

Human Fall Detection using multimodal datasets

Ana Janet Pacheco Jiménez

A project report submitted in conformity with the requirements
for the degree of Master's of Science in Information Technology

Department of Mathematical and Physical Sciences
Faculty of Graduate Studies
Concordia University of Edmonton



**HUMAN FALL DETECTION SYSTEM BASED ON MULTIMODAL
DATASET USING DEEP LEARNING ALGORITHMS**

ANA JANET PACHECO JIMÉNEZ

Approved:

Nasim Hajari, Ph.D.

Supervisor

Date

Committee Member Name, Ph.D.

Committee Member

Date

Dr. Alison Yacyshyn, Ph.D.

Dean of Graduate Studies

Date

Abstract

Falling events are the leading cause of fatal and non-fatal injuries among elderly [1], and without timely medical care people unable to recover by themselves, it may mean suffering from serious consequences. Designing reliable fall detection systems may help reduce the time for a person to receive an opportune medical care. Falls are events which might be described using multiple sensors, this is because the problem itself involves many features, such as the sudden change in human's body position which can be sensed by tracking acceleration changes in the three orthogonal directions, or while monitoring activities of daily living of elderly people. This project is aimed to compare the performance of a fall detection system based on a multimodal dataset using common deep learning algorithms, such as Convolutional Neural Networks and Multi-Layer models and feeding them with image data only (RGB and Depth Data), versus when adding signal data from an accelerometer sensor.

Keywords: multimodal-fall detection system, Convolutional Neural Network, Multi-Layer Perceptron, RGB and Depth data, accelerometer sensor.

Contents

1	Introduction	1
2	Theoretical framework	2
2.1	General pipeline for fall detection system	2
2.1.1	Data preprocessing	4
2.1.1.1	Image preprocessing	
		4
2.1.2	Feature extraction	5
2.1.3	Supervised deep learning algorithms	5
2.1.3.1	Convolutional Neural Network (ConvNet/CNN)	
		6
2.1.3.2	Traditional Machine Learning approach vs Deep Learning approach	
		7
2.1.4	Performance evaluation	8
3	Literature review	9
3.1	Vision-based systems	10
3.1.1	RGB information versus Depth information	11
3.2	Sensor-based database systems	12
3.2.1	Wearable sensors	14
3.3	Multimodal-based database	14
3.4	Limitations of existing datasets	14
4	Scope	15
5	Hypothesis	16
6	Methodology	16
6.1	Data Gathering	16
6.1.1	Dataset	16
6.2	Data preprocessing	17
6.2.1	Pre-processing categorical data	17
6.2.2	Pre-processing image data	18
6.2.3	Pre-processing accelerometer data	18
6.3	Convolutional Neural Network architecture	18
6.4	Multi-Layer Perceptron architecture	18
6.5	Combining deep learning models	18
7	Results and discussion	20

8	Conclusions	23
9	Future work	23
10	Recommendations	23
11	Acknowledgments	23
12	Appendices	24

List of Tables

1	Comparison of machine learning-based Human Activity Recognition (HAR) and deep learning-based HAR [22]	8
2	Characteristics of the most popular visual-based datasets from 2017 to 2020	11
3	Advantages and disadvantages of color and depth data [15, 14, 16] . .	12
4	Functionality of the most common motion sensors for fall detection .	15
5	Comparison between multimodal datasets and their performance. . .	16
6	UR dataset description	17
7	CNN architecture specifications	19
8	Multi-Layer Perceptron architecture specifications	19
9	Comparison of fall detection performance between system 1 and system 2	21
10	Comparison of fall detection performance between all the systems . .	23

List of Figures

1	Generalized working principle of fall detection systems	3
2	General architecture of Fall Detection Systems.	4
3	Confusion matrix	6
4	Confusion matrix	9
5	Percentage of the most implemented ML algorithms for fall detection and prevention from 2010 to 2021 [7].	10
6	RGB vs Depth images under different lighting conditions [14]. (a) RGB img. in normal lighting condition; (b) Depth img. in normal lighting condition; (c) RGB img. in low lighting condition; (d) Depth img. in low lighting condition	12
7	Architecture of deep learning system with multiple and mixed types of input data.	20
8	Confusion matrix using : (a) RGB and Depth (3 channels) and (b) RGB and Depth depth data (1 channel)	21
9	CNN performance of System 1. Accuracy value vs number of epochs .	22
10	CNN performance of System 2. Accuracy value vs number of epochs .	22
11	CNN architecture implementation in python code.	24
12	CNN architecture implementation in python code.	25
13	MLP architecture implementation in python code.	25
14	Concatenation phase in python code.	26

1 Introduction

A fall is defined as an event which results in a person to, inadvertently, rest on the floor or other lower level. Injuries from a fall may be fatal or non-fatal [1]. Having a fall is an unfortunate but also common event that can happen to anyone, but it usually becomes recurrent as people get older.

Causes of falls are multiple as they are the result of many risk factors involved, such as age, visual impairment, poor balance, unsafe environmental conditions, underlying medical conditions, side effects of medication [2, 1]. People aged 60 or 65 years or over, are at major high risk of suffering fatal falls or serious falls which may require subsequent long-term care and institutionalization. This is because sensory, and cognitive changes associated with aging and most environments are not well adapted for an aging population [1].

Elderly people is a growing demographic group, for example the United Nations [3] reported in 2019 that older people accounted for more than one fifth of the population in 17 countries, and its projections indicate this condition will be the same at the end of this century for 155 countries, which roughly represents the 61% of the world's population. Researchers have reported that around 69.5% of falls occurred at home [5] and in regions like Europe and Northern American where elderly individuals tend to live alone or with spouse only [4], this mean they are more prone to suffer a "long-lie", which is staying on the floor for an hour or more following a fall [6] and its consequences of a late medical care may lead to severe trauma or even lead to person dies.

Clearly, this is the reason why the development of health-care related hardware and software for fall detection devoted to this sector has been increasing during the last decades. The number of methods that have been proposed to detect falls can be classified into two broad categories ,wearable systems and non-wearable systems [7], or into main three categories by some other authors [8, 9]: wearable device-based, ambience sensor-based and vision or camera-based.

Wearable systems are as their name suggests, devices or sensors which are suitable to be attached to the human body for data collection. Examples of this kind of sensors are accelerometers, gyroscopes, magnetometers, Inertial Measurement Unit (IMU), Surface Electromyography (sEMG), among others. The three main motion sensors, accelerometer, gyroscope, and magnetometer, embedded in smartphones or smartwatches as micro-electromechanical systems (MEMS), have eased the data collection to assist researchers in developing systems to detect falls.

Vision-based devices comprehend systems able to identify a fall occurrence through artificial vision. Here, we can find conventional cameras (RGB or web cameras), motion capture devices, depth cameras such as Kinect, Laser Range Scanners (LRS), which take optical measurements to use them in image processing for later analysis [7]. The use of RGB cameras for human motion tracking, besides it has high privacy

concerns, it also has challenges in illumination changes, deformation, occlusion and background clutter [11].

Finally, ambience sensor-based systems or also known as context-aware systems consider all the sensors deployed in a surveillance environment. Examples of these sensors are infrared (IR), pressure, acoustic, radio-frequency sensors, floor sensors like Ground Reaction Floor (GRF).

Over the last years, computer vision has been evolving, mainly because of the implementation of machine learning algorithms in classifying falls and not falls while recognizing human actions in activities of daily living (ADLs).

Multisensor or multimodal approach detection systems assisted by Machine Learning algorithms for the data analysis, combine data collected from various sensors, wearable and non wearable, to detect falls in real time. In the attempt to enhance the reliability of fall prediction systems, this multimodal detection approach promises to compensate for the limitations of single-sensor based detection systems.

The document is organized as follows: firstly, in the Section 2 we present the general working flow, a fall detection system, and other common concepts found in deep learning models implementation. Section 3 we present an overview of the previous findings and studies using multimodal datasets and machine learning algorithms for fall detection. Limitations and challenges of the current problem are also discussed here. Section 4 and outlines the main objective and scope of this paper. Hypothesis is exposed in the Section 5. The Section 6 presents the methodology followed and a brief description of the dataset used. The results and discussion are shown in the Section 7, where performance comparisons between the two deep learning algorithms implemented, CNN and MLP, and the previous studies are done. Finally in the Section 8 and, Section 9, we discuss the conclusions extracted from the results retrieved and the future work projected for this research respectively.

2 Theoretical framework

2.1 General pipeline for fall detection system

All the existing fall detection systems have a sensing unit for data acquisition, then the data processing phase where the raw data collected feeds a traditional algorithm or a deep learning model to finally detect whether a fall happened or not. The Figure. 1 shows the generalized working principle of Fall Detection Systems (FDS).

