INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




Sistema de Deteção de Quedas Automático Baseado em Vídeo

JOÃO GONÇALO NUNES LEAL
Julho de 2023

Automatic Video-based Fall Detection System

An I3D/RNN approach

João Gonçalo Nunes Leal

Student No.: 1180723

A dissertation submitted in fulfillment of the requirements for the degree of Master of Science, Specialization Area of Artificial Intelligence Engineering

Supervisor: Dr. Zita Maria Almeida do Vale, Main Coordinating Professor, Polytechnic Institute of Porto - School of Engineering
Co-Supervisor: Hamed Moayyed, Researcher, Polytechnic Institute of Porto - School of Engineering
External Supervisor: Eng. David Mota, DevScope

Evaluation Committee:
President:
Dr. Maria Goreti Carvalho Marreiros, Coordinating Professor with Agreggation, Polytechnic Institute of Porto - School of Engineering

Members:
Dr. Zita Maria Almeida do Vale, Main Coordinating Professor, Polytechnic Institute of Porto - School of Engineering
Dr. Hugo Daniel Abreu Peixoto, Assisting Researcher at Minho University

Porto, July 3, 2023

Dedicatory

To my parents, who've made countless sacrifices to ensure that my path is filled with opportunities and promise.

Abstract

The elderly population faces difficulties in completing certain tasks independently, often requiring supervision to not only assist them but also to mitigate and notify about potential health risks. Falls, a prevalent and severe problem, pose a high risk of causing hospitalizations and fatalities. However, the aging population in developed countries is growing at an unprecedented rate, while the proportion of active age individuals continues to decline. Consequently, elderly care has become less accessible as caregivers are confronted with a larger number of patients. Nonetheless, conventional fall detection methods, typically triggered by victims themselves, are unreliable and inadequate. This thesis proposes an automatic alternative to existing methods, presenting a computer vision-based Fall Detection System (FDS) that utilizes a two-stream Inflated 3D Convolutional Neural Network (I3D) in conjunction with a Recurrent Neural Network (RNN). To enhance the available datasets, a new collection of simulated falls was created. Experimental evaluations demonstrate the superiority of this hybrid model over state-of-the-art fall detection models, achieving an accuracy of 94% and a recall value of 96%. By promptly and accurately detecting falls, a system employing this model could significantly reduce the risk of severe injuries posed to the elderly and physically disabled individuals.

Keywords: 3D CNN, automatic fall detection, computer vision, deep learning, I3D, RNN

Resumo

Os idosos enfrentam dificuldades em completar certas tarefas sozinhos e precisam de supervisão frequente, não só para assistí-los, mas também para mitigar e alertar para riscos potenciais de saúde. Quedas são problemas prevalentes e sérios, muitas vezes resultando em hospitalizações ou mortes. Contudo, nos países desenvolvidos, a população idosa está a crescer e a proporção de cidadãos de idade ativa a diminuir. Por consequência, cuidados a idosos tornam-se mais inacessíveis, já que enfermeiros são confrontados com um maior número de pacientes. Não obstante, métodos convencionais de deteção de quedas, que requerem, normalmente, a ativação por parte da vítima, não são confiáveis nem adequados. Esta tese propõe uma alternativa automática a estes métodos na forma de um sistema de deteção de quedas que incorpora uma rede neuronal convolucional 3D juntamente com uma rede neuronal recorrente. Para melhorar os datasets já existentes, uma nova coleção de vídeos de quedas foi criada. Este modelo híbrido revela ter performances superiores às de outros modelos, conseguindo uma acurácia de 94% e uma sensitividade de 96%. Ao ser capaz de detetar quedas precisa e imediatamente, um sistema que inclui este modelo poderá reduzir drasticamente o risco de ferimentos graves aos idosos e pessoas com deficiências físicas.

Palavras-chave: 3D CNN, automatic fall detection, computer vision, deep learning, I3D, RNN

Acknowledgement

I would like to thank Zita Vale and Hamed Moayyed for all the help they have provided me in the planning and writing of this thesis and articles; David Mota, for the indispensable guidance and for lending the camera used to record the original dataset; Luís Maia and Dewan Fayzur, for all their criticism and for pointing me in the right direction; Rita Sousa, for taking time out of her work to maintain and fix the AI Lab whenever it stopped working; finally, DevScope, for allowing me to pursue my project under their wing.

Contents

List of Figures

List of Tables

List of Abbreviations

ADL	**A**ctivities of **D**aily **L**ife
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
DL	**D**eep **L**earning
DT	**D**ecision **T**ree
FE	**F**eature **E**xtraction
FDDBg	**F**all **D**etection **D**ataset (from the University of) **B**our**g**ogne
FDDBm	**F**all **D**etection **D**ataset (from the University of) **B**ourne**m**outh
FDS	**F**all **D**etection **S**ystem
GRU	**G**ated **R**ecurrent **U**nit
I3D	**I**nflated **3**-**D**imensional (CNN)
IMU	**I**nertial **M**easurement **U**nit
k-NN	**k**-**N**earest **N**eighbors
LSTM	**L**ong-**S**hort **T**erm **M**emory
ML	**M**achine **L**earning
MLP	**M**ulti-**L**ayer **P**erceptron
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
RBF	**R**adial **B**asis **F**unction
ReLU	**Re**ctified **L**inear **U**nit
RGB	**R**ed, **G**reen, **B**lue,
RNN	**R**ecurrent **N**eural **N**etwork
SVM	**S**upport **V**ector **M**achine

List of Symbols

λ Set of hyperparameter values of a given model configuration

Chapter 1

Introduction

This chapter gives a contextualization to fall detection and its importance in elderly and disabled care, followed by a definition of the problem and its various restraints and requirements. Then, first, a list of the objectives of this thesis; second, the expected outcomes of this project are presented, as well as the chosen approach to solve the automatic fall detection problem. Finally, the overview of this report's structure is explained.

1.1 Context

As general standards of living increase so does life expectancy. The average, healthy person living in a developed country is expected to live well over 70 years, which, coupled with declining fertility rates, results in significant population aging [1]. Comparing the ratio between the elderly and young people segments of first-world populations over the last half-century allows us to observe a noticeable, steady, ongoing growth – today, there are fewer children per senior citizen than ever before [2]. Portugal has one of the most aged populations in Europe [3]. According to [4], in the year 2000, there were approximately 99 elderly (65 and older) per 100 children (14 and younger). That number has increased to 155 in 2017 and to 178 in 2021, meaning that the Portuguese elderly population is about to reach double the size of the children population in the following few years, if this growth continues. Furthermore, the number of active age (15 to 64) people per elderly person, referred to as the potential sustainability index, decreased from 4.2 in 2000, to 3.0 in 2017, to 2.7 in 2021 [4]. This trend is now starting to appear also in developing countries, which were once not as affected due to still maintaining high fertility rates. This creates a problem: the number of elderly people is beginning to dwarf the number of people that can effectively provide assistance and care to them. In 2019, 513,200 Portuguese seniors lived alone [5][1] and, in 2015, only 80,000 lived in nursing homes [7]. Portugal, in fact, has less solitary elderly than the European average [8]. This is not ideal, and results in many situations where senior citizens cannot look after themselves, but also do not have adequate access to elderly care.

Elderly care strives to meet the physical and social requirements pertaining to aged citizens. It can come in the form of social care, often performed by family members, friends and non-professional caregivers, or of medical care, provided by trained medical personnel in order to keep the elder person healthy and mobile. Impaired mobility is a major concern for senior citizens – the inability to climb stairs or to rise from a chair makes one disabled. This, unfortunately, is what makes falls more frequent and especially problematic, as a fall

[1] For reference, this number amounted to 5% of all Portuguese inhabitants and 21% of the elderly population in 2019 [6]

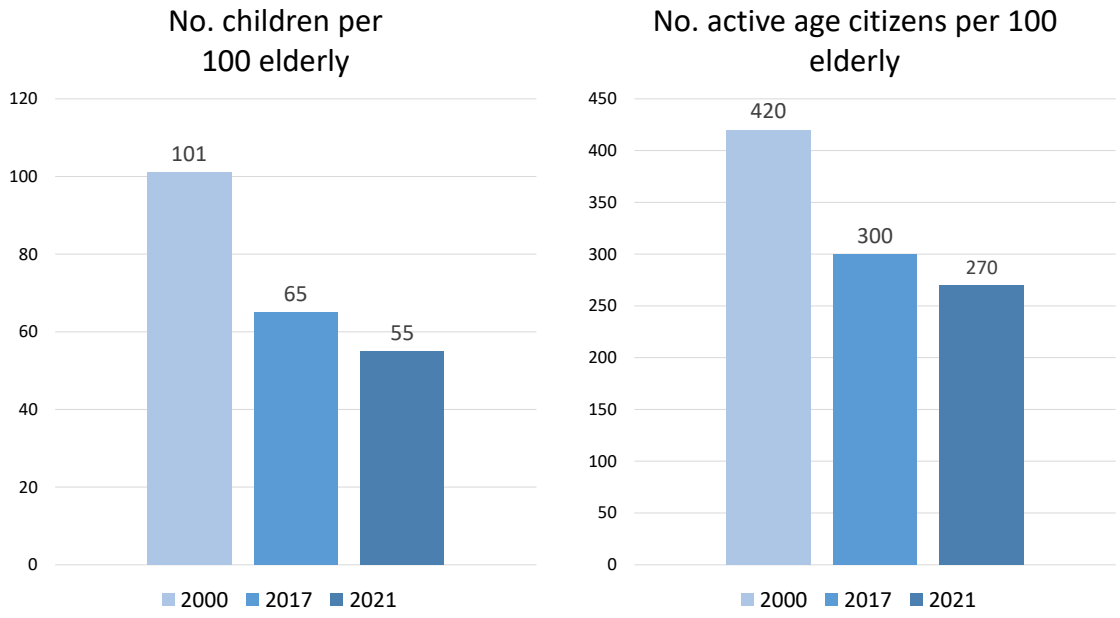

Figure 1.1: Indicators of population aging, Portugal [4].

can not only cause graver injuries than to younger adults but can also incapacitate them to the point when the act of getting up becomes impossible. Even if a senior has not suffered life-threatening impact trauma, the period of inactivity resulting from a fall is what often leads to severe medical complications [9].

A timely response to falls is thus necessary to prevent serious injuries, which requires, in the best-case scenario, the immediate detection of a fall event. Even in nursing homes, however, with dedicated staff looking over their elderly, falls can remain undetected for long periods of time, especially at secluded areas, for example, in a senior's room at night. The senior can resort to calling for help, if he can, but this is not sufficient. Nursing homes tend to place manual alarms in seniors' rooms, such as pull cord alarms, but these are stationary and often out of reach. Victims of falls cannot be expected to activate them on every occasion, much less in a timely fashion. As it stands, traditional, manual ways to detect falls, even in controlled environments like nursing homes, are impractical and ineffective. Falls are also a severe problem to the physically disabled, who might also suffer from debilitated mobility and require constant medical assistance and supervision.

Nonetheless, technology can help automate elderly care, lessening the effects of an eventual medical personnel shortage, and even improve seniors' sense of independence, which most hold in high regard [12]. For instance, instead of nurses periodically checking heart rate, body temperature and blood pressure, these vital signs could be automatically and constantly monitored by specialized sensors [13]–[16]; robots could assist in social interaction [17], [18] or facilitate mobility [19], [20]. Finally, there is the concept of automatic fall detection, which seeks to eliminate the human element in detecting and alerting to fall events. This is not a new concept, but no consensus currently exists in regard to its implementation.

In conclusion, elderly citizens require special care which can only be provided by professional personnel, which are derived from an ever-dwindling manpower pool. The probability that seniors suffer critical medical emergencies unaccompanied is thus increasing. Falls suffered by the elderly and physically disabled, for instance, tend to have drastic effects on their health

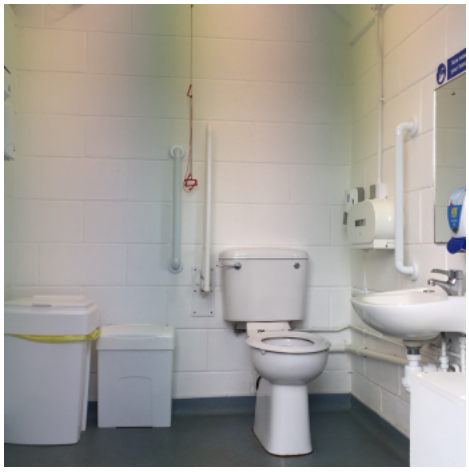

Figure 1.2: Pull cord placed next to a toilet. If the elderly falls outside the bathroom, this alarm would never be activated [10].

Figure 1.3: A wearable emergency button. Wearable alarms are widely used, but have their own set of issues [11].

and well-being. Being much more likely to result in severe injuries or death means that, in the case of a fall, its occurrence must be detected immediately, and medical assistance must be provided as soon as possible.

1.2 Problem

This thesis addresses the issue of elderly fall detection, as constant human supervision is not feasible nor practical, and manual alarms are insufficient. Therefore, a way to detect and alert to falls in real-time and automatically, meaning without any human interaction, is required, and tackled by this dissertation.

Such contraption must function as independently and as unobtrusively as possible while still providing accurate readings. The elder's daily routine and comfort should not be disrupted in any way, shape, or form either by the contraption's set up, functions or outputs, and should strive to threaten as little as possible the elder's sense of independence. This means that its presence should remain completely undetected unless the situation absolutely calls for some sort of intervention. It should not be time-specific, or, in other words, it should

function throughout the day, continuously and without interruptions. It must be capable of monitoring, at least, an entire average-sized indoor division and must also be capable of being used in a number of different room configurations. The cost of obtaining the sensors, as well as setting up and configuring the system and applying it to a standard living space, should be reasonably low.

This thesis seeks to address certain issues present in other state of the art automatic fall detection projects. These issues are mainly centered around the inefficient use of training data, which translates into inaccurate and misleading performance results. Additionally, the Artificial Intelligence (AI) model chosen to perform fall detection should be a clear improvement on the ones utilized by other researchers, either through performance or efficiency. To prove it, this model should be tested against other, more used models to see if there is a positive difference.

1.3 Objectives

The objectives set for this project are as follows:

1. Literature search and review
    - An investigation of the current advancements in the respective field, along with a comprehensive scrutiny of the tools, methodologies, and approaches employed by fellow researchers.
2. Analysis of existing Machine Learning (ML)/Deep Learning (DL) classification methods
    - An exploration of various model configurations not only in the domain of automatic fall detection but also in other related fields, with the objective of refining the approach to be adopted.
3. Collection of existing fall detection datasets
    - A compilation of openly accessible datasets, organized with the aim of generating a sufficient quantity of training data.
4. Procurement of sensor(s) for building a dataset
5. Creation of custom, private dataset to use in addition to those collected in objective 3.
    - While the compilation of available datasets offers a considerable volume of data, the development of an original dataset facilitates the incorporation of previously unexplored information. Consequently, it expands the existing collection further.
6. Data preprocessing and augmentation
    - Alterations to the final raw dataset with the goal of increasing data compatibility and diversity. Each data instance will be slightly altered, thus increasing variety in the training, validation and testing sets. Modifications to the final raw dataset in order to enhance data compatibility and diversity. Every data instance will undergo minor alterations, thereby augmenting variety in the train, validation, and testing sets.
7. Construction, training, and testing of fall detection models

- Multiple model configurations will be developed, trained, and validated. Subsequently, the model exhibiting optimal performance during the validation phase will undergo testing using new data. An analysis and discussion will ensue to evaluate the model's projected real-life performance.

1.4 Advocated approach and expected outcomes

AI has found consistent use in the detection and classification of human motions through visual data. Falls can, evidently, be identified by characteristic movements, and as such, this thesis advocates for the use of ML/DL techniques to develop a classification model capable of differentiating falls from other innocuous actions through visual means (video cameras). Models of this sort are specifically intended for elderly care, and must follow a few requirements especially regarding data privacy, but since this thesis is confined to the creation of AI models, practical implications are excluded from its scope.

ML/DL models need to be trained with a substantial amount of data to perform their intended task. However, the training a model learns for one task can be appropriated for a similar other, so called transfer learning. With this in mind, the hybrid model proposed by this thesis includes a two-stream Inflated 3-Dimensional (I3D) Convolutional Neural Network (CNN), pre-trained for general action recognition. It is theorized that its competency in that field is transferable to fall detection. It is also thought that, since it offers the best performance out of all CNNs in general action recognition, the I3D will outperform and be more efficient than other CNNs in this field as well. However, this network comprises only half of the complete model, being modified to only extract features from input videos. These features are meant to be passed as input to a Recurrent Neural Network (RNN), which is ultimately responsible for classification. Having two models work in conjunction this way has been attempted in works by others researchers and is meant to improve the detection rate of certain actions, of which falling is proposed to be one of them. Generally, this thesis affirms that the I3D/RNN hybrid model is capable of providing the best results out of any other model used for automatic fall detection at a more than adequate efficiency rate.

Furthermore, this exact hybrid model has not been used in any other visual Fall Detection System (FDS), and is hypothesized to provide not only greater performances, but also, when paired with a seldom seen method of data handling, more trustworthy results. As will be elaborated in the following chapters, several other fall detection projects fail to recognize certain limitations regarding training data, which, in turn, denigrate their system's stated results. The ensuing compilation of existing datasets alongside the creation of an original dataset will grant the advocated model an edge over other projects, but more importantly, the increased data amount gathered for this thesis allows the implementation of evaluation methods in such a way as to extract performance metrics that are much closer to a representation of the model's functioning in the real world.

1.5 Report structure

This thesis commences with Chapter 2, wherein the current advancements in the field are presented. Preceding this, a concise introduction outlines key theoretical topics such as AI, ML, and DL. Additionally, it encompasses various models employed by other researchers and the methodologies adopted to assess their efficacy. The latter portion of this chapter is dedicated to delineating the diverse applications and approaches pertaining to automatic

fall detection. Particularly, emphasis is placed on visual fall detection, encompassing an examination of existing datasets and a comprehensive review of systems that employ a similar approach.

Chapter 3 expounds upon the planned steps to be undertaken in order to fulfill the objectives of this thesis. This is further supplemented by providing an elaborate account of the tools, techniques, datasets, programs, and models utilized during the endeavor to construct an accurate fall detection model.

Chapter 4 offers an in-depth analysis of the implemented models and the meticulous scrutiny of their outputs. The findings from each detailed case study are duly presented. Subsequently, a comprehensive discussion ensues, encompassing an examination of the general outcomes attained as well as a comparative analysis vis-à-vis the findings reported by authors of other fall detection systems. Furthermore, salient remarks are made concerning the robustness of their training methods.

In conclusion, Chapter 5 meticulously outlines the extent to which all the stated objectives were accomplished, followed by a deliberation on potential avenues for future research. Concluding this chapter is a succinct section expressing final appreciations.

Chapter 2

State of the Art

This chapter is divided into two parts, starting with a more theoretical section which explains the concept of ML, how widely used classification models function, and how evaluation techniques are employed. This is useful for giving some semblance of context to the topics discussed in the second half, the exploration of existing fall detection systems and how they are constructed. The second half begins with general overview of existing automatic fall detection systems, followed by a subsection that offers descriptions of several public datasets for visual fall detection. The last subsection focuses on systems similar to the one presented by this thesis and, because of their resemblance, is separated from and goes more in depth into their explanation than the previous subsection.

2.1 Theoretical introduction

Before any real exploration of technologies, tools, and their implementations in the field of fall detection, it is worth going through a brief explanation of the concepts intrinsic to AI and, more specifically, ML, as to gain a better understanding of how these systems function, how they are developed, what are their strengths, shortcomings and how they compare to each other. In this chapter, then, a quick rundown on the theory behind ML technology will be provided, followed by a list of the more popular models used for classification and an explanation of the methods used to evaluate the performance of classification models.

2.1.1 Machine Learning

As AI becomes more advanced, what was once mainly an academic field, with few, niche practical applications, starts to become a valuable tool to tackle increasingly harder and more complex problems. Mainly, it is now able to solve problems involving a huge amount of data from an increasing number of sources. The 1970's saw the first trend towards using AI to solve real-life problems with the dissemination of expert systems, which, given input data, would follow a set of rules, known as a knowledge base, and in doing so would simulate the decision-making process of field experts [21]. This knowledge base would be constructed from interviews with actual experts who would give insight into their thought process – of course, this method is very subjective and is too dependent both on the human expert's availability and on their ability to adequately externalize that knowledge. This, added to an inability of an expert system to expand its knowledge base past a certain threshold due to performance issues, called for a novel approach to AI, one that would rely more on autonomous and sophisticated techniques and less on human interaction [22, pp. 10–11]. An approach which would gain prominence over expert systems, but not completely substitute, in these last few decades.

ML is strongly linked with AI but is also related to the fields of Data Mining, Computational Statistics and Probability, Mathematics, and, unlike rule-based systems, is based on induction – the set of rules or functions become a hypothesis created from and influenced by specific past experiences, which is then applied generally [23]. These experiences, grouped in collections called datasets, are fed into ML algorithms. These algorithms extract patterns and other useful knowledge from them in order to make predictions or descriptions, and as such they tend to need numerous experiences to learn from - this way, they can adequately represent relevant relations between features and a target output. However, as extensive as the dataset is, the algorithm's ability to adapt to unseen situations is still limited by the same factors that limit inductive reasoning: if a set of past experiences is finite it cannot hope to represent every possible future situation, so no inductive proposition can be proven implicitly true, ergo, an inductive logic can only be approximate to fact. As such, when training ML algorithms, it is necessary to minimize as much as possible bias, or failure to detect relevant patterns which leads to the creation of an unsuitable solution to the problem (underfitting), and variance, or hypersensitivity to the dataset's characteristics which results in the creation of a solution unable to adapt to situations outside that dataset (overfitting). Finding a hypothesis of matching complexity to the target function is the best way to assure the best performance and avoid under and overfitting.

The development and creation of more intricate, more sophisticated algorithms are still ongoing, and ML has found use in an increasing number of fields and applications, which include machine translation [24]–[26], autonomous vehicles [27]–[29], medical diagnosing [30]–[32], sentiment analysis [33]–[35], recommender systems [36], [37], market analysis [38], [39], and computer vision. Dividing ML are three main approaches:

- Reinforcement learning systems are based on the interactions between states, actions, policies, and rewards. As the environment changes states, policies dictate what actions the system ought to take, each resulting in a reward. The algorithm then analyzes the received rewards and alters its policies in order to maximize future ones. Programs like these can be found in self-driving cars and game AIs like chess [40] or Go [41].
- Unsupervised learning algorithms study unlabeled data and try to find recurring patterns in them. Used in finding the probability density function or performing cluster analysis, for instance.
- Supervised learning algorithms take training data made up of a set of instances which contain input features and an associated target feature, or label. The algorithm iterates through the train set and tries to learn a function that, when subject to new inputs, calculates the new instance's unseen target feature.

Supervised learning algorithms can themselves be divided into types: regression algorithms output numerical values in a continuous range, such as calculating the next iteration of a numerical sequence; similarity algorithms learn to compare two instances and measure their similarity – they are used in recommender systems, visual identity tracking/verification and so on; lastly, classification algorithms are much like regression algorithms, except these output one of a finite set of values, which are normally named classes. These can be binary, when the set of possible outputs contains only two values (for instance, a program that classifies an email as spam or not spam), or multiclass, when the set contains more than two values (a program that scans written text must associate every written symbol to a letter class).

