually, and 1 in 3 people over the age of 65 experience a fall annually. Falls remain one of the leading reasons for elderly people being admitted to the hospital at a rate of 38%, and the second largest cause being vehicle injuries at a rate of 13%.

The health direct [6] state that falls usually occur due to gradual changes in our bodies that make walking difficult, or having hazards inside of the current environments. The changes that occur due to aging tend to be balance issues, weakining in muscles that make it harder to walk, degradation in eyesight and slower reaction times.

Falls have a serious risk involved, the most common injuries being fractures to the thigh and hip at 38%, followed by injuries to the head at 20%. Falls cause more injury deaths in Australia than car collisions. To help visualise the impact of falls, A Western Australian dies every 26 hours, is admitted to a hospital every 19 minutes and presents to an emergency room every 12 minutes due to falls. According to the CDC [9], amongst people over the age of 65, $50 billion is spent on medical costs related to non-fatal fall injuries, and $754 million is spent related to fatal falls. These costs cover hospital feels, doctor checkups, rehabilitation, medical equipment, prescripted drugs and insurance.

These issues have led to the requirement for fall detection technology in order to acquire immediate response. If an individual has fallen over and is rendered unconscious they are at great risk and are unable in any way to seek help. Fall detection is performed in modern day and age through a variety of different techniques, illustrated in Table 2.1.

Fall Detection Method			
Technique	Type	Machine Learning Type	category
push button alarm system	wearable technology	N/A	non-automated
mobile phone/smartwatch	wearable technology	N/A	automated
wearable pendants	wearable technology	supervised learning	automated
camera based monitoring	surveillance	supervised learning	automated
IR monitoring	surveillance	semi-supervised learning	automated
mmWave monitoring	surveillance	semi-supervised learning	automated

Table 2.1: Fall detection techniques established in current date.

Research in the field of fall detection can be categorised into two major categories, automated and non-automated. The next section will outline the groundwork for both methods.

2.1 Non-Automated Fall Detection

2.1.1 Push Button Alarm Systems

Push button alarm systems simply work by sending an alarm when an individual presses the button on their supplied wearable technology. Typical push button alarms are displayed below in figure 2.1.

Upon falling the individual would press the button located on the device to signal for an alert. The location of the device is free to choose for the individual, as all responsibility for the use of this device is placed on the individual.

Figure 2.1: A typical wearable push button alarm worn around the neck. (SureSafe, 2021)

These methods however have been shown in studies from *Fleming et al.* [10] to be often ineffective at dealing with more dangerous fall. In the year of investigation on 265 fall reports, 82% of falls occurred when the individual was isolated. From the 60% of people who had fallen, 80% of them were unable to get up afterwards and 30% had laid on the floor for over an hour. A large portion of the study population were using push button alarms, however barriers of use arose. If an individual is unconscious after falling, they are unable to activate their alarm. There also arose the issue of many individuals who had a device available but were not wearing it during the time of their falls, this was often due to forgetfulness or the individuals choosing not to wear their alarms.

2.2 Automated Fall Detection

Automated fall detection attempts to takes responsibility off the user by automating the process of sending the alarm. This approach can be broken down into two different categories, sensor based approaches and machine learning based approaches.

2.2.1 Sensor Based Fall Detection

Sensor based fall detection is the evolution of push button alarm systems, they work in a similair regard of being worn by the individual, however algorithms in place can automatically detect falls and alert an emergency contact or carer. Typically these systems are in place on smartphones, smartwatches or pendants. [11] The way these systems work is through motion detection through the use of an accelerometer and a gyroscope located inside the devices. The sensors can detect everyday activity and are designed to detect the motion of a fall. The accelerometer measures the change in velocity divided by time, this is actually the technology that changes screen orientation on a phone when it is rotated. The gyroscope is a device used to determine the orientation using the earth's gravity, allowing the tracking of rotational information. Custom algorithms are in place to accurately sense when falls take place. Figure 2.2 shows the general flow of this fall detection methodology.

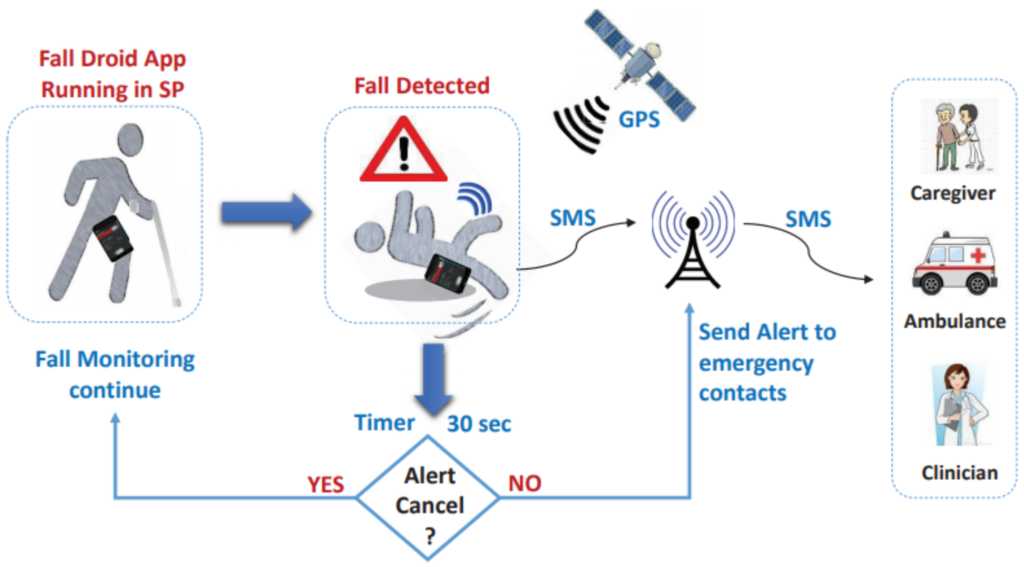


Figure 2.2: Fall detection flow for wearable technology. (Shahzad et al., 2018)

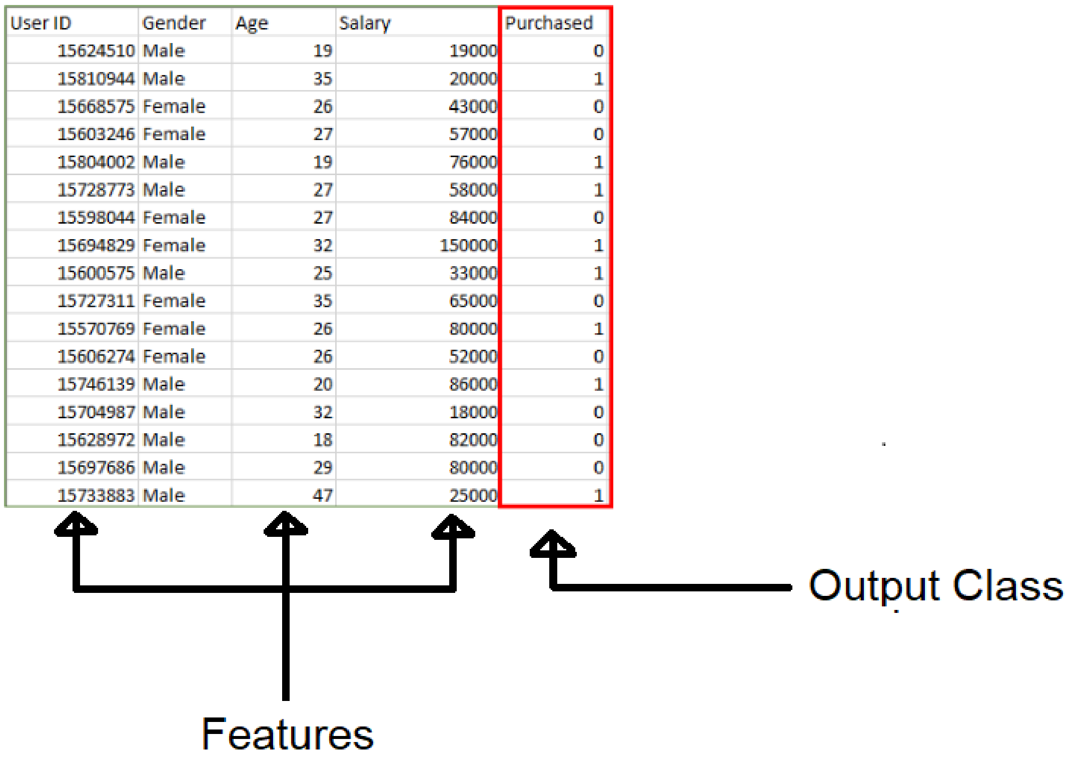
These fall detection methods are a significant improvement to push button alarm system as they take a majority of the responsibility off the user, however they still require the user to always have the devices on them and making sure their device is charged. In particular the pendants require the user to be in proximity to the device that sends the alert.