The overall pipeline for any fall detection system can be broken down into five major steps as shown in the Figure. 2. Collecting data is the first step to solving any machine learning problem, this is because gathering relevant and reliable data is crucial to get good predictions. Depending on the nature of the problem to solve, open-source datasets might be a good free and accessible source of information. Because real-world data is often inconsistent and noisy, a data preprocessing phase is needed to

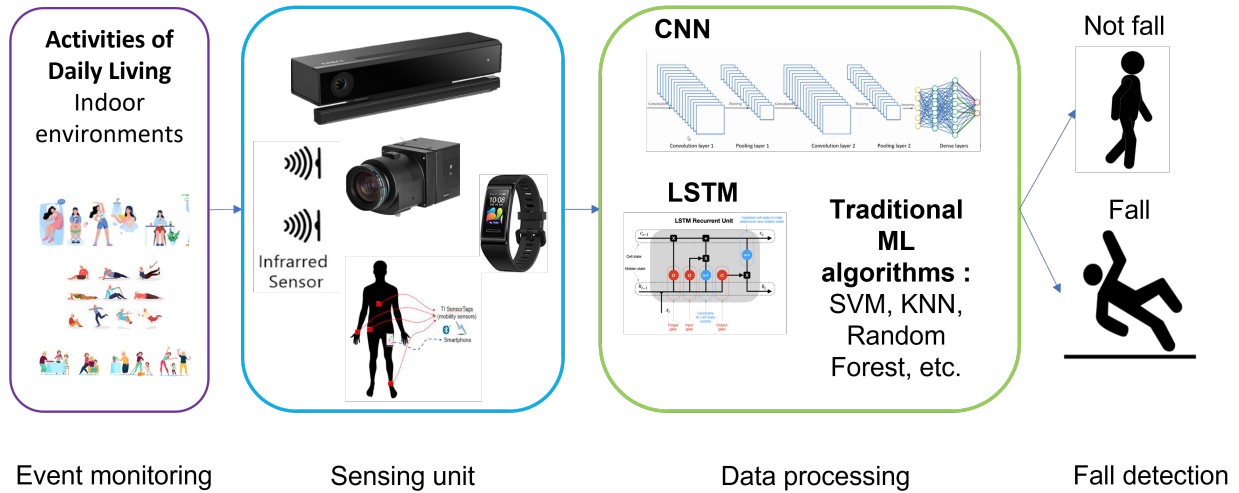


Figure 1: Generalized working principle of fall detection systems

clean this data and eliminate null and duplicate values, irrelevant features, etc.

The third step is feature engineering, where the preprocessed data is now selected and transformed into features that better represent the problem. Statistical tools are extensively implemented for feature extraction because it prevents redundant information. Based on given data input, various feature extraction methods may be applied.

Before starting to build a learning model, firstly our dataset has to be divided into a training set and a test set, although the ratio of each set depends on the application, it is common to use a ratio of 70 or 80 for training over 30 or 20 for testing. In the training phase the learning algorithm tries to "understand" how all the inputs contribute to form the target variable. Whereas the test set is used to validate the performance of the learning model.

Discriminative machine learning models, such as decision tree, evolutionary algorithms, K-nearest neighbors (KNN), Support Vector Machine (SVM), fuzzy logic, regression, and neural networks, are typically implemented for human fall detection [37].

Finally, for judging the performance of our model, we use function metrics, which are similar to loss functions, except that the results are not used when training the model. Selecting the right metric to evaluate our model will depend on the nature of the problem to solve.

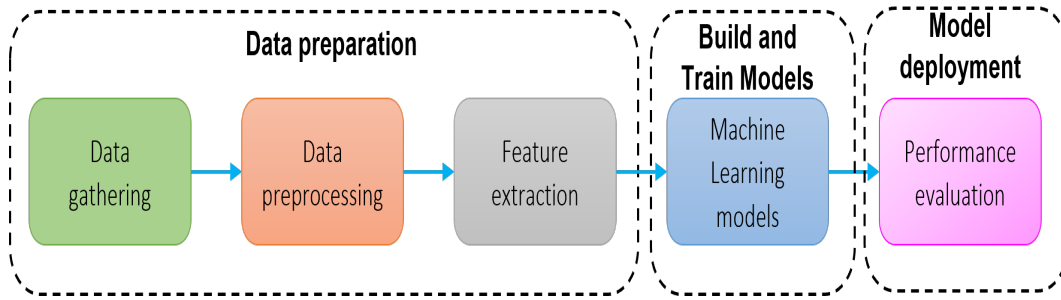


Figure 2: General architecture of Fall Detection Systems.

2.1.1 Data preprocessing

There are various basic techniques to make data ready to be processed by the next steps in the ML pipeline. Overall data preprocessing can be done in four different ways. Data cleaning, data integration, transformation and data reduction.

1. Data cleaning: Methods devised to fill in missing values, fix data discrepancies, identify outliers and reduce the noise.
2. Data integration: This technique combines multiple sources into a coherent data set.
3. Data transformation: It consists of three different processes, data normalization, aggregation and generalization
 - Normalization: It aims to change the values of numeric columns in a database to a common scale, without losing differences in the range of values.
 - Aggregation: It creates a summary of the raw data, identifying high correlations among features and then applying aggregation operations. It helps in decreasing high dimensionality
 - Generalization: Process to summarize data by replacing particular features with high level concepts.
4. Data reduction: It involves making smaller large sets of data into meaningful fragments.

2.1.1.1 Image preprocessing

Image preprocessing is devised to improve image data by enhancing specific visual features that may be meaningful for a certain application. It also helps in reducing undesired distortions.

There are 4 different techniques commonly used in image preprocessing, and they are listed as follows:

- Pixel brightness transformations or corrections : Among the most common operations we can find sigmoid stretching, histogram equation and gamma correction.
- Geometric transformations: It eliminates geometric distortion through operations like rotation, scaling, translation and shearing.
- Image filtering and segmentation:
 - Image filtering: It uses filters (also known as kernels) to modify, improve image properties or extract meaningful information, such as corners or edges. Low pass filtering, edge detection, directional filtering and Laplacian filtering are some of the common techniques.
 - Image segmentation: Technique used to break down into various image segments or subgroups a digital image to simplify its complexity.
- Fourier transform and image restoration: It decomposes the image into its sine and cosine components.

2.1.2 Feature extraction

This phase is devised to transform the preprocessed data into an optimal set of features that better represent the problem to learning models. The selected features are intended to be relevant and non-redundant. Dataset with a large number of features may cause overfitting in ML models. The common dimensionality reduction techniques are:

- Independent Component Analysis (ICA)
- Linear Discriminant Analysis (LDA)
- Principle Components Analysis (PCA)
- Locally Linear Embedding (LLE)
- t-distributed Stochastic Neighbor Embedding (t-SNE)
- Autoencoders

2.1.3 Supervised deep learning algorithms

Deep learning algorithms have rapidly become a common approach for fall detection, especially Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) which also have demonstrated to have good performance. Their use in this field is not a coincidence, because image processing itself is a high dimensional challenge to solve, and CNNs have proved to be very effective in reducing the number of parameters through filters or kernels which assign importance, learnable weights and biases, to objects or features in the image we do not want to lose. CNNs' filters are optimized through automated learning, whereas in traditional classification

algorithms these filters are hand-engineered. This means that pre-processing required in CNNs is much lower as compared to those traditional algorithms [38].

2.1.3.1 Convolutional Neural Network (ConvNet/CNN)

Convolutional Neural Networks are a class of Artificial Neural Networks (ANN), most commonly used in many image processing tasks. They are devised to automatically and adaptively learn spatial hierarchies of features through a backpropagation algorithm [32].

A ConvNet can be usually composed by:

1. Convolutional layer (Conv) : It is a type of linear operation for feature extraction. For image processing it gets as input a matrix of dimensions [height, width, depth] as shown in the Figure. 3. This layer contains filters also called kernels, that perform convolution operations to create an activation map or feature map. Kernel is aimed to scan the input image with respect to its dimension. Filter size and stride are its hyperparameters.