2.1.2 Popular classification models

The resulting set of functions and calculations of training a ML algorithm are contained in what is known as a model. This chapter is dedicated to a brief explanation of the algorithms that are most widely used for creating both binary and multiclass classification models.

Nearest Neighbor

Nearest Neighbor (NN) algorithms are one of the simplest and most straightforward classification algorithms. They work by measuring the proximity between attributes of an instance of unseen data and those of known instances and labeling that new instance according to those nearest to it, following the assumption that similar instances are distributed close to one another. They come in variations, depending on the number of neighbors considered for classification. The simplest only checks the closest known instance and classifies the new one as the same class, called 1-Neartest Neighbor, or 1-NN. Algorithms that take more than one neighbor into consideration are collectively referred to as k-Nearest Neighbors (k-NNs), with k standing for number of neighbors [42]. A 5-NN algorithm, for instance, calculates the five nearest instances to new instance p and assigns it the class that is most prevalent in its neighbors. If three neighbors are class "positive", then p is "positive" as well.

To calculate proximity between instances implies defining a metric that represents distance. If p and q are two instances of data and n is the number of features, this metric could be Euclidean distance,

$$d_e(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}, \tag{2.1}$$

or, to put it simply,

$$d_e(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}, \tag{2.2}$$

Manhattan distance,

$$d_e(p, q) = \sum_{i=1}^{n} |p_i - q_i|^2. \tag{2.3}$$

among other formulas like Hamming distance and Minkowski distance, which are not so often used [43].

k-NN is a "lazy" model, meaning it stores train data in memory and only performs computations at the prediction stage, foregoing the traditional protracted training stage like those of other algorithms. It is referred also as an instance- or memory-based method for this reason. Choosing k is up to the user's choice but some strategies like r-fold cross-validation[1] could be employed to somewhat optimize k, odd numbers being obviously favored in order to avoid ties. Generally, small k values are especially vulnerable to noise and outliers and large k values result in increased computational loads. It is up to the user to find that happy medium.

Despite its simplicity, this model has found its uses in rather complex problems and, due to being "lazy", training takes no time at all, which is a plus if the situation calls for learning additional train data. On the other hand, as the amount of data increases, so does the

[1] Discussed in section 2.1.3

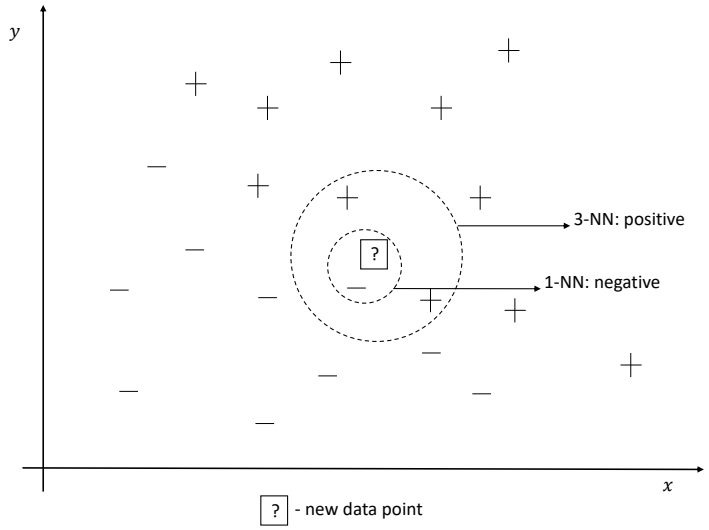

Figure 2.1: Example of Nearest Neighbour classification and k's influence on the model's performance.

computational load – past a certain dataset size, using k-NNs becomes impractical [22, pp. 76–82].

Decision Trees

Decision Trees (DTs) are made up of several interconnected nodes that form a tree-like structure, with end nodes being referred as "leaves", the first (top, conventionally) node as the "root node" and the connections as "branches". Each node represents a conditional operation that "tests" a variable and each branch coming out of that node represents one of its outcomes. Starting from the root, a data instance has its features tested repeatedly through several nodes and branches until a leaf is reached, which dictates the instance's class [44]. In a way, a DT decides which class an instance belongs to much the same way as an expert system rationalizes its thought process: by eliminating outcome possibilities through the application of a series of logical rules. Unlike expert systems, DTs form their own rules in the training process.

The algorithm takes train data and splits the set along the outcomes of variable tests. It first chooses a variable test to become the root node. Every outcome of that test is turned into a branch and other variables are chosen to become the following nodes. After all variables are tested, the final splits should contain only instances of a single class. If not, the process is repeated from the beginning. Optimally, the root node has maximum "split goodness" and allows the algorithm to gain the most amount of information and the algorithm does employ some heuristics to determine the usefulness of a split: if the proportion of each class remains the same after the split, it is useless; if all partitions contain instances of a single class, it has maximum usefulness.

The algorithm, then, must discriminate between attributes on their ability to maximize information gain and reduce uncertainty in predicting the target feature. Information gain can be defined as the entropy difference of a partition before and after a split. Entropy is defined as

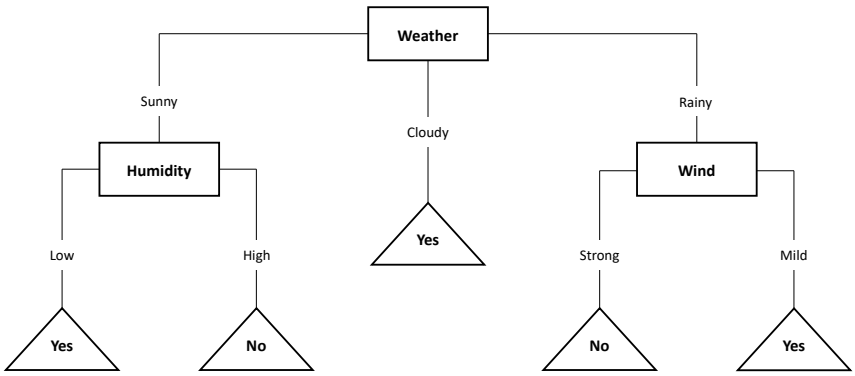

Figure 2.2: A simple decision tree that decides whether it is a good day to play tennis, with triangles as classes, rectangles as variable tests and branches as outcomes.

$$H(X) = -\sum_{i=1}^{n} P(x_i) \times \log_b P(x_i) \tag{2.4}$$

in which X is a set of values $x_1, x_2, ..., x_n$, $P(x_i)$ is the probability of observing a certain value and b is the logarithm base, most commonly 2, e or 10. The difference between entropies of a node n and a candidate split s at that node is thus defined as information gain:

$$IG(n, s) = H(n) - H(s, n) \tag{2.5}$$

DTs are robust, flexible models that stand out due to their great interpretability and efficiency ($O(n) = n \times \log n$). Their relatively weak accuracy can be remedied by constructing several DTs and selecting the most common output, a technique called Random Forest, although these are harder to interpret [22, pp. 103–111].

Support Vector Machines

Support Vector Machines (SVMs) calculate a separation line, called a hyperplane, between instances of data, consisting of vectors, in an p-dimensional space. When a new vector is fed to an SVM, it is projected on that same feature space and classified based on which side of the hyperplane the new vector lands on. If the problem is linearly separable, hard-margin linear SVMs are used. Given a dataset X with n objects belonging to a $y_i \in Y$ class, $Y = \{-1, +1\}$, the hyperplane is defined by

$$h(x) = w \cdot x + b \tag{2.6}$$

where $w \in X$ is its normal vector and $\frac{b}{||w||}$ is its offset from the origin. This separates the feature space into two regions, $w \cdot x + b > 0$ and $w \cdot x + b < 0$, and a signal function is used to classify new vectors

Table 2.1: Common types of kernels and their respective functions

Type of kernel	**k(x,y) function**
Linear	Equation (2.10)
Radial Basis Function (RBF) – Gaussian	$\exp(-\sigma\|\|x-y\|\|^2)$
RBF - Laplace	$\exp(-\sigma\|\|x-y\|\|)$
Polynomial	$(\delta \cdot (x-y)+\kappa)^d$
Sigmoidal	$\tanh(\delta(x \cdot y)+\kappa)$

$$g(x) = \text{sign}(h(x)) = \begin{cases} +1 \text{ if } w \cdot x + b > 0 \\ -1 \text{ if } w \cdot x + b < 0 \end{cases} \tag{2.7}$$

Equation (2.7) allows for the creation of an infinite number of hyperplanes. The optimal hyperplane, however, is said to be one with the largest possible margin and whose closest vectors satisfy the following equation:

$$|w \cdot x_i + b| = 1 \tag{2.8}$$

Therefore, no training vectors are located in $-1 \leq w \cdot x_i + b \geq 0$, hence the name "hard-margin".

Unfortunately, most problems are not linearly separable, requiring the use of soft-margin linear SVMs, which implement a hinge loss function, allowing vectors to violate equation (2.7) to a certain point, or non-linear SVMs, which map the original feature space to a higher-dimensional one where a linear hyperplane can be constructed with more ease. Of course, this mapping process usually implies unreasonably costly operations, but non-linear SVMs use what are called kernel tricks to keep computational costs down [45]. This mapping can be represented as $\varphi : R^N \to F$, F being the higher-dimensional feature space, which is defined as the dot product of the vector pair (x, y),

$$k(x, y) := (\varphi(x) \cdot \varphi(y)) \tag{2.9}$$

These vector pairs can be defined in kernel functions, however, such as

$$k(x, y) = (x \cdot y)^d \tag{2.10}$$

which correspond to equivalent mappings to linearly separable feature spaces, where the steps a linear SVM takes to find a hyperplane can be replicated. The type of kernel chosen does, in fact, affect the model's performance. Table 2.1 includes some of the most commonly used kernels.

SVMs generally have a good capacity for generalization, do a decent job when faced against high-dimensional data, as opposed to other algorithms which tend to under or overfit, and are efficient thanks the kernel trick technique. Nevertheless, they suffer from high sensibility to outliers and low interpretability [46], [22, pp. 153–161].

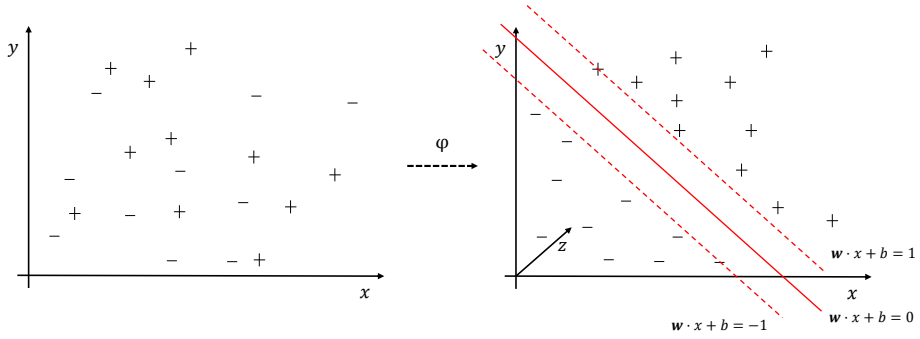

Figure 2.3: Mapping from a non-linearly separable feature space to a higher-dimensional feature space.

Neural Networks

In an attempt to replicate the functioning and structure of naturally occurring neural networks on a basic level, researchers in the 1940's and 1950's theorized the concept of Artificial Neural Networks (ANNs) and, in 1958, the perceptron was developed, becoming the first ANN. They consist of interconnected nodes, or neurons, in constructs of one or more dimensions. Each neuron comes equipped with a function that transforms input data into an output which is then transferred to the next neuron through connections called edges. These nodes are arranged in ordered layers, the first being called the input layer, which receives input data, the last being called the output layer, which presents the model's final output, and layers in between the two referred to as hidden layers, although these are not obligatory. If all neurons of layer n are connected to every neuron of layer $n+1$ those layers are considered fully-connected, if groups of neurons in n are connected to one neuron in $n+1$, those are called pooling layers. Data travelling between nodes is regulated by node and edge weights that increase or decrease signal strength, representing the importance of the operations performed by a certain connection. These weights are obtained, altered, and optimized in the training process [47].

A neuron accepts the value resulting from

$$u = \sum_{i=1}^{d} x_i w_i \tag{2.11}$$

with d being the number of incoming edges and x_i and w_i being an input value and a weight value, respectively, that belong to one of the neuron's incoming edges. u is then applied to the neuron's activation function, f_a, and its output is sent to the next neurons. There is a wide range of activation functions to choose from. For example, f_a can be linear, with $f_a(u) \in [-\infty, +\infty]$, binary step, $f_a(u) \in 0, 1$ or $-1, 1$, sigmoidal, $f_a(u) \in [0, 1]$, or Rectified Linear Unit (ReLU), $f_a(u) \in [0, +\infty]$.

The type of ANN discussed so far is called a feedforward NN, or Multi-Layer Perceptron (MLP), but some ANNs have special traits or constructs which allow them to perform operations that MLPs cannot. In a RNN, neurons can be connected to neurons in the following layer as well as to those in the same and even previous layers. These special connections are called feedback loops and allow RNNs to capture patterns in sequential and time-series data, making them especially suited to problems such as stock market predictions and language processing. Long-Short Term Memory network (LSTM) are RNNs which

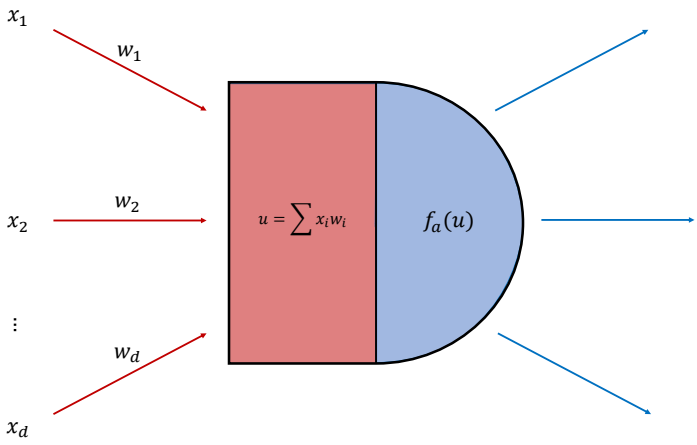

Figure 2.4: An artificial neuron.

possess a special neuron layer called memory cells, which preserve or discard past data after a certain amount of time, essentially "remembering" or "forgetting". This process is regulated with the help of input, output and forget gates. Gated Recurrent Unit networks (GRUs) function similar to LSTMs with forget gates but require less parameters due to the lack of an output gate. GRUs have been shown to perform better in some situations when compared to LSTMs [48].

CNNs contain convolutional layers in addition to traditional fully-connected and pooling layers. These NNs convolve filters, or kernels, across an image, for example, that activate when patterns are detected in the area being analyzed, forming an activation map. Unlike normal NNs, ANNs are capable of differentiating data based on features regardless of their spatial location. Each neuron belonging to a convolutional layer is related to a subregion of the input data, a receptive region. Despite this, CNNs are not more complex or larger than the average NN, which makes them quite efficient at processing large amounts of data without needing many additional computational resources [49].

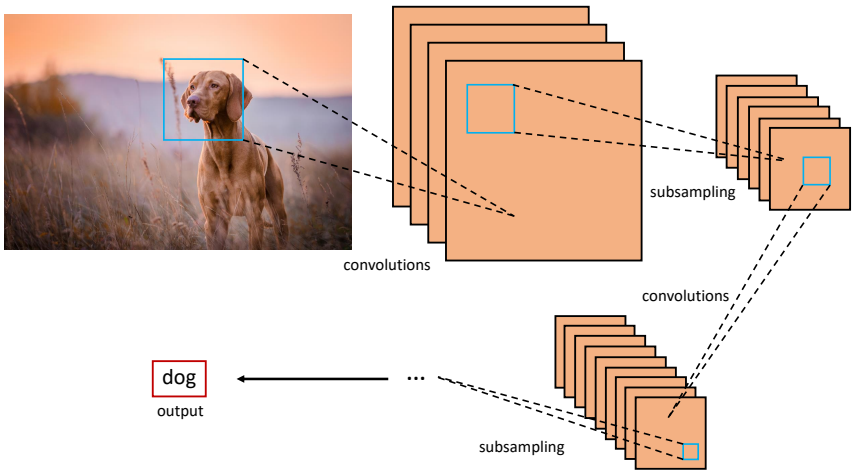

Figure 2.5: Typical CNN operations.

Back propagation is an algorithm that is used for adjusting edge weights while training NNs.

It works in two phases: a forward phase and a backwards phase. In the first phase, train data is fed to the NN until all output layer neurons produce an output value which is then compared to a desired value - the difference between the two values is the error. In the second phase, the error value is used to adjust the connection weights of the output layer and a new error value is calculated again for use in the previous layer. If calculating the deviation of an output neuron using a quadratic error function is defined as

$$e_l = \frac{1}{2}\sum_{i=1}^{k}(y_i - \hat{f}_i)^2, \tag{2.12}$$

with e_l as the error of the l-nth neuron of an output layer containing k neurons, y_i as the actual output and $\hat{f}_i$ as the desired output, then calculating the error value of the previous layer's neurons becomes

$$\delta_l = \begin{cases} f_a' e_l \text{ if } n_l \in c_{output} \\ f_a' \sum w_{lk}\delta_k \text{ if } n_l \in c_{hidden} \end{cases} \tag{2.13}$$

with δ_l being the error value of n_l neuron, f_a' being the derivative of the neuron's activation function (which is usually sigmoidal in NNs capable of backpropagation), w_{lk} being the connection weight between n_l and n_k of layer $c_{(l+1)}$. This phase starts all the way back from the output layer, moves on to the input layer and uses δ_l to slightly adjust edge weights. The entire process is then repeated.

ANNs are some of the best performing models for complex problems, especially deep NNs. The big downside is their "black-box"-like nature, since their knowledge is based on an enormous collection of parameters which are altered by complex mathematical formulas, making their decisions entirely incomprehensible to humans [22, pp. 132–149].

Additionally, an ANNs could be referred to as a deep neural network if it contains several hidden layers, at least two [50]. These networks are normally capable of performing more advanced operations than the traditional, "shallow" ML algorithms, such as convolution or recurrency. Therefore, CNNs and RNNs, including LSTM and GRU networks, as well as deep belief networks [51], [52] and deep reinforcement learning methods [53], [54], are part of a group of techniques called DL, a subset of ML. One of their core characteristics is the ability to take raw, unaltered data and extract more complex, higher-level patterns [55].

Due to this ability, these algorithms are ideal solutions for problems that require the analysis of large, high-dimensional data, such as speech and text recognition, computer translation or computer vision. The scientific field of computer vision concerns itself with finding ways to convert visual data into meaningful symbolic information [56] – in the form of object, motion or event detection, video tracking or image restoration, for instance. While researchers are not just limited to using DL techniques to accomplish these tasks, CNNs and RNNs have been instrumental in advancing the field, producing results that far outclass other methods.

2.1.3 Quality evaluation methods of classification models

Once built and trained, these models have their performance measured using test data, which is usually partitioned from the original dataset alongside train and, optionally, validation data. The holdout method involves the separation of the whole dataset into a train set of

proportion p and a test set of proportion $(1-p)$. The proportion of the test set depends on the type of model, size of dataset, task specific variables and is defined on a case-by-case basis but is between 5%-20% the size of the train set in most cases. Unlike the other sets, data from the test set is, supposedly, never exposed to the model during any step of the model's training stage – it is unseen data, in order to provide as much of an accurate portrayal of the model's behavior in a real-life situation, where all data is unknown to it.

The holdout method, however, is not perfect and might result in inadequate data partitions to test the model with, either because of a lack of variety in data, imbalance in the number of class instances or the overrepresentation of "easy"-to-predict data. Methods that seek to rectify this include random subsampling and r-fold cross-validation, which consist in measuring the model's quality by the result average of several training and evaluation processes done on random, repeated partitions of the same dataset. In the case of the former, holdout is performed several times by random sampling to obtain several sets and, the latter, involves splitting the dataset into r parts (folds) and choosing $r-1$ folds to train the model and the remaining one to test it, repeating the process until all folds have been used as the test set [22, pp. 194–196].

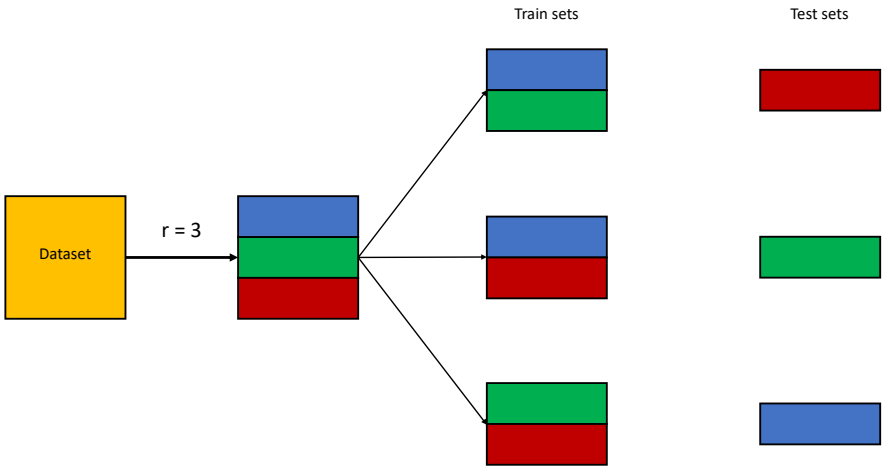

Figure 2.6: r-fold cross validation.

Classifiers can be divided into binary and multiclass. The former assign either one of two classes to a specific object and these classes, even if already named to suit the task these models are made for by the researchers for the sake of convenience, can nevertheless be defined as Positive and False. Which class is which is at the discretion of the researcher but should be self-evident (in the case of fall detection, most researchers consider a Positive classification as an occurrence of a fall). After the model is tested and has classified all test data, the results can be observed and analyzed using a confusion matrix:

Table 2.2: Confusion matrix

Predicted/Real	**P**	**N**
PP	TP	FP
PP	FN	TN

In which True Positives (TP) is the number of instances of Positive predictions (PP) correctly assigned to Positive objects (P), True Negatives (TN) are instances of Negative predictions (PN) correctly assigned to Negative objects (N), False Positives (FP) are instances of Positive predictions (PP) incorrectly assigned to Negative objects (N) and False Negatives (FN) are instances of Negative predictions (PN) incorrectly assigned to Positive objects (P). There are several metrics which allow researchers to evaluate and compare classifiers on their performance and efficacy. Evaluation metrics are calculated from the values above:

Table 2.3: Binary evaluation metrics

Accuracy (Acc.)	$ACC = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+TN+FP+FN}$
Sensitivity, recall or true positive rate (Sens.)	$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$
Specificity or true negative rate (Spec.)	$TNR = \frac{TN}{N} = \frac{TN}{TN+FP}$
Precision or positive predictive value (Prec.)	$PPV = \frac{TP}{TP+FP}$
F1 score	$F_1 = 2 \times \frac{PPV \times TPR}{PPV+TPR} = \frac{2TP}{2TP+FP+FN}$
Miss rate or false negative rate (miss.)	$FNR = \frac{FN}{P} = \frac{FN}{FN+TP} = 1 - TPR$

Table 2.3 only includes a small sample of all useful metrics, but these are the most common. The first five are widely used by researchers when evaluating their models, such that, in most of their works, one or more of these is always used to gauge model performance, with accuracy being the most common. In the scope of fall detection, however, miss rate is usually absent and there are even cases when the researchers omit sensitivity values, despite both being very important metrics which researchers should prioritize, since false negatives, or failures to detect an actual fall, are much more serious errors than false positives, or detecting falls when none has occurred.