2.2.2 Machine Learning Based Fall Detection

Machine learning fall detection is the recent development of fall detection technology. Machine learning is the methodology of learning through examples, and attempting to classify new examples based on what was learnt during training. We can often categorise machine learning into three major sections, supervised learning, semi-supervised learning and unsupervised learning. The next sections will outline the different approaches to supervised and semi-supervised learning.

2.2.3 Supervised Learning

Supervised learning is a branch of machine learning that learns how to fit incoming example data to a certain class output. This can also be seen as predicting an output class based on the incoming data. A very simple example would be a model that predicts whether a person earns a salary over $50,000 based on the person's occupation, age, and other features that describe the person. This process however does require data in order to train up a model, and the data must be labelled. Labelled data is where each feature is defined for the example data input. Figure 2.3 displays an outline of data used in supervised learning. From the figure, gender is the first feature, age is the second feature and salary is the final feature, and this combination is used in order to predict the output class which is purchased (whether or not the person has purchased the item).



User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure 2.3: Labelled data for a simple supervised learning classifier. (GeeksForGeeks, 2020)

For fall detection, typically the input would be a video stream passed into a model, where each frame of the videos pixel data is split into 3 colour channels for RGB, and other times the data would be motion information. The following section will explain the types of models used for supervised learning for fall detection.

Convolutional Neural Networks

In the case of fall detection using convolutional neural networks (CNN), many approaches can be taken. The first approach is to take the optical flow of an image to capture motion history, and using a 3D CNN combined with transfer learning. Another approach would be to use a 3D CNN, and stacking multiple frames together as input for the CNN. Finally another approach is to use mmWave data in a deep 2D CNN.

In a study performed by *Ji et al.* [14] 3D CNN's were used to detect human activities on TRECVID data which consists of 49 hours of video footage from London airport. The study had focused on recognising three different human actions, putting a cellphone to the ear, pointing and placing an object somewhere. Alongside having data on these cases, a large amount of negative samples were placed in where none of these actions were present in the samples. The model used for this study is shown below in figure 2.4.

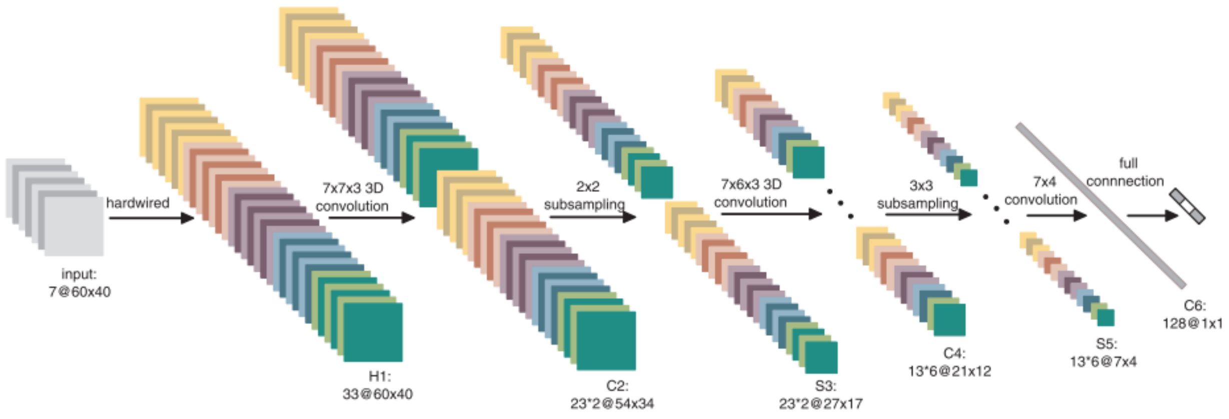


Figure 2.4: CNN structure for the study on human activity recognition. (Ji, 2013)

Whilst this study did not directly detect falls, the extension is that human activity recognition can be extended to also include falls amongst the activities listed and potentially even more. The results from this study had shown that under a false positive rate of 0.1%, the model had on average 71.48% precision, 2.43% recall and an AUC value of 0.0139. Using a false positive rate of 1% the model had an average 55.84% precision, 11.47% recall and an AUC value of 0.6993. This approach had outperformed all other state of the art models such as 2D CNN frame-based model and spatial pyramid matching SVM's. It is worth mentioning however that for the results, the CellToEar precision and recall was typically much lower than ObjectPut and Pointing. This could possibly be due to the action of bringing a cellphone to the ear being much more complex than simply putting down an object or pointing at something. It is also worth mentioning that for fall detection, having as high specificity as possible is the main priority, misclassifying a fall as a false negative could be the difference between life and death. Due to the low accuracy of this model it would be unsuitable for fall detection, especially when other models have been shown to perform with much higher accuracies.

In another study performed by *Núñez-Marcos et al.* [15] 3D CNN's were used to detect falls on simulated fall data through the use of transfer learning. This study had first converted each video frame into an optical flow image, and the horizontal and vertical vector field were seperated and passed into the CNN as a stack simijair to the previous study. The model used for this study is displayed below in figure 2.5, and is very deep with many convolving layers, and finally 3 fully connected layers that are then softmax'd in order to predict the binary classification if the example was a fall or not.

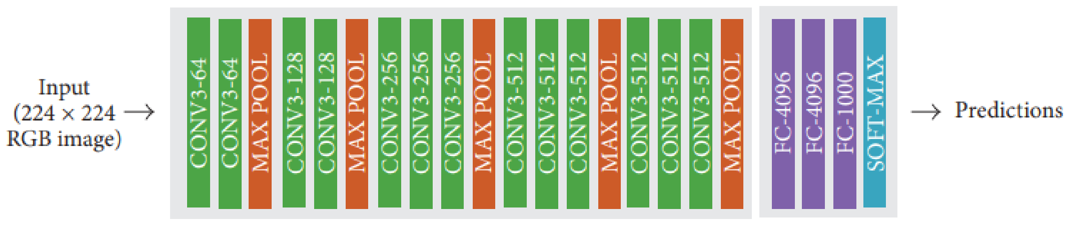


Figure 2.5: CNN structure for the study on fall detection using optical flow. Keep note the input is the stacked horizontal and vertical optical flow images. (Núñez-Marcos, 2017)

Transfer learning was used in order to account for the lack of public fall data, first the CNN was trained on the Imagenet images dataset. Next the model was trained on the UCF101 action recognition dataset, using optical flow images for input. Finally the CNN was frozen during training (no weight updates) in order to fine tune the final fully connected layers. A sequence of frames were extracted using a sliding window approach to extract the frames where fall or no fall sequences occur.

A sliding window approach was used in order to bunch incoming sequence of frames to perserve the motion history within the sequence of frames. The specific approach taken was to use a step of 1 which makes sure that no motion history is lost, however this causes training and classification to be slower. Figure 2.6 displays a visual representation of the sliding window approach. It would be interesting to further research the performance vs efficency trade-off between different step sizes for a sliding window approach.

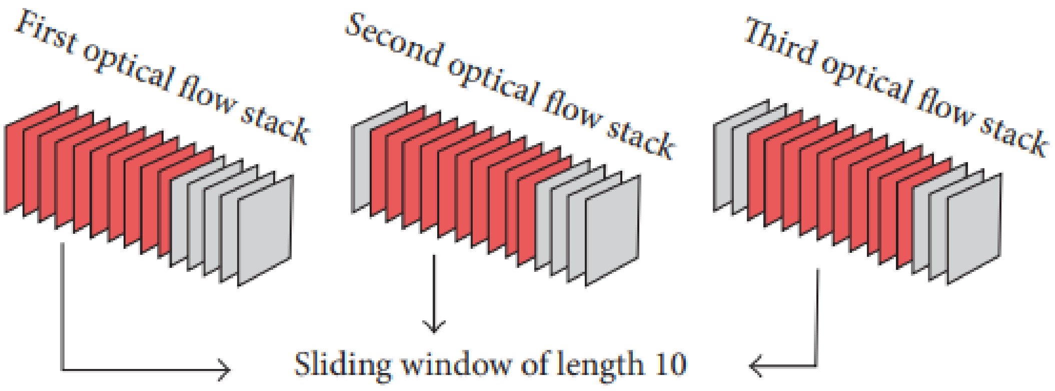


Figure 2.6: Sliding window approach used to maintain motion history. (Núñez-Marcos, 2017)

The result of this study had shown that it was difficult to learn how to classify the ”fall” class in particular, attempts were made to improve the importance of the fall class such as adding bias to the fall class in the learning, and whilst this proved to greatly increase the classification of falls, it did come at a cost of more false positives. Three different fall datasets were used for testing, UR fall dataset (URFD), fall detection dataset (FDD) and multiple cameras fall dataset (Multicam). the model had shown extremely high sensitivity which is the measure of how good the system predicts falls, and the model has shown high specificity. On the URFD dataset the model had a 100% sensitivity rate, and 94.86% specificity rate. On the FDD dataset the model had a 93.47% sensitivity rate, and 97.23% specificity rate. On the Multicam dataset the model had a 98.07% sensitivity rate, and 96.20%. This method produces extremely high accuracy and would be a suitable fall detection model, however it does come with the issue of privacy concerns. Figure 2.7 shows the data involved with this type of classification, and this flags privacy concerns as this data is stored by a third-party, constantly monitoring individuals.