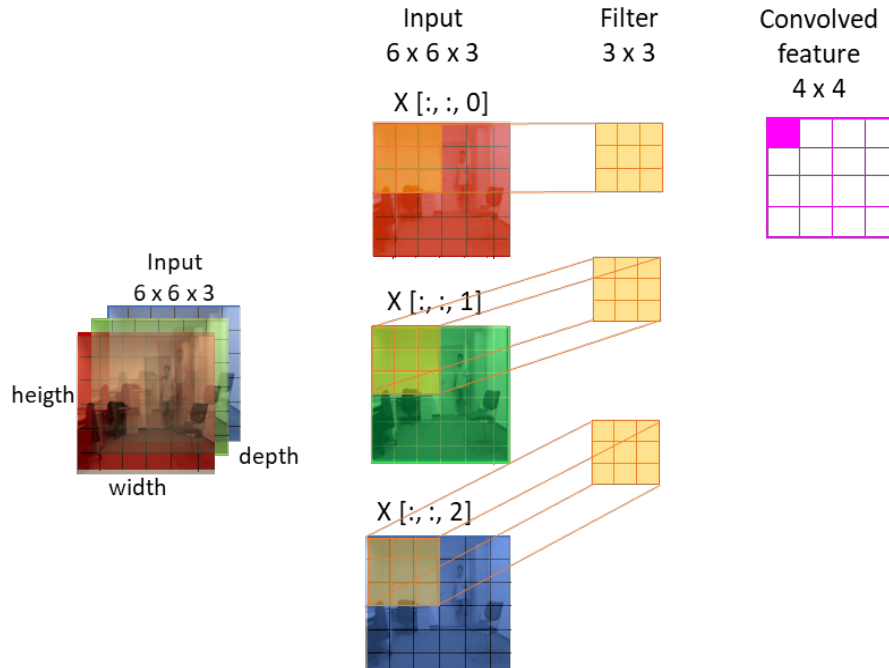


Figure 3: Confusion matrix

2. Nonlinear activation function: The Rectified Linear Unit (ReLU) is the most common nonlinear activation function used. ReLU computes the value provided as input directly, or the value 0.0 if the input is 0.0 or less. We can describe this function mathematically as $f(x) = \max(0, x)$.

3. Pooling Layer (Pool): It is a down sampling operation, which summarizes the presence of features in the patches of the activation map, which does some spatial invariance.
 - Max pooling: Selects the maximum value of the feature map
 - Average pooling: Calculates the average for each patch on the activation map.
4. Fully Connected Layer (FC) / Hidden Connected Layer: It is a feed forward neural network which operates on a flatten input (array of number of 1D). Each input is connected to all the neurons. This layer is the combination of affine function and nonlinear function.
 - Affine function: $f(x) = Wx + b$ where W is the weight and b the bias.
 - Non-linear functions: Sigmoid, TanH, and ReLu
5. Output Layer: Here is where the desired outputs are obtained. Linear, Sigmoid, and Softmax, are the most common activation functions implemented.

2.1.3.2 Traditional Machine Learning approach vs Deep Learning approach

Usually a typical deep learning work flow to analyze a dataset follows these four sequential stages: data understanding and pre-processing, feature-engineering, and deep learning model building, training and finally performance evaluation.

Before feeding a traditional machine learning model or a deep learning algorithm, the raw data is preprocessed or transformed to speed up the training process and improve the overall performance. There are many

preprocessing techniques which can be applied, for instance data cleaning, dimensionality reduction, sampling data, data transformation, imbalanced data, and so forth.

Although, pre-processing phase is required in a Convolutional Neural Networks, this is much lower as compared to other classification algorithms.

The Table 1 summarizes the differences between a traditional machine learning and a deep learning approach on featuring-engineering and learning process.

Recurrent Neural Networks architectures, and particularly LSTMs have feedback connections, exploit time series data, such as a sequence of images, to exhibit temporal dynamic behaviour. LSTMs process entire sequences of information, keeping significant information of previous data in the sequence to support the processing of new data points [39].

Table 1: Comparison of machine learning-based Human Activity Recognition (HAR) and deep learning-based HAR [22]

Process	Machine learning	Deep learning
Feature engineering	<ul style="list-style-type: none"> • Relies on manually extracted features. • Depends on applications. • Fails to deal with complicated activities. • Requires feature selection, and dimensionality reduction approaches. • Fails to handle the inter-class variability and inter-class similarity. 	<ul style="list-style-type: none"> • Learns abstract features from raw input data automatically. • Discovers spatial, temporal dependencies and scale invariant features from the input data automatically.
Learning process	<ul style="list-style-type: none"> • Works well on small training data. • Requires limited computation time and memory usage. 	<ul style="list-style-type: none"> • Requires large dataset to prevent overfitting. • High computational complexity. • Requires specialized hardware to accelerate the training process.

2.1.4 Performance evaluation

Performance evaluation is the last step in the pipeline of any machine learning project implementation. This phase tells us how effective our trained learning model is classifying or predicting.

The common practice to assess the performance of this classification task come from the the confusion matrix, as shown in the Figure. 4, and also listed as follows:

Where TP : *True Positive*, FP : *False Positive*, and FN : *False Negative*.

$$Accuracy = \frac{TP + TN}{TN + FP + TP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{TotalPredictedPositive} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{TotalActualPositive} \quad (3)$$

$$Recall = 2X \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

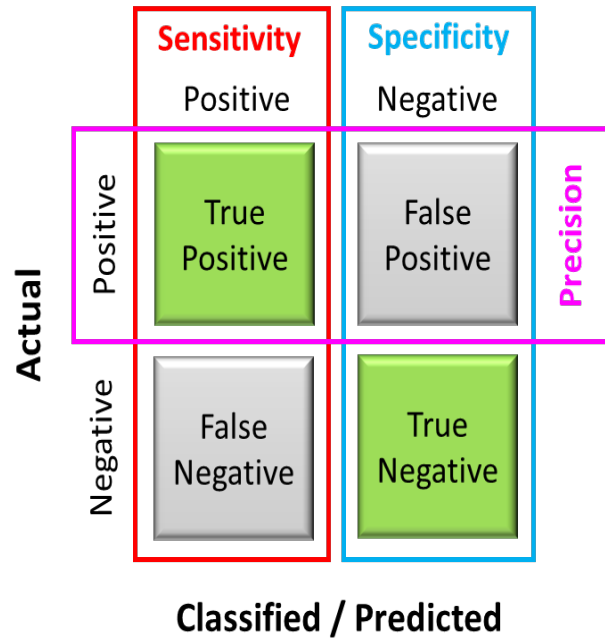


Figure 4: Confusion matrix

3 Literature review

Existing studies for human fall detection can be divided into three main categories, based on the type of the database used.

The first category is visual-based dataset which may contain images retrieved from RGB, and depth cameras, data from motion capture devices, or other optical devices. Sensor-based dataset where data from wearable and ambient sensors might be gathered. The third category is a multimodal-based dataset where the two first categories are combined.

The graph, Figure 5, shows the most implemented machine learning algorithms. Overall, SVM and KNN have been the most implemented for fall detection and prevention [7], however, during the last years there has been a trend towards exploring deep learning algorithms specially for image-based fall detection systems. A deep learning approach can provide unseen input data from the example of images.

The main learning outcomes of SVM and KNN, a traditional ML approach, have been that SVM algorithms can manage high-dimensional data and set a wide boundary between a standing and falling person; it is also memory-efficient in nature that is optimal for wearable devices. KNN works well on making real time predictions but the systems based on it, might require a lot of memory to store the training data.

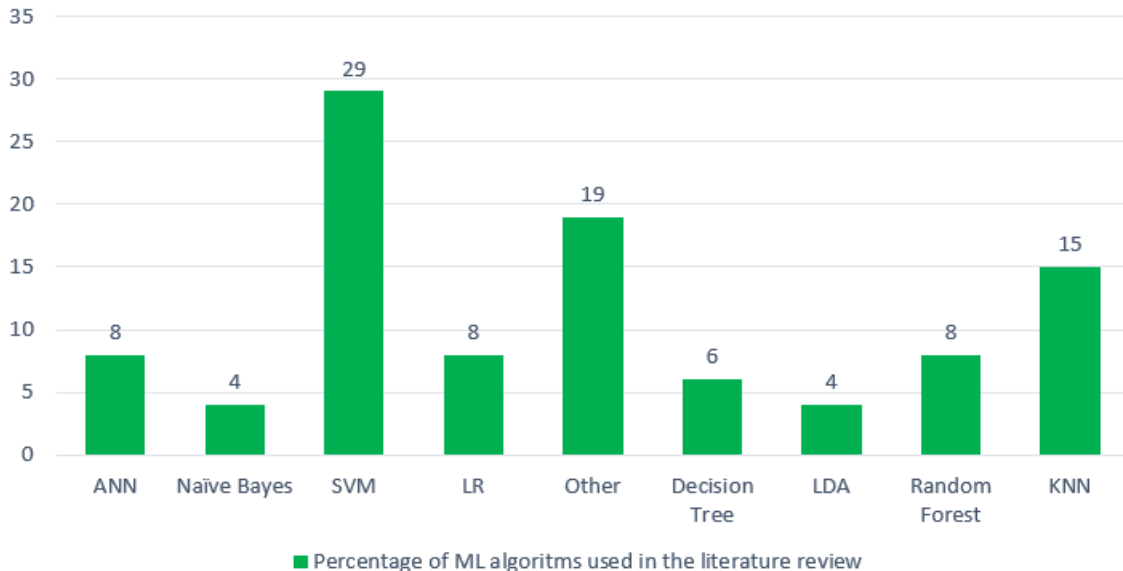


Figure 5: Percentage of the most implemented ML algorithms for fall detection and prevention from 2010 to 2021 [7].

3.1 Vision-based systems

Human activity recognition has been a topic of interest for several fields, going from industrial to healthcare applications. This is because awareness of human motion is crucial for people’s assisted living with underlying medical conditions and those industrial environments where there is a close interaction between humans and robots.