Multiclass classifiers, on the other hand, cannot have the same metrics calculated in the same manner, since there are more than two classes. A multiclass problem, however, while not as obvious, could be seen as multiple binary classification problems instead. For instance, a model could be trained to differentiate between pictures of dogs, cats, and horses, each of these animals being one class. In this example, the model is classifying images simultaneously as '*dog*' or '*other*', as '*cat*' or '*other*' and as '*horse*' or '*other*'. This way, we can calculate the same metrics by combining each binary classification's measures and taking their average. Some metrics can also be calculated on a macro level, where each class is weighed equally, and on a micro level, where each instance has equal weight and any discrepancy in the number of instances of each class is ignored. Accuracy, however, is not calculated on either level, and F1 score is only measured on the macro level. If k is the number of classes, M is macro level and μ is micro level, then the same six metrics could be calculated as shown in table 2.4.

2.2 Fall detection systems

Without any way to detect falls automatically, elderly, and disabled people who have suffered a fall can only receive medical attention if the fall was witnessed by an observer, if someone hears their calls for help, if they set off some sort of manual alarm such as a pull cord or

Table 2.4: Multiclass evaluation metrics

Metric	M	μ
Accuracy (Acc.)	$ACC = \frac{\sum_{i=1}^{k} \frac{TP_i+TN_i}{TP_i+TN_i+FP_i+FN_i}}{k}$	
Sensitivity (Sens.)	$TPR_M = \frac{\sum_{i=1}^{k} \frac{TP_i}{TP_i+FN_i}}{k}$	$TPR_\mu = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k}(TP_i+FN_i)}$
Specificity (Spec.)	$TNR_M = \frac{\sum_{i=1}^{k} \frac{TN_i}{TN_i+FP_i}}{k}$	$TNR_\mu = \frac{\sum_{i=1}^{k} TN_i}{\sum_{i=1}^{k}(TN_i+FP_i)}$
Precision (Prec.)	$PPV_M = \frac{\sum_{i=1}^{k} \frac{TP_i}{TP_i+FP_i}}{k}$	$PPV_\mu = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k}(TP_i+FP_i)}$
F1 score	$F_{1M} = 2 \times \frac{PPV_M \times TPR_M}{PPV_M+TPR_M}$	-
Miss rate (miss.)	$FNR_M = 1 - TPR_M$	$FNR_\mu = 1 - TPR_\mu$

if they got up and went to search for it themselves. Obviously, this is not ideal, as falls are usually incapacitating, and caregivers cannot be expected to be constantly present. Pull cords and other manually activated alarms are quite common, especially in nursing homes, but are clearly not sufficient since, even if the affected person is not totally incapacitated, these do not guarantee a timely activation. However, a system that could detect or even predict falls without the need for human intervention would be certainly lifesaving.

There has been a substantial amount of research in this field in order to create and improve such systems. Nowadays, this technology is still in its infancy and very few systems have passed the prototype phase and progressed on to markets, with researchers producing new implementations fairly regularly with vastly different approaches.

This section is dedicated to explaining these systems which, to present them in a more organized manner, have been categorized as being either a wearable system, an environmental system, or a visual system, based on what kind of sensors they use. After this, a description of visual public datasets (as in, free of charge and accessible without the need for any registration or permission) is given, as it pertains to the later section about systems which more closely resemble this project and whose choice of dataset is important in their discussion.

2.2.1 Overview of fall detection technologies

Wearable systems

Wearable systems base their operations on data originating from wearable devices, or devices that must be on one's body to perform their intended function. These usually consist of accelerometers, which measure a body's acceleration in its own instantaneous rest frame, or gyroscopes, which measure orientation and angular velocity. These sensors can be embedded in belts, wrist, or ankle bracelets and recently, as they become more common, smartwatches and smartphones.

A big advantage of using these kinds of sensors is that systems based on them tend to be rather simple and, as a result, cheap and accessible. Some of these use basic threshold algorithms with some degree of success, such as in the work of Fudickar et al. [57]. Nowadays, app stores feature a selection of threshold-based fall detection applications for smartphones and smartwatches usually come packaged with one already built-in. A few systems even use threshold and ML algorithms at the same time and in conjunction to improve accuracy, like in Wolfe et al. [58]. However, it is apparent that ML algorithms are more robust and precise. Waheed et al. [59] used RNNs to perform noise-tolerant detection of falls through the use of Inertial Measurement Units (IMUs), which are wearables containing accelerometers, gyroscopes and, occasionally, magnetometers, in order to achieve an accuracy of 97.21%. Leu et al. [60] installed triaxial accelerometers and gyroscopes on moblie phones, whose signals are classified by a DT, along with being subdivided into six different types of falls, with an accuracy of Activity of Daily Life (ADL) classification of 98.46% and fall classification of 96.57%. Yu et al. [61] used Hierarchical Attention-based CNNs with 98.59% precision and 97.58% sensitivity rates. These systems can also be used to predict falls, not just detect them: in Shibuya et al. [62], subjects wear sensors on their backs or waists that can tell if a fall is about to occur based on the subject's gait.

Figure 2.7: Wireless Gait Analysis Sensor used by Shibuya et al. [62].

Despite being the most commercially abundant systems, wearables are inherently flawed. Besides their lower accuracy, and oftentimes the cause for it, these systems are overly sensitive to noise, especially if they are placed on the wrists or ankles, as the extremities are subject to sudden jerking motions more frequently which confuse the sensors. On top of that, they only work if they are worn, which is an issue as they tend to be uncomfortable to wear and elderly people are rather prone to forget or refuse to put them on. Even if they do not, they have to come off at some point during the day. Lastly, there is also a chance that the device is broken as a result of the fall.

Environmental systems

Systems that use acoustic, vibration, pressure or (active, short wavelength) infrared sensors, radars, and WiFi signals to detect falls are called environmental and surveil a single room or division. Some researchers consider camera-based systems, the ones explained later, also as being environmental, however, although there is some merit to their logic, it is worth separating the two, since computer vision methods cannot be used in this type of systems.

Adnan et al. [63] and Droghini et al. [64] are examples of systems that use acoustic sensors that pick up on sound wave patterns to differentiate falls from ADLs. Vibration

sensors, which include piezoelectric and accelerometric sensors, are quite similar but capture pressure waves transmitted through the floors and walls, such as in Alwan et al. [65], who use a piezoelectric sensor, or in Muheidat et al. [66], who use pressure sensor pads covered by a carpet to read the movements of the subject who treads on them. Infrared sensors have been used by Fan et al. [67] to detect falls with LSTMs and GRU models and by Nishiguchi et al. [68] not to detect falls, but to assess fall risk according to step speed and accuracy measurements. More recently, radars and wireless signals have shown to have great potential to detect falls by systems such as the ones by Tang et al. [69], which measures distance between shoe-mounted radars and objects in front of them to prevent falls, Amin et al. [70], which analyses the Doppler effect present in the radar signals corresponding to a human's motions, and Wang et al. [71], which performs activity recognition by measuring Channel State Information phase difference between two common WiFi devices.

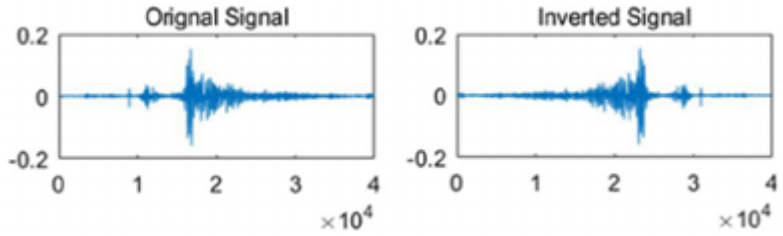

Figure 2.8: Acoustic signal analysis in Adnan et al. [63].

Environmental sensors are not intrusive nor obstructive and, unlike wearables, can detect falls without the need to be adjusted or turned on every day, due to their "place-and-forget" nature. On the other hand, their detection range is limited and their performance tends to suffer from the existence of "blind spots" and a hypersensitivity to noise: infrared sensors capture other heat sources and so readings can be often obfuscated by gases, unlike thermal imaging; vibration sensors are too dependent on floor material and carpets to give accurate readings and, like acoustic sensors, frequently mistake the fall of objects and miscellaneous sounds both inside and outside a division as actual, human falls. Unfortunately, these disadvantages are normally ignored by researchers, and not nearly enough steps are taken to remedy them.

Visual systems

Visual systems use computer vision algorithms to capture falls through camera feeds. These feeds could originate from normal cameras, which fill every frame with pixels containing color values (Red, Green, Blue (RGB), tentatively), thermal cameras or depth cameras. Fall detection could function through frame/image analysis, such as pose recognition, or frame sequence analysis, by examining the dynamic of certain movements frame by frame. The possibility of using characterization, Feature Extraction (FE) and activity recognition models, which have been perfected by implementations in other areas, and, through transfer learning, adapting them for fall detection grants these systems an edge performance-wise. Cameras are now cheaper and better than ever, which means implementation costs do not have to be impractically high as before, as shown in De Miguel et al. [72].

However, costs are still high by comparison, especially in the case of thermal and, to a lesser extent, depth cameras. In order to provide a better accuracy and eliminate blind spots, more than one camera should be used which, together with having to place them in several divisions (these systems are obviously limited by visual line-of-sight), drives up the

cost significantly. DL algorithms, which are otherwise perfectly suitable to image data, are also notoriously resource hogs, needing more computational power to function properly than the average ML program, and, on top of that, they tend to be slow. If RGB cameras are used then a big issue arises in the form of a serious breach of privacy, which has indeed put many researchers off using them in favor of less intrusive methods. Thermal and depth imaging are more than adequate solutions to this problem, but if the use of RGB cameras is insisted on, then data protection methods must be employed. The choice of using thermal or depth cameras over RGB cameras does alleviate the privacy issue, but comes at the cost of effective range and image definition, which may result in worse accuracy scores. But unlike thermal vision, whose main purpose (reading the temperatures of objects) is almost entirely useless for fall detection, depth cameras possess the ability to better capture changes in distances, which could be an advantage if the model is trained to adequately process that information.

Nevertheless, while obtaining train data is difficult regardless of camera type, as the following section explains, there are very few existing thermal or depth fall detection datasets, limiting implementations of thermal/depth FDSs. A more in-depth exploration of visual FDSs is made in section 2.2.3, after an enumeration of the currently available, public visual fall detection datasets.

2.2.2 Public datasets for visual systems

As computer vision technologies become more advanced, the need for more extensive datasets increases. Since the millennium, several of these were compiled from numerous sources for a lot of different end uses. Some public datasets for image classification are truly massive: the National Institute of Standards and Technology (NIST) Special Database 19 [73] contains handprinted sample forms from 3699 writers which add up to 814,255 images of isolated handwritten letters, originally released in 1995 by the american NIST [74] but released again in 2016 as a second edition, and is widely used in testing computer recognition of handwritten text, constructed from Special Database 1, 2 and 7; MNIST [75], itself a 70,000 images sized, normalized subset of NIST SD-1 and -3, is very popular among beginner AI developers; ImageNet [76] contains 14,197,122 labeled images organized in the WordNet hierarchy with bounding box annotations and, while the full dataset can only be used with permission from its developers, a subset of 2,025,721 images is available on Kaggle [77]; the 80 Million Tiny Images dataset [78] is made up of 79,302,017 32x32 color images labeled with 75,062 nouns, but has since been taken down by the developers.

There are plenty of video classification datasets as well, although not quite as big: Kinectics-700 [79] has at least 700 video clips, extracted from Youtube, for each 700 classes; the Cityscapes dataset [80] offers stereo video sequences from 50 cities with pixel annotations of 5,000 frames and 20,000 weakly annotated frames; UCF-101 [81] has 13,320 videos of 101 action classes, collected, again, from Youtube. Unfortunately, even though these and most general action recognition datasets cover a substantial number of different classes, 'falling' is seldom included. In fact, vision-based FDSs, whether these are based on RGB, thermal or depth cameras, are dependent on publicly available datasets made specifically for fall detection, of which there are few and whose quality is, excluding some, subpar.

None of the fall detection datasets include more than 1,200 videos, which is a hinderance in a computer vision project, as DL algorithms in general need a huge amount of data to be acceptably robust. What these datasets lack in size they should make up for it in diversity

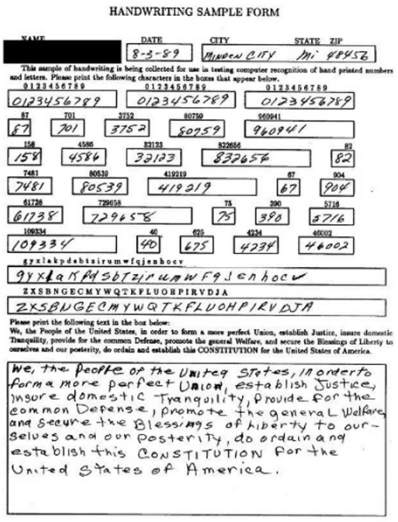

Figure 2.9: Handprinted sample form, NIST SD-19 [73].

of data, which means including different kinds of falls (falling backwards, forwards, sideways, falling from standing up, from sitting, from lying, while holding objects, while walking with canes, crutches and walkers, so on), different backgrounds (different rooms, lightings, camera angles, other people), different actors (body shapes, clothes, gaits), different kinds of innocuous actions, or ADLs, obscured actions and active attempts to simulate falls as realistically as possible, without putting the actor in the risk of injury – but in these aspects most datasets also fall short.

Table 2.5 contains characteristics of all publicly available visual datasets designed around fall detection in mind. Some, like UP-Fall [82], offer wearable data as well, making them multimodal. By numbers alone, these pale in comparison to the aforementioned general action recognition datasets: UP-Fall, the biggest, contains 510 falls and 612 ADLs for a total of 1,122 videos; the smallest dataset, FOE.MMU [83] contains 21 fall events and 30 ADLs recorded with just one camera; altogether the collection of datasets has an average of 178 falls and 101 ADLs per dataset and a total of 1,685 falls and 1,118 ADLs, not counting Fall Detection Dataset from the University of Bournemouth [84][2] since, although it is meant for detecting falls, offers no sequence of images of falls occurring, only 22,636 isolated frames of lying, standing, sitting, bending and crawling poses. Since falls cannot be gathered from actual surveillance systems, due to privacy concerns, or reliably extracted from Youtube videos, for instance, so fall events have to be painstakingly simulated and recorded manually, so this shortcoming is to be expected from these kinds of datasets.

Fall diversity also leaves a lot to be desired: Multicam [85] only has 22 different scenarios which include falls and ADLs and 2 scenarios dedicated solely to ADLs, which is compensated by having eight cameras record every scenario from different angles. In fact, other datasets make up for the lack of variety by having multiple cameras record the same events. Most dataset creators also did not pay much attention to having a lot of fall type variety beyond simple characteristics like direction, exceptions being eHomeSeniors [86], High Quality Fall Simulation Data (HQFSD) [87] and FDDBg [88], which feature a great amount of variety.

[2]Since there are two datasets named "Fall Detection Dataset" from two universities, henceforth they will be abbreviated as such: Fall Detection Dataset from the University of Bourgogne (FDDBg),Fall Detection Dataset from the University of Bournemouth (FDDBm).

Table 2.5: Existing visual datasets (cont. next page)

Dataset	Number of Actors	Falls	ADLs	Backgrounds	Fall diversity	Sensors	Additional information
UP-Fall [82]	17	510	612	1	Limited	- 5 IMUs (wearables equipped with accelerometer, gyroscope, and luminosity sensor) placed on wrist, neck, leg pocket, waist, and ankle - 1 NeuroSky MindWave Mobile (EEG) - 2 RGB cameras	N/A
UR-Fall [89]	5	60	40	7	Very limited	- 2 Kinect cameras (RGB/depth) - IMU/PS Move Sensor (accelerometric)	N/A
FDDBm [84]	5	22,636*		5	None	1 Kinect (RGB/depth)	Images only portray different poses, and no fall sequence is included
EDF [90] OCCU [90]	5	40 30	30 80	1	Very limited	2 Kinects (depth)	EDF & OCCU are usually paired as one dataset, OCCU contains only obscured falls and ADLs

Dataset	Number of Actors	Falls	ADLs	Backgrounds	Fall diversity	Sensors	Additional information
HQFSD [87]	10	275	85	1 (2 lightings)	Diverse	5 RGB cameras	Actors imitate real-life falls and ADLs from nursing home footage, using canes, crutches, and walkers, occasionally
FDDBg [88]	9	146	79	5	Diverse	1 RGB camera	N/A
Multicam [85]	1	176	192	1	Diverse	8 RGB cameras	N/A
eHomeSeniors [86]	6	448	0	1	Diverse	- 1 Melexis thermal sensor - 4 Omron thermal sensors	N/A
TST [91]	11	132	132	1	Very limited	- 1 Kinect (depth) - 1 IMU	N/A
FOE.MMU [83]	1	21	30	1	Limited	1 RGB camera	N/A

Figure 2.10: Frame from FDDBg [88].

HQFSD and eHomeSeniors even go the extra mile in terms of authenticity as the former is made up of accurate simulations of real-life events captured from surveillance cameras in an elderly home and the latter includes falls performed by actors directed by a physiotherapist.

Thankfully, all but Multicam and FOE.MMU use more than one actor, which goes a long way to ensure an adaptable classification model. HQFSD used 10 actors and UP-Fall used 17. As for background variety, which is particularly important, UR-Fall and FDDBg in particular feature videos recorded in several different rooms and lightings.

2.2.3 Visual/Deep Learning-based Fall Detection Systems

This section is dedicated to listing existing FDSs that make use of ML techniques, more specifically DL or other computer vision methods, which are more closely related to the system developed for this thesis. Table 2.6 lists fourteen systems which, in one way or another, provided insight and reference for this project. The datasets used, FE/characterization methods, classification models, type of cameras and test results are presented for each. All of them are either visual systems or perform fall detection using one or more DL techniques. They are sorted in ascending order by the average of their stated performance metrics, which are almost unilaterally high - Khraief et al. [92] manage to achieve an accuracy of 99.72% as their system sits on the bottom of the table, accordingly, and most systems place in the 90-100% bracket in at least one metric. That deep-learning-based visual systems are the standard for accuracy seems to be evidenced by these results.

RGB-based systems are featured most prominently in this list, and all suffer from the issue of intrusiveness. Depth and thermal cameras present a logical alternative, as these types of imaging are less revealing but still detailed enough to be of use. Rafferty et al. [93] took three approaches to detecting falls through thermal images: via logical processes, which first detect and isolate blobs of thermal signatures that most likely represent a human and then compare 3 sequential scenes in search of rapid blob expansions which, if its size increases past a certain threshold, indicate the occurrence of a fall; via scene analysis, in which a three layer CNN is trained to classify instances of single images on fall condition; via composite scene analysis, in which a four layer CNN is trained to classify sequences of three images, much like the first approach. The scene analysis approach provided the best results with a detection rate of 80%, which pales in comparison to other systems, but might be excused

by the lack of an adequate amount of train data. Composite scene analysis fared much worse, with a detection rate of just 35%, which is justified by the authors, again, as the result of having a limited dataset. The use of extremely simple three- and four-layered CNNs is possibly also due to the small dataset size and in order to avoid overfitting, but perhaps increasing the number of layers might have provided better results, especially for the composite scene analysis approach, which indicates an obvious occurrence of underfitting.

These relatively poor results place this system at the top of the table but serve as an example of how thermal cameras have good potential, since they capture images that are detailed enough to have computer vision methods applied on them without threatening the individual's privacy.

The use of infrared sensors makes a system environmental and not conducive to computer vision techniques, unlike thermal imaging. As mentioned earlier, environmental infrared sensors typically use active infrared imaging by illuminating an area with a short wavelength light. Thermal, on the other hand, operates in mid-to-long wavelengths and only captures heat energy emanating from objects, and is thus called passive. This means that thermal cameras do not record reflected lights and are unaffected by smoke, haze, dust, and other sources of illumination. Infrared imaging can be detailed enough to match thermal cameras but only from high-end, industrial sensors, which are not usually present in these systems. The work of Fan et al. [67] appears in this section because, despite using one low-resolution 8x8 IR sensor, they experimented with and tested several DL neural networks, namely LSTMs and GRU networks with and without attention mechanisms, and compared them to a conventional MLP and a pre-built k-NN model. The authors present two sets of test results, one where the test data included only falls parallel to the sensor and the other where all falls were perpendicular to sensor – as is to be expected, results vary a substantial amount between the "easier" set and the "harder" set, respectively. In Table 6, the displayed precision and sensitivity values had to be calculated by taking the average from these two evaluations to get a sense of the system's performance overall . The conclusions drawn from this were that both LSTM and GRU models are roughly on par with each other, and both outperform the MLP, introducing attention mechanisms does not necessarily improve their detection rates and all models fared better in all aspects compared to the k-NN model.

Equally non-intrusive are depth cameras. Rahnemoonfar and Alkittaw [94] used these to significant effect in training a 3D CNN with the SDUFall dataset [95], which contains 240 videos of falls and ADLs recorded with a Microsoft Kinect sensor. Two models were created, a binary "fall or not" model and a 6-class model. Unsurprisingly, the binary model achieved a better result of 97.58% accuracy, but, more importantly and more relevant to this thesis, both accept as input a sequence of ninety-nine 160x120 frames. It would have been interesting to see how fast and how many computational resources it takes to classify one whole sequence, unfortunately the authors do not provide those numbers.

However, depth imaging is more commonly used in conjunction with RGB cameras. This combination is possible and practical as the Kinect sensor, which is often used, provides both types of imaging, being very cheap and accessible, even if its production was stopped by Microsoft in favor of the Azure Kinect DK [96]. In theory, using multimodal data has its advantages and could lead to an increase in classification quality. In this case both types of images complement each other: human silhouettes are less affected by light and visual noise in depth than in RGB images, but depth cameras have lesser range than RGB ones.