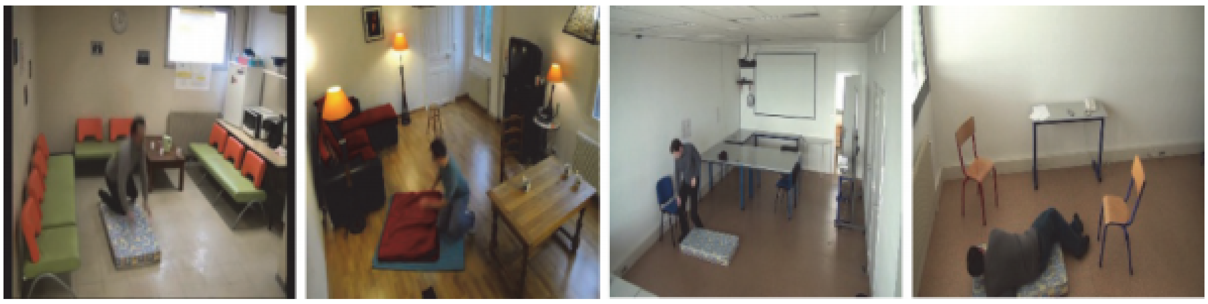


Figure 2.7: Camera feed footage for the proposed CNN model. (Núñez-Marcos, 2017)

Jin et al. [15] performed a study to detect patient activity, primarily fall detection using millimeter wave (mmWave) Radar and Deep CNNs. This model developed was able to distinguish patients, and perform activity recognition on all the patients. The model would first detect the multiple patients from the radar using custom built algorithms. Doppler features for each patient is then passed into a deep CNN. This model is shown below in figure 2.8

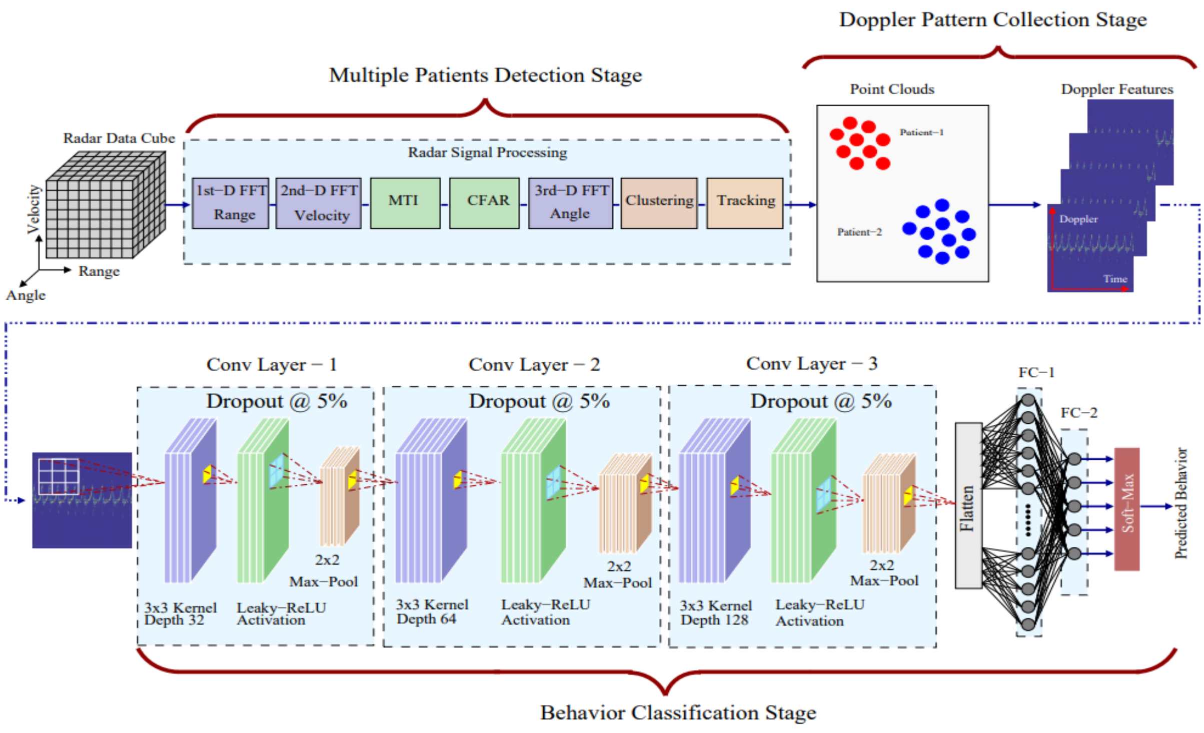


Figure 2.8: CNN structure and data extraction used for proposed model. (Jin, 2019)

The model in this study was able to classify behaviour into the following categories: walking, falling, swinging hand for help, seizure and restless movement. The data used for this study was simulated falls with only one patient.

One issue that arose with this method of fall detection was the requirement to define a strong dataset that covers all kinds of situations, as when a fall was in another direction, the fall was misclassified.

The result of this study was based on the number of patients in the frame. when performing behaviour prediction on one patient, specifically for falling, the inference accuracy was 84.49%, however adding another user to the frame had significantly decreased the inference accuracy to 66.02%. This is likely due to the increased complexity of the task to classify seperately multiple people, and possibly deepening the network or training for more epochs may be needed to improve this accuracy.

A major advantage of this approach was the ability to maintain privacy of individuals due to the data used for classifcation. No video footage is taken from subjects, and subjects are displayed as a bunch of points, illustrated in the data collection portion of figure 2.8.

Data Availability Issues for Fall Detection

In the field of fall detection using machine learning, acquiring fall data has been a major challenge. In a study conducted by *Schwickert et al.* [4] it was shown that 94% of studies use simulated falls for training models, validating models or both. This is likely due to the fact that fall data is rare, as falls are rare events, and simulating a fall cannot necessarily encapsulate all aspects of a natural fall.

For supervised learning techniques this can be problematic for a few reasons. If a model has only been trained to work on simulated falls it may overfit during training and perform poorly in a real world event, falls from healthy young subjects are not the same as a fall from an elderly person [3]. Another issue is that supervised learning methods require a large amount of data for training in order to make sure the model does not overfit to the training dataset, this is especially true in more complex models that must learn more weights. The data provided must also be evenly distributed in output classes to make sure the model is not biased towards a specific output class. Due to the rarity of falls, it would be unrealistic to train a supervised learning model with real fall data due to the lack of real falls present. This is less problematic for anomaly detection methods as they primarily use ADL data, however when it comes to testing the models, often simulated falls are used.

2.2.4 Semi-Supervised Learning

Semi-supervised learning is a branch of machine learning that combines a few labelled data examples with a bunch of unlabelled data, and attempts to group similair data together based on known information from the labelled data. This approach is powerful as it combines benefits from both unsupervised learning and supervised learning.

The key benefit to semi-supervised learning is that it removes the need to perform data labelling on a majority of the data collected, which can be cumbersome, exhausting and time consuming [3].

A typical approach to semi-supervised learning for fall detection, is through the use of anomaly detection models. These models are trained on activities of daily living (ADL) data, rather than fall data. When the model encouters data it is unfamilair with such as falls or anomalies, then the loss function spikes indicating an anomaly. The next section will go into more detail on anomaly detection models.

One Class Classifiers

One class classifiers (OCC) are a method of machine learning that focuses primarily on classifying a single class, and treating any other output as the same. Typically these methods employ traditional machine learning techniques such as K-Nearest Neighbour (KNN) or Support Vector Machines (SVM).

In a study performed by *Medrano et al.* [22] falls were detected as anomalies through the use of a smartphone. ADL data on ten volunteers was recorded, including 8 type of falls. two types of one class classifiers were compared to determine which outlier model performed better. The better model was then compared to state of the art supervised learning models.