Although it has been argued that visual sensors, and particularly RGB cameras, may lead to exposing older people’s private life when monitoring activities of daily living. Usage of depth cameras in most of the published articles is considered protective and private because individuals’ faces are hidden [19]. Some other approaches to avoid these privacy concerns have been proposed such as instead of directly using the raw images, but extracting numerical information from them [20].

From the thirty-four vision-based fall detection systems, published in articles from 2015 to 2020, studied by Gutiérrez’s, J. et al.[12], we can see that the most popular databases used for fall detection were: UR fall detection, LE2I, and Fall detection datasets. Table. 2 shows the main characteristics of those datasets.

The UR Fall Detection dataset was used in at least twenty-eight of the thirty-four articles summarized in [12]. Although this dataset includes depth, RGB and accelerometer information, researchers only used RGB or Depth information to feed their learning models.

Using only RGB information, K. Sehairi et al. [33] implemented foreground extraction and feature extraction associated with each silhouette, then they used SVM, KNN and a fully connected ANN, which demonstrated to have the best accuracy of

Table 2: Characteristics of the most popular visual-based datasets from 2017 to 2020

Signal Type	Dataset Name	Characteristics
Depth and Accelerometer data	UR fall detection [21]	30 falls and 40 ADLs recorded by RGB-D and accelerometer systems
RGB	LE2I [29]	191 different activities including ADLs and 143 falls
Depth	Fall detection dataset [30]	5 volunteers execute 5 different types of fall

99.61%.

A. Abobakr et al. [34], extracted silhouette information by using depth information. Random decision forest for pose recognition and SVM for movement identification, using UR Fall Detection and CMU Graphics Laba motion capture library, they got an accuracy 96%, precision 91% and Sensitivity 100%.

In 2020, Qi Feng et al. [35], using RGB features maps from CNN and LSTM and then for classification they used a softmax based features vector from ANN implemented in its last layer. Using the Multicam Fall Dataset, they got a specificity 93.5%, and a precision of 94.8% on the UR Fall Detection dataset.

3.1.1 RGB information versus Depth information

Depth images have been used to detect, identify and localize human body parts in real time [13] and have demonstrated to have high reliability on real time gaming systems. Some studies have underlined the benefits of not using RGB images in some visual-sensor-based systems to track human motion, where color invariance has been convenient. This is because for some classification tasks, depth images can actually encode enough relevant information. Sangheon Park et al.[14] in Figure. 6 shows a set of RGB and depth images under different lighting conditions. Frames (a) and (b) are images under normal illumination conditions. In contrast, frame (c) shows the same man posing as in the previous image, but because of the poor lighting conditions it is tougher to appreciate it, whereas the depth camera seems to be resilient to this lighting variation. On the other hand, most of the previous color-based works were based on shape features extracted from human region candidates, e.g. real-time hand tracking where skin color and motion information were broadly used [17].

Depth data has been used in many research papers either solely or supplementing RGB data, and has been verified to be resilient in changing illumination conditions. Its broad use for fall detection is justified by the number of advantages in classification tasks, and particularly in those with segmentation problems, such as human motion recognition. Where the person needs to be detected and then subtracted from the background to give us a far more granular understanding about the person’s shape and posture in the image.

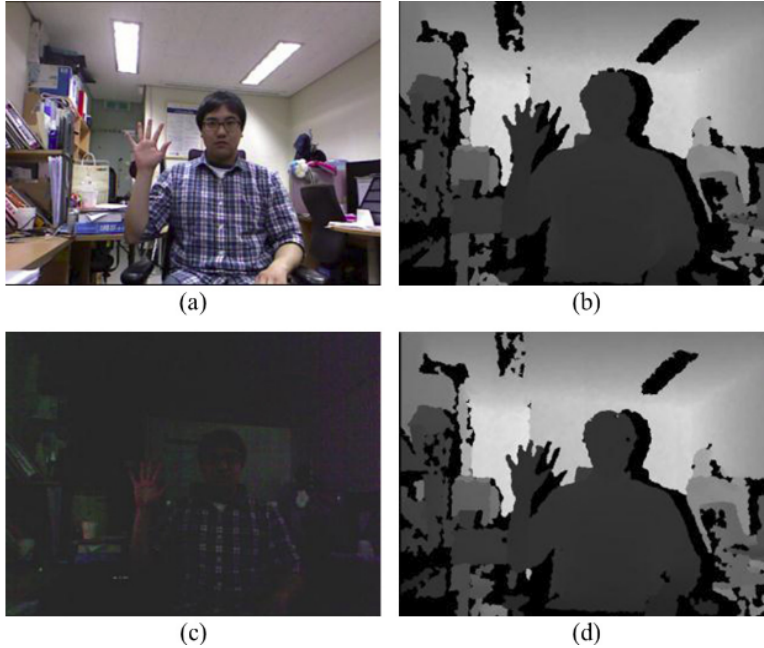


Figure 6: RGB vs Depth images under different lighting conditions [14]. (a) RGB img. in normal lighting condition; (b) Depth img. in normal lighting condition; (c) RGB img. in low lighting condition; (d) Depth img. in low lighting condition

The Table 3 summarizes some of the advantages and disadvantages of depth and RGB information in solving segmentation tasks, such as human pose recognition and motion tracking.

Table 3: Advantages and disadvantages of color and depth data [15, 14, 16]

	RGB data	Depth data
Advantage	Easy to find feature	Resilient to light variation Getting real depth value Remove or mitigate sources of ambiguity Shorter time of processing
Disadvantage	Sensitive to changing light conditions Occlusion	Hard to find features Noise in edges Occlusion

3.2 Sensor-based database systems

The great availability, portability and relative invariability to environmental changes of wearable sensors have enabled researchers to collect relevant information to help to detect when a fall is happening. Among the most cited sensor-based datasets are summarized in [18]. In light of the privacy concerns video surveillance may raise, ambient and wearable sensors have been good allies in fall detection because they keep individuals' identity undisclosed. Some of the most cited public sensor-based datasets for fall detection are summarized as follows:

MobiFall [23] has been a popular dataset cited in many research papers created by the Biomedical Informatics and eHealth Laboratory of Technological Educational Institute of Crete. This dataset collects information from an inertial module (3D accelerometer and gyroscope) of a smartphone located in a trouser pocket. Twenty-four volunteers, men and women between 22 to 47 years old, performed 4 different types of falls (Forward-lying, Front-knees-lying, Sideward-lying, Back-sitting-chair) and 9 ADLs. ADLs were chosen based on real-life frequency and on their affinity to actual falls, which might lead to false positive cases. K-nearest neighbours classifier, 10-fold cross-validation and subject-wise percentage split, were the methods used in this paper.

Another publicly available dataset is the extended version of MobiAct [24] which is based on MobiFall, and includes 4 different types of falls and 12 different ADLs from a total of 66 volunteers with more than 3200 trials, all captured by a smartphone located in a trouser pocket. Actually MobiAct is considered to be suitable for both HAR or HFD research. The classifiers IBk (with 1 nearest neighbor), and J48 decision tree had the best performance in classifying ADLs, getting an accuracy of 99.88% and 99.3% respectively.

tFall dataset by Medrano et al.[25] uses the embedded accelerometer of two smartphones positioned in the person's pocket and hand bag. It collects data from seven male and 3 women with ages ranging between 20 to 42 years old. The activities were differentiated as Fall and ADL. The samples were gotten every 0.02ms(50Hz). This study implemented One-Class SVM and Neural Networks for learning models and for evaluation, they implemented 10-fold cross-validation. SVM sensitivity 92.9%, 1NN sensitivity 90.0%, SVM specificity 91.7%, geometric mean $\text{sq}(\text{SP} \cdot \text{se})$ SVM 92.2%, 1NN 86.9% , SVM AUC 96.8%, 1NN 91.5%. The authors showed that when the algorithms were tested with part of her or his own data, personalized, it boosted their performance.

The UMAFall dataset [26] includes 531 files (322 ADLs and 209 falls), collected from a smartphone located in the right thigh pocket, and other wearable sensors positioned in the right wrist, ankle, hip, waist and chest. This paper analysed the relationship between the sensing point location and the performance of threshold-based approach for fall detection.

The SisFall [31] database used two self developed devices with two accelerometers and one gyroscope positioned on subjects' waist. They collected information from 15 types of falls and 19 ADLs, from 38 subjects ranging from age 19 to 75 years old. The authors followed the common pipeline to process the data: pre-processing, feature extraction, classification, and validation. Threshold-based classification and 10-fold cross-validation set-up to validate the robustness of the classification.

The DLR [27] dataset was generated by the German Aerospace Center which gathers information from an IMU sensor worn on the belt by 16 participants. This research paper designed and evaluated a system for activity recognition for seven ADLs. The

authors showed that a Hidden Markov Model (HMM) based on Bayesian Network achieved a precision of 80% and 100% for recall.

