UNIVERSITA' DEGLI STUDI DI PAVIA

FACOLTA' DI INGEGNERIA
DIPARTIMENTO DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

DOTTORATO DI RICERCA IN BIOINGEGNERIA E BIOINFORMATICA
XXXV CICLO - 2022

ARTIFICIAL INTELLIGENCE SOLUTIONS FOR POSTURE RECOGNITION SYSTEM IN AMBIENT ASSISTED LIVING

PhD Thesis by

BRUNA MARIA VITTORIA GUERRA

Advisor:
Prof. Micaela Schmid
Prof. Stefano Ramat

PhD Program Chair:
Prof. Silvana Quaglini

Sintesi

L'attività di ricerca descritta in questa tesi di dottorato è stata condotta presso il Laboratorio di Bioingegneria dell'Università di Pavia. Il tema principale della tesi riguarda l'implementazione di soluzioni di Intelligenza Artificiale (IA) per l'analisi di dati di un sistema di Ambient Assisted Living (AAL). Questo sistema è in fase di realizzazione e si propone di "assistere nel vivere" persone fragili, monitorandole all'interno del loro ambiente domestico.

La ricerca di forme adeguate e sostenibili di promozione del benessere e di salvaguardia della salute delle persone fragili, quali anziani e/o persone disabili, è oggi un argomento molto dibattuto. La popolazione di età pari o superiore ai 65 anni è in continuo aumento e previsionalmente negli anni a venire il trend relativo all'aumento della popolazione over 65 nei paesi della UE-28 continuerà a crescere tanto da raggiungere, secondo le proiezioni demografiche di Eurostat fatte sul periodo 2018 – 2100, un picco di 525 milioni intorno al 2040. La percentuale delle persone di età pari o superiore agli 80 anni passerà tra il 2018 e il 2100 dal 5,6 % al 14,6 %. Ne segue che i cittadini europei presentano un'aspettativa di vita sempre più alta e quindi, in futuro, si avrà un numero sempre maggiore di persone autosufficienti o parzialmente autosufficienti a cui dover assicurare un invecchiamento attivo, ed un numero di persone totalmente dipendenti a cui dover assicurare un'assistenza h24. Parallelamente, la percezione sociale della persona con disabilità ha avuto una forte evoluzione con conseguente adeguamento legislativo a garanzia dei diritti portando ad una sempre maggior inclusione della persona diversamente abile in tutti gli ambiti di vita.

Negli ultimi anni numerose sono state le proposte di nuove soluzioni assistenziali che assicurino una vita dignitosa e sicura all'interno delle proprie mura domestiche, ritardando il più possibile il ricovero in strutture specializzate. In questo quadro le Tecnologie dell'Informazione e della Comunicazione (Information and Communication Technologies , ICT), l'Internet delle Cose (Internet of Things, IoT) e l'Intelligenza Artificiale (IA) sono state la chiave di volta nella sviluppo di sistemi intelligenti che permettono alla persona fragile di vivere nelle propria abitazione in sicurezza, offrendo la possibilità di mantenere corretti stili di vita, migliorare la socialità e preservare l'autonomia nello svolgimento delle diverse attività quotidiane. Numerosi sono gli IoT Lab, Future Technology Lab, Living lab il cui interesse è rivolto alla progettazione di sistemi assistenziali di complessità differente. Si passa dal semplice sensore per il rilevamento di

segnali vitali e la caduta, a veri e propri sistemi di tele-assistenza domiciliare costituiti da reti di sensori utilizzate per il monitoraggio della persona, utili per individuare situazioni pericolose e stili di vita, a reti ancor più complesse, in cui i sensori sono affiancati da attuatori a supporto dell'autonomia della persona. Queste due ultime soluzioni rendono l'ambiente domestico intelligente (Smart Home) e necessitano di sensori smartobject e i cui dati vengono memorizzati ed elaborati per fornire informazioni sulle attività giornaliere dell'ospite, anomalie e variazioni della routine quotidiana utili ad individuare variazioni sensibili nei comportamenti. I dati, se elaborati per fornire prospetti a medio e lungo termine, possono essere interpretati in un'ottica di prevenzione. Alternativamente, se processati on-line, possono essere impiegati per un tempestivo intervento in risposta ad un segnale di allarme. Le diverse soluzioni proposte si differenziano per l'HW (sensori indossabili, sistemi, ambientali (sistemi *Sensor-based)* e telecamere (sistemi *Vision-Based*), il SW di analisi dei dati basato principalmente su algoritmi di natura statistica o di Machine e Deep Learning (riconoscimento di azioni attraverso le tecniche proprie della Human Activity Recognition (HAR)), ed infine per i protocolli e le architetture di comunicazione. A soluzioni Wireless, che eliminano costosi e complicati cablaggi e permettono di distribuire nell'ambiente sensori a corto raggio, si affiancano architetture più complesse, a più livelli, in cui si prevede una preelaborazione locale dei dati in mini-data center ed una successiva elaborazione ed archiviazioni su Cloud.

L'obiettivo di questa tesi riguarda, lo sviluppo di soluzioni di IA, atte ad identificare specifiche posture assunte da un soggetto fragile all'interno di un ambiente domestico, in situazioni di vita quotidiana. La postura identificata costituisce uno degli ingressi ad un sistema più complesso che riceve, in aggiunta, i dati grezzi o elaborati provenienti da una rete di sensori ambientali dislocati nella stanza di interesse. Il sistema ha il compito di integrare ed analizzare i diversi ingressi al fine di distinguere uno scenario di vita quotidiana da una potenziale situazione di pericolo (ad esempio, persona sdraiata a letto→probabile situazione di vita quotidiana; persona sdraiata a terra→potenziale situazione di pericolo). I dati di ingresso al modello di IA sono stati acquisiti da sensori di tipo *Vision-based*, ovvero quattro dispositivi Kinect V2, disposti nell'ambiente, secondo una geometria che consenta di monitorare un'area sufficiente ampia del locale. La preelaborazione dei dati così, come la costruzione dei database di analisi, sono stati implementati nell'ambiente Matlab, mentre le architetture proposte, così come la loro validazione, sono state realizzate in linguaggio Python.