The results of this study had shown that one class nearest neighbour (OCNN) had performed best with an AUC of 0.9554, whereas one class SVM (OSVM) had an AUC of 0.9439. However when compared to state of the art SVM models, it had performed worse, where OSVM had an AUC of 0.968. This is likely due to the rate of false alarms that occur with an outlier detection model.

In another study performed by *Yu et al.* [23] falls were detected using an online OSVM. Video data frames were used for this study, first the background of the frame would be subtracted, then feature extraction would be performed, and the processed input would be sent into the OSVM for classification. Further algorithms were in place to reduce the rate of false positives, such as checking the duration spent on the floor and the amplitude of the movement.

The result of this study had shown extremely high accuracy with a 100% true positive rate and a 3% false positive rate. However the data used for this method puts user privacy at risk, since camera feed is used.

Generative Models and Autoencoders

Generative models are a section of machine learning that attempt to either create or recreate data from given inputs. There are two main models used for generative models, variational autoencoders (VAE) and generative adversarial networks (GAN).

GAN networks typically would not be used for the task of classification, as the aim of this network is to fool the critic in the actor-critic scenario and convince the critic the output generated is real data.

For the task of fall detection VAE networks can perform exceptionally well. VAE models provide the ability to reconstruct data from input data and compress data. the aim of VAE models is to reproduce the input as close as possible based on the knowledge learnt during training. The output of VAE models is a classification with a probability likelihood as shown in figure 2.9. This makes VAE networks exceptional at anomaly detection, by using a threshold confidence value to determine if an input is an anomaly.

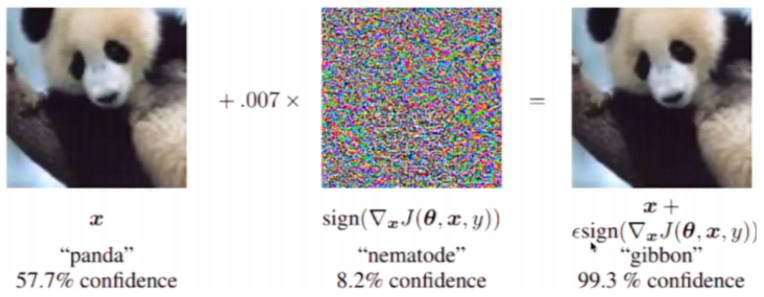

+ .007 ×
=
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 2.9: The output of a generative adversarial network (Goodfellow, 2015)

The structure of a typical VAE model is shown below in figure 2.10. Image data input is fed into the model, and the model begins to encode the image by compressing the data to only represent the important features which are learnt during training. At the bottleneck which is the red box on the figure, this is where the final encoded data representation of the input is located, this specific image went from 4 features down to 2 features. The generative section of the model comes after the bottleneck where the model attempts to reconstruct the original input image from the encoded input.

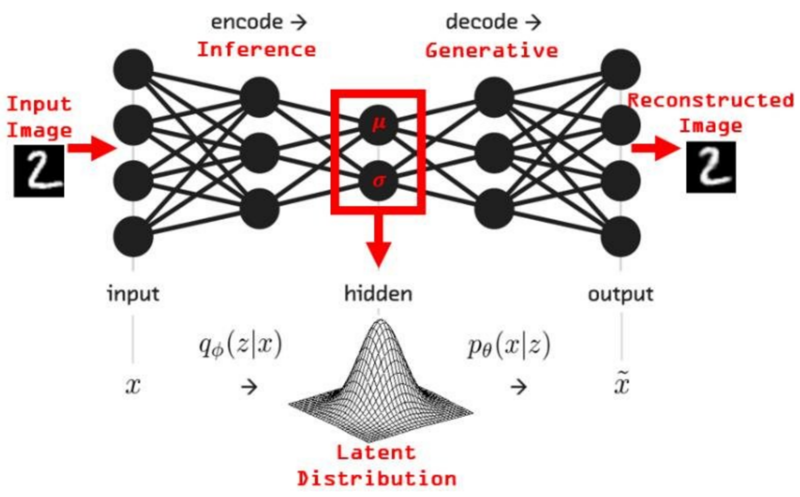


Figure 2.10: The basic structure of a variational autoencoder (A. Al, 2020)

Sparse autoencoders (SAE) are a type of autoencoder that follow the general autoencoder structure, however the key difference is in how the hidden activations work. In an SAE they can include equal or more nodes at the bottleneck, however only a small number of hidden units are allowed to be active at the same time, creating artificial dynamic bottlenecks. This sparsity constraint forces the model to respond to unique statistical features of the training data, where nodes specialise in categorising particular features. One key advantage from this type of autoencoder is that it has improved performance on classification tasks. Figure 2.11 displays a visual representation of a SAE.

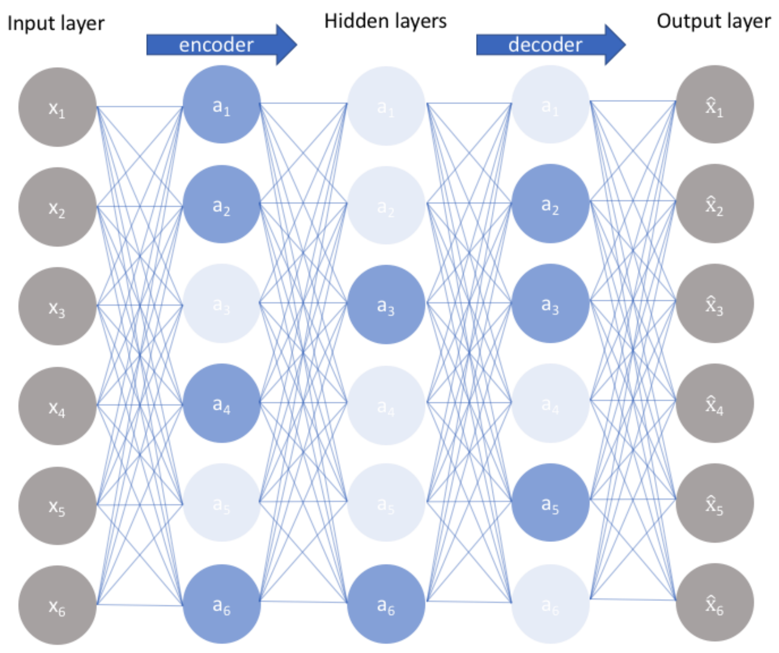


Figure 2.11: Visual representation of a sparse autoencoder, light blue nodes are inactive (Z. Syoya, 2018)

In a study performed by *Jin et al.* [21] VAE networks alongside LSTM networks were used in order to detect falls using mmWave radar data. The study had first processed mmWave data performed on ADL in order to capture doppler features and centroid data for the input frame. This data would then be input fed into a hybrid model that modifies the typical bottleneck of a VAE model.

Rather than employing the use of a typical bottleneck with hidden layers, a sequence to sequence model is used instead in order to capture history motion data. The model used is shown in figure 2.12.

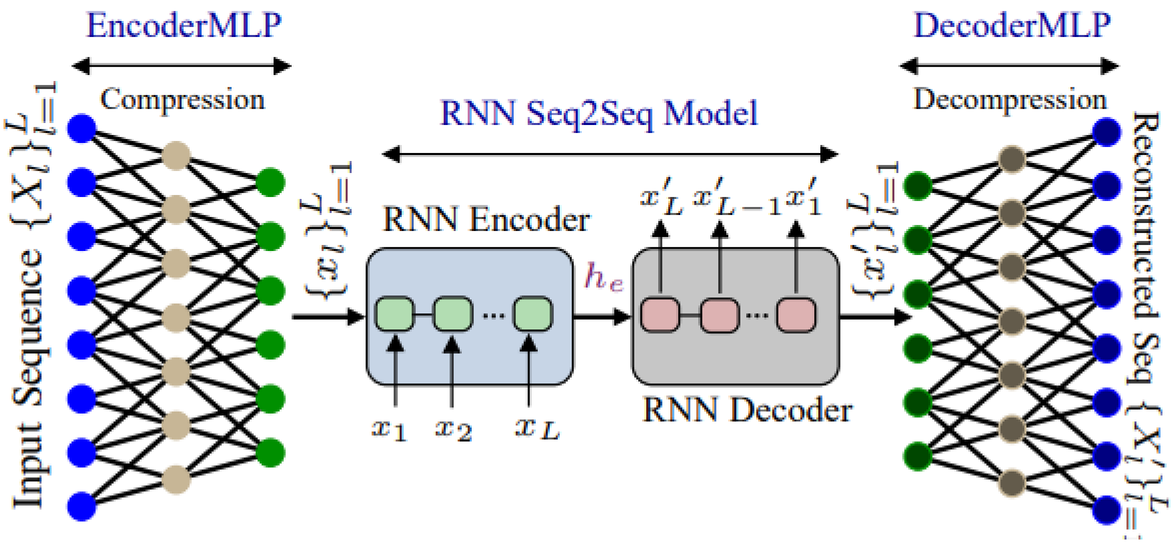


Figure 2.12: Variational autoencoder with the use of LSTM (Jin, 2020)

The model would output the likelihood of the image reconstruction, and when the model output was below a threshold value, this would indicate an anomaly had occurred. Centroid information is used to further investigate anomalies and reduce false positive rates. When the centroid had decreased in height rapidly, alongside detecting an anomaly this would indicate a fall has occurred.