In Vilarinho's et al. [28] implemented a system based on threshold and pattern recognition techniques for fall detection and daily activities recognition.

Finally, TST [36] Fall detection dataset contains depth frames, skeleton joints, and acceleration samples collected during simulation of ADLs and falls.

3.2.1 Wearable sensors

The Table. 4 shows an overview of the functionality of the most common wearable devices used in falls detection. MEMS wearable sensors have been widely used in this field in part due to their affordability, global adoption of these smart devices, and also because there are less privacy concerns compared to RGB cameras.

Although existing sensor-based methods have proved to have good results identifying falls, the batteries of these wearable sensors need to be recharged or replaced, which may lead to stop wearing them if there is not someone else who takes care of it. Besides, some users may express some discomfort after wearing them for a long time, which is most likely to happen among elderly because they are simply not accustomed to it.

3.3 Multimodal-based database

Many existing methods are based on wearable devices with motion sensors, such as accelerometers and gyroscopes, which have got reasonably good results in fall detection. However, users could feel uncomfortable after wearing the device for a long time, or sometimes forget to wear them. Besides, wearable devices consume batteries that need to be replaced or recharged frequently. Visual monitoring hence has some advantages, where users are freed from wearing devices with minimum disturbance to their daily lives. The Table. 5 shows some of the relevant multimodal-based systems.

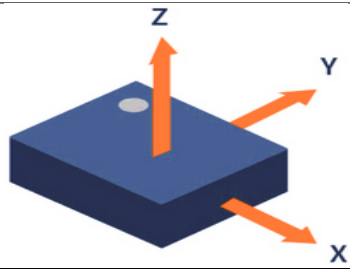
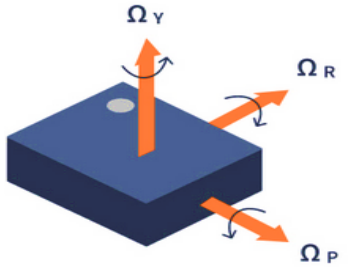
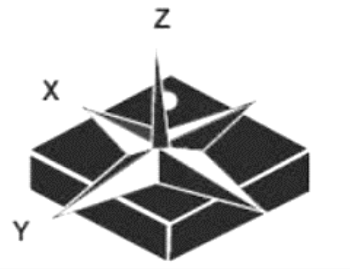
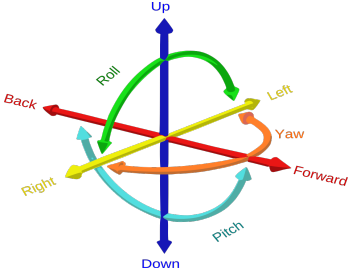
3.4 Limitations of existing datasets

From the literature review we can see that researchers still having some limitations when developing automatic systems for Human Fall Detection. Overall the majority of the published articles worked with data collected from lab conditions [7], and not in day-living environment or from publically available datasets.

The availability of multimodal datasets is still being less compared to single-based sensor modality. Around 88% [7] of the studies conducted for fall detection are wearable-sensor-based, captured by either IMU sensors or embedded sensors in smart-phone.

There is only one dataset [31] with more than eight fall styles. Finally, it is important to stress that most of subjects of study were young healthy adults instructed to

Table 4: Functionality of the most common motion sensors for fall detection

Sensor name	Functionality	Image
Accelerometer	Accelerometer is an electromechanical device that detects linear acceleration along an axis. Its measure units are meters per second squared (m/s^2) or in G-forces (g).	
Gyroscope	A device used for measuring rotational changes (angular velocity along an axis) or maintaining orientation. The units of angular velocity are measured in degrees per second ($^\circ/s$) or revolutions per second (RPS).	
Magnetometer	A magnetometer measures the power and direction of magnetic fields. Magnetometers measure a magnetic field or flux density in metric units of gauss (G) or the international system (IS) unit tesla (T).	
IMU	An Inertial Measurement Unit is essentially the three sensors embedded in one system, that is, Accelerometer + Gyroscope + Magnetometer sensor. It has 2 to 6 degrees of freedom, which describe the movements of an object in 3D space	

simulate falls. Moreover, there are few datasets that consider the impact of sensors placement on the performance of whatever machine learning algorithm used.

4 Scope

This paper aims to provide a comparison of the performance of deep learning algorithms (Convolutional Neural Network and Multi-Layer Perceptron) for fall detection systems addressing two different approaches. Firstly, a system where its inputs are only image data (RGB + Depth) and learning model is CNN. And secondly, a system

Table 5: Comparison between multimodal datasets and their performance.

Dataset	Type of sensors or Cameras	Subjects	No.Samples	Learning model	Performance
UR Fall Detection	1 IMU, 2 Depth cameras	5	70 sequences	SVM for depth SVM for depth + acc Threshold UFT and LFT	Depth: accuracy 90.0%, precision 83.3%, sensitivity 100%, specificity 91.25%. Depth + cc: accuracy 98.33%, precision 96.77%, sensitivity 100%, specificity 96.67%
UP-Fall	5 IMU, 1 electroencephalograph (EEG) , 6 infrared in grid 2 cameras	17	3 repetitions	Random Forest, SVM, neural networks, KNN	IR+IMU+EEG : 69.38% (RF) and 68.19% (MLP)
MHAD	6 accelerometers, 4 microphones, 1 motion capture system, 4 multiview cameras, 1 Depth camera	12	5 repetitions	SVMS, KNN	accuracy 98.24%
Dovgan et al [*25]	Ubisense system, 1 Accelerometer	10	3 for ADL 2 for falling	SVM C4.5	Ubsinse : accuracy 95.58% Accelerometer: accuracy 57.96%
CMDFALL	7 overlapped Kinect sensors 2 accelerometers	50	400 Falls 600 ADLs	C3D convnet on RGB DMM-KDES on Depth Res-TCN on skeleton 2D ConvNet on Acceleration	RGB + Depth + Skeleton + Acc: F1-Score: 98.29% RGB +Skeleton: 98.29%

where accelerometer data is added implementing MLP model.

5 Hypothesis

It is not possible to fully describe when a fall is happening using a single sensor, this is because either wearable, ambient or visual sensors have their own limitations. Developing a fall detection system based on multiple inputs and mixed data information coming from different sensors improves the performance of deep learning algorithms because it compensates for sensors disadvantages.

6 Methodology

6.1 Data Gathering

The our proposed FDS almost followed the general pipeline described in the Figure. 2. For data gathering, this study used the public multimodal dataset, UR Fall Detection Dataset. We selected this dataset because its availability and also because it has been used by many research papers we can later compare the results of this proposed approach.

6.1.1 Dataset

The UR Fall Detection Dataset was generated by the University of Rzeszow [21], when studying traditional machine algorithms for a system which integrated depth maps and accelerometer data to reduce false-alarm errors.

There are two types of falls: falling from standing position and falling from sitting

on a chair, and common activities like walking, sitting down, crouching down and lying. Motion data was synchronized with all RGB and depth sequences.

Activities of daily living were recorded using only the camera parallel to the floor, front view, and x-IMU (256Hz) sensor. The 70 sequences were recorded in typical indoor environments, like offices, classroom, rooms, etc. The Table. 6 summarizes the main features of the UR Fall Detection dataset.

Table 6: UR dataset description

No. subjects	No. of samples (ADLs/Falls)	Types of sequences (ADLs / Fall)	Types of sensors	Sensors position	No. of sensor points	Views
5 males	40 / 30	4 / 2	Depth camera RGB camera IMU	Waist	1	Front view Top view

The extracted features from depth maps are stored in CSV format. Each record contains the following information:

- sequence name
- frame number
- labels :
 - 1 : subject is not lying
 - 1 : subject is lying on the ground
 - 0 : is temporary pose, when subject "is falling"

And some other depth features for subject detection, such as bounding box height to width ratio, major to minor axis ratio computed from blob segmentation of segmented person, bounding box occupancy by person’s pixels, actual human height in [mm] and distance of person center to the floor in [mm].

6.2 Data preprocessing

6.2.1 Pre-processing categorical data

The UR Fall dataset required minimal preprocessing. For labelling we change the original labels described previously, and we considered the following categorical labels:

- 0 : when subject is not lying
- 1 : when subject "is falling" or lying on the ground

6.2.2 Pre-processing image data

1) Data selection

In total 2995 images were selected from the UR Fall Detection Dataset (URFD), all of them part of the set of fall sequences. That is, we have 11,980 images in total because of the four cameras (2 Depth and 2 RGB cameras).

2) Image resizing

Image resizing was performed to accelerate the training process of deep learning models. Therefore, we reshaped our images, from 640x480 to 128x128 pixels.

3) Image normalization

Because the original range of values for depth and RGB images go from 0 to 1, it was not necessary to normalize them.