La tesi è stata così organizzata: nel **Capitolo 1** è fornita una breve panoramica sull'AAL basata su un'introduzione generale, una descrizione delle finalità e del suo sviluppo negli ultimi anni. Il capitolo si chiude con un esempio di possibile Smart Home; nel **Capitolo 2**, si è descritto in generale il concetto di HAR, illustrando il processo che porta dal dato

acquisito alla identificazione di un'azione umana. Sono state illustrate le diverse tipologie di sensori, le tecniche di analisi dei dati acquisiti con particolare attenzione alla definizione delle feature di interesse e la costruzione del dataset. Infine, particolare attenzione è stata posta alla descrizione dei diversi algoritmi di IA solitamente proposti nelle soluzioni di HAR; nel **Capitolo 3** viene presentato il lavoro di tesi riguardante la parte di set-up sperimentale in termini di strumentazione utilizzata, protocollo di acquisizione delle prove sperimentali, preelaborazione dei dati e costruzione del dataset; nel **Capitolo 4** viene presentato un excursus dei diversi algoritmi e le diverse architetture di rete proposte, riportando per ciascuna i risultati ottenuti ed una breve discussione volta ad analizzare le prestazioni del classificatore, le criticità emerse e le possibili azioni da intraprendere. Le architetture proposte passano da una prima rete neurale di tipo Multi-Layer Perceptron (MLP) accompagnata successivamente da un algoritmo di preprocessing dei dati acquisiti, a modelli di rei neurali ricorrenti quali Long-Short Term Memory (LSTM) e Gated Recurrent Unit (GRU).

L'elaborato termina con le conclusioni sull'intera attività di tesi e i possibili sviluppi futuri.

Abstract

The research activity described in this thesis was conducted at the Bioengineering Laboratory of the University of Pavia. The main theme of the thesis concerns the implementation of Artificial Intelligence (AI) solutions for data analysis of an Ambient Assisted Living (AAL) system. This system is being implemented and aims to "assist in living" frail people by monitoring them within their home environment.

The search for appropriate and sustainable forms of promoting the well-being and safeguarding the health of frail persons, such as the elderly and/or persons with disabilities, is a much-debated topic today. The population aged 65 years and older is continuously increasing and projected in the years to come, the trend regarding the 'increase of the population over 65 in the EU-28 countries will continue to grow so much that, according to Eurostat's demographic projections made on the period 2018 - 2100, it will reach a peak of 525 million around 2040. The proportion of people aged 80 and older will increase between 2018 and 2100 from 5.6% to 14.6%. It follows that European citizens have an increasing life expectancy, and therefore, in the future, there will be an increasing number of self-sufficient or partially self-sufficient people to whom we will have to provide active aging, and a number of totally dependent people to whom we will have to provide 24-hour care. At the same time, the social perception of the person with disabilities has undergone a strong evolution with consequent legislative adjustment to guarantee rights leading to an increasing inclusion of the disabled person in all spheres of life.

In recent years there have been numerous proposals for new care solutions that ensure a dignified and safe life within one's own home, delaying hospitalization in specialized facilities as much as possible. Within this framework, Information and Communication Technologies (ICT), the Internet of Things (IoT) and Artificial Intelligence (AI) have been key in the development of intelligent systems that allow the frail persons to live in their homes safely, offering the possibility of maintaining correct lifestyles, improving sociability and preserving autonomy in the performance of various daily activities. There are numerous IoT Labs, Future Technology Labs, Living labs whose focus is on designing assistive systems of different complexity. They range from the simple sensor for vital signs and fall detection, to true home tele-assistance systems consisting of sensor networks used to monitor the person, useful for detecting dangerous situations and lifestyles, to even more complex networks, in which sensors are joined by actuators to support the person's autonomy. The latter two solutions make the home environment smart (Smart Home) and require smart object sensors whose data are stored and processed to provide information on the host's

daily activities, anomalies and variations in daily routines useful for detecting sensitive changes in behavior. The data, if processed to provide medium- and long-term prospects, can be interpreted from a prevention perspective. Alternatively, if processed online, they can be used for early intervention in response to an alarm signal. The different solutions proposed differ in HW (wearable sensors, environmental systems, (Sensor-based systems) and cameras (Vision-Based systems)), the data analysis SW based mainly on statistical or Machine and Deep Learning algorithms (action recognition through the techniques inherent to Human Activity Recognition (HAR)), and finally in communication protocols and architectures. Wireless solutions, which eliminate costly and complicated cabling and allow for the deployment of short-range sensors in the environment, are flanked by more complex, multi-layered architectures in which there is local pre-processing of data in mini-data centers and subsequent processing and storage on the Cloud.

The objective of this thesis concerns the development of AI solutions, designed to identify specific postures assumed by a frail subject within a home environment, in everyday life situations. The identified posture constitutes one of the inputs to a more complex system that receives, in addition, raw or processed data from a network of environmental sensors located in the room of interest. The system is tasked with integrating and analyzing the different inputs in order to distinguish an everyday life scenario from a potentially dangerous situation (e.g., person lying in bed → probable everyday life situation; person lying on the floor → potentially dangerous situation). The input data to the AI model were acquired from Vision-based sensors, i.e., four Kinect V2 devices, arranged in the environment according to a geometry allowing to monitor a sufficiently large area of the room. The pre-processing of the data, as well as the construction of the analysis databases, were implemented in the Matlab environment, while the proposed architectures, as well as their validation, were carried out in the Phyton language.