Table 2.6: Similar works (cont. next 2 pages)

System	Datasets	FE/Char.	Class. model	Sensors	Results
Rafferty et al. [93]	Private	CNN	CNN	Thermal	Acc.: - Images: 80% - Sequences: 35%
Adhikari et al. [84]	FDDBm	CNN	CNN	RGB/depth	Acc.: 74%
Fan et al. [97]	Multicam, FDDBg, HQFSD, private	CNN	CNN	RGB	Sens.: - FDDBg: 98.4% - Multicam: 97.1% - HQFSD: 74.2% - Priv.: 63.7%
Fan et al. [67]	Private	Basic filtering methods	MLP, LSTM(-ATT), GRU(-ATT)	Infrared	MLP - Prec.: 82%; Sens.: 86%; F1: 84% LSTM - Prec.: 92%; Sens.: 91%; F1: 91% LSTM-ATT - Prec.: 83%; Sens.: 97%; F1: 89% GRU - Prec.: 88%; Sens.: 86%; F1: 87% GRU-ATT - Prec.: 86%; Sens.: 91%; F1: 88%
Chen et al. [98]	UR-Fall	R-CNN, Open Pose	CNN	RGB	Acc.: 87.7% Prec.: 94.1%

System	Datasets	FE/Char.	Class. model	Sensors	Results
Wang et al. [99]	Multicam, FOE.MMU	Histograms of Oriented Gradients, Local Binary Pattern and feature maps, CNN	SVM	RGB	Sens.: 93.7% Spec.: 92.0%
Ma et al. [100]	Private	3D CNN	3D CNN, Autoencoder	RGB	Sens.: 93.3% Spec.: 92.8%
Espinosa et al. [101]	UP-Fall	Optical flow	CNN	RGB	Acc.: 95.64% Sens.: 97.65% Prec.: 96.91%
De Miguel et al. [72]	Private	Kalman filtration, background removal	k-NN	RGB	Acc.: 96.9% Sens.: 96.0% Prec.: 97.6%
Hasan et al. [102]	UR-Fall, FDDBm, Multicam	Open Pose, LSTM	LSTM	RGB	UR-Fall - Sens.: 99%; Spec.: 96% FDDBm - Sens.: 99%; Spec.: 97% Multicam - Sens.: 98%; Spec.: 96%
Rahnemoonfar and Alkittaw [94]	SDUFall [95]	3D CNN	3D CNN	Depth	Binary - Acc.: 97.58% Multiclass - Acc.: 93.20%

System	Datasets	FE/Char.	Class. model	Sensors	Results
Zou et al. [103]	FDDBg, private	ANN	ANN	RGB	Sens.: 100% Spec.: 97.04% Acc.: 97.23%
Lu et al. [104]	Sports-1 M [105] (FE), UCF-101 (FE), Multicam, FDDBg, UR-Fall	3D CNN	LSTM	RGB	Acc.: 99%
Khraief et al. [92]	UCF-101 (FE), Multicam, UR-Fall, FDDBg	Optical flow, pose recognition	CNN	RGB/depth	Acc.: 99.72%

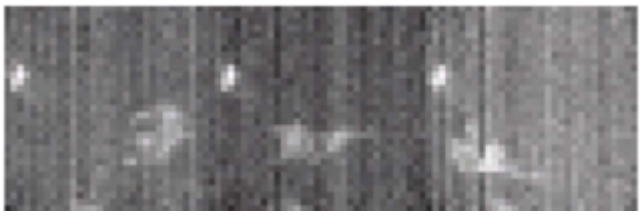

Figure 2.11: Thermal sequence which depicts a human first standing, in the process of falling and, finally, lying on the floor [93].

As part of their project, Adhikari et al. [84] created the FDDBm and their system is built around it, consisting of a CNN trained specifically to do pose recognition with just RGB, just depth, and RGB/depth images with and without background subtraction. The model performed best with RGB/depth imaging with background subtraction and scored 74% accuracy on the test set. The authors admit that pose recognition is just part of their concept of a complete, robust FDS: the next step would be developing a way to analyze sequences of poses, their transition speed and end states, which would be the determining factor in detecting falls. A couple of years later, in 2019, Adhikari would dedicate his doctorate thesis [106] to continuing his previous work, abandoning, however, the idea of pose estimation in favor of detecting groupings of joints belonging to stable regions of the human body and feeding them to an LSTM to analyze their movements and detect falls. This new system was able to achieve 88.33% accuracy.

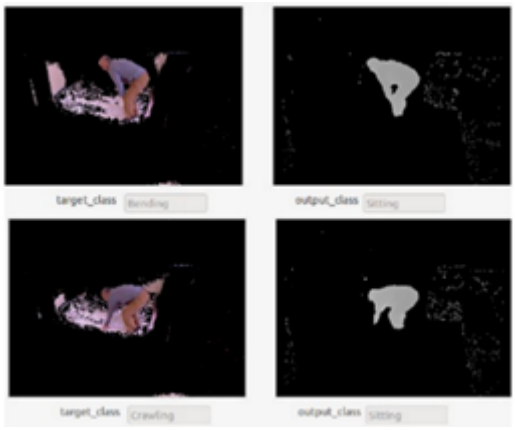

Figure 2.12: Fall detection using RGB/depth imaging, Adhikari et al. [84].

Khraief et al. [92] combine the results of shape, RGB, depth, and motion analysis with a four-stream CNN: the first stream, much like Adhikari et al., performs pose recognition on a sequence of frames and models pose deformations; the second stream extracts visual information from RGB and depth images and takes advantage of their complementarity; the third stream focuses on optical flow and occurrences of sudden movements; the fourth stream complements the third by measuring motion velocity and direction, based also on optical flow. This method achieved excellent, almost perfect results, which the authors expect to be improved further by adding an additional stream analysis of wearable data.

The remaining systems are all based on simple RGB imaging and do not differ significantly in their approaches. De Miguel et al. [72] offer a relatively robust system without relying on DL techniques, instead choosing simple FE methods and a k-NN for classification, surpassing some more complex systems in performance and in being by far one of the cheapest to build. Wang et al. [99] use a CNN on top of non-DL techniques for FE and an SVM for

classification. The rest all use some sort of neural network for classification at least. The model created by Lu et al. [104] is of special importance as it consists of a 3D CNN to perform FE and an LSTM to classify sequences of frames. Its structure closely resembles this project's model, and the results presented by its authors definitely set a standard to look up to.

Yet, despite the values indicating high performance, one must take these results given by researchers with a grain of salt. Especially in the context of fall detection, traditional dataset splicing (holdout and similar) by itself cannot recreate the necessary conditions to adequately test and portray the model's actual performance. This is mostly due to how fall detection datasets are constructed - data variety is so low that using "unseen" test data from the same dataset also used for training can produce misleading results. In other words, the model's capacity to adapt to new situations is rarely adequately tested. One way to abate, not solve, this problem is to use an entire separate dataset just for validation and testing (for example, using FDDBg for training, UR-Fall for validation, and HQFSD for testing). This method is seldom used and is found even less commonly in the lower rows of Table NEED REF.

Adding to this, earnest comparisons between systems performance-wise cannot be made in good faith unless the two systems use the same instance or collection of datasets for testing. Contrasting general action recognition and image recognition, no standard dataset for fall detection exists, like UCF101 or Kinetics. Researchers train and test their models on whichever datasets they deem more appropriate, usually resulting in varying levels of accuracy when introducing data from other datasets.

2.3 Privacy and data management guidelines in visual Fall Detection Systems

As visual FDSs rely on constant and systematic video surveillance of a subject and possibly public or third-party spaces, the development and use of these systems must be scrutinized and made to comply to regional or national legal frameworks regarding privacy laws, since there are no international level set of regulations or guidelines that restrict general behavior towards the handling of private data. In the European Union, the regulations stated in the General Data Protection Regulation (GDPR) [107] dictate how sensitive data should be collected, handled, and stored. The 3/2019 European Data Protection Board (EDPB) guidelines on processing of personal data through video devices [108] provides substantial insight into how to process and manipulate personal data in compliance with existing regulations. According to Article 5 of the GDPR, visual data collected by an FDS, which must be considered intensely intimate personal data, can:

- only be processed for the sole purpose of detecting human falls (Article 5 (1) (b))
- only contain information relevant for the detection of falls (Article 5 (1) (c), 'data minimization')
- stored for long periods of time only if for scientific/statistical purposes, perhaps with the end goal of improving the classification algorithm (Article 5 (1) (e))
- be processed only in a manner that ensures its security and protection from unlawful and unauthorized access (Article 5 (1) (f))

Furthermore, on a more ethical note, data processing is lawful only with the full consent of each data subject, who must be fully aware of its purpose (Article 6 (1) (a)). The

controller (in this case, the FDS's developer/distributor or an entity responsible for elderly care) must be able to demonstrate that consent was, in fact, given by the data subject, who has also been made aware of their right to, at any time and for any reason, withdraw their consent (Article 7 (1), (3)). In the case that the data subject is physically or legally incapable of providing consent and the controller is able to prove this, as well as justify that this processing is in accordance with the subject's vital interests, Article 9 (2) (c) could theoretically be invoked [108, p. 17] (69).

FDSs should either fall in the category of black-box or real-time monitoring systems. The difference between the two being that, in the case of the former, data is placed in storage temporarily before deletion and can only be accessed in the event of an incident, while in the latter no data is stored at all. Black-box solutions have the advantage of preserving some data as evidence which, in the context of fall detection, could help researchers better understand the algorithm's performance and possibly improve it. Real-time monitoring solutions, conversely, cannot recover data post analysis, but also do not run the risk of unauthorized access to stored data. In both types of systems, it might still be possible for a third party to illegally access the video feed. It is up to the system's developers to choose what design fits best to a given situation [108, p. 11] (29).

Regarding data minimization, several FE methods already involve eliminating extraneous information from frames, such as background removal and optical flow. It could be argued that the use of these techniques does ensure a less intrusive experience, but being operations that occur during or after processing, the unaltered captured footage is still vulnerable. Ma et al.'s [100] system goes one step further and automatically obscures any of the subject's identifiable facial traits at the video capturing stage. This method involves finding facial features in thermal imaging first and generating a mask that is then applied to a spatial light modulator, preventing light originating from the subject's face from reaching the camera's RGB sensor. As result, the captured images themselves feature black shapes that hide the subject's face, whose identity is concealed throughout the fall detection process.

Figure 2.13: Different mask shapes in Ma et al. [100].

Although RGB-based FDSs' main issue is their inherent breach of privacy, almost no project article mentioned so far includes an in-depth description of procedures that prevent unauthorized access to data (either stored or captured). This is one aspect that researchers should pay special attention to, particularly if the FDS in question is designed for actual real-life use in mind.

Chapter 3

Method and Implementation

This chapter provides a detailed account of the implementation process for an automated fall detection system. It begins by announcing the method devised for this project. Following this is a section elucidating the tools and technologies employed in this project, and sections that address the subject of datasets, including the inherent limitations of fall detection datasets, the compilation of multiple exisitng datasets, the creation of a new dataset, and the techniques employed for data preprocessing and augmentation. A subsequent section focuses on the software programs employed to extract segments from video recordings and modify video characteristics. Furthermore, the chapter explores each model employed for FE and classification, along with their respective functionalities. Finally, a brief explanation is provided regarding the integration between the FE models and the classification model.

3.1 Method

This thesis proposes the implementation of DL algorithms and techniques, specifically computer vision, to create a model capable of distinguishing falls from ADLs, the latter term emcompassing all human actions and interactions that do not consist of falls. It should make this distinction based on real-time data streams originating from standard RGB cameras and should strive to be accurate, avoiding false predictions.

A careful exploration of the state of the art, including the of the state of both the fields of camera-driven fall detection and general action recognition, was made with the objective of encountering an optimal approach to this problem. This analysis resulted in the theory that the implementation of the combination of two models - one responsible for FE, another for classification - would be a promising subject, worthy of research. It was theorized that the I3D, which recently has outperformed other CNNs on action recognition tasks in accuracy and efficiency, would be the best candidate for an accurate, yet efficient FE model, and that its use in conjunction with an RNN would provide the best accuracy results out of all other techniques seen in other FDSs. This hybrid model must be built, trained and have its performance compared to other FDSs.

Since DL algorithms need to be trained and tested with data and obtaining said data from actual recorded falls of seniors is both challenging and unethical, said data must be created from simulations of falls. These simulations must be numerous and of good variety, representing a good range of possible fall and ADL scenarios, in order to improve the model's performance and make it adequately robust for actual real-life use. A custom private dataset must then be created, in tandem with existing public datasets, to provide even more data diversity. This, of course, requires the procurement of cameras to record falls and ADLs

with, in addition to the development of programs and algorithms to access the cameras' feed, as well as to store and organize the newly captured videos.

Following its acquisition, the data must be preprocessed, as to, for one, be compatible with the expected DL algorithm's input. This includes truncating each video to a set duration, discarding any unnecessary information. Several data augmentation techniques were also used to also further improve the data's quality and variety, thereby increasing the model's performance. This will be accomplished with the creation of other programs specifically designed to slightly alter some of the videos' characteristics such as contrast and orientation.

Once all data is collected and processed, the I3D/RNN hybrid model must be tested rigorously to prove its usefulness for real-life use. This involves putting it up against other models features in other FDSs: a 2D CNN and a more traditional 3D CNN, two very common models in this field, were chosen to be compared to the I3D on their FE capabilities; two RNNs, an LSTM and a GRU, along with a 2D CNN, will all be tested in classifying the output of the FE models. Each experimented model being subject to various modifications and tuning processes to increase their performance as much as possible.

The flux diagram shown in figure 3.1 presents the functioning of a hypothetical FDS which would be capable of receiving the video stream originating from a camera, extracting video segments from it, and send those to the model that has been incorporated into the system.

3.2 Tools and Technologies

In order to develop and implement an FDS based on RGB imaging, several tools and technologies were used. Concerning physical instruments, one Xiaomi home security camera was acquired and modified to both record a dataset and for testing purposes, the author's own PC was used to access the camera feed and to store footage, and DevScope's AI R&D lab, which enables access to several computer units equipped with CUDA-compatible[1] GPUs, was used to build, train, and test DL models. Additionally, but less noteworthy, several household objects served as props for the constructed dataset, as were some rooms of the author's house also used as background.

Regarding virtual technologies and programs:

- The programming language Python features heavily in the programs used to access the video feed, store, and edit footage as well as in code related to the model's construction and testing.
- The OpenCV[2] library was used to record, store, and edit videos and frame manipulation.
- The camera was modified to create a RTSP (Real Time Streaming Protocol) server at startup capable of streaming the camera's feed. Scripts and other files were created by GitHub user Filipowicz251 and five other contributors[3] and altered as seen fit.
- The code related to building and testing the models was contained in Jupyter Notebooks.

[1] `https://developer.nvidia.com/cuda-toolkit`
[2] `https://opencv.org/`
[3] `https://github.com/Filipowicz251/mijia-1080P-hacks`

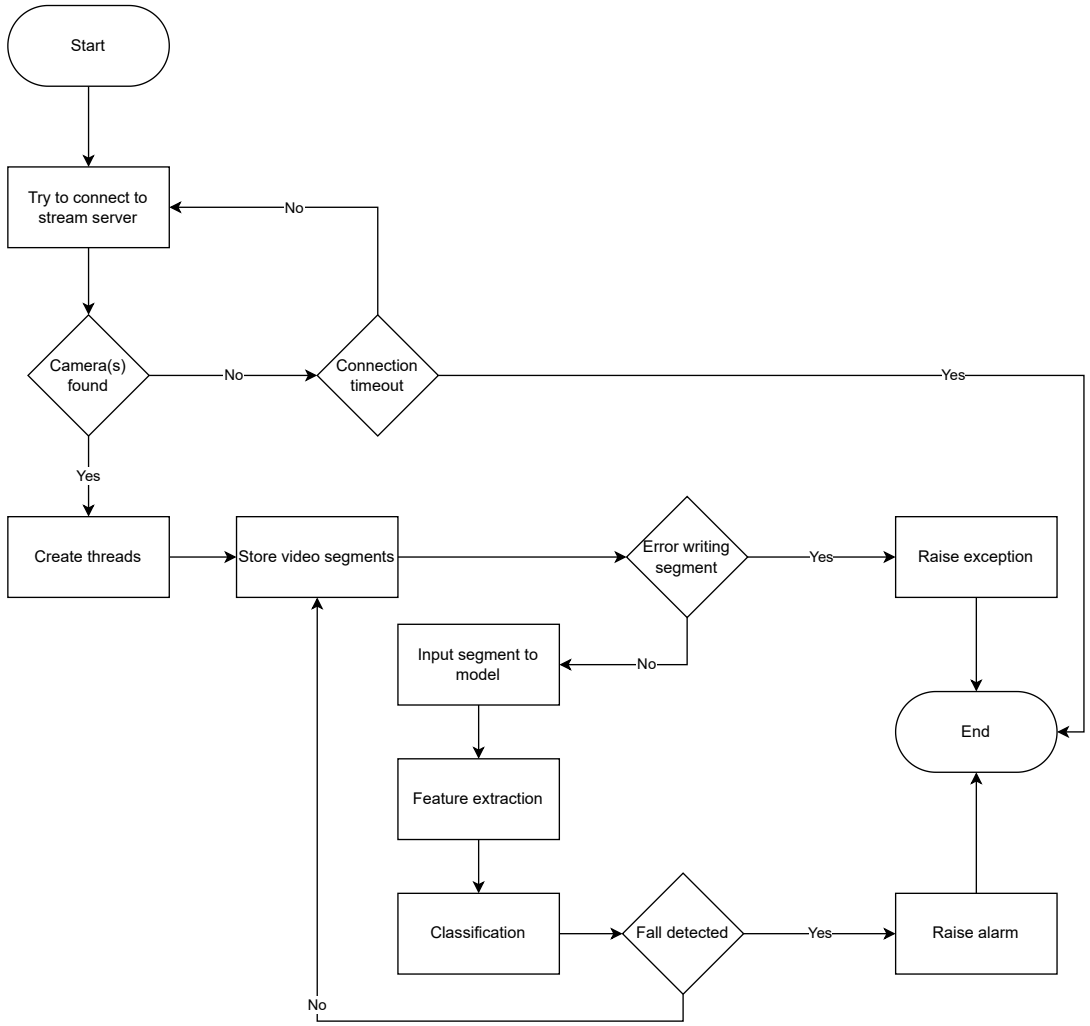

Figure 3.1: Proposed system flow.

- PyTorch[4], Tensorflow[5], GluonCV[6], their libraries and extensions, and frameworks built on top of them, including Keras[7], were used to develop both models.
- The models ran on CUDA-enabled GPU acceleration, greatly shortening training and classification times.

3.3 Datasets

The development of a DL model requires the assembly of a dataset to train and test its validity for real-world use. As discussed in section 2.2.2, one of the biggest hurdles in constructing fall detection systems via visual sensors is obtaining enough data and ensuring its quality. On one hand, existing public datasets are small and relatively homogeneous, while, on the other, creating one must be done by methodically staging and simulating falls, which implies employing actors and using several backgrounds and room layouts. Building

[4]PyTorch is "an open-source machine learning framework...": `https://pytorch.org/`
[5]Tensorflow is "an end-to-end machine learning platform": `https://www.tensorflow.org/`
[6]"GluonCV: a Deep Learning Toolkit for Computer Vision": `https://cv.gluon.ai/contents.html`
[7]Keras is a deep learning framework built on top of Tensorflow: `https://keras.io/`

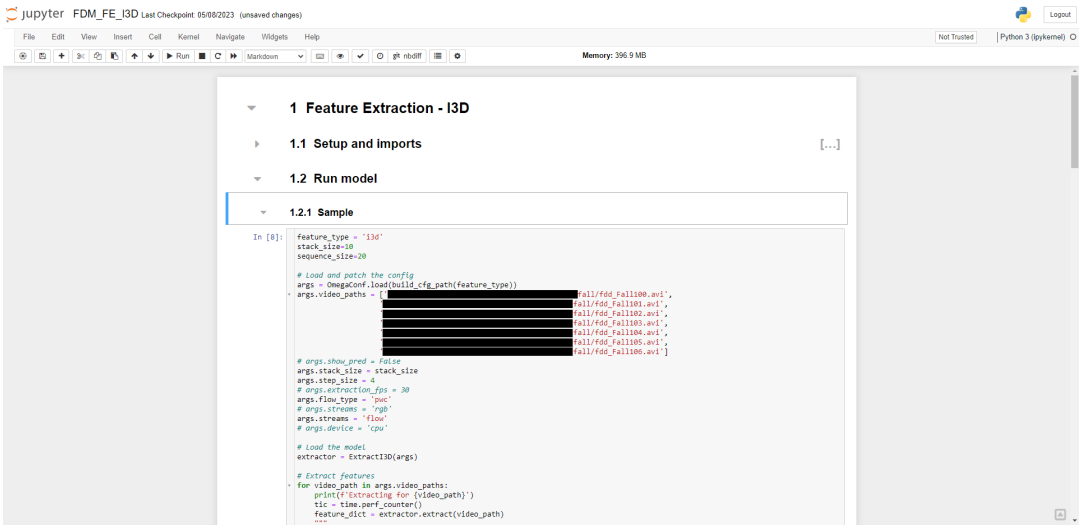

Figure 3.2: A Jupyter Notebook. This instance allows for feature extraction using the I3D model

a dataset that is as big and as varied as the popular datasets for general action recognition would necessitate such a level of preparedness and organization that it would require its own expensive and time-consuming project. Yet, as it stands, a fall detection model trained on one dataset would invariably lack generalization capabilities – one dataset cannot hope to capture even a fraction of all possible variations in fall scenarios as to train a model effectively.

One thing that can alleviate both issues of dataset size and variety is simply combining several existing datasets into one. This has been carried out by several projects as pointed out in section 2.2.3 and the same has been done here. But even this method might not be foolproof: combining UP-Fall and HQFSD would provide more than 800 falls, not counting ADLs, but only 2 different backgrounds. There is no way to ensure that an NN trained and tested on these two datasets would not learn background patterns which are useless or even detrimental to detecting falls in other instances, but which are over-represented due to how repetitively they appear in data. A few datasets show actors falling exclusively on mattresses or carpets, for instance, which might give the model "the wrong idea".

Another point also briefly discussed in section 2.2.3 is a problem that arises from this lack of variety. In using a homogeneous dataset, one cannot guarantee the validity of the test results, as the data present in the test split, supposedly completely unknown to the model, is too similar to the train data to reflect the model's generalization performance - the model is possibly already "familiar" with the test data, in other words. Even when combining several datasets, in the case of fall detection this might be insufficient. One way to remedy this would be to use one or more datasets in the train split and one or more whole separate datasets for the validation and test split each. For instance, using FDDBg and HQFSD for training, Multicam for validation and UR-Fall for testing. This cannot be done haphazardly, however, as reserving an entire dataset for testing deprives the model of much needed data and variety in training; while trying to avoid overfitting, one might be enabling underfitting instead.

For this project, it was decided to combine several datasets in order to maximize variety, number, and quality of data. On top of this, it was deemed necessary to record an original

dataset to complement the compilation of existing datasets and to provide a more realistic training process. In essence, two separate datasets were constructed - the first, a compilation of publicly available datasets, and the second, the original, could supplement the training set and "free" two smaller datasets for validation and testing. The question of which datasets might be excluded from the training set is tackled in the following chapters.

3.3.1 Compilation of existing datasets

Videos from five datasets, UR-Fall [89], UP-Fall [82], HQFSD [87], Multicam [85], and FDDBg [88], were selected, half being videos depicting a fall and the other half including some kind of ADL. Videos depicting what are considered to be "critical" ADLs were prioritized; these are ADLs which might most visually resemble the movements of a fall, such as sitting, bending down, kneeling, and lying.