6.2.3 Pre-processing accelerometer data

1) Signal data normalization

Signal data was normalized into range [0,1] using MinMaxScaler sklearn function.

6.3 Convolutional Neural Network architecture

We selected Convolutional Neural Network as our learning model because it has proved to be a power algorithm for image processing. The CNN implemented for image processing was compiled using categorical cross entropy as loss function with a batch size of 64 as optimizer we used "adam".

The CNN used when adding the signal data, was almost the same but here we did not compile our model, using the parameter described previously, Figure. 11 .

The Table. 7 shows the name of the layer, type and the output shape of each layer. The code implementation can be seen in the Section 12,

6.4 Multi-Layer Perceptron architecture

The Multi-Layer Perceptron model is based on Keras Sequential API. The MLP architecture comprises one fully connected input layer and a hidden layer, using ReLu as activation function for both layers. The Table. 8 illustrates the technical specifications of the architecture implemented.

6.5 Combining deep learning models

In order to create a model able to accept multiple inputs, and also mixed data types like numerical and image information, we define a Keras model capable to receive

Table 7: CNN architecture specifications

Layer	Type	Output shape
input_1	(InputLayer)	[(None, 128, 128, 3)]
conv_1	Conv2D	(None, 128, 128, 32)
pool1	MaxPooling2D	(None, 64, 64, 32)
norm1	BatchNormalization	(None, 64, 64, 32)
drop1	Dropout	(None, 64, 64, 32)
conv2	Conv2D	(None, 64, 64, 32)
pool2	MaxPooling2D	(None, 32, 32, 32)
norm2	BatchNormalization	(None, 32, 32, 32)
drop2	Dropout	(None, 32, 32, 32)
flat	Flatten	(None, 32768)
hidden1	Dense	(None, 512)
norm3	BatchNormalization	(None, 512)
drop3	Dropout	(None, 512)
hidden2	Dense	(None, 256)
norm4	BatchNormalization	(None, 256)
drop4	Dropout	(None, 256)
out	(Dense)	(None, 2)

Table 8: Multi-Layer Perceptron architecture specifications

Layer	Type	Output shape
dense_3	(Dense)	(None, 8)
dense_4	Conv2D	(None, 4)

numerical data (accelerometer data) and image data from RGB and Depth cameras at the same time.

Multiple and mixed types of data refers to have multiple types of independent data.

The Figure. 7 shows the two branches of our model to handle each type of data. The branch number one is the Muti-Layer Perceptron model, which is aimed to handle numerical data whereas our second branch is the CNN model devised to process RGB and depth image information. Both branches are concatenated together to build the final model.

For the concatenation step we used the concatenate function of Keras, where we passed the outputs of our two branches. Then added 4 dense layers to classify our image. The output layer of our model is formed by a fully connected layer and a linear activation function.

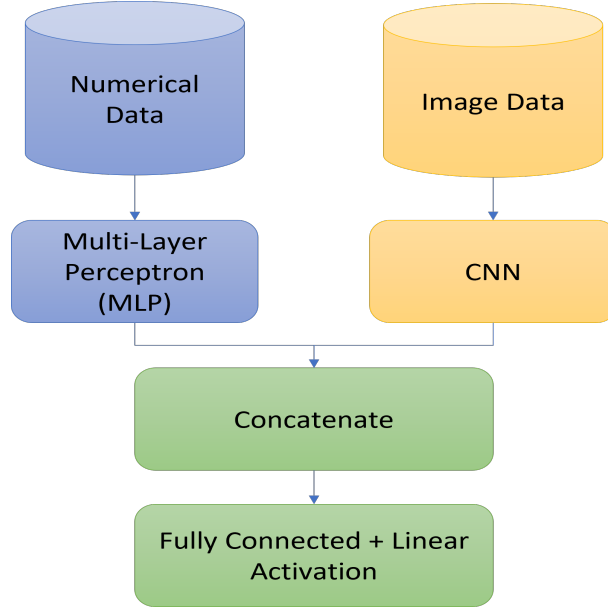


Figure 7: Architecture of deep learning system with multiple and mixed types of input data.

7 Results and discussion

The test experiments are described as follows:

1. System 1: Image data
 - Inputs : RGB (3 channels) + Depth (3 channels)
 - Learning model: CNN
2. System 2: Image data
 - Inputs : RGB (3 channels) + Depth (1 channel)
 - Learning model: CNN
3. System 3: Image data + Accelerometer data
 - Inputs : RGB (3 channels) + Depth (3 channels)
 - Learning model: CNN and MLP
4. System 4: Image data + Accelerometer data
 - Inputs : RGB (3 channels) + Depth (1 channel)
 - Learning model: CNN and MLP

The results showed that CNN model performed well when it was fed either with RGB and depth data considering 3 channels and 1 channel, Figure. ??, respectively,

or when we treated depth image as RGB image.

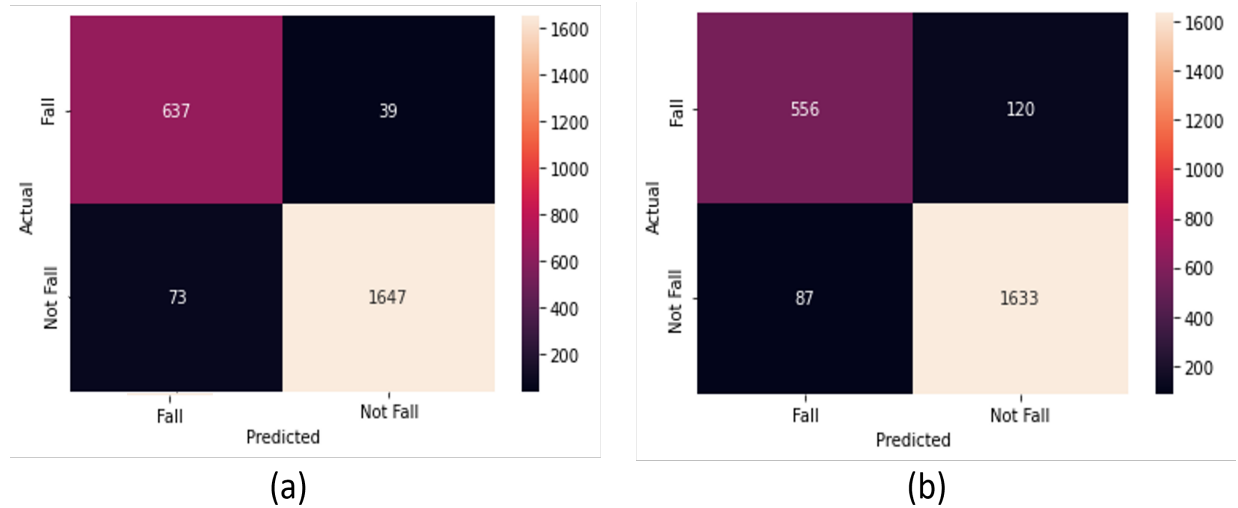


Figure 8: Confusion matrix using : (a) RGB and Depth (3 channels) and (b) RGB and Depth depth data (1 channel)

Table 9: Comparison of fall detection performance between system 1 and system 2

	Precision	Recall	F1-score	Accuracy
System 1	0.88	0.94	0.91	0.9549
System 2	0.89	0.81	0.85	.9149

However, the overall performance of our model decreased dramatically when we added the signal data of the accelerometer, going from 95.24% of accuracy to 71.79% for our first test case. And from 91.49% to 68.20% for the second test case, which after 5 to 6 epochs the improvement of the accuracy halted. We tried to solve this issue changing some parameter in the definition of the MLP architecture but we did not see positive impact on the final result.

The Figure. 9 and Figure. 10 show the accuracy value versus the number of epochs of System 1 and System 2 respectively. Overall, it can be seen that the value of accuracy for System1 in test phase fluctuated dramatically specially during the first 35 to 40 epochs, before started increased to just under 96%. On the other hand, accuracy of System 2 became steady around epoch 26.

Although the loss value plot versus epoch for System 1 showed some important peaks in two times, the loss value always seemed to decreased. In contrast, it was not possible to visualize a decreasing trend of the loss value in System 2.