To test the model used in this study, 50 simulated falls alongside negative samples were created, and passed into the model for classification. The result of this study had shown that it had achieved a specificity of 98% (49/50) and only 2 false alarms were triggered amongst the negative samples. This method proves extremely promising as it maintains the privacy of the subjects, whilst also providing high specificity. However this method has only been tested on simulated falls which often times cannot represent a real fall situation, and the model has not been tested or designed to work in a real-time environment.

In another study performed by *Nogas et al.* [24] a deep spatio-convolutional autoencoder (DSCAE) was used to detect falls as anomalies. Video frames stacked together were used as input to the network, where 3D convolutions take place for encoding inputs. Three convolutional layers are used to encode the 8x64x64x1 data into 2x8x8x8 data. Encoded inputs are then up-sampled and sent through further convolutional layers in order to reconstruct the encoded inputs. Figure 2.13 illustrates a simplified representation of the network structure.

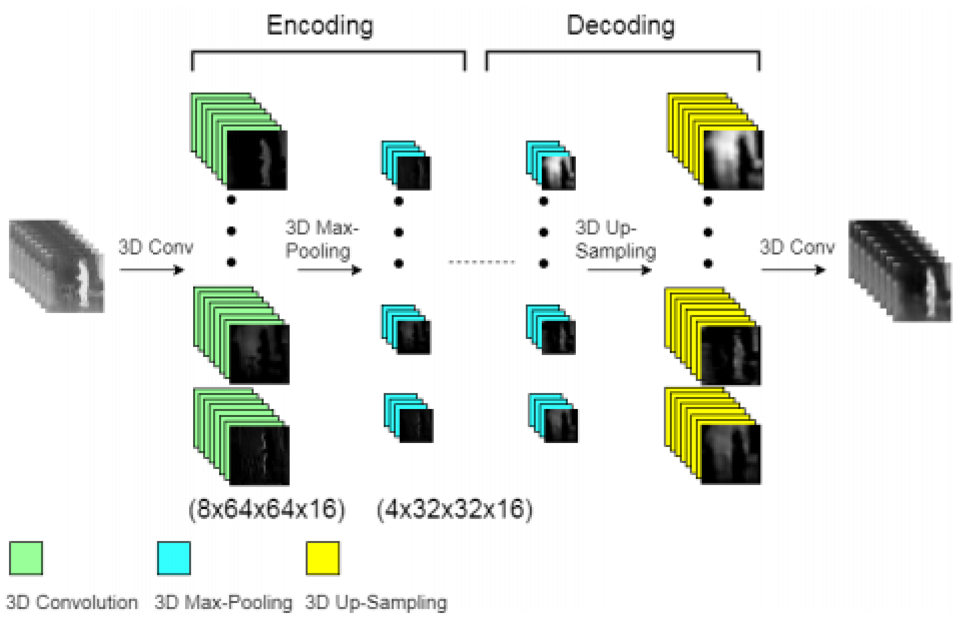


Figure 2.13: deep spatio-convolutional autoencoder structure (Nogas, 2020)

The model uses a novel threshold system in order to determine anomalies. Rather than computing an anomaly score on a frame by frame basis, the model computes an anomaly score for each frame alongside reconstruction error information in adjacent video frame stacks. This was shown to perform better on real world scenarios.

To test the model three different public datasets were used, thermal fall dataset (TFD), URFD and the SDU dataset. The result of this study shows that this model outperforms other autoencoder methods and other machine learning approaches. the AUC of the model on the three datasets averaged 0.91 whereas a traditional deep autoencoder approach had an average AUC of 0.78 and a K-nearest neighbour approach had an AUC of 0.58.

2.3 mmWave Data

Millimeter-wave radar is a new sensing technology that allows for the detection of objects. These radars provide information regarding range, velocity and angle of these objects. mmWave is a contactless technology that operates between 30GHz and 300GHZ. The use of small wavelengths allows sub-mm range accuracy and is able to penetrate through materials such as drywall, plastic or clothing [25, 26]. Another key feature to mmWave sensing technology is the ability to preserve user privacy whilst maintaining motion and positional information. Figure 2.14 shows the processed data from a mmWave radar alongside a camera view. On the left of the image you can see the person has fallen, however on the right we see a point cloud representation of that motion history, abstracting the individual from the motion entirely.

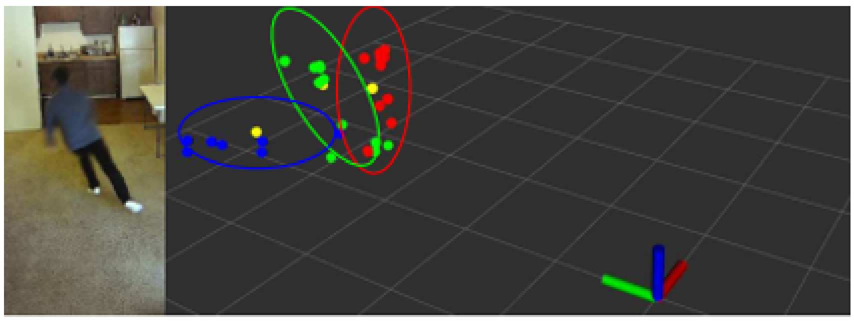


Figure 2.14: Data captured from mmWave sensor visualised. (Jin, 2020)

2.4 Problem Statement

Previous studies have shown the difficulty in detecting falls in real time, often times a study would excel in one aspect of fall detection but present drawbacks that cannot be ignored. Supervised learning models would present a high specificity, however this is done through the use of simulated falls and often overfit when it comes to real world scenarios. Supervised learning models also present the concern for privacy as often camera feed data is used, placing subjects under 24/7 surveillance. Due to the lack of data available, these models cannot be trained with real fall data.

Semi-supervised learning approaches seem to do an excellent job when it comes to producing models that obtain high specificity and do not require fall data for training. Whilst these models are almost ideal, they still use simulated falls to validate the dataset, and are not designed to work in real-time. Another issue discovered in the study was that determining a threshold value that minimises false alarms whilst maximising the correctly classified true positive case was a difficult task.

In this thesis, the goal is to create a generative fall detection model that is able to detect falls as anomalies in realtime, whilst maintaining the subject's privacy and achieving a high specificity on real falls, with a low rate of false positives. Privacy will be maintained as mmWave radar data will be used (see figure 2.14). This model will also work 24/7 without placing any responsibility on the subject.

Tensorflow and Keras API will be used in order to develop the model planned. Edge computers will run locally to collect and process data, and pass it onto a pretrained model that is able to detect falls in real-time.

Chapter 3

Requirements, Use Cases and Testing

In this chapter, the aim is to outline the requirements of the project, use cases and testing methodology. Further changes will be made in thesis part B & C upon discovering new requirements, or evaluating feedback from testing.

3.1 Requirements

The following requirements are done in correspondence to MoSCoW requirement structure:

- P1 (Priority 1, must have)
  - All these requirements form the MVP and are crucial for launch
- P2 (Priority 2, should have)
  - All these requirements should be done within the time frame but are not crucial to launch and can be delayed to further releases

- P3 (Priority 3, could have)
  - All these requirements are featured we would like but in the time frame we would delay these to focus on priorities 1 and 2
- P4 (Priority 4, would have)
  - Features we do not expect to have but show potential extensions of our product.