The selection and arrangement of datasets for compiling was guided by three decision factors: data structure, quality, and balance. Firstly, the model exclusively accommodates video segments, rendering the inclusion of FDDBm unfeasible due to its data consisting of disjointed frames lacking temporal continuity. Secondly, certain datasets exhibit fall scenarios that more closely resemble real-life occurrences, and these were prioritized. Lastly, the integration of a large dataset with a relatively smaller one may introduce an imbalance, potentially compromising the model's learning process. This concern was particularly relevant in the case of UP-Fall, where its full inclusion would create said imbalance. The exact distribution of each dataset (after preprocessing) is as follows:

- UR-Fall: 120 segments
- HQFSD: 504 segments
- FDDBg: 302 segments
- Multicam: 352 segments
- UR-Fall: 600 segments

Some fall videos from HQFSD and FDDBg were excluded, since these depicted falls that were too obscured, not acceptably apparent, or even entirely absent: for instance, as every fall in HQFSD was recorded with 5 cameras simultaneously, due to the actor's positioning some falls occurred completely out of some of the cameras' field of vision. A little less than half of UP-Fall was left out, due to the conclusion that these videos did not contain vital, essential data and that their inclusion would create too much imbalance. A considerable amount of ADL segments was extracted from the same ADL video and some even from fall videos in order to increase the dataset's size and to even out fall/ADL distribution, in a process that is discussed more thoroughly in section 3.3.3. In total, this compilation contains 1,878 videos.

3.3.2 Construction of the original dataset

A Xiaomi MI Home Security Camera Basic 1080p [109][8] was procured and configured to create a streaming server at startup through RTSP. This enables any computer within the camera's local network to access its feed without having to access it through Xiaomi

[8] This specific camera is not representative of the type of camera that should be used in an actual FDS, or in anything but a prototype.

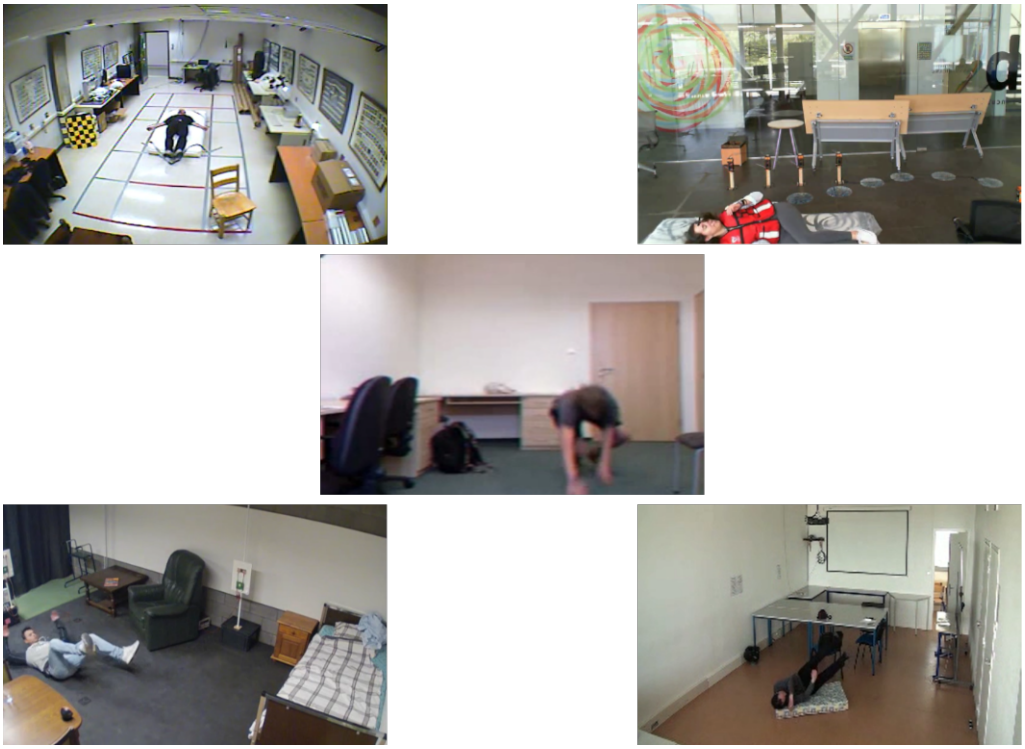

Figure 3.3: Frames from all compiled datasets. Clockwise, starting from top left, ending at middle: Multicam, UP-Fall, FDDBg, HQFSD, UR-Fall.

proprietary and closed software, making video capturing and editing possible. It was with this camera that all videos used to construct the new, original dataset were recorded.

Besides the camera, the materials used are as follows: the only actor was this thesis' author, the six backgrounds were two bedrooms, bathroom, the dining room, the living room, and foyer of the author's residence. Crutches and several household items (clothing, blankets, books) were also featured in some scenarios. Using the table in section 2.2.2 as reference, this dataset would be described in the following manner:

- Number of
  - Subjects: 1
  - Falls: 514
  - ADLs: 514
  - Backgrounds: 6 (at least 2 lightings and angles each)
- Fall diversity: Diverse
- Sensors: 1 RGB camera

In terms of size, this dataset was held back by not having more than one camera. If one more camera was available to capture an additional angle of each scenario it would be larger than any other public dataset. Comparatively, however, this one is surpassed by no other dataset in number of distinct fall and ADL scenarios, as every video contains a unique sequence of movements. HQFSD, for instance, contains five videos for each of the mere 55 fall scenarios and 17 ADL scenarios.

Figure 3.4: Frame from original dataset.

The dataset's size was limited by the absence of multiple cameras. Incorporating an additional camera to capture an extra angle for each scenario would make this dataset surpass the size of any existing other. Nevertheless, when considering the number of distinct fall and ADL scenarios, or videos in the dataset that showcase a distinct sequence of movements, this dataset remains unparalleled. The presence of a singular subject throughout all the videos hurts data diversity. In order to address this limitation, various attires were worn during the filming process over multiple days, thus introducing a modest yet noteworthy level of visual diversity. Recording videos on different days and at different hours of the day also resulted in distinct levels of natural and artificial lighting and varying levels of illumination intensity, from fully illuminated to completely dark. Furthermore, the camera's night vision capability was employed, resulting in some videos being black-and-white.

Special care was taken to walk and move while simulating physical impairment in order to act out more realistic scenarios. Additionally, interactions with objects were included, such as throwing, shaking, dropping, picking, manipulating, and otherwise interacting with books, clothes, blankets, doors, drawers, furniture, crutches, and even a dog. These features are seldom seen in other datasets. Some scenarios occurred in the scenic background and a few obscured falls were incorporated, such that either only the subject's body hitting the ground can be seen or only the movements before hitting the floor are visually present. These scenarios were expected to be very difficult for any model to accurately classify, especially considering they rarely appear in the other datasets. However, it was deemed important to include these videos, even if an actual FDS would make use of more than one camera specifically to counteract the effects of blind spots and almost-out-of-frame movements, as their use in training data increases adaptability at little to no cost to accuracy.

Most ADL videos were extracted from fall scenarios through preprocessing, in order to match the number of fall videos, resulting in 1,028 videos in total. Again, this process is explained more in depth in the following section.

3.3.3 Data preprocessing and augmentation

Before introducing the collected videos to the model, all videos were subject to an editing process with the goal to 1) transform them into data resembling what is considered the expected input of the model, 2) to reduce the amount of worthless or misleading information, and 3) to further improve data variety.

Figure 3.5: Frame from original dataset using the camera's night vision function.

First, all videos were edited to only be of a set duration or shorter. This duration was decided on the basis that all video segments, including those going to eventually be fed to the model in final stage of the FDS's development, should be just long enough to contain the subject's posture before, their movements during, and their position after the fall. Simultaneously, segments should not be unnecessarily long and, by being so, decrease system performance. Three seconds, as it stands, was found to be a perfectly adequate duration, and does have the fortunate side effect of decreasing the dataset's storage size. Quite a few HQFSD 1920x1080 videos were several minutes long, reaching over 20 minutes length in some cases. Excluding many UR-Fall videos that were shorter than three seconds, and which were left unedited, all other videos had to be shortened. With falls and "critical" ADLs, videos were cut to only contain the most important motions, at slightly different starting points.

As stated previously, several videos, despite being labeled as falls by their respective dataset's original creators, contained severely or entirely obscured falls. If included in the train set, these videos would negatively impact the model's learning process. For this reason, if no useful data could be otherwise extracted, the videos were simply excluded from the "fall" label. Videos that had no human presence were labelled as "ADL".

Lastly, in another effort to increase data diversity, several image augmentation methods specific to each video were employed that alter the video's hue, contrast, and color warmth values, as well as occasionally mirroring and tilting every frame belonging to a single video segment. This step was crucial in increasing the dataset's variety.

After preprocessing, the two datasets add up to 2,906 video segments, 4.14 GB in storage space, 1 hour, 12 minutes, 21 seconds of fall and 1 hour, 12 minutes, 23 seconds of ADL footage, totaling 2 hours, 24 minutes, 44 seconds of 30 FPS footage, consisting of 260,520 individual frames.

3.4 Video capturing and editing programs

Several Python programs were written in the course of this project: one was used to access the Xiaomi's camera feed, record, and store footage to build the original dataset; another was used to apply data augmentation techniques to the videos in both datasets; and a third was developed to capture the camera's feed, extract three second segments, pass these on

to the fall detection model and emit an alarm, if necessary. The rest were written to aid in preprocessing or aid in dataset construction and compilation.

Both programs responsible for accessing the video stream do so through OpenCV's *VideoCapture* function, which takes the camera's RTSP server IP as parameter. As soon as a connection is established the capture stream becomes active, the frame size is determined, and OpenCV's *VideoWriter* is used to save the video in memory. *VideoWriter* is run before the feed is terminated by manual input, in the case of the first program, or called repeatedly in three second intervals by two parallel threads, one of which is 1.5s ahead of the other. This way, video segments overlap each other, minimizing the chance that a fall occurs between segments, too late in one segment and too early in the following segment for it to be detected. This "sliding window" approach was tested and found to be sufficiently fast that the model is able to classify segments slightly after the following segment is stored, ensuring a timely, as close as real-time that is possible, alarm activation.

The second program starts by separating each video by frames. Following that, it performs image augmentation techniques on all video frames, including randomly changing hue, contrast, and brightness values in addition to random image mirroring and perspective shifts. These operations are performed on frame batches, which ensures that all the frame's frames are altered the exact same way. Tensorflow's *Image* library was used to accomplish this.

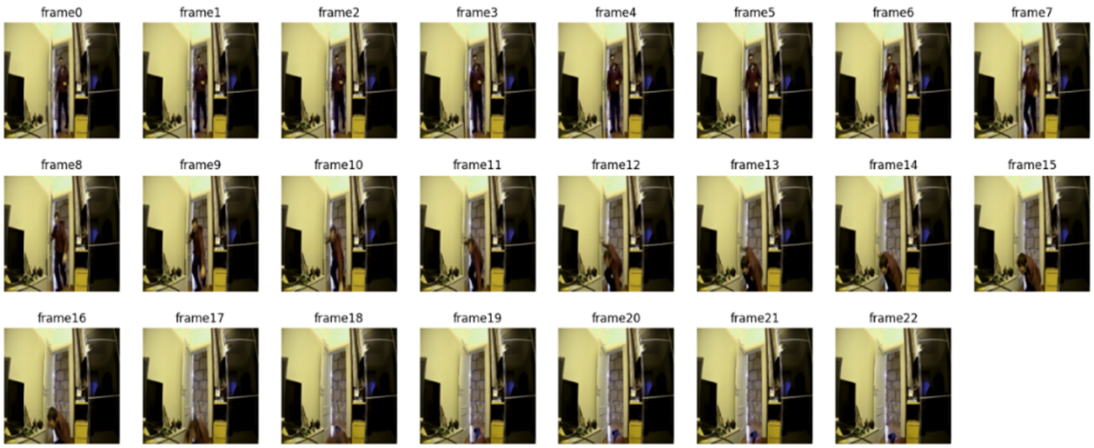

Figure 3.6: Frames of video segment that underwent image augmentation.

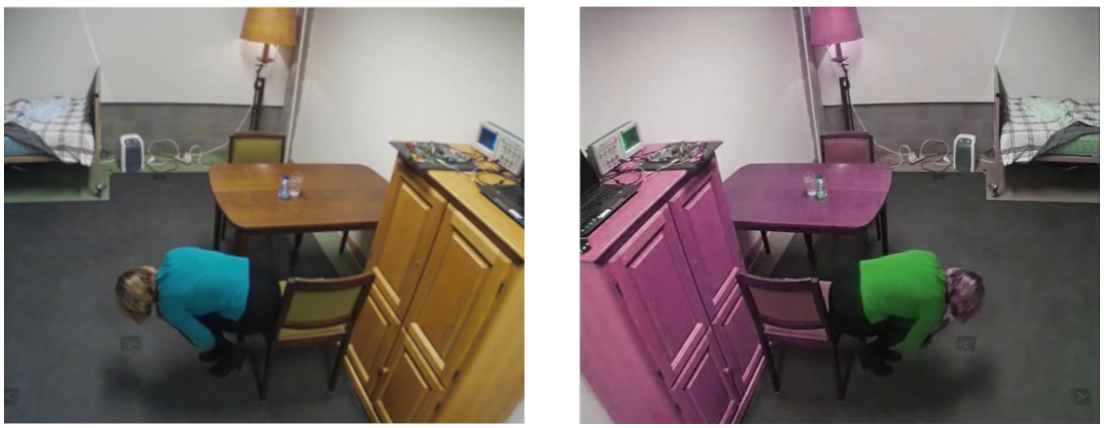

Figure 3.7: Original frame and augmented frame side-by-side.

The rest of the programs are very minor, serving as a way to automatically alter a large number of videos' frame rate, length, and filenames – essentially performing tasks that would be too time-consuming if done manually.

3.5 Model structures

In this section, insight is given on how the fall detecting model is structured, what compelled the choice of these types of DL models, and how they function. The first subsection is dedicated to the I3D CNN, the two other alternative models, and their role as feature extractors. The second lists the several classification models tested, namely GRU, LSTM, CNNs, and how they function in order to classify the extracted features. The last subsection presents a small summary explaining how these two parts interact and operate as a whole.

3.5.1 Feature extraction models

Videos are simply sequences of frames, so in order to classify human motions contained in video, a computer program could, at some point, perform computations and analysis on the image level and reach a conclusion drawn from them regarding the whole sequence. Initially, the ML algorithm in charge of detecting falls (or other human acts) on video has available to it, in essence, a sequence of matrices containing the color values of each of the frame's pixels. Unchanged, data formatted this way could be processed and combed for signs indicating specific movements. However, FE helps single out and only consider patterns that, in theory, are most important for the task at hand, ensuring cleaner and simpler data. If the purpose is to detect human motion, it would be helpful for the algorithm to only focus on human figures, for example.

CNNs are especially suited for FE due to their ability to find shapes and patterns in images regardless of their position. Adequately trained, these models can consistently and accurately detect a person's outline and pose. But when using a CNN to process sequences of frames for specific movements, on the other hand, there's a risk that it will not be quite as precise because the CNN is not able to learn time-specific information. The model combs through frames individually and separately, ignoring any patterns that might exist "in-between" them.

To remedy this problem, adding a recurrent layer to a CNN model, such as an LSTM, would allow it to process time distributed sequences and enable spatio-temporal classifications. Another solution, three-dimensional CNNs (3D CNNs) are, unlike traditional 2D CNNs, specifically designed to perform the typical CNN operations on ordered sequences – the added dimension being temporal in nature. Carreira and Zisserman's work in "Quo Vadis, Action Recognition?" [110] demonstrates the performance difference between the two aforementioned solutions, a 3D-fused two-stream network, and their implementation of a two-stream I3D on the popular HMDB-51 [111] and UCF-101 [81] action recognition datasets. The results presented show clearly that the I3D outperforms any other type of NN in action recognition.

For this reason, coupled with the fact that using a 3D CNN for FE is rare in prevailing FDSs (and none has used the I3D), the exploration of an I3D and a RNN combination was thought of as a promising subject of investigation in the field of fall detection. Consequently, this hybrid model was implemented and subjected to rigorous testing, alongside other models employed as benchmarks, to facilitate the evaluation of their respective performances. Thus, three pre-trained FE models were tested: an I3D, a conventional 3D CNN, and a 2D CNN.

Given that the I3D model has not been subject to extensive usage and evaluation like the other two types of models, it is crucial for it to exhibit superior performance in order to establish itself as the optimal solution for visual fall detection. In comparison to the I3D model, these alternative models have been thoroughly employed and tested by numerous researchers. Hence, for the I3D model to claim its position as the most effective solution, it is imperative that it outperforms these established alternatives.

Regardless of the chosen model, each video produced a output structured as a tensor comprising a sequence of *n* elements. Each element within the tensor represented an array consisting of 1,024 values respective to a frame of the input sequence. In instances where the duration of a video was less than three seconds, resulting in a lower number of frames, the output tensor would contain a reduced number of arrays. In such cases, the last array was duplicated to ensure the tensor matched the expected length. On the contrary, in the unlikely event that the video contained extra frames, and, as such, the output sequence contained more than *n* elements, the sequence was truncated accordingly to conform to the desired length. Initially, the raw video files automatically undergo a conversion process wherein they are transformed into a series of image frames and subsequently subjected to cropping and resizing operations, and then fed into the FE model.

```
Extracting for [redacted]/fall/fdd_Fall100.avi
flow
(20, 1024)
[[0.15973701 0.01743325 0.15376072 ... 0.44875908 0.03439462 0.0137872 ]
 [0.11320019 0.00660586 0.15720423 ... 0.4299725  0.14558244 0.01019125]
 [0.09089721 0.         0.09795841 ... 0.28809968 0.         0.00811647]
 ...
 [0.07521354 0.04193142 0.16333315 ... 0.37444225 0.35496739 0.00266913]
 [0.09018589 0.01972731 0.09874976 ... 0.35556263 0.10437635 0.00278772]
 [0.0629118  0.03234569 0.12622653 ... 0.42003074 0.15564311 0.0059069 ]]
fps
()
30.0
timestamps_ms
(20,)
[ 333.33333333  466.66666667  600.          733.33333333  866.66666667
 1000.         1133.33333333 1266.66666667 1400.         1533.33333333
 1666.66666667 1800.         1933.33333333 2066.66666667 2200.
 2333.33333333 2466.66666667 2600.         2733.33333333 2866.66666667]
```

Figure 3.8: The I3D's standard output in *'flow'* mode. The input video was divided into 20 equidistant frames.

Two-stream Inflated 3D CNN (I3D)

Training 3D CNNs from scratch can be computationally expensive due to the increased number of parameters. The I3D model tackles this problem by taking pre-trained 2D CNNs on large-scale image datasets, such as ImageNet, and inflating them to 3D. It initializes the weights of the 3D CNN by copying the learned weights from the corresponding 2D CNN layers. The I3D model is an extension of the Inception architecture specifically designed for video understanding tasks, such as action recognition. Simply, by using the inflated weights, and applying 3D convolutions to capture temporal dependencies across frames and 2D convolutions to extract spatial features within each frame, the I3D model can efficiently learn spatio-temporal features from video clips. The network architecture consists of multiple convolutional layers, followed by pooling, non-linear activation functions, and fully connected layers for classification.

The I3D model has achieved impressive performance on various action recognition benchmarks, including Kinetics [29]. It demonstrates the ability to recognize and classify actions in videos by effectively modeling both spatial and temporal information. The pre-training strategy with inflated 2D CNN weights enables the model to benefit from large-scale image datasets, enhancing its ability to generalize and learn robust representations for video understanding tasks.

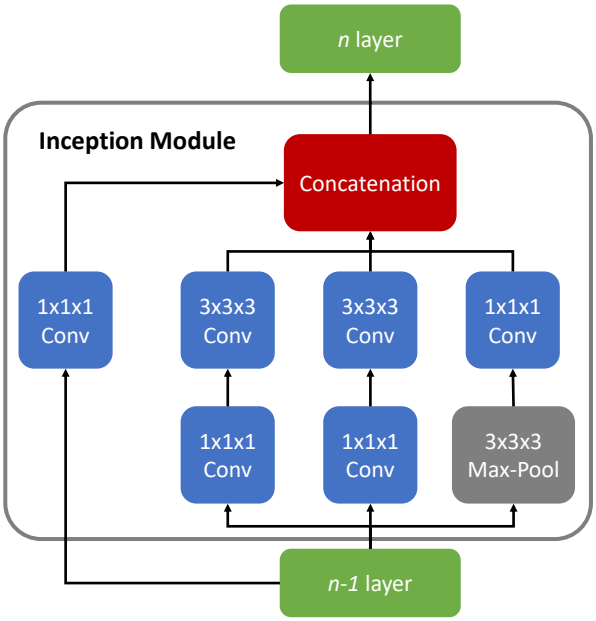

Figure 3.9: The inception module - the building block of the I3D [110].

The implementation of a pre-trained I3D model from the Kinetics-400 dataset, using PyTorch, derived from the work of Vladimir Iashin and three other contributors [112], was utilized and modified for the purposes of this project. Aiming to use it as a feature extractor, its final layers were removed. Its two streams are capable of producing both RGB features, in other words, features obtained from the RGB channels of images and videos, and optical flow features, or features extracted through analysis of the velocity and direction of objects present in the videos. The I3D can output one of these feature types at a time, but it is meant to do both simultaneously. When the single stream mode is selected, the model generates an output in the form of a tensor consisting of n arrays, with n representing the number of frames within the sequence. Conversely, if both streams are chosen, the I3D model produces two tensors for each type of feature.

Another I3D was also implemented with the intent to fine-tune it to also detect falls, as opposed to merely extracting features. Architecture-wise, the only difference to the original is the last layer, which was included, made trainable and whose output was changed to fit the binary classification problem. For the sake of convenience, the GluonCV implementation of a Kinetics-400 pre-trained I3D provided a quick and easy way of setting up, training, and testing the altered model, but regarding the model's functioning, it is exactly the same as the previously mentioned PyTorch implementation – minus the output. Since it eschews a dedicated FE model, the Fine-Tuned I3D (FT-I3D) was the only model that functions in isolation. This FT-I3D was only tested with both feature types.

3D ResNeXt-101 (3D CNN)

ResNeXt-101 is a deep learning model that belongs to the family of convolutional neural networks (CNNs). It is an extension of the ResNet (Residual Network) architecture, which has been widely used and highly successful in various computer vision tasks, such as image classification and object detection. The ResNeXt-101's original model structure is based on the ResNet (Residual Network) architecture, and introduces a novel concept called "cardinality" to enhance the representational power of the network. Cardinality refers to the number of parallel pathways, or "cardinal groups," in a network layer. In a ResNeXt-101, each layer is composed of multiple cardinal groups, where each group operates independently and learns diverse feature representations. The increased cardinality allows ResNeXt to capture a wider range of feature patterns by jointly learning from multiple pathways. This design choice improves the network's ability to model intricate details and boosts its overall performance. The "101" refers to the number of layers in the network, making this model a very deep one.

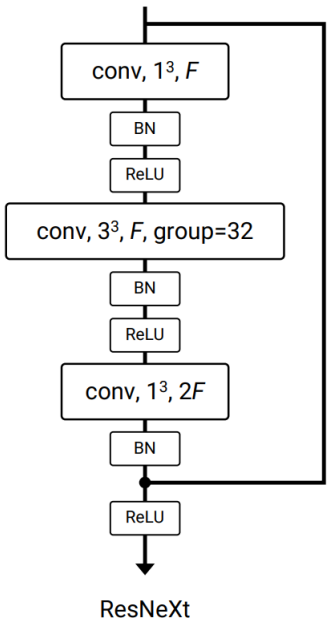

Figure 3.10: The 3D ResNeXt-101 block [113].