The thesis was organized as follows: in **Chapter 1**, a brief overview of AAL is provided based on a general introduction, a description of its purpose and its development in recent years. The chapter closes with an example of a possible Smart Home; in **Chapter 2**, a general description of HAR was given, illustrating the process leading from acquired data to the identification of a human action. The different types of sensors, techniques for analyzing the acquired data with a focus on defining features of interest, and dataset construction were explained. Finally, special attention has been paid to the description of the different AI algorithms usually proposed in HAR solutions; **Chapter 3** presents the thesis work regarding the experimental set-up in terms of instrumentation used, acquisition protocol for experimental testing, data pre-processing and dataset construction; **Chapter 4** presents an excursus of the different algorithms and the different network architectures proposed, reporting for each the results obtained and a brief discussion aimed at analyzing the performance of the classifier, the critical

issues that emerged and the possible actions to be taken. The proposed architectures go from an initial Multi-Layer Perceptron (MLP) type neural network, later accompanied by a pre-processing algorithm of the acquired data, to recurrent neural network models such as Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU).

The thesis ends with conclusions about the entire thesis activity and possible future developments.

Contents

Chapter 1

A gentle introduction to Ambient Assisted Living

The term Ambient Assistive Living (AAL) generally refers to the use of information and communication technologies (ICT), stand-alone assistive devices and smart home technologies, in a person's daily living and working environment to enable individuals to stay active longer, remain socially connected, and live independently [1]. Moreover, AAL system encourages healthy lifestyle, and supports disease prevention strategies based on personalized risk assessment and continuous monitoring.[2]–[4]. AAL systems are primarily targeted to frail people, i.e., elderly and/or disabled people, trying to maintain continuous support and to prolong their independent living actively and healthily [5].

Aleksic, et al. proposed a subdivision in four different generations of AAL systems based on the differences between the technologies involved, namely [4], [6] (Figure 1.1):

- *First Generation of AAL Systems*. These systems mainly consist of alert and alarm systems based on pendant or button alarm devices, worn by the subject, or installed in the environment. If a dangerous situation occurs, the monitored individual presses the button or pendant to send an alarm to a call center or a caregiver. Some examples of these solutions are the SafeLife Beghelli [7] and the LifeAlert [8]. These devices, although offering several benefits, had specific weaknesses, especially when the person is incapacitated either physically or mentally or he/she may not have the capacity to trigger the alarm. Moreover, very often the individual forgets to wear and recharge the device.
- *Second Generation of AAL Systems*. These systems are characterized by a more technological device (worn by the subjects or installed in indoor places) as they integrate sensors able to detect dangerous conditions and react accordingly without relying on the user to trigger the alarm. An example of these solutions, particularly useful for older adults with mild cognitive impairment, was the device able to detect a gas leak and send an automatic alarm by contacting the appropriate authorities [9]. Despite the potential benefits, a weakness

associated with this generation is the fact that some users feel it is intrusive.

- *Third Generation of AAL Systems*. These systems have been extended by the improvements of ICT solutions moving towards a more comprehensive concept of AAL. Specifically, the sensors are designed to detect problems and to prevent the worst scenarios, the actuator to provide the assisted person with support, and smart interfaces provide information, support, and encouragement. The idea is to build non-intrusive home arrangements, composed by several sensors, actuators and computing system, not only to monitor the home environment but also to control vital signs, changes in mobility and activity patterns of the frailty person and to support the execution of the daily living activities [10].
- *Fourth Generation of AAL Systems*. This generation is characterized by the use of Artificial Intelligence (AI) algorithms for the analysis of AAL solutions data. These systems are designed to be intelligent, able to learn and adapt to the requirements of the assisted people, and match with their specific needs [4], [11]–[16].

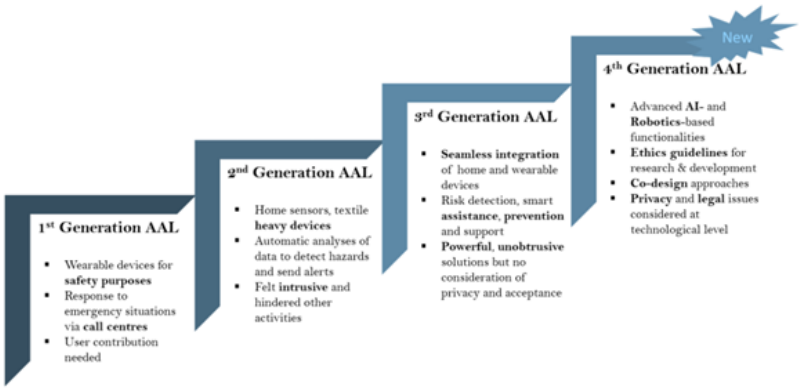

Figure 1.1: The four generations of AAL systems [4].

The subject monitoring could be realized in *outdoor* and/or *indoor* environments. In *outdoor* environments, the elderly may be exposed to various risks, such as falls or excessive heat or cold. Also, in the case of people with early symptoms of dementia, wandering and confusion or getting lost are common risks. In these contexts AAL systems aim at providing support to frail people in various aspects, such as in checking the routes, recognizing anomalous behaviors, evaluating motion activities, and so on [17], [18]. On the contrary, in *indoor* scenario, frail people are exposed to a number of risks that are closely related to the place where they live. In particular, the *indoor* environment has been found to be a factor contributing to most falls [19]. Uneven or slippery floor surfaces (including the presence of rugs and mats), tripping obstacles, inadequate lighting, poorly designed or maintained stairs without handrails and inappropriate furniture are cited

as increasing the risk of falling, tripping or slipping for frail people. Other hazards relate to the absence of safety or preventative devices such as night lights and grab rails [18], [19].

Generally, *indoor* environments can be further differentiated in homes and in retirement residences. In the private home the individuals live alone or with another inhabitant (*caregiver*) and the AAL systems mainly prevent domestic accidents and deliver health care services. Whereas, in the retirement residences, several people live together and move to common spaces, the AAL systems are basically designed to carry out group activities and controlled physical activities [5], [20].