1. **I will create an algorithm that is able to detects falls.** [**P1**]

   1.1. I will create a deep learning model that is able to detect falls. [**P1**]

   1.1.1. I will create a deep learning model that is able to detect ADL [**P1**]

   1.1.1.1. I will define ADL as a single output class and abnormalities as the other output class. [**P1**]

   1.2. I will create a generative model that is able to compute the probability of an event being generated with the input given. [**P1**]

   1.2.1. I will create a function that is able to under sample abundant ADL data and oversample lacking fall data. [**P1**]

   1.2.2. I will create a model that contains A Variational Autoencoder to encode inputs. [**P1**]

   1.2.3. I will create a model that connects the encoder from requirement 1.2.2 to a LSTM to process the encoded inputs. [**P1**]

   1.2.4. I will create a model that connects the outputs from requirement 1.2.3 to a decoder to determine likelihood of reconstruction. [**P1**]

   1.2.5. I will set a threshold to trigger for abnormality based on the output of requirement 1.2.4. [**P1**]

   1.3. I will create a function that is triggered on requirement 1.2.5 that will check more detailed information on the abnormality. [**P1**]

   1.3.1 I will create a function that determines the velocity and difference in centroid height and compares to a set threshold. [**P1**]

1.3.1.1. I will create a function that triggers an alert from requirement 1.3.1. [**P1**]

2. **I will collect data from mmWave Radars.** [**P1**]

   2.1. Data will be processed locally. [**P1**]

   2.1.1. mmWave radar data will be cleaned to remove outliers, anomalies and incorrect camera values. [**P1**]

   2.1.2. mmWave radar data will be grouped to give XYZ coordinates, Doppler information. [**P1**]

   2.1.3. mmWave radar data will be processed to distinguish multiple people in a frame. [**P3**]

   2.2. Data will be collected locally on edge computers. [**P1**]

   2.3. Processed data will be sent to the model in real-time to detect falls. [**P1**]

   2.3.1. Processed data will be sent to the model over the cloud and the output of the model will be sent to an azure function. [**P1**]

3. **I will create a backend system to detect falls as abnormalities in real-time.** [**P2**]

   3.1. I will create a backend system that works with requirement 2 in order to collect data. [**P2**]

   3.2. I will create a backend system to pass the collected data to the model in requirement 1. [**P2**]

   3.3. I will create a backend system that uses the alert from requirement 1.3.1.1 to alert a carer. [**P3**]

   3.3.1. I will create an azure function that is triggered upon requirement 1.3.1.1. [**P3**]

   3.3.1.1. I will create a function that sends an alert to a caretaker via SMS. [**P3**]

4. **I will create a model that is able to detect other dangerous events such as seizures alongside requirement 1. [P4]**

   4.1. I will create further algorithms similar to requirement 1.3.1.1 that will be able to distinguish between different anomalies. **[P4]**

   4.2. Using the same system as Requirement 3, correct emergency contacts will be notified when specific abnormalities occur. **[P4]**

3.2 Use Cases

The following use cases cover all priority 1 requirements as well as some priority 2 requirements

Use Case 1	
Requirements	1.3, 3.3
Outline	An alert is sent to an azure function that sends an SMS or phonecall to emergency contact upon fall detection.
Users	emergency contacts, falling individual, internal software.
Overview	Upon detecting a fall the system will send an alert to an azure function to signal that an individual has fallen over, the azure function will be designed to contact the emergency contact of choice for the individual.
Trigger	individual has fallen over.
Precondition	The model has returned that an individual has fallen.
Postcondition	The emergency contact has been notified and alerted.

Use Case 2	
Requirements	2
Outline	Data from mmWave radars are processed locally.
Users	internal software.
Overview	Upon installation of the system, mmWave radars begin automatically sensing objects nearby. The data collected is sent locally to an edge computer where data processing occurs. Data is cleaned and to remove anomalies and outliers and a python script begins processing the incoming data. information regarding the coordinates, doppler and centroid are extracted and sent to the model in real time.
Trigger	The system has been integrated.
Precondition	The equipment is satisfactory and all hardware connections have been made.
Postcondition	Processed data is sent to the model in real time.

Use Case 3	
Requirements	1.2
Outline	Processed data is sent to the model in real time to classify each batch of frame inputs as a fall or not fall.
Users	Internal software.
Overview	The model has been trained and receives data in batches of frame using a sliding window approach, upon detection of an abnormality from the batch, the model will perform further fall detection algorithms such as determining the motion of the subject in frame sequence.
Trigger	None
Precondition	The model has been pretrained and successfully integrated.
Postcondition	If a fall is detected, an alert is sent in the flow of use case 2. otherwise the frame sequence is thrown away.

3.3 Testing

extensive testing of the system for this thesis project will be conducted to ensure the model works at a high specificity amongst multiple datasets including real fall data if the possibility arises. And to ensure all systems work as expected

First the model will be trained on a large amount of ADL data, and the saved model will be tested. The model will first be ran through local testing to determine specificity, this will be done using existing fall data and ADL data. If the model proves a high accuracy on the local testing, three different fall datasets made public which are, URFD, FDD, Multicam datasets will then be used to further test the model.

If the model shows promise ideally I would like to test it in a real-word scenario, potentially at a facility, alongside other fall detection measures to account for potential failure.

In order to test the azure function in place, unit tests will be created to exhaust all possible scenarios for receiving an alert from the model. These unit tests will employ a mock server to ensure that the handling of responses from the azure function is also appropriate. If for instance the internet connection drops and there is no reply, the system will continuously resend the alert until it receives verification from the azure function. Alongside this cases where emergency contact is unavailable or issues arise will be tested extensively. Unit testing will be performed using PyTest.

3.4 Planned Architecture

To perform this task table 3.1 below outlines more detailed information on the software and technologies for the system, and figure 3.1 and 3.2 outlines a software architecture diagram for the proposed system.

Planned Software Architecture	
Component	Technology
Data collection	mmWave radar and edge computer, sliding window
Data processing	Python3.7 code
Model creation	TensorFlow using Keras, Variational Autoencoders and LSTMs
Fall detection algorithms	python3.7 code
Signalling alerts	azure function
response from azure function handling	python3.7 code

Table 3.1: Planned software architecture

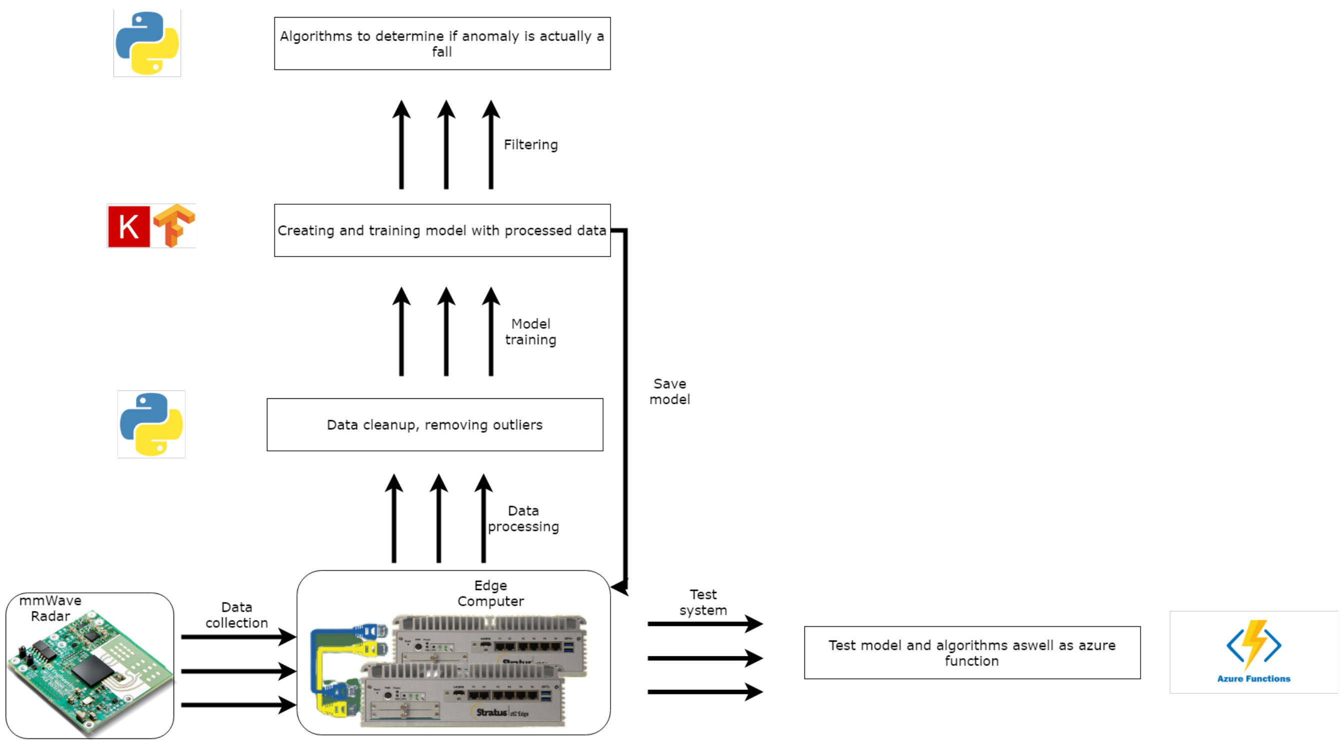

Algorithms to determine if anomaly is actually a fall
Filtering
Creating and training model with processed data
Model training
Save model
Data cleanup, removing outliers
Data processing
Edge Computer
mmWave Radar
Data collection
Test system
Test model and algorithms aswell as azure function
Azure Functions