With the exception of modifications made to ensure package compatibility and to the input algorithm, the only alteration made to the model was the incorporation of an average pooling layer.

The 3D CNN tested was adapted from the work of Hara et al. [113]. Specifically, the model used was a ResNeXt-101, modified to incorporate 3D convolutional layers, and pre-trained on Kinetics-400, which has achieved great results on various visual recognition tasks, including image classification and object detection benchmarks like ImageNet - a capability which might be transferable to fall detection. It has been widely adopted as a powerful backbone architecture in many computer vision applications due to its impressive performance and ability to learn rich representations from complex visual data.

Its projected performance was thought to be better than the 2D CNN's, as the resulting tensor would not only contain spatial, but also temporal features, providing a more comprehensive representation of the input data. Unlike the I3D, the ResNeXt is one-stream,

indicating that the only feature type it is capable of generating is RGB type features.

EfficientNetV2 (2D CNN)

The EfficientNetV2 is a family of deep learning models designed to achieve state of the art performance in image classification tasks while maintaining computational efficiency. It is an extension and improvement over the original V1 models, which were already well known for their accuracy to model size ratio. They are built using a compound scaling method that optimizes the architecture's depth, width, and resolution simultaneously. This compound scaling allows the models to be more accurate by scaling up the dimensions while keeping the model computationally affordable. The main idea behind EfficientNetV2 is to leverage AutoML (Automated Machine Learning) techniques to search and optimize the architecture automatically. This approach helps to identify the most effective network designs and model configurations for a given task.

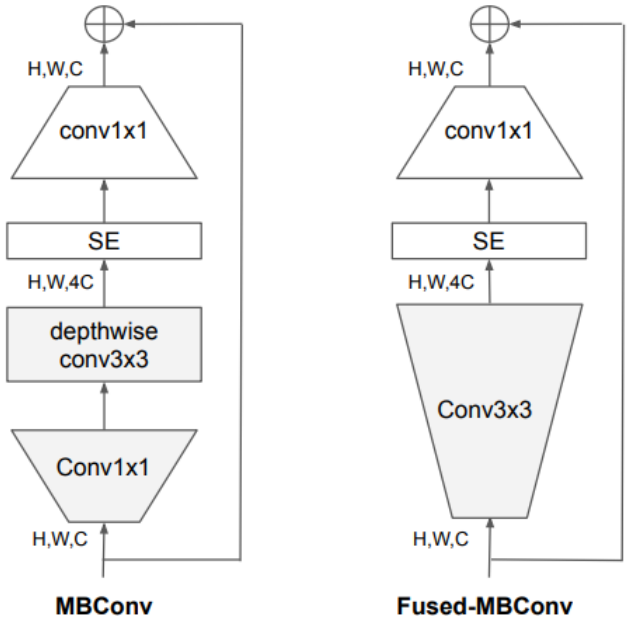

Figure 3.11: The MBConv and Fused-MBConv blocks that make up the EfficientNet [114].

There exist several EfficientNetV2 models defined by their size, and the one specifically used in this project is the "small" variant. The "medium" and "large" sizes are also available, but all models are designed to be lightweight, making them suitable for deployment on computationally weaker devices like mobile phones. Their proficiency is similar, although there is a slight increase in accuracy towards the "heavier" models when tested on ImageNet [115]. EfficientNetV2 models also incorporate various advanced techniques such as stochastic depth, drop-connect, and improved training schemes. These techniques enhance the model's generalization capabilities, reduce overfitting, and improve overall performance. As a result from all of these qualities, these models have achieved state of the art results on popular image classification benchmarks, ImageNet included, outperforming previous models. They have, as such, become widely adopted in both research and practical applications.

The 2D CNN selected for this thesis was an EfficientNetV2 [114], pre-trained on the ImageNet dataset. With an expanded and better-organized dataset, it was deemed valuable

to compare the accuracy scores not just of the other FE models, but also of a previously attempted fall detection approach, which saw the use of a MobileNet [116] 2D CNN for FE.

Despite the aforementioned accolades, it was theorized that, out of the three FE models, the 2D CNN would fare the worst, in large part due to its inability to capture temporal patterns, even when working in conjunction with an RNN. In fact, it was the only model that required being inserted into a TimeDistributed layer, as its input only allows singular images. Thanks to this workaround, it was then possible that from a sequence of frames a sequence of feature tensors of the same size could be extracted, each element corresponding to one input frame. Like the ResNeXt, this model only outputs RGB features.

3.5.2 Classification models

While 3D CNNs are well suited for image data, it has been found that they excel at extracting low-level spatio-temporal features, information between sets of adjacent frames. Meanwhile, recurrent networks such as LSTMs or GRUs are capable of modelling much higher-level, but only temporal, features, being possibly more appropriate for detecting certain actions for this reason. Wang et al. [117] demonstrate in their study that a combination of a pre-trained I3D network, trained on the Kinetics dataset, and an LSTM model achieves superior accuracy in classifying actions present in videos compared to other state of the art models. Notably, the accuracy of this combined approach surpasses even that of an isolated I3D network.

For this thesis, it was deemed that such a pairing would give the best results, specifically in terms of accuracy. However, there some doubts arose regarding the choice of DL model that should be responsible for analyzing the extracted features and performing classification. Two obvious alternatives, both examples of RNNs, were LSTM and GRU networks. Since these are very similar in the way they are built and how they function, this was an opportunity to see if their minor architectural differences would be reflected in some way in the accuracy scores, in the context of fall detection.

Table 3.1: RNN initial structure (GRU, 12-length input sequence)

Layer type	**Output shape**
Input Layer	(None, 12, 1,024)
GRU	(None, 12, 16)
GRU	(None, 8)
[Dropout]	(None, 8)
Dense (ReLU)	(None, 8)
[Dropout]	(None, 8)
Dense (softmax)	(None, 2)

The initial layer composition, inspired by previous endeavors in visual action recognition, was designed to resemble a relatively straightforward yet efficient RNN, as illustrated in table 3.1, which shows a GRU network configured to accept sequences of length 12. Dropout layers are included in the table but were not always present in the models themselves, as they were contingent upon the detection of overfitting during training. Should the need arise to enhance the capacity of this structure in detecting pertinent patterns within the input tensors, additional RNN and/or Dense layers may have been incorporated, or their outputs modified. All parameters within the model were trainable. The output of this model consists

of two numerical values ranging from 0 to 1, where the larger value signifies the class to which the input video belongs.

The initial hypothesis suggested that RNNs would possess superior capabilities in detecting fall-related motions due to their sequential nature. However, to examine this conjecture, CNN models were also employed. Similar to RNNs, these CNN models had to remain simple in order to prevent overfitting, considering the utilization of relatively uncomplicated tensors as input objects. The existing sophisticated CNN architectures, such as ResNet [118], EfficientNet, and Inception [119], were deemed unsuitable due to their complexity. Consequently, a CNN model was constructed from scratch for this purpose.

Table 3.2 displays an initial configuration of the CNN. Similar to the RNNs, the composition of the CNN underwent several iterations, involving the addition or removal of convolution and dense layers. Additionally, the inclusion or exclusion of dropout layers was determined based on the specific circumstances. The present architecture consists of a total of 3,204,178 trainable parameters.

Table 3.2: CNN initial structure (12-length input sequence)

Layer type	**Output shape**
Conv2D	(None, 12, 1,024, 32)
MaxPool2D	(None, 6, 512, 32)
Conv2D	(None, 6, 512, 64)
MaxPool2D	(None, 3, 256, 64)
Conv2D	(None, 3, 256, 64)
Flatten	(None, 49,152)
[Dropout]	(None, 49,152)
Dense (ReLU)	(None, 64)
[Dropout]	(None, 64)
Dense (ReLU)	(None, 32)
[Dropout]	(None, 32)
Dense (ReLU)	(None, 16)
[Dropout]	(None, 16)
Dense (softmax)	(None, 2)

Naturally, both of these structures were altered more substantially throughout the course of testing by changing the number of Dense, RNN, or convolutional layers. Table 3.3 exemplifies one such variation, which was an attempt to build a slightly more complex LSTM compared to the initial structure.

3.5.3 Final hybrid structure

Combining the truncated version of the FE models with either an RNN or CNN results in an hybrid structure with upwards of 23,000,000 parameters. The program that incorporates this model and used in the FDS is expected to access the camera's stream and extract three-second video segments which are then introduced to the model unchanged. The model receives the video file and performs FE. According to observations taken during testing several models on a computer running a NVIDIA GeForce GTX 1080 Ti, the time it takes to feature extract three-second video files, assuming GPU acceleration is being applied, averages at $\sim$1.3 seconds per video, which is well within the acceptable computing time

Table 3.3: A more complex LSTM structure, nicknamed LSTM-MCQ2 (20-length input sequence)

Layer type	**Output shape**
Input Layer	(None, 20, 1,024)
LSTM	(None, 20, 64)
LSTM	(None, 20, 32)
LSTM	(None, 16)
[Dropout]	(None, 16)
Dense (ReLU)	(None, 16)
[Dropout]	(None, 16)
Dense (ReLU)	(None, 8)
[Dropout]	(None, 8)
Dense (softmax)	(None, 2)

range. Following that, the resulting tensor is inputted to the GRU network, which classifies it – again assuming with GPU acceleration, this process is nigh instantaneous, but it is expected that even without it the computing time should not be excessively long. The two output values are compared, and class is immediately ascertained. If the video is classified as a "fall", then the program can proceed with raising the alarm. The model's workflow can be visualized in figure 3.12. Although the integration of two models in this way is a commonly employed technique in current research, none of the existing systems have specifically utilized the I3D model, despite its proven effectiveness.

Figure 3.12: Fall detection hybrid model workflow.

Chapter 4

Testing and Evaluations

This chapter focuses on the crucial component of an FDS, namely the fall detection model, as it has the most influence on system performance. The analysis of various models and the assessment of their accuracy, along with other pertinent metrics, are discussed herein. The methodology employed for conducting these tests is initially elucidated, followed by an exploration of the most meaningful metrics for fall detection. Subsequently, the case studies are presented, along with the corresponding outcomes obtained.

4.1 Methodology

The datasets underwent preprocessing and augmentation prior to being used as input for the FE models. These models produced tensors, which were combined with their corresponding labels and filenames, resulting in separate dictionaries for each dataset. To determine the allocation of datasets for training, validation, and testing, certain considerations were taken into account. The original dataset, UP-Fall, and Multicam were deemed too large to exclude from the train set, while HQFSD contained crucial quality scenarios that should not be omitted. Consequently, out of the remaining datasets, FDDBg, for being the bigger dataset, was designated as the validation set, and UR-Fall was chosen as the test set.

It is important to highlight that the datasets employed in this study still do not provide a complete assessment of the model's performance. Swapping the validation and test splits would yield entirely different performance scores. Nonetheless, given the limited availability of sufficient data, this approach was considered the minimum requirement. Prior to utilization, all sets had their data shuffled to ensure randomness in the input to the model.

The primary focus of scrutiny in this investigation pertained to the classification models. Regarding the functional aspects of the FE models, the only modification made was the adjustment of the frame skip value, determining the number of frames processed and the input size of the classifier. The implementation of model checkpoints, facilitated by the utilization of Keras' callback library, allowed for automatic preservation of the model's parameters in case the validation results exhibited improvement during training. Consequently, at the conclusion of the training process, the weights acquired at the best epoch, based on validation accuracy and loss, were loaded onto the model for the subsequent testing. Additionally, the number of training epochs was subject to modification. Lastly, the optimizer and type of loss were regarded as mutable parameters within the model's compiler.

To summarize, the variables monitored or adjusted to assess and enhance the model's performance encompassed the following factors:

- FE models - Number of input frames sequences

- Classification model structure, including:
  - Number of dropout layers and percentage of neurons dropped
  - Number and format of conv/RNN/dense layers
- Classification model compiler:
  - Loss type
  - Optimizer
  - Learning rate (decay)
- Number of epochs
- Class weights

The collection of hyperparameters for a specific model configuration is designated as λ. λ was altered as seen fit in order to increase validation accuracy as much as possible. The configuration which managed to perform above all others was then tested on the test set. When constructing an FDS, certain evaluation metrics hold greater importance than accuracy alone. The avoidance of false negatives (undetected falls) is of utmost importance. Specifically, recall (also known as true positive rate) and its counterpart, miss rate (false negative rate), were given higher priority during result analysis, following the logic laid out in section 2.1.3. Nevertheless, accuracy, precision, specificity, and F1 score were also considered and exerted some influence when comparing different models.

The testing procedure entailed the utilization of one of the FE models, which processed all datasets. This model was configured to partition each video into a predetermined number of frames. The resulting outputs were then organized into train, validation, and test sets. To ensure randomness, the order of elements within each set was randomized. Subsequently, an initial iteration of the classification model was trained using the train set, and its performance was assessed on the validation set. Various minor modifications were introduced to the structure and/or compiler of the classification model, as well as adjustments to the training process, such as altering the number of epochs or adjusting class weights. The modified model was then trained and validated accordingly. This iterative process continued until all possible modifications were exhausted. The validation outcomes across all model iterations were compared, and the model that exhibited the best performance was subsequently evaluated using the test set. This comprehensive evaluation was conducted for each classification model. Once the testing of all three classification models was completed, the FE model was then modified, and the entire process was repeated. This iterative procedure was conducted for all FE models employed in the study.

4.2 Case studies

One case study was developed for each of the tested FE models. These case studies include comprehensive tests utilizing all the classification models previously mentioned so that every possible feature model/classification model combination is tested. The output structure of all FE models were homogenized, so that the classification models could be used in any combination with minimal alterations. However, while the EfficientNetV2 and ResNeXt-101 are one-stream, in its related case study, the I3D was configured to use both its streams to output RGB frames, optical flow frames, and both. Each feature type was tested with all

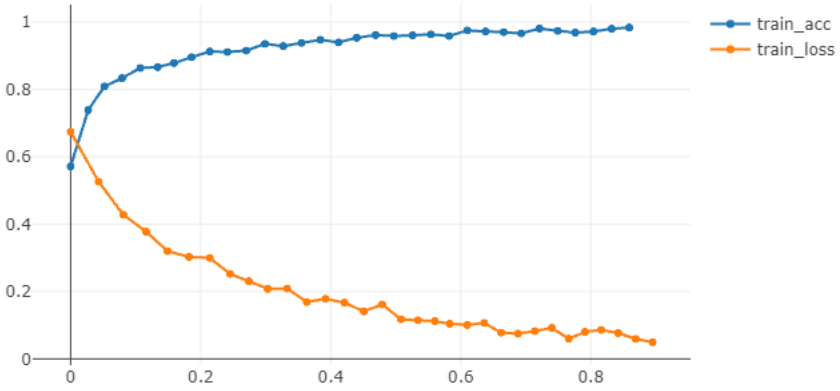

Figure 4.1: Evolution of training accuracy (blue) and loss (orange) over 89 epochs - ResNeXt-101/LSTM(-MCQ).

classification models. Additionally, experiments with an intact, fine-tuned I3D were included, but only using both feature types. The RNN models were fitted with a *Flatten*-imbued *TimeDistributed* layer when necessary, as the extra dimension created by both feature types is incompatible with its possible input shapes, unlike with the CNN and I3D.

As such, case studies include the following model combinations:

- 1st case study - EfficientNetV2
  - ENet/LSTM
  - ENet/GRU
  - ENet/CNN
- 2nd case study - ResNeXt-101
  - RNeXt/LSTM
  - RNeXt/GRU
  - RNeXt/CNN
- 3rd case study - I3D
  - I3D-RGB
    - I3D-RGB/LSTM
    - I3D-RGB/GRU
    - I3D-RGB/CNN
  - I3D-OF (optical flow)
    - I3D-OF/LSTM

    - I3D-OF/GRU
    - I3D-OF/CNN
- I3D-Both
    - I3D-Both/LSTM
    - I3D-Both/GRU
    - I3D-Both/CNN
- FT-I3D

4.2.1 1st case study - EfficientNetV2

Tables 4.1 and 4.2 show the results of each model combination's validation and testing processes, respectively. All metrics were taken from the same model, chosen by the greatest overall validation performance belonging to each model combination: for instance, the results presented by the ENet/LSTM in table 4.1 are from the same model as the metrics attributed to ENet/LSTM in table 4.2. The model combination which achieved the highest F1 score on the test set is highlighted in both tables.

Table 4.1: EfficientNetV2 - Validation results on FDDBg

Model	Accuracy	Recall	Specificity	Precision	F1 score
ENet/LSTM	82.8%	87.4%	78.1%	80.0%	83.5%
ENet/GRU	90.0%	88.7%	91.4%	91.2%	89.9%
ENet/CNN	**90.7%**	**90.0%**	**91.4%**	**91.3%**	**90.7%**

Table 4.2: EfficientNetV2 - Test results on UR-Fall

Model	Accuracy	Recall	Specificity	Precision	F1 score
ENet/LSTM	80.0%	85.0%	75.0%	77.3%	81.0%
ENet/GRU	75.0%	80.0%	70.0%	72.7%	76.2%
ENet/CNN	**83.3%**	**81.7%**	**85.0%**	**84.5%**	**83.1%**

Globally, all metrics average at a little below of 90% and present what could be considered acceptable performances, barring the ENet/GRU which barely reaches a test accuracy of 80%. However, as can be seen in the studies to follow, these results are unusual in the sense that the classification CNN managed to slightly outperform both RNN models. Compared to the ENet/LSTM, the second-best model, the ENet/CNN achieves a 2% higher F1 score, but also exhibits a 3.3% lower recall value. Since recall is a prioritized metric in this context, one could argue that the LSTM is the preferable choice, despite the lackluster specificity and precision scores. Unsurprisingly, the test results are overall lower than the validation results.

4.2.2 2nd case study - 3D ResNeXt-101

Tables 4.3 and 4.4, similar to the previous two, show the results of each model combination's validation and testing processes. The best-performing feature type for each model combination (F1 score) is again highlighted in both tables.

Table 4.3: ResNeXt-101 - Validation results on FDDBg

Model	Accuracy	Recall	Specificity	Precision	F1 score
RNeXt/LSTM	92.4%	93.4%	91.4%	91.6%	91.6%
RNeXt/GRU	**94.4%**	**92.1%**	**96.7%**	**96.5%**	**94.2%**
RNeXt/CNN	92.4%	86.1%	98.7%	98.5%	91.9%

Table 4.4: ResNeXt-101 - Test results on UR-Fall

Model	Accuracy	Recall	Specificity	Precision	F1 score
RNeXt/LSTM	83.3%	86.7%	80.0%	81.3%	83.9%
RNeXt/GRU	**87.5%**	**90.0%**	**85.0%**	**85.7%**	**87.8%**
RNeXt/CNN	81.7%	76.7%	86.7%	85.2%	80.7%

Unlike the previous case study, the classification CNN does slightly outperform the LSTM on the validation set, but is the worst model on the test set, which is more in line with the notion that the RNNs have a supposed edge when classifying data of this type. The GRU now takes first place in both sets, displaying an 87.8% test F1 score and just breaking the 90% barrier in recall. The difference in validation and test scores is more pronounced.

4.2.3 3rd case study - I3D

The most effective feature type for each combination of models (as measured by F1 score) is indicated in both 4.5 and 4.6. The performance observed on the validation set consistently surpasses that of the test set, with the I3D/GRU combination attaining an impressive 98% F1 score. Conversely, this same model configuration only achieves a 94.3% F1 score on the test set. Although inferior to the I3D/LSTM combination in F1 score, the I3D/GRU exhibits a recall score surpassing the latter by over 5%. Notably, while all models perform better with both feature types or optical flow on the validation set, the test set yields the best results exclusively with RGB features. This discrepancy further supports the notion that outcomes significantly vary across datasets. The fine-tuned I3D model obtains relatively lower F1 scores of 91% and 90.5%.

Table 4.5: I3D - Validation results on FDDBg

Model	Feature type	Accuracy	Recall	Specificity	Precision	F1 score
I3D/LSTM	RGB features	93.4%	92.1%	94.7%	94.6%	93.3%
	Optical flow	94.4%	94.7%	94.0%	94.1%	94.4%
	Both	**95.7%**	**94.0%**	**97.4%**	**97.3%**	**95.6%**
I3D/GRU	RGB features	92.1%	94.7%	89.4%	89.9%	92.3%
	Optical flow	97.0%	96.0%	98.0%	98.0%	97.0%
	Both	**98.0%**	**97.4%**	**98.7%**	**98.7%**	**98.0%**
I3D/CNN	RGB features	92.4%	90.7%	94.0%	93.8%	92.3%
	Optical flow	**94.7%**	**94.7%**	**94.7%**	**94.7%**	**94.7%**
	Both	94.4%	92.7%	96.0%	95.9%	94.3%
FT-I3D	Both	91.4%	86.8%	96.0%	95.6%	91.0%

Table 4.6: I3D - Test results on UR-Fall

Model	Feature type	Accuracy	Recall	Specificity	Precision	F1 score
I3D/LSTM	**RGB features**	**95.8%**	**91.2%**	**100%**	**100%**	**95.6%**
	Optical flow	90.0%	93.3%	86.7%	87.5%	90.3%
	Both	92.5%	95.0%	90.0%	90.5%	92.7%
I3D/GRU	**RGB features**	**94.2%**	**96.7%**	**91.7%**	**92.1%**	**94.3%**
	Optical flow	82.5%	80.0%	85.0%	84.2%	82.1%
	Both	89.0%	90.0%	88.3%	88.5%	89.2%
I3D/CNN	**RGB features**	**92.5%**	**90.0%**	**95.0%**	**94.7%**	**92.3%**
	Optical flow	85.0%	76.7%	98.3%	92.0%	83.6%
	Both	85.0%	81.7%	88.3%	87.5%	84.5%
FT-I3D	Both	90.0%	95.0%	85.0%	86.4%	90.5%

4.3 Discussion of results

As hypothesized, the I3D model demonstrates superior performance compared to the other FE models, while the RNN models exhibit higher accuracy than the CNN models (with the the exception of the first case study) - this is thought to be due to the latter's limited ability to capture temporal patterns. A preliminary examination of misclassified videos by the CNN model indicates a tendency to incorrectly classify actions involving getting up and lying down, suggesting a difficulty in discerning these actions from falls. Despite the overall better results achieved by the I3D-RGB/LSTM, the I3D-RGB/GRU exhibits a 5.5% higher recall value, which can be considered a suitable compensation for the 1.3% difference in F1 score and comparatively lower specificity and precision values. Consequently, the I3D-RGB/GRU model should be regarded as the superior fall detection model.

However consistent throughout the last two case studies the results of each model combination were, the first case study stands out as being a striking exception. If the data contains primarily temporal distinctions, and the EfficientNet is not capable of effectively capturing it, it would be expected that the RNNs, which are designed to model temporal dependencies, would be able to compensate for this limitation to some extent. The discrepancy could be explained by realizing that, while the RNNs can capture temporal dependencies, both still require compatible features as input. If the features extracted by the EfficientNet do not contain relevant temporal information, the RNNs may not be able to fully exploit their capabilities. In this case, the EfficientNet/CNN combination, which focuses on spatial features, might perform better due to the nature of the available features.