Depending on the application domains, different technologies can be used starting from simple Internet of Things (IoT) devices (especially for *outdoor* environment) [15], [21], [22], to more complex sensor networks composed by environmental sensors, smart devices and cameras that are the core of the modern smart home. A smart home is a home equipped with many sensors and actuators, connected in a network and remotely controlled, capable of detecting the opening of the doors, the brightness of the rooms, the temperature, the humidity, the CO_2 level etc. Moreover, the sensors are installed to monitor subjects' daily life activities in order to guarantee security and safety. The sensors, actuators, smart objects or devices can be installed at different locations of the environment and interconnected via communication protocols. Each system is deployed using a communication technology such as ZigBee, Bluetooth, ZWave, USB, and Ethernet, among others [23]–[25]. The selection of sensors depends on several factors: the aim of the AAL system, the cost of the sensors, their intrusiveness, their acceptability from users, and the privacy issues.

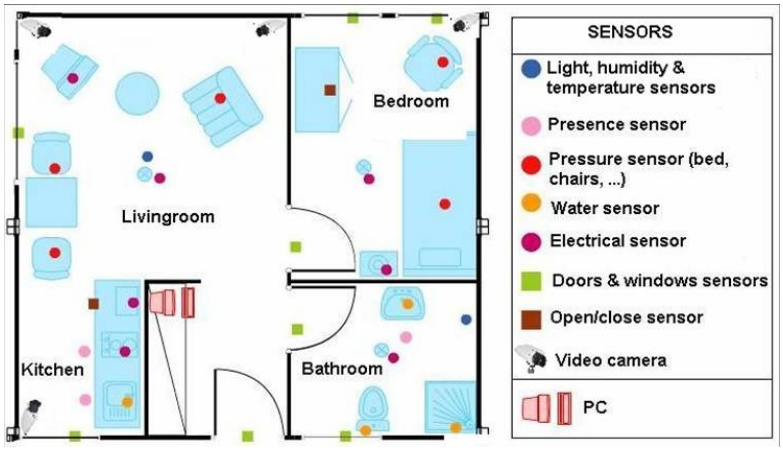

Figure 1.2: sample schematic setup of a AAL smart home [26].

Figure 1.2 shows a schematic example of an AAL smart home for frail people. It is possible to distinguish a set of sensors involved to guarantee comfort and the energy efficiency of the home as for example the light, the humidity, the temperature sensors (blue dots, Figure 1.2), the water sensors

(orange dots, Figure 1.2) used to detect hot/cold water consumption, power sensors (violet dots, Figure 1.2) and the contact sensors (brown and green dots, Figure 1.2). Beside these, a set of sensors are placed to monitor the inhabitant and to guarantee security and safety. For example, the presence sensors (pink dots, Figure 1.2), together with the pressure sensors (red dots, Figure 1.2) and the video-cameras are installed to investigate the resident habits (i.e., to investigate in which room the subject spends most hours of the day or if he/she leads a sedentary lifestyle spending most of the time lying in bed or sitting in an armchair). The presence sensors and the video-cameras are also used to have the subject's position in the home in case of a fast-medical rescue and to detect the presence of an intruder.

Usually, all the raw or pre-processed data are sent to a local or remote collection center able to integrate and to analyze them implying robust and intelligent algorithms [27]. Moreover, in the smart homes in which automatic dangerous situation detection or voluntary request of help trigger an alarm toward third parties, a real and robust analysis of data are mandatory. Therefore, according to the task of the AAL system and on the type of the technologies employed, two different data analysis methodologies are usually used. The first one is based on threshold analysis method. This simple technique can be enough to trigger alerts when dangerous events are detected [5], [28]–[30]. The second one, more recent, is based on Artificial Intelligence (AI) solutions (i.e., *machine* and *deep learning* algorithms) for Human Activity Recognition (HAR). The different steps at the basis of the HAR process will be detailed in the next Chapter (Chapter 2) [31].

A last consideration as regard of the AAL systems must be made; these cannot be closed, as the needs and habits of people change over time as well as the parameters to be observed. The methodologies for data analysis must consider the possibility of differently weighting or customizing some parameters rather than others, dynamically [5].

Chapter 2

Human Activity Recognition approaches in Ambient Assisted Living

In recent years *Human Activity Recognition* (HAR) has become one of the trendiest research topics. HAR techniques aim at the automatic identification of people's everyday activities, in a given context, through the combination of technologies and the implementation of *Artificial Intelligence* (AI) solutions [32]. In the last years HAR has become widely used in several relevant and heterogeneous application fields, from the most commercial to the most assistive ones, especially in the AAL domain [33]. Furthermore HAR, in AAL domain, can provide an array of solutions for improving the quality of individuals' life, allowing elderly and/or frail people to live healthier and independently for longer, helping individuals with disabilities and supporting caregivers and medical staff.

In order to implement a HAR system, as a first step, it is necessary to identify the type of activities to be identified. Human activities are grouped into four categories depending on the engaged body parts [34], [35]:

- *gestures* include primitive actions performed by a person's body part, i.e., hand gestures like "okay gesture" or "thumbs up";
- a*ctions* are related to a set of basic movements carried out by a single person, such as walking, standing, sitting, running etc.;
- *interactions* are a type of activities performed by two people and it could include also the relation between a person and objects (i.e., as playing guitar or hugging each other);
- *group of activities* are the most complex, because are mostly composed by a combination of *gestures*, *actions* and *interactions*, like group meeting or group walking.