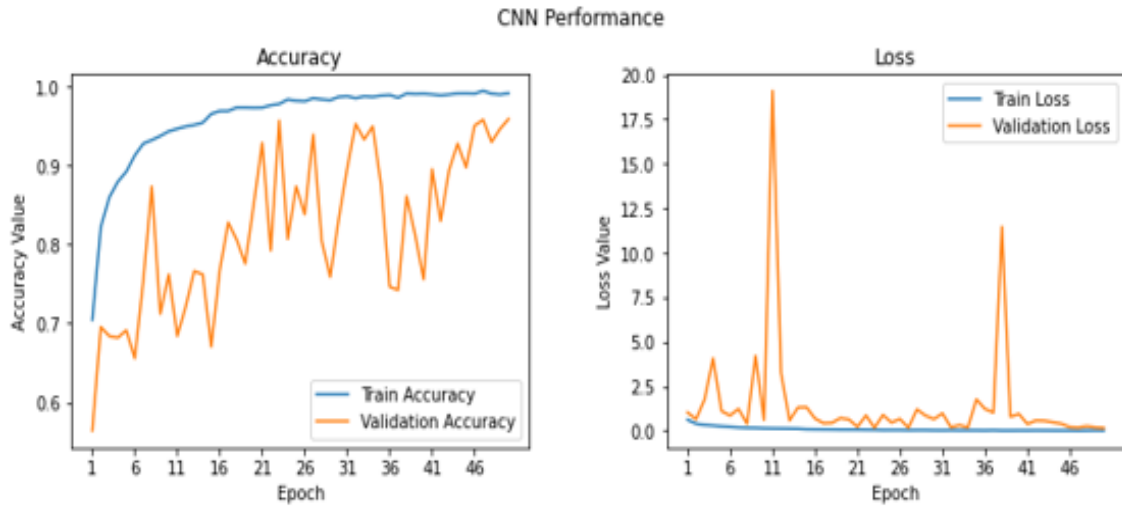


Figure 9: CNN performance of System 1. Accuracy value vs number of epochs

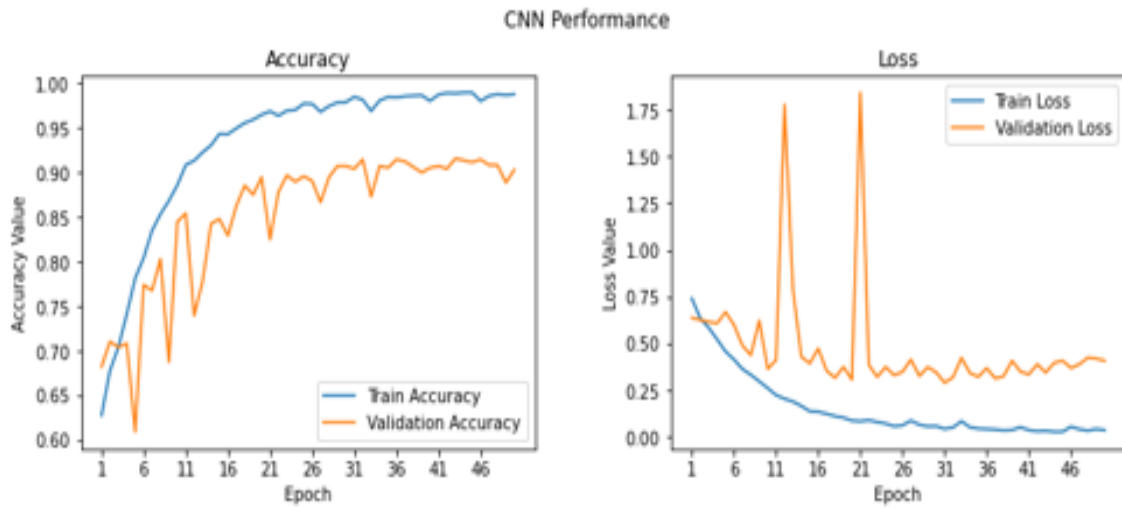


Figure 10: CNN performance of System 2. Accuracy value vs number of epochs

Table 10: Comparison of fall detection performance between all the systems

	Accuracy
System 1	95.49
System 2	91.49
System 3	71.79
System 4	68.20

8 Conclusions

Things that have been learned as a result of the work you have done. You should check once again how well the major components are linked: research questions, literature review, choice of appropriate method and techniques, and findings.

Recommendations are usually found in reports. They include a set of actionable steps with goals associated along with possible solutions.

9 Future work

In order to enhance the performance of the proposed FDS in this paper, we plan to analyse and redefine the architecture of the Multi-Layer Perceptron model and the process used to concatenate these two architectures, which may be the reason of diminishing the performance.

We also want to implemented Long-Short Term Memory model to exploit temporal information from accelerometer data and image sequences contained in UR Fall Detection dataset.

10 Recommendations

In the literature review, preprocess image data has proved to improved the overall performance of whatever ML implemented for fall detection, then, we recommend try to apply one of the image preprocessing techniques described in our theoretical framework.

11 Acknowledgments

I would like to express my special thanks of gratitude to my supervisor Dr. Nasim Hajari member of Faculty of Science, who gave me the golden opportunity to do this wonderful project which also helped me in doing a lot of research and I came to know about so many new things.

12 Appendices

```
# ===== CNN Image Data only =====
INPUT_SHAPE = (width_shape, height_shape, 3)
inp = keras.layers.Input(shape=INPUT_SHAPE)

conv1 = keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same')(inp)
pool1 = keras.layers.MaxPooling2D(pool_size=(2, 2))(conv1)
norm1 = keras.layers.BatchNormalization(axis = -1)(pool1)
drop1 = keras.layers.Dropout(rate=0.2)(norm1)

conv2 = keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same')(drop1)
pool2 = keras.layers.MaxPooling2D(pool_size=(2, 2))(conv2)
norm2 = keras.layers.BatchNormalization(axis = -1)(pool2)
drop2 = keras.layers.Dropout(rate=0.2)(norm2)

flat = keras.layers.Flatten()(drop2)

hidden1 = keras.layers.Dense(512, activation='relu')(flat)

norm3 = keras.layers.BatchNormalization(axis = -1)(hidden1)
drop3 = keras.layers.Dropout(rate=0.2)(norm3)

hidden2 = keras.layers.Dense(256, activation='relu')(drop3)
norm4 = keras.layers.BatchNormalization(axis = -1)(hidden2)
drop4 = keras.layers.Dropout(rate=0.2)(norm4)

out = keras.layers.Dense(2, activation='sigmoid')(drop4)

model = keras.Model(inputs=inp, outputs=out)
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
print(model.summary())
```

Figure 11: CNN architecture implementation in python code.

References

- [1] Falls, 2022 WHO, <https://www.who.int/news-room/fact-sheets/detail/falls>, 26 April 2021.
- [2] Falls: Older people: Royal college of nursing, <https://www.rcn.org.uk/clinical-topics/Older-people/Falls>, The Royal College of Nursing, 30 March 2022.
- [3] <https://www.un.org/en/development/desa/population/publications/pdf/ageing/WorldPopulationAgeing2019-Highlights.pdf>
- [4] <https://www.un.org/en/development/desa/population/publications/pdf/ageing/LivingArrangements.pdf>
- [5] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4932831/>
- [6] Bourke, A. K., O, B. J. V., & Lyons, G. M. (2007). Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. *Gait & Posture*, 26(2), 194–199. <https://doi-org.pbidi.unam.mx:2443/10.1016/j.gaitpost.2006.09.012>


```

286 # ===== CNN =====
287
288 def create_cnn(width_shape, height_shape, depth):
289     INPUT_SHAPE = (width_shape, height_shape, 3)
290     inp = keras.layers.Input(shape=INPUT_SHAPE)
291
292     conv1 = keras.layers.Conv2D(32, kernel_size=(3, 3),
293                                 activation='relu',
294                                 padding='same')(inp)
295     pool1 = keras.layers.MaxPooling2D(pool_size=(2, 2))(conv1)
296     norm1 = keras.layers.BatchNormalization(axis = -1)(pool1)
297     drop1 = keras.layers.Dropout(rate=0.2)(norm1)
298     conv2 = keras.layers.Conv2D(32, kernel_size=(3, 3),
299                                 activation='relu',
300                                 padding='same')(drop1)
301     pool2 = keras.layers.MaxPooling2D(pool_size=(2, 2))(conv2)
302     norm2 = keras.layers.BatchNormalization(axis = -1)(pool2)
303     drop2 = keras.layers.Dropout(rate=0.2)(norm2)
304
305     flat = keras.layers.Flatten()(drop2)
306
307     hidden1 = keras.layers.Dense(512, activation='relu')(flat)
308     norm3 = keras.layers.BatchNormalization(axis = -1)(hidden1)
309     drop3 = keras.layers.Dropout(rate=0.2)(norm3)
310     hidden2 = keras.layers.Dense(256, activation='relu')(drop3)
311     norm4 = keras.layers.BatchNormalization(axis = -1)(hidden2)
312     drop4 = keras.layers.Dropout(rate=0.2)(norm4)
313
314     out = keras.layers.Dense(2, activation='sigmoid')(drop4)
315
316     model = keras.Model(inputs=inp, outputs=out)
317
318     return model

```

Figure 12: CNN architecture implementation in python code.

```

327 # ===== Multi-layer perceptron =====
328
329 def create_mlp(dim, regress = False):
330     model = Sequential()
331     model.add(Dense(8, input_dim=dim, activation="relu"))
332     model.add(Dense(4, activation="relu"))
333     if regress:
334         model.add(Dense(1, activation="linear"))
335     return model
336

```

Figure 13: MLP architecture implementation in python code.

```

# ===== Concatenate =====
mlp = create_mlp(X_train_acc.shape[1], regress=False)
cnn = create_cnn(128, 128, 3)

# Concatenate the two streams together
combinedInput = concatenate([mlp.output, cnn.output])
x = Dense(4, activation="relu")(combinedInput)
x = Dense(1, activation="relu")(x)
x = Dense(1, activation='linear')(x)
model = Model(inputs=[mlp.input, cnn.input], outputs=x)

opt = Adam(learning_rate=1e-3, decay=1e-3 / 200)
model.compile(loss = 'binary_crossentropy', optimizer='adam', metrics=['accuracy'] )

# train the model
print("[INFO] training model...")