Surprisingly, among the hyperparameters in λ, the number of frames in the input sequences appears to have a negligible effect. The manipulation of the model's input size, whether increased or decreased, does not yield any noticeable improvement or deterioration in performance, contrary to the theorized expectations. Various sequence lengths ranging from 10 to 20 were evaluated, yielding no discernible distinctions, except for the variation in FE times, resulting in either slower or faster processing.

In contrast, the structure of the classification models holds significant sway over their performance, favoring simpler models overall. The most successful model, incorporating a GRU, mirrors the structure provided in table 3.1. It was compiled using an *Adam* optimizer with a slightly augmented learning rate. Conversely, as the model complexity increased, with the

inclusion of additional layers, its performance suffered unless accompanied by dropout layers or reduced learning rates. The impact of learning rate decay was also minimal, exhibiting limited discernible influence. Nonetheless, it was observed in other tests, most notably in the second case study, that certain models exhibited improved performance when incorporating more complex structures. Case in point, the most effective iteration of the GRU model coupled with the ResNeXt employed the MCQ structure, which consisted of two GRU layers and four dense layers, and used the Stochastic Gradient Descent (SGD) optimizer with a slightly lower learning rate than default.

Regarding the influence of class weights, the outcomes were mixed. While in some cases they succeeded in augmenting recall rates, in others, no discernible difference was observed, or they even had an unintended inverse effect. It seems that, despite their overall simplicity, the classification models were still complex enough to fit data despite the weights. An examination of the training logs revealed there to be a small increase in effectiveness of the class weights when applied to simpler models.

As previously mentioned, when examining all FE models, it is observed that the test results generally exhibit inferior performance compared to the validation results. Nevertheless, it is noteworthy to point out that the best validation outcomes do not correspond to the best test outcomes. In the case of the I3D, the model's accuracy on the test set is lower when applied to optical flow or both features, which contradicts the behavior observed during validation. This discrepancy emphasizes the significant impact of using different datasets, as altering the datasets would likely yield considerably distinct results. Although the model has demonstrated promising potential, it is crucial to acknowledge this aspect. The relatively unsatisfactory outcomes of the fine-tuned I3D model can be attributed to its complexity, whereby the limited availability of data hampers its effective tuning, resulting in poorer performance when compared to the simpler RNN and CNN models.

Table 4.7 contains the results of this approach's best model against those stated by the authors of some of the more similar visual state of the art FDSs.

Table 4.7: Comparison between this approach and (some) state of the art systems

FDS	Model	Accuracy	Recall	Specificity	Precision	F1 score
This approach	**I3D/GRU**	**94.2%**	**96.7%**	**91.7%**	**92.1%**	**94.3%**
Adhikari et al.	CNN	74%	-	-	-	-
Fan et al.	Filters/LSTM	-	91%	-	92%	92%
Ma et al.	3D CNN/ Autoencoder	-	93.3%	92.8%	-	-
Rahnemoonfar and Alkittaw	3D CNN	97.6% 93.2%	-	-	-	-
Lu et al.	3D CNN/LSTM	99.1%	-	-	-	-
Khraief et al.	Optical flow/CNN	99.7%	-	-	-	-

In general, the I3D/GRU demonstrates a good performance, surpassing several other FDSs. However, as mentioned earlier, the FDSs listed in table 4.7 were either trained on a singular dataset or neglected to implement appropriate measures for separating training and testing sets, typically relying on the holdout method or similar approaches. Among the few FDSs that did address this concern, such as Fan et al. [67], their results largely align with the findings of this study. Notably, they achieved an impressive accuracy of 98% when evaluating

the FDDBg dataset, yet their performance dropped to 74% and 63% when testing against HQFSD and their own dataset, respectively. These outcomes emphasize the significant impact of dataset variations on the apparent performance of alternative fall detection systems. It is essential to recognize that the handling of data, along with the intrinsic characteristics of the data itself during the stages of training, validation, and testing, can greatly influence the performance metrics of such systems.

It is important to highlight that the presented results do not explicitly demonstrate the EfficientNet's ability to generate features more rapidly, thereby necessitating fewer computational resources, in comparison to 3D CNNs. The efficiency is so pronounced that utilizing GPU acceleration to maintain computation times under 3 seconds becomes unnecessary. This advantage, which makes up for its inferiority in other aspects, holds considerable apparent significance. Conversely, it should be noted that even the most basic contemporary computers are equipped with GPUs capable of acceleration. Consequently, the significance of this advantage is contingent upon the manner in which a FDS implements the model.

To summarize, the outcomes obtained unequivocally demonstrate the potential of the hybrid I3D/RNN model in the realm of visual fall detection, aligning with its outstanding performance in general action recognition tasks. In direct comparison to other popular models that underwent testing, the proposed model surpasses them in terms of both accuracy and recall rates. Implementing a FDS based on this model would significantly enhance the safety and quality of life for elderly individuals. By meticulously handling the training data, while being mindful of its limitations, we can express greater confidence in the model's ability to generalize and perform effectively in unfamiliar real-world environments, surpassing the capabilities of other FDSs.

Chapter 5

Conclusions

In this final chapter, the project is reviewed as a whole first by examining if and to what extent the goals set in the inception of this thesis were met. Then, comments are made regarding the possible, eventual continuation of this project and the research herein, either for the purpose of correcting any defects in the final result or improving its quality further. Lastly, there are some observations and personal remarks concerning the results themselves, reflections on the work done throughout this thesis and a few statements on the hurdles met along the way.

5.1 Accomplished goals

In the course of preparation and planning for this thesis, the author set several objectives with the end goal of building of an AI model which could consistently and accurately detect human falls in videos. All objectives were accomplished, and this section details how and to what extent each objective was met.

5.1.1 Objectives 1 and 2 - State of the art

An exploration of the state of the art of the automatic fall detection field allowed for the formation of an hypothesis. This hypothesis states that the combination of the I3D's FE capabilities with a RNN would provide the greatest overall performance out of all other models already implemented. The state of the art analysis revealed also glaring flaws related to the methods with which most other researchers handled their training data. These mistakes were caused by ignoring the inherent flaws of available data, flaws for which, presently, there are no solutions, but only remedies.

5.1.2 Objectives 3, 4 and 5 - Datasets

To avoid repeating the aforementioned mistakes, to properly recognize these limitations, it was first deemed necessary to compile data to create as varied, balanced and big a dataset as possible. This was accomplished by accessing publicly available datasets, created by other researchers, for fall detection in mind. On top of this, made possible by acquiring a suitable RGB video camera, a substantial amount of videos were recorded and added to the compilation datasets. The resulting dataset encompassed more than 2,900 videos, and it is, to date[1], the biggest and most diverse collection of fall detection video segments ever used in the training of a fall detection model.

[1] June, 2023

5.1.3 Objective 6 - Preprocessing and video augmentation

The videos, or sets of images, collected underwent an editing process with the goal to further increase the dataset's size, to remove unwanted data, and to standardize the remaining segments. Image augmentation techniques were applied to all videos, increasing much needed diversity of data. In both of these steps, the compilation and creation of data, several programs were written to automatize and streamline the processes of data collection and organization.

5.1.4 Objective 7 - Model construction, training and testing

With data having been gathered, the model structure devised initially could then be built and trained. In an effort to solidify the final results and challenge the I3D/RNN hypothesis, several models were obtained or created with performance comparison in mind. A pre-trained I3D was configured to extract features in RGB, optical flow, and two-stream modes, and was paired with three types of classification models, LSTMs, GRUs, and CNNs. An additional pre-trained I3D was fine-tuned to also perform classification. Two other FE models, prevalent in other FDSs, were put up against the I3D: a 2D CNN, an EfficientNetV2, and a traditional 3D CNN, a modified ResNeXt-101, two models which have achieved impressive results in general action recognition and fall detection tasks alike. The extra feature extractors were used in conjunction with the three classifiers as well. To ensure that the results obtained during testing were as representative to real-life use as possible, two entire datasets were reserved for validation and testing - this was the only way one could ensure completely unseen data at this stage of development.

5.1.5 Final results

The test results clearly demonstrate the potential of the novel I3D/RNN hybrid model in the context of visual fall detection. In comparison with other popular visual-based models, the proposed model often equals or outperforms in both accuracy and recall rates, but, by going the extra mile to handle training data carefully, conscious of its limitations, the results obtained by this model can be attested to with much more confidence than other FDSs. The author is certain in stating that this model is a visible improvement on others similar, and the implementation of an FDS utilizing this model would have a significant positive and distinct impact on the safety and quality of life for elderly individuals and the physically impaired.

5.2 Future work

Automatic fall detection, as a field of research, is very recent and using AI to achieve it is a very novel idea still, but one that has shown promise over dependence on conventional techniques. As ML technology itself starts to mature, new accurate, robust, and practical solutions to old problems start to appear – when computers begin to be able to process imagery intelligently, a myriad of new possibilities arise. But the problem plaguing the state of automatic fall detection is the lack of data, which completely undermines ML-driven solutions. It is possible to use ML for this goal to great effect as it does have the potential to out-perform any other type of implementation, but without the data, it is hard to prove it.

Image or object recognition and, to a lesser but similar extent, general action recognition technologies, on the contrary, allow researchers access to not just huge datasets, but datasets capable of setting an "industry" standard. Of course, researchers do not all use the same datasets in the exact same way, but these datasets – UCF-101, Kinetics, ImageNet, to name a few – offer the opportunity for accurate and transparent comparison. They are good enough to train and test all kinds of models and allow for an accurate portrayal of the model's capabilities in the real world. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[2], for example, pits models trained on ImageNet (alone) against one another to see which is best at object detection and to give researchers a chance to visualize scientific progress in that field.

There is no ImageNet for visual fall detection so the field cannot hope to replicate this any time soon, or at least with what it is available now. It is admittedly difficult to build a fall dataset: data cannot be extracted online, nor outsourced, nor requested from hospitals and institutions. Falls must be simulated, which necessitates planning, equipment, manpower, time, even creativity. To create a dataset of the same level of size and quality as Kinetics or UFC-101 requires a years-long project with actual funding. It is a herculean task, but one that could be a gigantic step forward in progressing visual fall detection. Additionally, perhaps creating a multimodal dataset would further the fall detection as a whole, not just the visual subfield. But until this is done, it is unlikely that ML fall detection systems will find use in the real world.

That being said, progress is possible, even if by brute force. The model developed for this thesis, regardless of its performance, can never be too close to perfect. There is always work to be done to tweak it, to obtain better train data, all in the name of improving accuracy and sensitivity. Maybe an entirely different model is more suitable, perhaps a new FE technique is the missing piece. Repeated experimentation and validation and keeping abreast of the state of the art is a necessity in continuous ML development.

One more possibility for future work is discarding RGB imaging in favor of thermal, depth, or radar sensors to preserve the elderly's sense of privacy. With computer vision algorithms becoming increasingly more robust, using them on imaging that is traditionally considered less feature-rich or detailed than RGB becomes a real possibility.

5.3 Final appreciations

This project is influenced greatly by a previous fall detection project, developed two years ago as part of the author's licence degree in Software Engineering course. The techniques and knowledge applied there had to be significantly improved upon in this one. In that project, a few existing datasets were also compiled, although in much lesser number, with the total amount of videos barely surpassing 500. Despite this, no original dataset was recorded to compensate. Even worse, the entire dataset was simply divided, randomly, into train, validation, and test sets. The model was a hybrid 2D CNN (MobileNet)/GRU, which, at the time, was unprecedented in fall detection, but its performance was not compared to any alternative model. Finally, testing was shallow and did not go much further than changing the learning rate and inserting dropout layers. Nevertheless, the model was rated as 97% accurate, surpassing most other FDSs. In hindsight, the amazing results obtained were

[2]Old website: `https://www.image-net.org/challenges/LSVRC/`, Kaggle: `https://www.kaggle.com/competitions/imagenet-object-localization-challenge/overview`

clearly the consequence of severe overfitting brought about by an extremely small dataset, haphazardly distributed throughout the three sets.

Going in to this thesis, it was clear that if the same problem was to be tackled, that some aspects had to be significantly improved. In an effort to be innovative, it was thought from the get go that the amount of data had to be increased, and that an original dataset was to be created. Following the state of the art's exploration, the new model would similar to the last, but the 2D CNN would be upgraded to 3D. Instead of just a GRU, several classification models would be tried, and tested more thoroughly.

These measures by themselves were a big step up. However, after consulting with an expert, it later became clear that even after enlarging the dataset, the validation/test sets had to have more degrees of separation from the train set, which prompted the way data is used in this project. The past project also became evidence that the accuracy of visual fall detection models cannot be taken entirely at face value, not even with a large enough training set, if no steps were taken to ensure the complete unfamiliarity of test data - an observation repeated in many of the other FDSs. It was almost to be that the same mistakes would be made here, if not for this correction from said expert, whose input caused a dramatic change to course of this project.

Superficially, the I3D/GRU fall detection model, 94.2% accurate and 96.7% sensitive, might seem to perform worse than the MobileNet/GRU, but, in reality, it is leaps and bounds superior to its previous form. Thanks to this project, the newer model can be considered, with great confidence, to have improved on the body of work preceded by it, of the author and others, and have a use in actual elderly care. In conclusion, this author genuinely hopes that, in the near future, even better models, tested responsibly, can transcend the theoretical and experimental phase they find themselves in still and find their use in real-life applications.

Bibliography

[1] United Nations, Population Division, *Home Page \textbar Data Portal*, en, Publication Title: Population Division Data Portal. [Online]. Available: https://population.un.org/dataportal/home (visited on 05/31/2023).

[2] *Age dependency ratio*, Publication Title: Our World in Data. [Online]. Available: https://ourworldindata.org/grapher/age-dependency-ratio-of-working-age-population (visited on 05/31/2023).

[3] *Europe: Share of elderly population by country 2021*, en, Publication Title: Statista. [Online]. Available: https://www.statista.com/statistics/1105835/share-of-elderly-population-in-europe-by-country/ (visited on 05/31/2023).

[4] *Índice de envelhecimento e outros indicadores de envelhecimento*. [Online]. Available: https://www.pordata.pt/portugal/indice+de+envelhecimento+e+outros+indicadores+de+envelhecimento-526 (visited on 05/31/2023).

[5] *Agregados domésticos privados unipessoais: Total e de indivíduos com 65 e mais anos*. [Online]. Available: https://www.pordata.pt/portugal/agregados+domesticos+privados+unipessoais+total+e+de+individuos+com+65+e+mais+anos-822 (visited on 05/31/2023).

[6] *População residente: Total e por grupo etário*. [Online]. Available: https://www.pordata.pt/portugal/populacao+residente+total+e+por+grupo+etario-10 (visited on 06/15/2023).

[7] J. F. D. Mendes, "Lares e idosos : Perspetiva bioética da pastoral da saúde," por, doctoralThesis, Universidade Católica Portuguesa, Feb. 2016. [Online]. Available: https://repositorio.ucp.pt/handle/10400.14/21227 (visited on 05/31/2023).

[8] *Statistics \textbar Eurostat*. [Online]. Available: https://ec.europa.eu/eurostat/databrowser/view/ilc%5C_lvps30/default/table?lang=en (visited on 05/31/2023).

[9] C. S. Palma, "Quedas nos idosos: Do risco à prevenção," PhD Thesis, Instituto Politécnico de Beja, Escola Superior de Saúde, 2012. [Online]. Available: https://comum.rcaap.pt/bitstream/10400.26/3975/1/Relat%5C%c3%5C%b3rio%5C%20final.pdf (visited on 05/31/2023).

[10] *All about Emergency Cords*, en, Feb. 2019. [Online]. Available: https://accessible-toilets.co.uk/2019/02/07/all-about-emergency-cords/ (visited on 06/19/2023).

[11] *Best Medical Alert Systems With Fall Detection*, en. [Online]. Available: //www.usnews.com/360-reviews/services/medical-alert-system/fall-detection (visited on 06/19/2023).

[12] M. Ratnayake, S. Lukas, S. Brathwaite, J. Neave, and H. Henry, "Aging in Place:" *Dela J Public Health*, vol. 8, no. 3, pp. 28–31, Aug. 2022, issn: 2639-6378. doi: 10.32481/djph.2022.08.007. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9495472/ (visited on 05/31/2023).

[13] U. Anliker, J. Ward, P. Lukowicz, *et al.*, "AMON: A wearable multiparameter medical monitoring and alert system," *IEEE Transactions on Information Technology in*

Biomedicine, vol. 8, no. 4, pp. 415–427, Dec. 2004, issn: 1558-0032. doi: 10.1109/TITB.2004.837888.

[14] V. L. Richmond, S. Davey, K. Griggs, and G. Havenith, "Prediction of Core Body Temperature from Multiple Variables," eng, *Ann Occup Hyg*, vol. 59, no. 9, pp. 1168–1178, Nov. 2015, issn: 1475-3162. doi: 10.1093/annhyg/mev054.

[15] G. Schwartz, B. C.-K. Tee, J. Mei, *et al.*, "Flexible polymer transistors with high pressure sensitivity for application in electronic skin and health monitoring," en, *Nat Commun*, vol. 4, no. 1, p. 1859, May 2013, issn: 2041-1723. doi: 10.1038/ncomms2832. [Online]. Available: https://www.nature.com/articles/ncomms2832 (visited on 05/31/2023).

[16] Z. Wang, Z. Yang, and T. Dong, "A Review of Wearable Technologies for Elderly Care that Can Accurately Track Indoor Position, Recognize Physical Activities and Monitor Vital Signs in Real Time," en, *Sensors*, vol. 17, no. 2, p. 341, Feb. 2017, issn: 1424-8220. doi: 10.3390/s17020341. [Online]. Available: %5Cunderline%7Bhttps://www.mdpi.com/1424-8220/17/2/341%7D (visited on 05/31/2023).

[17] M. Begum, R. Wang, R. Huq, and A. Mihailidis, "Performance of daily activities by older adults with dementia: The role of an assistive robot," eng, *IEEE Int Conf Rehabil Robot*, vol. 2013, p. 6 650 405, Jun. 2013, issn: 1945-7901. doi: 10.1109/ICORR.2013.6650405.

[18] S. C. Kramer, E. Friedmann, and P. L. Bernstein, "Comparison of the Effect of Human Interaction, Animal-Assisted Therapy, and AIBO-Assisted Therapy on Long-Term Care Residents with Dementia," *Anthrozoös*, vol. 22, no. 1, pp. 43–57, Mar. 2009, issn: 0892-7936. doi: 10.2752/175303708X390464. [Online]. Available: https://doi.org/10.2752/175303708X390464 (visited on 05/31/2023).

[19] M. Hersh, "Overcoming Barriers and Increasing Independence – Service Robots for Elderly and Disabled People," en, *International Journal of Advanced Robotic Systems*, vol. 12, no. 8, p. 114, Aug. 2015, issn: 1729-8806. doi: 10.5772/59230. [Online]. Available: https://doi.org/10.5772/59230 (visited on 05/31/2023).

[20] E. G. Christoforou, A. S. Panayides, S. Avgousti, P. Masouras, and C. S. Pattichis, "An Overview of Assistive Robotics and Technologies for Elderly Care," en, in *XV Mediterranean Conference on Medical and Biological Engineering and Computing – MEDICON 2019*, J. Henriques, N. Neves, and P. de Carvalho, Eds., ser. IFMBE Proceedings, Cham: Springer International Publishing, 2020, pp. 971–976, isbn: 978-3-030-31635-8. doi: 10.1007/978-3-030-31635-8_118.

[21] P. JACKSON, *Introduction to expert systems*, English, 2nd edition. Wokingham, England ; Reading, Mass: Addison Wesley, Jan. 1990, isbn: 978-0-201-17578-3.

[22] J. Gama, A. C. Lorena, K. Faceli, M. Oliveira, and A. P. d. L. Carvalho, *Extração de Conhecimento de Dados*, por, 3rd edition. Edições Sílabo, 2017, isbn: 978-972-618-914-5.

[23] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, English. Cambridge, MA: The MIT Press, Aug. 2012, isbn: 978-0-262-01825-8.

[24] P. F. Brown, J. Cocke, S. A. D. Pietra, *et al.*, "A statistical approach to machine translation," *Comput. Linguist.*, vol. 16, no. 2, pp. 79–85, Jun. 1990, issn: 0891-2017.

[25] F. Stahlberg, "Neural Machine Translation: A Review," *Journal of Artificial Intelligence Research*, vol. 69, pp. 343–418, Oct. 2020. doi: 10.1613/jair.1.12007.

[26] H. Somers, "Machine Translation: History, Development, and Limitations," in Jan. 2012. doi: 10.1093/oxfordhb/9780199239306.013.0029.

[27] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and Decision-Making for Autonomous Vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, May 2018. doi: 10.1146/annurev-control-060117-105157.

[28] A. Faisal, T. Yigitcanlar, M. Kamruzzaman, and G. Currie, "Understanding autonomous vehicles: A systematic literature review on capability, impact, planning and policy," en, *J Transp Land Use*, 2019, Accepted: 2019-12-16T20:43:26Z. doi: 10.5198/jtlu.2019.1405. [Online]. Available: http://conservancy.umn.edu/handle/11299/209218 (visited on 06/15/2023).

[29] S. Liu, L. Li, J. Tang, S. Wu, and J.-L. Gaudiot, *Creating Autonomous Vehicle Systems, Second Edition*, English, 2nd edition. Springer, Sep. 2020, isbn: 978-3-031-00677-7.

[30] Z.-H. Zhou and Y. Jiang, "Medical diagnosis with C4.5 Rule preceded by artificial neural network ensemble," eng, *IEEE Trans Inf Technol Biomed*, vol. 7, no. 1, pp. 37–42, Mar. 2003, issn: 1089-7771. doi: 10.1109/titb.2003.808498.

[31] M. Jamshidi, A. Lalbakhsh, J. Talla, *et al.*, "Artificial Intelligence and COVID-19: Deep Learning Approaches for Diagnosis and Treatment," *IEEE Access*, vol. 8, pp. 109 581–109 595, 2020, Conference Name: IEEE Access, issn: 2169-3536. doi: 10.1109/ACCESS.2020.3001973.

[32] A. Barragán Montero, U. Javaid, G. Valdes, *et al.*, "Artificial intelligence and machine learning for medical imaging: A technology review," *Physica Medica*, vol. 83, pp. 242–256, Mar. 2021. doi: 10.1016/j.ejmp.2021.04.016.

[33] N. Altrabsheh, M. Cocea, and S. Fallahkhair, "Sentiment Analysis: Towards a Tool for Analysing Real-Time Students Feedback," in *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, ISSN: 2375-0197, Nov. 2014, pp. 419–423. doi: 10.1109/ICTAI.2014.70.

[34] S. K. Khatri, H. Singhal, and P. Johri, "Sentiment analysis to predict Bombay stock exchange using artificial neural network," in *Infocom Technologies and Optimization Proceedings of 3rd International Conference on Reliability*, Oct. 2014, pp. 1–5. doi: 10.1109/ICRITO.2014.7014714.

[35] P. Yang and Y. Chen, "A survey on sentiment analysis by using machine learning methods," in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Dec. 2017, pp. 117–121. doi: 10.1109/ITNEC.2017.8284920.

[36] R. Mu, "A Survey of Recommender Systems Based on Deep Learning," *IEEE Access*, vol. 6, pp. 69 009–69 022, 2018, Conference Name: IEEE Access, issn: 2169-3536. doi: 10.1109/ACCESS.2018.2880197.