Independently of the type of human activity to classify, the HAR process follows a typical schema characterized by a series of phases summarized in each block of Figure 2.1 and briefly described in the next paragraphs [35]. The blocks are:

1. *data acquisition*: in this phase the acquisition setup is defined taking into account the environment constraints, the human activity to be identified (i.e., simple activity or a set of more complex activities carried out at home or outside) and the final aims of the HAR process;
2. *data pre-processing*: in this phase the pre-processing of data is carried out in order to make the data useful for the classification models. This phase concerns data noise and redundancy reduction, data normalization and segmentation of data;
3. *feature extraction:* in this phase the attributes, able to characterize the human activity to be classified, are defined and computed;
4. *feature selection*: in this phase the most informative attributes are selected through automatic feature selection techniques;
5. *dataset construction:* in this phase the collected data are divided into training, validation and test data, in order to train, validated and test the HAR algorithm;
6. *pattern recognition*: in this phase the algorithms to infer activities are designed and implemented. This is accomplished through the employment of AI techniques, i.e., *machine learning* and *deep learning* algorithms.

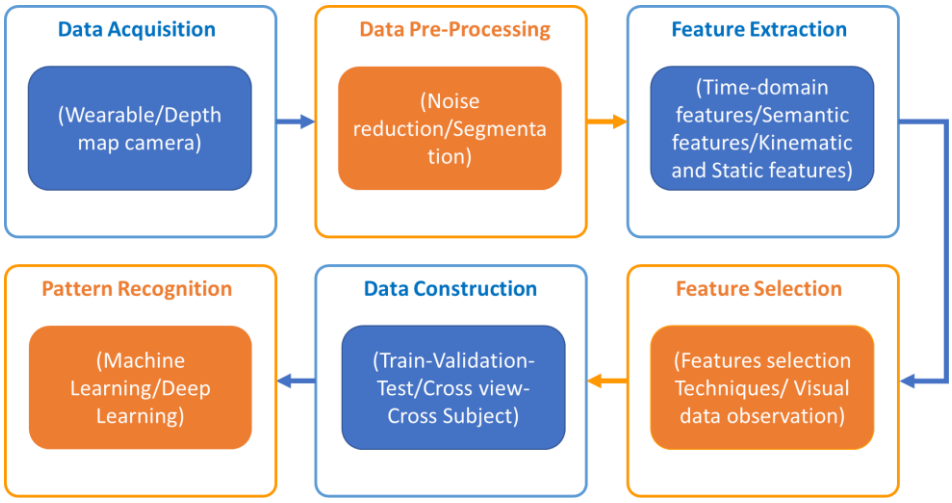

Figure 2.1: block diagram of a HAR general process.

2.1. Data Acquisition

Activity recognition can be defined as the ability to recognize/detect current activity depending on the information received from a single or a network of several sensors. With the advancements in technology and the reduction of device costs, the monitoring of daily activities has become very popular and practical. Nowadays, people may be monitored during their daily life activities, such as cooking, eating, sleeping, or watching TV. To acquire the data useful for human activities recognition, different approaches can be used, depending on the type of information required, the environment and the subject constraints, and where the actions are carried out (*indoor* or

outdoor). Accordingly, HAR techniques can be broadly classified in two different categories: *Vision-based* systems and *Sensor-based* systems. The first solution uses visual sensing devices, e.g., camera-based systems, to monitor a person's behavior and the environment, the second one, instead, employs sensor like IoT or smart devices for activity monitoring [36] (Figure 2.2).

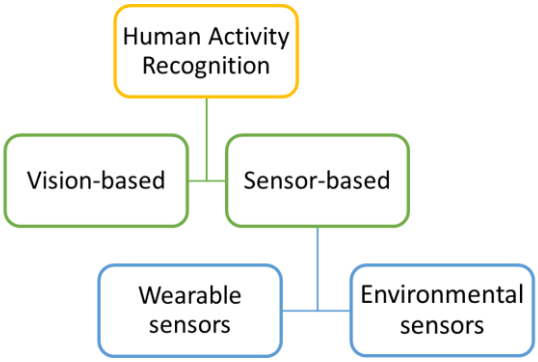

Figure 2.2: classification of Human Activity Recognition techniques.

Despite the focus of this thesis is on HAR techniques with *Vision-based* solutions, to provide the reader with a more comprehensive view on the HAR topic, a short description of the *Sensor-based* modalities is also given.

The *Sensor-based* HAR approach has been applied to various real-world applications, especially smart home, healthcare and AAL solutions [35]. In general, the *Sensor-based* HAR approaches are split into two categories according to the type and the installation position (worn by the subject or installed in the environment or objects) of the sensors, namely [35], [36]: wearable, and environmental sensors. Wearable sensor systems are device with HW (hardware) and SW (software) embedded, that can be incorporated into clothing or worn by the user's body like accessories [32]. HW usually involves wearable inertial sensors providing data from a triaxial accelerometer, gyroscope and magnetometers which can be conveniently worn by users or integrated into portable devices, such as *smartphones*, *smartwatches*, *smart bands*, glasses, or helmets. Mohd Noor et el. developed an algorithm, employing data collected from a waist-mounted *smartphone* (Samsung Galaxy S II) to identify three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs) [37]. [. In the same way, Roberge et al. collected inertial data from a wristband equipped with triaxial accelerometer and gyroscope for hand gesture recognition (drink, cut, mix, etc.…). The aim of this HAR system was to monitor the routines of an inhabitant of a smart home in order to recognize the on-going activities and to provide support when required [38]. Alternatively, to the inertial sensors some passive wearable devices based on Radio Frequency Identification (RFID) technology receivers are

also used [35], [39]. In this case the subject wears the sensor (markers), called tag, and an electromagnetic field emitter is installed in the environment. The subject's movements cause changes in the electromagnetic field, that are detected by a receiver and analyzed for implementing emergency situations identification systems [40]–[43]. Paolini et al. suggest a portable RFID reader designed to simultaneously perform 3-D tracking of multiple tagged entities (worn by the subjects) in harsh electromagnetic indoor environments. This customized RFID system interacts with the wearable tags, worn by the subjects, and with a monitoring platform to locate in real time the tagged people in their daily life environments, prevent their entrance in environment zones considered unsafe, control their position, identify their activities and remotely detect their potential falls [44].