Figure 3.1: Architecture used for training the model

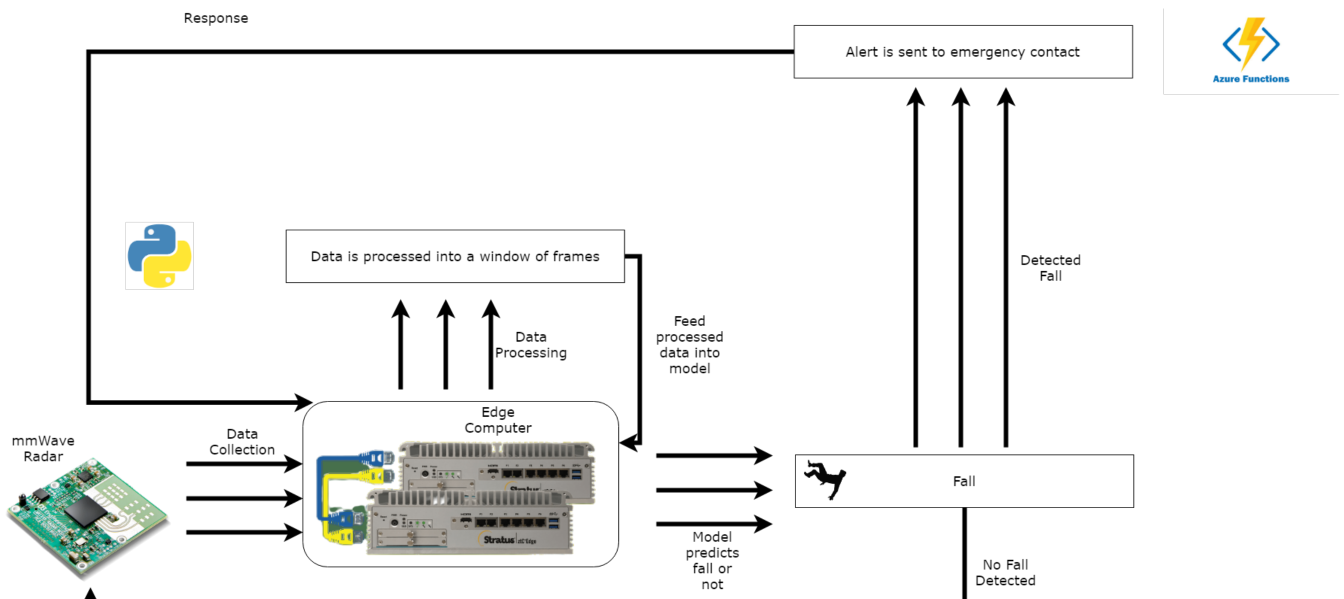

Response
Alert is sent to emergency contact
Azure Functions
Data is processed into a window of frames
Detected Fall
Feed processed data into model
Data Processing
Edge Computer
mmWave Radar
Data Collection
Fall
Model predicts fall or not
No Fall Detected

Figure 3.2: Architecture used for real time detection

Chapter 4

Current work and Future Plans

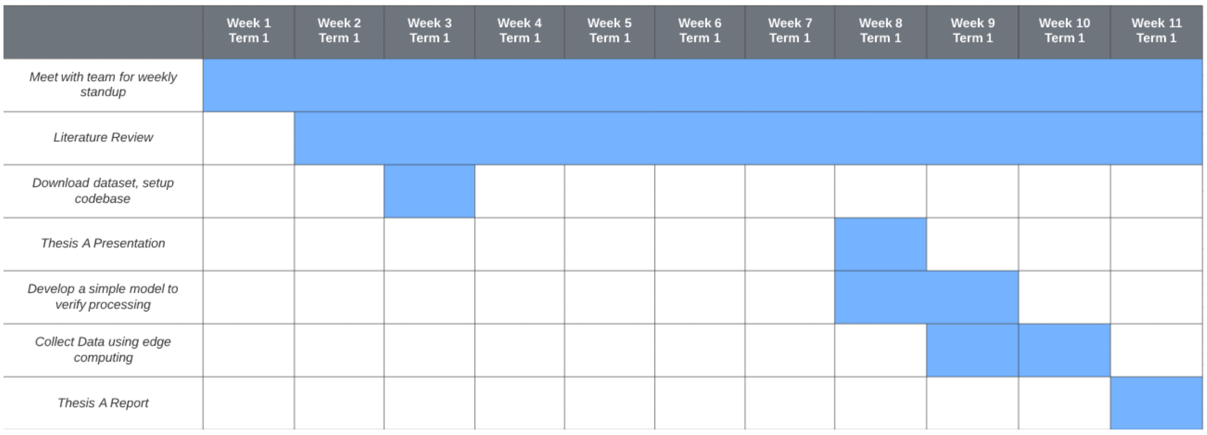
During this term i have done the following:

- Researched exisiting fall detection methodologies and the current gaps in the field.
- Determined what problem was to be adressed.
- Researched the major issue in fall detection, which is the lack of fall data.
- Researched more into depth models that do not rely on fall data.
- Downloaded the existing dataset.
- Setup the codebase to be run locally.
- acquired a katana account for developing machine learning code on a super computer.
- began coding a model mentioned in the paper to see if the approach would be valid for our task.

My current plan is to develop a working real-time model that achieves over 90% specificity and a low false positive rate by the end of thesis B. Thesis B will be mainly focused on the design process where I will code, analyse and test the code, and using the feedback improve the current code.

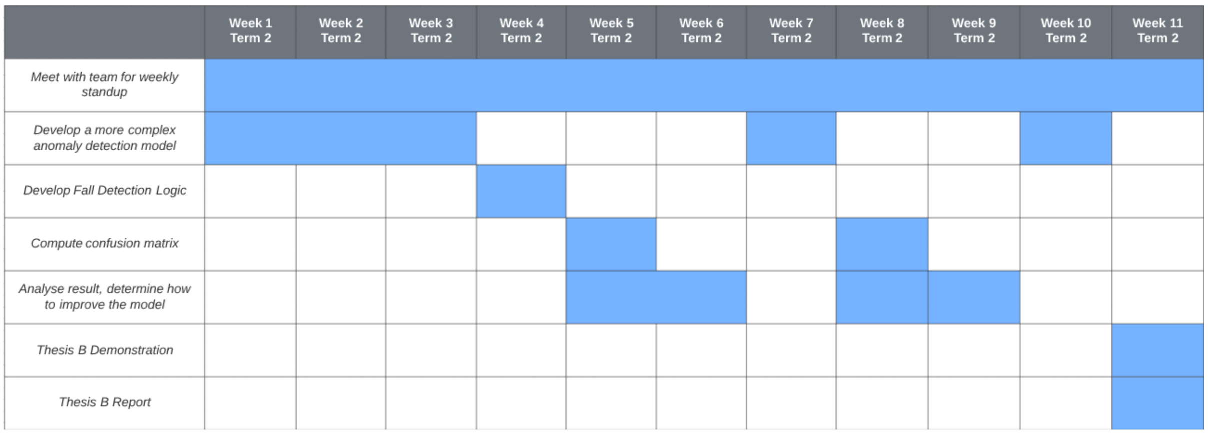
For thesis C my plan will be to improve the model, and begin working on the backend system that accompanies the model.

Figure 4.1, 4.2 and 4.3 show a gantt chart for thesis A, B & C.