```

Figure 14: Concatenation phase in python code.

- [7] Latest Research Trends in Fall Detection and Prevention Using Machine Learning: A Systematic Review
- [8] Vallabh, P.; Malekian, R. Fall detection monitoring systems: A comprehensive review. *J. Ambient. Intell. Humaniz. Comput.* 2018, 9, 1809–1833.
- [9] Mubashir, M.; Shao, L.; Seed, L. A survey on fall detection: Principles and approaches. *Neurocomputing* 2013, 100, 144–152.
- [10] Rizk, H.; Yamaguchi, H.; Youssef, M.; Higashino, T. Gain without pain: Enabling fingerprinting-based indoor localization using tracking scanners. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, Seattle, WA, USA, 3–6 November 2020; pp. 550–559
- [11] http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture2.pdf
- [12] <https://www.mdpi.com/1424-8220/21/3/947>
- [13] <http://www.robotics.stanford.edu/~koller/Papers/Plagemann+al:ICRA10.pdf>
- [14] https://www.researchgate.net/publication/257879792_3D_hand_tracking_using_Kalman_filter_in_depth_space/fulltext/027a2ba80cf2195fcb2a162c/3Dhandtracking-using-Kalman-filter-in-depth-space.pdf?origin=publication_detail
- [15] <https://ieeexplore.ieee.org/document/6385968>
- [16] <http://www.joig.net/uploadfile/2021/1124/20211124052740953.pdf>
- [17] S. Ghidoni and M. Munaro, “A multi-viewpoint feature-based re-identification system driven by skeleton keypoints,” *Robot. Auton. Syst.*, vol. 90, no. C, pp. 45–54, Apr. 2017. [Online]. Available: <https://doi.org/10.1016/j.robot.2016.10.006>
- [18] 10 Lourdes Martínez-Villaseñor

- [19] https://www.researchgate.net/publication/361676991_Addressing_Privacy_Concerns_in_Dept
- [20] https://www.sciencedirect.com/science/article/abs/pii/S1077314215002659?fr=RR-2&ref=pdf_downloadrr=7274f09d8822420f
- [21] Kepski, M.; Kwolek, B. Embedded system for fall detection using body-worn accelerometer and depth sensor. In Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Warsaw, Poland, 24–26 September 2015; Volume 2, pp. 755–759. <http://fenix.univ.rzeszow.pl/mkepski/ds/uf.html>
- [22] L. Minh Dang, Kyungbok Min, Hanxiang Wang, Md. Jalil Piran, Cheol Hee Lee, Hyeonjoon Moon, Sensor-based and vision-based human activity recognition: A comprehensive survey
- [23] Vavoulas, George Pediaditis, Matthew Chatzaki, Charikleia Spanakis, Emmanouil Tsiknakis, Manolis. (2016). The MobiFall Dataset: Fall Detection and Classification with a Smartphone. *International Journal of Monitoring and Surveillance Technologies Research*. 2. 44-56. 10.4018/ijmstr.2014010103.
- [24] Vavoulas, George Chatzaki, Charikleia Malliotakis, Thodoris Pediaditis, Matthew Tsiknakis, Manolis. (2016). The MobiAct Dataset: Recognition of Activities of Daily Living using Smartphones. 143-151. 10.5220/0005792401430151. www.bmi.teicrete.gr <https://www.scitepress.org/papers/2016/57924/57924.pdf>
- [25] Medrano C, Igual R, Plaza I, Castro M (2014) Detecting Falls as Novelties in Acceleration Patterns Acquired with Smartphones. *PLoS ONE* 9(4): e94811. <https://doi.org/10.1371/journal.pone.0094811> <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0094811> pone-0094811-g004
- [26] Casilari, E.; Santoyo-Ramón, J.A.; Cano-García, J.M. UMAFall: A multisensor dataset for the research on automatic fall detection. *Procedia Comput. Sci.* 2017, 110, 32–39.
- [27] Frank, K.; Vera Nadales, M.J.; Robertson, P.; Pfeifer, T. Bayesian recognition of motion related activities with inertial sensors. In Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing-Adjunct, Copenhagen, Denmark, 26–29 September 2010; pp. 445–446.
- [28] Vilarinho, T.; Farshchian, B.; Bajer, D.G.; Dahl, O.H.; Egge, I.; Hegdal, S.S.; Lønes, A.; Slettevold, J.N.; Weggersen, S.M. A combined smartphone and smart-watch fall detection system. In Proceedings of the 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool, UK, 26–28 October 2015; pp. 1443–1448.

- [29] Charfi, I.; Miteran, J.; Dubois, J.; Atri, M.; Tourki, R. Optimized spatio-temporal descriptors for real-time fall detection: Comparison of support vector machine and Adaboost-based classification. *J. Electron. Imaging* 2013, 22, 041106.
- [30] Adhikari, K.; Bouchachia, H.; Nait-Charif, H. Activity recognition for indoor fall detection using convolutional neural network. In *Proceedings of the 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, Nagoya, Japan, 8–12 May 2017.
- [31] https://mdpi-res.com/d_attachment/sensors/sensors-17-00198/article_deploy/sensors-17-00198.pdf?version=1484917986
<https://www.mdpi.com/1424-8220/17/1/198/htm>
- [32] Yamashita, R., Nishio, M., Do, R.K.G. et al. Convolutional neural networks: an overview and application in radiology. *Insights Imaging* 9, 611–629 (2018). <https://doi.org/10.1007/s13244-018-0639-9>
- [33] Sehairi, K.; Chouireb, F.; Meunier, J. Elderly fall detection system based on multiple shape features and motion analysis. In *Proceedings of the 2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, Fez, Morocco, 2–4 April 2018; pp. 1–8.
- [34] Abobakr, A.; Hossny, M.; Nahavandi, S. A Skeleton-Free Fall Detection System From Depth Images Using Random Decision Forest. *IEEE Syst. J.* 2017, 12, 2994–3005.
- [35] Feng, Q.; Gao, C.; Wang, L.; Zhao, Y.; Song, T.; Li, Q. Spatio-temporal fall event detection in complex scenes using attention guided LSTM. *Pattern Recognit. Lett.* 2020, 130, 242–249.
- [36] <https://iee-dataport.org/documents/tst-fall-detection-dataset-v2>
- [37] Sensor-based and vision-based human activity recognition: A comprehensive survey
- [38] <https://shadab-hussain.medium.com/building-a-convolutional-neural-network-male-vs-female-50347e2fa88b>
- [39] <https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9>
- [40] Charfi, I.; Miteran, J.; Dubois, J.; Atri, M.; Tourki, R. Optimized spatio-temporal descriptors for real-time fall detection: Comparison of support vector machine and Adaboost-based classification. *J. Electron. Imaging* 2013, 22, 041106.
- [41] Sehairi, K.; Chouireb, F.; Meunier, J. Elderly fall detection system based on multiple shape features and motion analysis. In *Proceedings of the 2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, Fez, Morocco, 2–4 April 2018; pp. 1–8.

- [42] Thummala, J.; Pumrin, S. Fall Detection using Motion History Image and Shape Deformation. In Proceedings of the 2020 8th International Electrical Engineering Congress (iEECON), Chiang Mai, Thailand, 4–6 March 2020; pp. 1–4.
- [43] Htun, S.N.; Zin, T.T.; Tin, P. Image Processing Technique and Hidden Markov Model for an Elderly Care Monitoring System. *J. Imaging* 2020, 6, 49. [
<https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/> https://www.hihstore.com/?product_id=10846191143 https://www.hihstore.com/?product_id=10846192845 <https://www.analyticsvidhya.com/blog/2021/05/introduction-to-supervised-deep-learning-algorithms/> <https://oakland.edu/Assets/upload/docs/AIS/Syllabi/TaylerResearchHypothesis.pdf>
- [44] <https://commons.wikimedia.org/wiki/File:6DOF.svg> CMOS:complementary metal oxide semiconductor
- [45] Kirsty Williamson et al, *Research Methods for Students* (2002). Academics and Professionals, Chandos Publishing.
- [46] Trench, W. F., *Elementary Differential Equations* (2013). Faculty Authored and Edited Books & CDs. 8.
- [47] Nicholson, W. K., *Linear Algebra with Applications* (2019). Creative Commons License (CC BY-NC-SA), Lyryx Learning.