Wearable sensors have numerous advantages, including their small size, the low energy demand necessary for their operation, the direct acquisition of information on the subject's activity and the full respect of subject's *privacy*. At the same time, they have also some drawbacks. For example, they need to be worn by the subjects and to operate for long time periods. This issue could be a significant problem for the monitored subject and for the battery life of the devices. Also, to fully capture the 3D motion associated with a human action, a single sensor may not be adequate. It may be necessary to utilize multiple sensors, thus increasing the intrusiveness of the devices worn by the subject [45]–[47].

Environmental sensors consist of ambient sensors installed in the subject's home, which passively monitors the occupants without the need for the user to manually operate on the installed devices (i.e., pressure, microphone, RFID and presence). This approach is more practical since it does not require the user to carry any device while performing an activity. Sensors can be installed directly in the environments or within everyday objects or furniture like doors, beds, chairs, washstand, toilet, and cupboards (object sensor). While wearable sensors measure human activities directly, object sensors detect specific objects movement to infer human activity [48]–[50]. For instance, Chaccour et al. realized a smart carpet based on piezoresistive pressure sensors to detect falls of the inhabitant [51], [52]. Furthermore, power meters can be used to monitor appliance usage, such as TV set or lamps [53], whereas smart pill box devices can be very useful for checking medication intake [54]. For example, Roland et al. installed an accelerometer attached to a smart drinking cup to efficiently identify the user's drinking movement [55]. At the same time, Bassoli et al. installed sensors directly on the furniture of the subject's house for the HAR. For instance, pressure pads are exploited to monitor bed (or chair) occupancy, sensors, installed inside the fridge, are used to get indirect information about feeding habits, etc. [48].

The main advantages of environmental sensors are their being privacy-preserving and their unobtrusiveness since the person does not have to wear them. However, there are some challenges. The main problem regarding these sensors is infrastructure dependency and their limitation in the identification of few movements of a specific activity especially as regards the object sensors (i.e., smart cup recognized only drinking action or sensors

tagged inside cushion or bed that identified only a set of sleeping posture). Moreover, environmental sensors, are utilized less often than wearable sensors due to high costs and setup challenges and also, like wearable approach, this solution may also not be feasible all the time because it bounds the users to use tagged objects or be within the ambient in which the sensors are installed [35].

In the last few decades, *Vision-based* HAR has become a trending topic thanks to the possibilities of application in various real-world scenarios.

In *Vision-based* HAR systems the data are acquired by one or multiple cameras, located in the home monitored space, to capture the person's daily life activities. Moreover, these devices allow a continuous monitoring of the person without any involvement of the user. Figure 2.3 shows an example of *Vision-based* HAR application: Domingo et. al propose a non-intrusive system based on RGB-D cameras (Kinect V2) installed in different area of the ambient, in order to monitor the inhabitant. In general, regardless of whether they are installed *indoors* or *outdoors*, the cameras can provide more comprehensive environmental information than other sensors. Moreover, low-cost cameras, can be easily installed for monitoring the daily activity of people in order to keep an active and healthy life under observation and to notify caregivers in the case of calls for help [31], [35], [56]–[60]. The most widely used *Vision-based* technologies are: RGB camera and Depth camera [61].

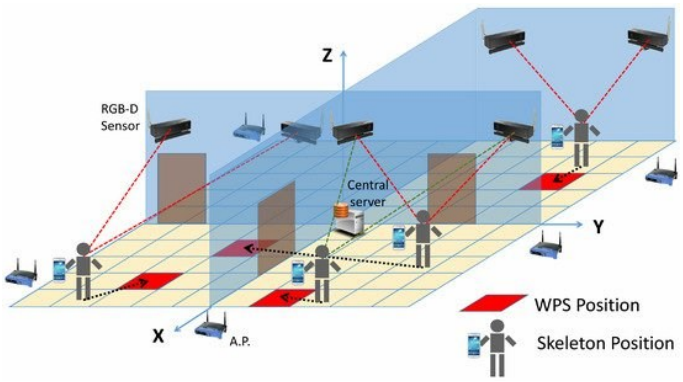

Figure 2.3: an example of *Vision-based* HAR approach for AAL scenario [62].

An RGB image contains red, green, and blue bands in the visible spectrum, which can be recorded using cameras equipped with a regular complementary metal-oxide-semiconductor (CMOS) sensor (Figure 2.4). The output matrices contain sensors encoding the RGB signals received when the shutter is open for a fraction of a second. The product of such a process produces images with a value representing the intensity of the light received in the picture element (pixel) at each coordinate of the sensor matrix [4]. RGB cameras are highly available and affordable. These devices can provide information about the characteristic of the shape, color and texture

of the scene. For instance, Zerrouki et al., implemented an AI algorithm for HAR, based on variation in body shape on RGB videos. For each frame, they calculated area ratios of the body shape segmentation for the identification of six different human activities (walking, standing, lying etc….)[63]. The drawbacks of the RGB devices are the limited area of the camera sight, the not preserving individual's privacy, the complex calibration procedure and high sensibility to the variations of the environmental conditions that can occur during data acquisitions, depending on e.g., lighting conditions, type of illumination and cluttered background [35], [39], [64], [65]

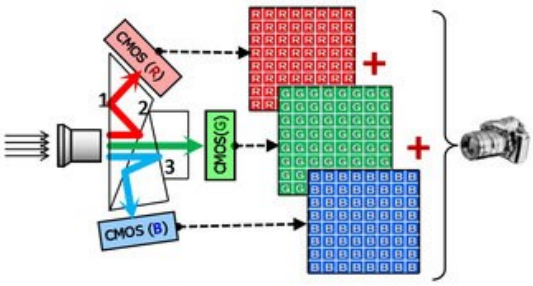

Figure 2.4: the imaging principle of traditional of 3×CMOS RGB camera

Differently from RGB image, a depth image contains the distance (depth) information from the sensor to an element in the scene, namely each acquired pixel value represents the distance from the camera. Depth camera components include optics, depth sensors, such as IR projectors IR sensors, phase detectors and imaging pipeline.