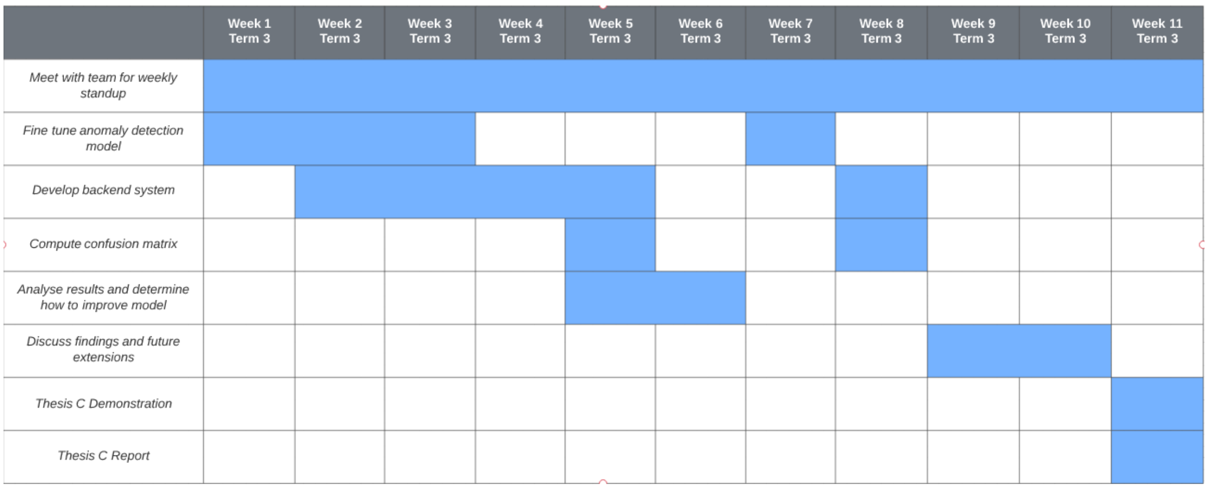
	Week 1 Term 1	Week 2 Term 1	Week 3 Term 1	Week 4 Term 1	Week 5 Term 1	Week 6 Term 1	Week 7 Term 1	Week 8 Term 1	Week 9 Term 1	Week 10 Term 1	Week 11 Term 1
Meet with team for weekly standup	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Literature Review		🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Download dataset, setup codebase			🟦								
Thesis A Presentation								🟦			
Develop a simple model to verify processing								🟦	🟦		
Collect Data using edge computing									🟦	🟦	
Thesis A Report											🟦

Figure 4.1: Gantt chart for thesis A



	Week 1 Term 2	Week 2 Term 2	Week 3 Term 2	Week 4 Term 2	Week 5 Term 2	Week 6 Term 2	Week 7 Term 2	Week 8 Term 2	Week 9 Term 2	Week 10 Term 2	Week 11 Term 2
Meet with team for weekly standup	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Develop a more complex anomaly detection model	🟦	🟦	🟦				🟦			🟦	
Develop Fall Detection Logic				🟦							
Compute confusion matrix					🟦			🟦			
Analyse result, determine how to improve the model					🟦	🟦		🟦	🟦		
Thesis B Demonstration											🟦
Thesis B Report											🟦

Figure 4.2: Gantt chart for thesis B



	Week 1 Term 3	Week 2 Term 3	Week 3 Term 3	Week 4 Term 3	Week 5 Term 3	Week 6 Term 3	Week 7 Term 3	Week 8 Term 3	Week 9 Term 3	Week 10 Term 3	Week 11 Term 3
Meet with team for weekly standup	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Fine tune anomaly detection model	🟦	🟦	🟦				🟦				
Develop backend system		🟦	🟦	🟦	🟦			🟦			
Compute confusion matrix					🟦			🟦			
Analyse results and determine how to improve model					🟦	🟦					
Discuss findings and future extensions									🟦	🟦	
Thesis C Demonstration											🟦
Thesis C Report											🟦

Figure 4.3: Gantt chart for thesis C

Bibliography

[1] WHO, Falls, World Health Organization, 2018, [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/falls

[2] R. Luque, E. Casilari, M.-J. Mor´on, G. Redondo, Comparison and Characterisation of Android-Based Fall Systems, Sensors 14, IEEE, 2014, 18543-18574

[3] S. S. Khan, J. Hoey, Review of Fall Detection Techniques: A Data Availability Perspective, University of Waterloo, 2016, pp.6, 8-9, 12, 19

[4] L. Schwickert, C. Becker, U. Lindemann, C. Mar´echal, A. Bourke, L. Chiari, J. Helbostad, W. Zijlstra, K. Aminian, C. Todd, S. Bandinelli, J. Klenk, Fall detection with body-worn sensors: A systematic review, Zeitschrift f¨ur Gerontologie und Geriatrie, IEEE, 2013, 706–719

[5] B. Marr, Facial Recognition Technology: Here Are The Important Pros and Cons, Forbes, 2019, [Online]. Available: https://www.forbes.com/sites/bernardmarr/2019/08/19/facial-recognition-technology-here-are-the-important-pros-and-cons

[6] healthdirect, Falls and The Elderly, healthdirect, 2020, [Online]. Available: https://www.healthdirect.gov.au/falls

[7] Stay On Your Feet, Falls facts , Stay On Your Feet, 2021, [Online]. Available: https://www.stayonyourfeet.com.au/over60/what-can-cause-a-fall/falls-facts/

[8] R. W. Broadley, J. Klenk, S. B. Thies, L. P.J. Kenney, M. H. Granat, Methods For The Real-World Evaluation of Fall Detection Technology: A Scoping Review, MDPI, 2018, pp. 1-3

[9] CDC, Cost of Older Adult Falls, Centers for Disease Control, 2020, [Online]. Available: https://www.cdc.gov/homeandrecreationalsafety/falls/data/fallcost.html

[10] J. Fleming, C. Brayne, Inability To Get Up After Falling, Subsequent Time On Floor, and Summoning Help: Prospective Cohort Study In People Over 90, BMJ, 2008, [Online]. Available: pp. 1, 3-4

[11] Medical Alert Advice, Is The Apple Watch Series 4, 5 or 6 With Fall Detection Right For You?, Medical Alert Advice, 2021, [Online]. Available: https://www.medicalalertadvice.com/articles/apple-watch-fall-detection/

[12] A. Shahzad, K. Kim, FallDroid: An Automated Smart Phone Based Fall Detection System Using Multiple Kernel Learning, IEEE, 2018, pp. 1-2, 9

[13] G. W. Lindsay, Convolutional Neural Networks As A Model of The Visual System: Past, Present, and Future, J Cogn Neurosci, 2020, pp. 3-5

[14] S. Ji, M. Yang, K. Yu, 3D Convolutional Neural Networks For Human Action Recognition, IEEE, 2013, pp. 221-230

[15] A. Núñez-Marcos, G. Azkune, I. Ignacio Arganda-Carreras, Vision-Based Fall Detection With Convolutional Neural Networks, Hindawi, 2017, pp. 221-230

[16] J. Arunnehru, G. Chamundeeswari, S. P. Bharathi, Human Action Recognition Using 3D Convolutional Neural Networks With 3D Motion Cuboids In Surveillance Videos, ScienceDirect, 2018, pp. 472-477

[17] F. Jin, R. Zhang, A. Sengupta, S. Cao, S. Hariri, N. K. Agarwal, S. K. Agarwal, Multiple Patients Behavior Detection in Real-time Using mmWave Radar and Deep CNNs, IEEE, 2019, pp. 1-5

[18] L. Palmerini, J. Klenk, C. Becker, L. Chiari, Accelerometer-Based Fall Detection Using Machine Learning: Training and Testing on Real-World Falls, MDPI, 2020, pp. 2-15

[19] I. J. Goodfellow, J. Shlens C. Szegedy, Explaining and Harnessing Adversarial Examples, ICLR, 2015, pp. 1-6

[20] Abacus.AI, The Intuition Behind Variational Autoencoders, Abacus.AI, 2020, [Online]. Available: https://medium.com/@realityenginesai/understanding-variational-autoencoders-and-their-applications-81a4f99efc0d

[21] F. Jin, A. Sengupta, S. Cao, mmFall: Fall Detection Using 4D MmWave Radar and A Hybrid Variational RNN AutoEncoder, IEEE, 2020, pp. 1-10

[22] C. Medrano, R. Igual, I. Plaza, M. Castro, Detecting Falls As Novelties In Acceleration Patterns Acquired With Smartphones, PloS one, 2012, pp. 1-9

[23] M. Yu, Y. Yu, A. Rhuma, S. M. R. Naqvi, L. Wang, J. A. Chambers, An Online One Class Support Vector Machine-Based Person-Specific Fall Detection System For Monitoring An Elderly Individual In A Room Environment, IEEE, 2013, pp. 1002–1014

[24] J. Nogas, S. S. Khan, A. Mihailidis, DeepFall – Non-invasive Fall Detection With Deep Spatio-Temporal Convolutional Autoencoders, arXiv, 2020, pp. 1–18

[25] C. Lovescu, S. Rao, mmWave Radar Sensors - What Is mmWave, Texas Instruments, 2020, [Online]. Available: https://www.ti.com/sensors/mmwave-radar/what-is-mmwave.html

[26] C. Lovescu, S. Rao, The Fundamentals of Millimeter Wave Sensors, Texas Instruments, 2020, pp. 1-9