[37] Y. Afoudi, M. Lazaar, and M. Al Achhab, "Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network," en, *Simulation Modelling Practice and Theory*, vol. 113, p. 102 375, Dec. 2021, issn: 1569-190X. doi: 10.1016/j.simpat.2021.102375. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X21000836 (visited on 06/15/2023).

[38] P. Donepudi, "Machine Learning and Artificial Intelligence in Banking," *Engineering International*, vol. 5, pp. 83–86, Jan. 2017. doi: 10.18034/ei.v5i2.490.

[39] S. Kalyoncu, A. Jamil, J. Rasheed, M. Yesiltepe, and C. Djeddi, "Machine Learning Methods for Stock Market Analysis," Jun. 2020.

[40] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362,

no. 6419, pp. 1140–1144, Dec. 2018, Publisher: American Association for the Advancement of Science. doi: 10.1126/science.aar6404. [Online]. Available: https://www.science.org/doi/10.1126/science.aar6404 (visited on 06/17/2023).

[41] B. Bouzy and G. Chaslot, "Monte-Carlo Go Reinforcement Learning Experiments," in *2006 IEEE Symposium on Computational Intelligence and Games*, ISSN: 2325-4289, May 2006, pp. 187–194. doi: 10.1109/CIG.2006.311699.

[42] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, May 2019, pp. 1255–1260. doi: 10.1109/ICCS45141.2019.9065747.

[43] *What is the k-nearest neighbors algorithm? | IBM*, en-us. [Online]. Available: https://www.ibm.com/topics/knn (visited on 06/15/2023).

[44] *STAT 451 – Introduction to Machine Learning and Statistical Pattern Classification (Fall 2020) - Dr. Raschka*. [Online]. Available: https://pages.stat.wisc.edu/~sraschka/teaching/stat451-fs2020/ (visited on 06/15/2023).

[45] K. Hornik, D. Meyer, and A. Karatzoglou, "Support Vector Machines in R," *Journal of Statistical Software*, vol. 15, Apr. 2006. doi: 10.18637/jss.v015.i09.

[46] B. Scholkopf, "SVMs: A practical consequence of learning theory," *IEEE Intelligent Systems*, 13(4):36–40, 1998. [Online]. Available: https://www.semanticscholar.org/paper/SVMs%E2%80%94a-practical-consequence-of-learning-theory-Scholkopf/b2773c7f8ff48d33b88f8e4c33bedcbe18074c73 (visited on 06/15/2023).

[47] Z. Zhang, "A gentle introduction to artificial neural networks," eng, *Ann Transl Med*, vol. 4, no. 19, p. 370, Oct. 2016, issn: 2305-5839. doi: 10.21037/atm.2016.06.20.

[48] L. C. Jain and L. R. Medsker, *Recurrent Neural Networks: Design and Applications*, 1st. USA: CRC Press, Inc., 1999, isbn: 978-0-8493-7181-3.

[49] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," *ArXiv e-prints*, Nov. 2015.

[50] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," en, *Electron Markets*, vol. 31, no. 3, pp. 685–695, Sep. 2021, issn: 1422-8890. doi: 10.1007/s12525-021-00475-2. [Online]. Available: https://doi.org/10.1007/s12525-021-00475-2 (visited on 06/15/2023).

[51] Y. Hua, J. Guo, and H. Zhao, "Deep Belief Networks and deep learning," in *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, Jan. 2015, pp. 1–4. doi: 10.1109/ICAIOT.2015.7111524.

[52] G. Hinton, "Deep Belief Nets," en, in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2010, pp. 267–269, isbn: 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_208. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_208 (visited on 06/15/2023).

[53] K. Arulkumaran, M. Deisenroth, M. Brundage, and A. Bharath, "A Brief Survey of Deep Reinforcement Learning," *IEEE Signal Processing Magazine*, vol. 34, Aug. 2017. doi: 10.1109/MSP.2017.2743240.

[54] S. Mousavi, M. Schukat, and E. Howley, "Deep Reinforcement Learning: An Overview," Jun. 2018, pp. 426–440, isbn: 978-3-319-56990-1. doi: 10.1007/978-3-319-56991-8_32.

[55] P. P. Shinde and S. Shah, "A Review of Machine Learning and Deep Learning Applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Aug. 2018, pp. 1–6. doi: 10.1109/ICCUBEA.2018.8697857.

[56] R. Klette, *Concise Computer Vision: An Introduction into Theory and Algorithms* (Undergraduate Topics in Computer Science), en. London: Springer, 2014, isbn: 978-1-4471-6319-0. doi: 10.1007/978-1-4471-6320-6. [Online]. Available: https://link.springer.com/10.1007/978-1-4471-6320-6 (visited on 06/15/2023).

[57] S. Fudickar, A. Lindemann, and B. Schnor, "Threshold-based Fall Detection on Smart Phones," in *International Conference on Health Informatics (HEALTHINF) 2014*, Mar. 2014.

[58] M. Wolfe, J. Caguioa, A. Nguyen, and J. Cheun, "Fall Detection: Threshold Analysis of Wrist-Worn Motion Sensor Signals," *SMU Data Science Review*, vol. 3, no. 2, Sep. 2020. [Online]. Available: https://scholar.smu.edu/datasciencereview/vol3/iss2/10.

[59] M. Waheed, H. Afzal, and K. Mehmood, "NT-FDS—A Noise Tolerant Fall Detection System Using Deep Learning on Wearable Devices," en, *Sensors*, vol. 21, no. 6, p. 2006, Jan. 2021, issn: 1424-8220. doi: 10.3390/s21062006. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2006 (visited on 05/31/2023).

[60] F.-Y. Leu, C.-Y. Ko, Y.-C. Lin, H. Susanto, and H.-C. Yu, "Chapter 10 - Fall Detection and Motion Classification by Using Decision Tree on Mobile Phone," en, in *Smart Sensors Networks*, ser. Intelligent Data-Centric Systems, F. Xhafa, F.-Y. Leu, and L.-L. Hung, Eds., Academic Press, Jan. 2017, pp. 205–237, isbn: 978-0-12-809859-2. doi: 10.1016/B978-0-12-809859-2.00013-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128098592000139 (visited on 06/19/2023).

[61] S. Yu, Y. Chai, H. Chen, R. A. Brown, S. J. Sherman, and J. F. Nunamaker, "Fall Detection with Wearable Sensors: A Hierarchical Attention-based Convolutional Neural Network Approach," *Journal of Management Information Systems*, vol. 38, no. 4, pp. 1095–1121, Oct. 2021, issn: 0742-1222. doi: 10.1080/07421222.2021.1990617. [Online]. Available: https://doi.org/10.1080/07421222.2021.1990617 (visited on 05/31/2023).

[62] N. Shibuya, B. Nukala, A. Rodriguez, *et al.*, "A real-time fall detection system using a wearable gait analysis sensor and a Support Vector Machine (SVM) classifier," in *2015 Eighth International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, Jan. 2015, pp. 66–67. doi: 10.1109/ICMU.2015.7061032.

[63] S. M. Adnan, A. Irtaza, S. Aziz, M. O. Ullah, A. Javed, and M. T. Mahmood, "Fall detection through acoustic Local Ternary Patterns," en, *Applied Acoustics*, vol. 140, pp. 296–300, Nov. 2018, issn: 0003-682X. doi: 10.1016/j.apacoust.2018.06.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003682X18302834 (visited on 06/15/2023).

[64] D. Droghini, E. Principi, S. Squartini, P. Olivetti, and F. Piazza, "Human Fall Detection by Using an Innovative Floor Acoustic Sensor," en, in *Multidisciplinary Approaches to Neural Computing*, ser. Smart Innovation, Systems and Technologies, A. Esposito, M. Faudez-Zanuy, F. C. Morabito, and E. Pasero, Eds., Cham: Springer International Publishing, 2018, pp. 97–107, isbn: 978-3-319-56904-8. doi: 10.1007/978-3-319-56904-8_10. [Online]. Available: https://doi.org/10.1007/978-3-319-56904-8%5C_10 (visited on 05/31/2023).

[65] M. Alwan, P. Rajendran, S. Kell, *et al.*, "A Smart and Passive Floor-Vibration Based Fall Detector for Elderly," in *2006 2nd International Conference on Information & Communication Technologies*, vol. 1, Apr. 2006, pp. 1003–1007. doi: 10.1109/ICTTA.2006.1684511.

[66] F. Muheidat, L. Tawalbeh, and H. Tyrer, "Context-Aware, Accurate, and Real Time Fall Detection System for Elderly People," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, Jan. 2018, pp. 329–333. doi: 10.1109/ICSC.2018.00068.

[67] X. Fan, H. Zhang, C. Leung, and Z. Shen, "Robust unobtrusive fall detection using infrared array sensors," in *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Nov. 2017, pp. 194–199. doi: 10.1109/MFI.2017.8170428.

[68] S. Nishiguchi, M. Yamada, K. Uemura, *et al.*, "A novel infrared laser device that measures multilateral parameters of stepping performance for assessment of fall risk in elderly individuals [corrected]," eng, *Aging Clin Exp Res*, vol. 25, no. 3, pp. 311–316, Jun. 2013, issn: 1720-8319. doi: 10.1007/s40520-013-0042-9.

[69] Y. Tang, Z. Peng, L. Ran, and C. Li, "iPrevent: A novel wearable radio frequency range detector for fall prevention," in *2016 IEEE International Symposium on Radio-Frequency Integration Technology (RFIT)*, Aug. 2016, pp. 1–3. doi: 10.1109/RFIT.2016.7578162.

[70] M. G. Amin, Y. D. Zhang, F. Ahmad, and K. D. Ho, "Radar Signal Processing for Elderly Fall Detection: The future for in-home monitoring," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 71–80, Mar. 2016, issn: 1558-0792. doi: 10.1109/MSP.2015.2502784.

[71] H. Wang, D. Zhang, Y. Wang, J. Ma, Y. Wang, and S. Li, "RT-Fall: A Real-Time and Contactless Fall Detection System with Commodity WiFi Devices," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 511–526, Feb. 2017, issn: 1558-0660. doi: 10.1109/TMC.2016.2557795.

[72] K. De Miguel, A. Brunete, M. Hernando, and E. Gambao, "Home Camera-Based Fall Detection System for the Elderly," en, *Sensors*, vol. 17, no. 12, p. 2864, Dec. 2017, issn: 1424-8220. doi: 10.3390/s17122864. [Online]. Available: https://www.mdpi.com/1424-8220/17/12/2864 (visited on 05/31/2023).

[73] P. J. Grother, "NIST Special Database 19. NIST Handprinted Forms and Characters Database," en, *NIST*, Oct. 2008, Last Modified: 2008-10-16T10:10-04:00 Publisher: Patrick J. Grother. [Online]. Available: https://www.nist.gov/publications/nist-special-database-19-nist-handprinted-forms-and-characters-database (visited on 06/15/2023).

[74] *About NIST*, en, Last Modified: 2022-01-11T14:13-05:00, Jul. 2009. [Online]. Available: https://www.nist.gov/about-nist (visited on 06/15/2023).

[75] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, Nov. 2012, Conference Name: IEEE Signal Processing Magazine, issn: 1558-0792. doi: 10.1109/MSP.2012.2211477.

[76] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[77] *ImageNet Object Localization Challenge*, en. [Online]. Available: https://kaggle.com/competitions/imagenet-object-localization-challenge (visited on 06/15/2023).

[78] A. Torralba, R. Fergus, and W. T. Freeman, "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008,

Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, issn: 1939-3539. doi: 10.1109/TPAMI.2008.128.

[79] L. Smaira, J. Carreira, E. Noland, E. Clancy, A. Wu, and A. Zisserman, *A Short Note on the Kinetics-700-2020 Human Action Dataset*, en, Oct. 2020. doi: https://doi.org/10.48550/arXiv.2010.10864. [Online]. Available: https://arxiv.org/abs/2010.10864v1 (visited on 06/15/2023).

[80] M. Cordts, M. Omran, S. Ramos, *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," Jun. 2016. doi: 10.1109/CVPR.2016.350.

[81] K. Soomro, A. Zamir, and M. Shah, "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild," *CoRR*, Dec. 2012.

[82] L. Martínez-Villaseñor, H. Ponce, J. Brieva, E. Moya-Albor, J. Núñez-Martínez, and C. Peñafort-Asturiano, "UP-Fall Detection Dataset: A Multimodal Approach," en, *Sensors*, vol. 19, no. 9, p. 1988, Jan. 2019, Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, issn: 1424-8220. doi: 10.3390/s19091988. [Online]. Available: https://www.mdpi.com/1424-8220/19/9/1988 (visited on 06/15/2023).

[83] J.-L. Chua, Y. C. Chang, and W. K. Lim, "A simple vision-based fall detection technique for indoor video surveillance," en, *SIViP*, vol. 9, no. 3, pp. 623–633, Mar. 2015, issn: 1863-1711. doi: 10.1007/s11760-013-0493-7. [Online]. Available: https://doi.org/10.1007/s11760-013-0493-7 (visited on 06/15/2023).

[84] K. Adhikari, H. Bouchachia, and H. Nait-Charif, "Activity recognition for indoor fall detection using convolutional neural network," in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, May 2017, pp. 81–84. doi: 10.23919/MVA.2017.7986795.

[85] E. Auvinet, L. Reveret, A. St-Arnaud, J. Rousseau, and J. Meunier, "Fall detection using multiple cameras," eng, *Annu Int Conf IEEE Eng Med Biol Soc*, vol. 2008, pp. 2554–2557, 2008, issn: 2375-7477. doi: 10.1109/IEMBS.2008.4649721.

[86] F. Riquelme, C. Espinoza, T. Rodenas, J.-G. Minonzio, and C. Taramasco, "eHome-Seniors Dataset: An Infrared Thermal Sensor Dataset for Automatic Fall Detection Research," en, *Sensors*, vol. 19, no. 20, p. 4565, Jan. 2019, issn: 1424-8220. doi: 10.3390/s19204565. [Online]. Available: https://www.mdpi.com/1424-8220/19/20/4565 (visited on 05/31/2023).

[87] G. Baldewijns, G. Debard, G. Mertes, B. Vanrumste, and T. Croonenborghs, "Bridging the gap between real-life data and simulated data by providing a highly realistic fall dataset for evaluating camera-based fall detection algorithms," *Healthc Technol Lett*, vol. 3, no. 1, pp. 6–11, Mar. 2016, issn: 2053-3713. doi: 10.1049/htl.2015.0047. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4814805/ (visited on 05/31/2023).

[88] I. Charfi, J. Miteran, J. Dubois, M. Atri, and R. Tourki, "Optimized spatio-temporal descriptors for real-time fall detection: Comparison of support vector machine and Adaboost-based classification," *Journal of Electronic Imaging*, vol. 22, pp. 041 106–041 106, Oct. 2013. doi: 10.1117/1.JEI.22.4.041106.

[89] B. Kwolek and M. Kepski, "Human fall detection on embedded platform using depth maps and wireless accelerometer," en, *Computer Methods and Programs in Biomedicine*, vol. 117, no. 3, pp. 489–501, Dec. 2014, issn: 0169-2607. doi: 10.1016/j.cmpb.2014.09.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260714003447 (visited on 05/31/2023).

[90] Z. Zhang, C. Conly, and V. Athitsos, "Evaluating Depth-Based Computer Vision Methods for Fall Detection under Occlusions," en, in *Advances in Visual Computing*,

G. Bebis, R. Boyle, B. Parvin, *et al.*, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2014, pp. 196–207, isbn: 978-3-319-14364-4. doi: 10.1007/978-3-319-14364-4_19.

[91] S. Gasparrini, E. Cippitelli, E. Gambi, *et al.*, "Proposal and Experimental Evaluation of Fall Detection Solution Based on Wearable and Depth Data Fusion," en, in *ICT Innovations 2015*, S. Loshkovska and S. Koceski, Eds., ser. Advances in Intelligent Systems and Computing, Cham: Springer International Publishing, 2016, pp. 99–108, isbn: 978-3-319-25733-4. doi: 10.1007/978-3-319-25733-4_11.

[92] C. Khraief, F. Benzarti, and H. Amiri, "Elderly fall detection based on multi-stream deep convolutional networks," en, *Multimed Tools Appl*, vol. 79, no. 27, pp. 19 537–19 560, Jul. 2020, issn: 1573-7721. doi: 10.1007/s11042-020-08812-x. [Online]. Available: https://doi.org/10.1007/s11042-020-08812-x (visited on 05/31/2023).

[93] J. Rafferty, J. Medina-Quero, S. Quinn, *et al.*, "Thermal Vision Based Fall Detection via Logical and Data driven Processes," in *2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD)*, May 2019, pp. 35–40. doi: 10.1109/BCD.2019.8884820.

[94] M. Rahnemoonfar and H. Alkittawi, "Spatio-Temporal Convolutional Neural Network For Elderly Fall Detection In Depth Video Cameras," in *2018 IEEE International Conference on Big Data (Big Data)*, Dec. 2018, pp. 2868–2873. doi: 10.1109/BigData.2018.8622342.

[95] X. Ma, H. Wang, B. Xue, M. Zhou, B. Ji, and Y. Li, "Depth-Based Human Fall Detection via Shape Features and Improved Extreme Learning Machine," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 6, pp. 1915–1922, Nov. 2014, issn: 2168-2208. doi: 10.1109/JBHI.2014.2304357.

[96] *Azure Kinect DK – Develop AI Models | Microsoft Azure*, en-US. [Online]. Available: https://azure.microsoft.com/en-us/products/kinect-dk (visited on 06/15/2023).

[97] Y. Fan, M. D. Levine, G. Wen, and S. Qiu, "A deep neural network for real-time detection of falling humans in naturally occurring scenes," en, *Neurocomputing*, vol. 260, pp. 43–58, Oct. 2017, issn: 0925-2312. doi: 10.1016/j.neucom.2017.02.082. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217304666 (visited on 05/31/2023).

[98] Y. Chen, Y. Zhang, B. Xiao, and H. Li, "A framework for the elderly first aid system by integrating vision-based fall detection and BIM-based indoor rescue routing," en, *Advanced Engineering Informatics*, vol. 54, p. 101 766, Oct. 2022, issn: 1474-0346. doi: 10.1016/j.aei.2022.101766. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034622002245 (visited on 05/31/2023).

[99] K. Wang, G. Cao, D. Meng, W. Chen, and W. Cao, "Automatic fall detection of human in video using combination of features," in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Dec. 2016, pp. 1228–1233. doi: 10.1109/BIBM.2016.7822694.

[100] C. Ma, A. Shimada, H. Uchiyama, H. Nagahara, and R.-i. Taniguchi, "Fall detection using optical level anonymous image sensing system," en, *Optics & Laser Technology*, Special Issue: Optical Imaging for Extreme Environment, vol. 110, pp. 44–61, Feb. 2019, issn: 0030-3992. doi: 10.1016/j.optlastec.2018.07.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030399218310612 (visited on 06/15/2023).

[101] R. Espinosa, H. Ponce, S. Gutiérrez, L. Martínez-Villaseñor, J. Brieva, and E. Moya-Albor, "A vision-based approach for fall detection using multiple cameras and convolutional neural networks: A case study using the UP-Fall detection dataset," en, *Computers in Biology and Medicine*, vol. 115, p. 103 520, Dec. 2019, issn: 0010-4825. doi: 10.1016/j.compbiomed.2019.103520. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482519303816 (visited on 05/31/2023).

[102] M. M. Hasan, M. S. Islam, and S. Abdullah, "Robust Pose-Based Human Fall Detection Using Recurrent Neural Network," in *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*, Nov. 2019, pp. 48–51. doi: 10.1109/RAAICON48939.2019.23.

[103] S. Zou, W. Min, L. Liu, Q. Wang, and X. Zhou, "Movement Tube Detection Network Integrating 3D CNN and Object Detection Framework to Detect Fall," en, *Electronics*, vol. 10, no. 8, p. 898, Jan. 2021, issn: 2079-9292. doi: 10.3390/electronics10080898. [Online]. Available: https://www.mdpi.com/2079-9292/10/8/898 (visited on 05/31/2023).

[104] N. Lu, Y. Wu, L. Feng, and J. Song, "Deep Learning for Fall Detection: Three-Dimensional CNN Combined With LSTM on Video Kinematic Data," *IEEE Journal of Biomedical and Health Informatics*, vol. 23, no. 1, pp. 314–323, Jan. 2019, issn: 2168-2208. doi: 10.1109/JBHI.2018.2808281.

[105] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, Jun. 2014, pp. 1725–1732. doi: 10.1109/CVPR.2014.223.

[106] K. Adhikari, "Computer vision based posture estimation and fall detection.," en, doctoral, Bournemouth University, Jan. 2020. [Online]. Available: https://eprints.bournemouth.ac.uk/33227/ (visited on 05/31/2023).

[107] *General Data Protection Regulation (GDPR) – Official Legal Text*, en-US. [Online]. Available: https://gdpr-info.eu/ (visited on 06/15/2023).

[108] *Guidelines 3/2019 on processing of personal data through video devices | European Data Protection Board*. [Online]. Available: https://edpb.europa.eu/our-work-tools/our-documents/guidelines/guidelines-32019-processing-personal-data-through-video_en (visited on 06/15/2023).

[109] Mina, *MI Home Security Camera Basic 1080P User Manual*, en-us, Publication Title: Manuals+, Jul. 2021. [Online]. Available: https://manuals.plus/%5C_mi/home-security-camera-basic-1080p-manual (visited on 05/31/2023).

[110] J. Carreira and A. Zisserman, *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset*, Feb. 2018. doi: 10.48550/arXiv.1705.07750. [Online]. Available: http://arxiv.org/abs/1705.07750 (visited on 05/31/2023).

[111] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2556–2563. doi: 10.1109/ICCV.2011.6126543.

[112] *I3D (RGB + Flow) - Video Features Documentation*. [Online]. Available: https://iashin.ai/video%5C_features/models/i3d/ (visited on 05/31/2023).

[113] K. Hara, H. Kataoka, and Y. Satoh, "Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?" In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, ISSN: 2575-7075, Jun. 2018, pp. 6546–6555. doi: 10.1109/CVPR.2018.00685.

[114] M. Tan and Q. V. Le, *EfficientNetV2: Smaller Models and Faster Training*, Jun. 2021. doi: 10.48550/arXiv.2104.00298. [Online]. Available: http://arxiv.org/abs/2104.00298 (visited on 05/31/2023).

[115] K. Team, *Keras documentation: Keras Applications*, en. [Online]. Available: https://keras.io/api/applications/ (visited on 06/17/2023).

[116] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *MobileNetV2: Inverted Residuals and Linear Bottlenecks*, Mar. 2019. doi: 10.48550/arXiv.1801.04381. [Online]. Available: http://arxiv.org/abs/1801.04381 (visited on 05/31/2023).

[117] X. Wang, Z. Miao, R. Zhang, and S. Hao, "I3D-LSTM: A New Model for Human Action Recognition," en, *IOP Conf Ser Mater Sci Eng*, vol. 569, no. 3, p. 032 035, Jul. 2019, issn: 1757-899X. doi: 10.1088/1757-899X/569/3/032035. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/569/3/032035 (visited on 05/31/2023).

[118] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[119] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jun. 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.