Generally, the depth information extraction from depth camera can be performed with different acquisition techniques: Structured Light, Time of Flight (**TOF**), Light Detection and Ranging (**LIDAR**) and Stereo Camera Sensing (**SCS**) [64], [66]–[68]. Structured Light technique rely on a projection of IR light in which the depth information is calculated using Triangulation method. As is possible to see in Figure 2.5, this method finds the position of the projected point in the image plane starting from the distance between the camera (CMOS) and the IR projector, the position in space of the IR projector, the internal calibration camera parameters. All these data concur to define the triangle which the height represents the distance (depth) between the object and camera [66]. Most Structured Light sensors do not work under direct sunlight since they rely on light projection in a scene. Therefore, they are usually suitable for indoor scene applications [64], [66].

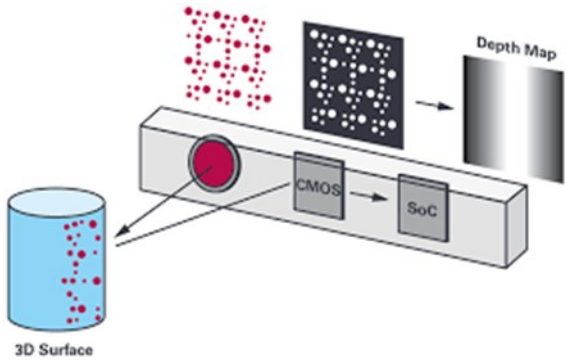

Figure 2.5: example of how a Depth map image is obtained though Structured Light technique.

TOF technique relies on the time that an IR light wave takes to hit an object's surface and return to the sensor. A simple version of a TOF sensor uses a single light pulse. This latter is reflected by the scene objects in the field of view. The incoming light pulse is delayed, depending on the distance to the object, and the delay is measured by a high-speed electric device. There are multiple strategies for capturing the TOF of light. The most straightforward strategy is using Pulse Modulation, where a very fast pulse of light is emitted and then received by the sensor. Continuous Modulation is another technique, where the light is modulated by its intensity, and the distance is measured by calculating the shift in phase of the original emitted light and the received light. This technique is employed by Kinect V2 sensor for depth image reconstruction. This device, being the system used in the experimental setup of this thesis, will be discussed in detail in a successive paragraph (Chapter 2, paragraph 2.2.1). Similar to Structured Light sensors, TOF sensors are generally not recommended under strong sunlight conditions. TOF sensors are more commonly applied to indoor scenes [67], [69], [70].

LIDAR technique uses the same techniques as TOF. An emitted IR light is received by a sensor, but they rely on focused laser beams, which allow them to collect distance measurements as far as a few kilometers. LIDAR sensors emit IR light; therefore, they work in difficult lighting conditions, such as dark environments. They are suitable for indoor and outdoor application scenes, but the available models are usually limited to specific applications, such as aerial measurements, outdoor/driving applications, and small indoor spaces depth estimation [64].

SCS technique is based on the analysis of two or more image coming from two or more cameras. In the case of two cameras (Figure 2.6) the triangulation is made on the position of the object projection on the plane of the camera A and B, respectively, the distance between the two cameras together with the camera calibration data. A limitation of this technique occurs when the point of interest has no texture. Therefore, it is difficult to determine a point's depth with acceptable accuracy without the correspondence of the pixels in both image planes [66], [67].

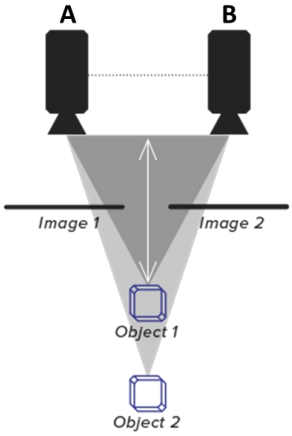

Figure 2.6: example of how a Depth map image is obtained though Stereo Camera Sensing technique.

In general, unlike RGB information, the depth image is quite invariant to illumination, color and texture changes, consequently becoming more reliable for estimating body silhouette, providing 3D structural information of human actions. On the other hand, sometimes, noisy measurements can cause some problem in the correct identification of the objects/subjects in the scene, and, for that reason, the data need to be processed and refined. [65].

Finally, RGB and Depth sensors can also be combined into a single device and are called RGB-D camera. i.e., Kinect V1, Kinect V2, Intel Real Sense D400 and so on. The Kinect V2 device will be described in detail in paragraph 2.1.1, as it was used for data acquisition in this thesis. The data acquired with RGB-D camera can be analyzed with Computer Vision techniques to extrapolate the position of specific points of the human body, often corresponding to the bumps of bones or joints (Figure 2.7 and Figure 2.8). The number of joints, usually variable from 15 to 25, depends on the biomechanical model of the human body defined for the skeleton tracking analysis. Starting from the joints position data, it is possible to compute body segments positions and joint angles [34]. Often, in the AAL domain, the skeleton tracking data are stored and analyzed instead of RGB or depth data to preserve privacy [65], [71].

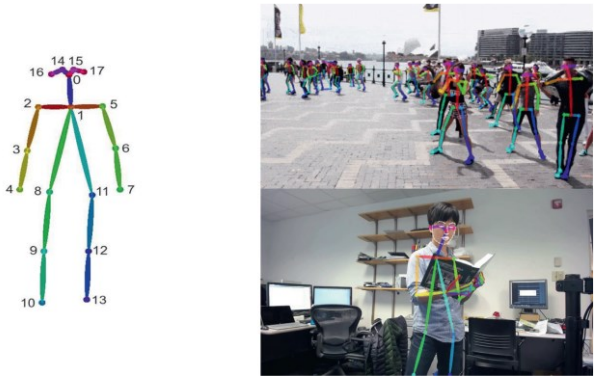

Figure 2.7: example of 2D skeleton tracking on RGB data.

Tu et al. took advantage of the spatial coordinates, calculated from the 3D skeletal tracking, with the goal of implementing an automatic AI algorithm for HAR [72]. A different approach is used from Li et al. They proposed a skeletal trajectory shape image (STSI) and skeletal pose image (SPI) sequences for modeling 3D skeletal sequences [73].

Figure 2.8: The three different types of *Vision-based* data, starting from left to right: RGB data, Depth map data, Skeleton data.

Sensor and *Vision-based* approaches are often combined to obtain detailed data about human activity. These applications allow the recognition of extremely complex activities, even in contexts where the presence of multiple individuals is expected. For instance, Taiwo et al. suggest an elderly people's residence or home, in which a large number of sensors are installed in different rooms in order to monitor different aspects of the subject daily activities (Figure 2.9). In detail, they developed a ubiquitous, cloud-based intelligent smart home, installing environment sensors (gas sensor, motion sensor, camera and so on, Figure 2.9). The system controls, monitors, and oversees the person safety and the ambient security via a smartphone mobile application (Mobile Application Interface, Figure 2.9). One module controls and monitors electrical appliances and environmental factors (Ecological Condition Interface, Figure 2.9), while another module oversees the home security and inhabitant's safety by detecting motion and capturing images

(Home Camera, Smart Graphical View and Home Control Interface, Figure 2.9) [74].

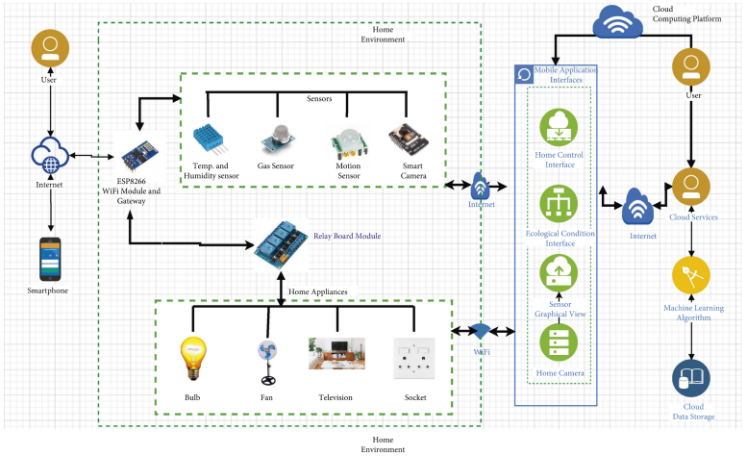

Figure 2.9: an example of Hybrid HAR approach in AAL home environment [74].

2.1.1. Kinect V2

Recently, the progress of sensor technologies has led to affordable high-definition depth cameras, such as Microsoft Kinect (Microsoft, USA). In 2010 the Kinect V1 has been a revolution in affordable 3D sensing. Initially meant only for the gaming industry, it was soon to be used by scientists, robotics enthusiasts and hobbyists all around the world. It was later followed by the release of another Kinect (Kinect V2, Figure 2.10) in 2013. Depth sensors could be a portable, affordable, marker-less alternative to three-dimension motion capture systems. The spread of these sensors has intensely influenced the human motion analysis, especially in the topic of HAR for AAL.

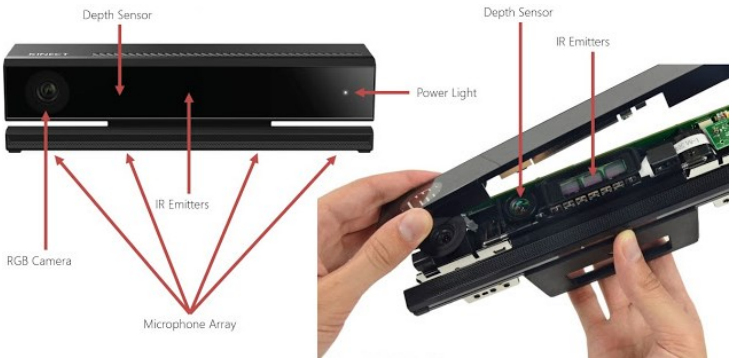

Figure 2.10: Kinect V2 components.

The Kinect V2 device contains a RGB camera, an infrared emitter (IR emitter), a depth sensor and a microphone array that can be used for speech recognition and voice control (Figure 2.10). All the technical specifications are summarized in Table 2.1 [75].

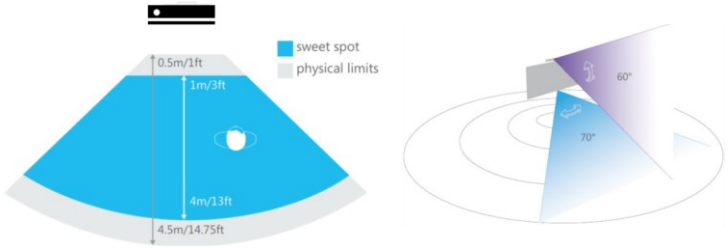

Figure 2.11: Kinect V2 depth sensor range.

Table 2.1. Kinect V2 technical specification.

Feature	Kinect V2
RGB Camera	1920 x 1080 x 16 bit per pixel 16:9 YUY2 @ 30 Hz (15 Hz in low light, HD)
Depth Camera	512 x 424 x 16 bits per pixel 16-bit TOF depth sensor
Range	Only one configuration (Figure 2.11): 0.5m to 8m (1.6 ft.–26.2 ft.) Quality degrades after 4.5m (14.7 ft.)
Angular Field of View	70° Horizontal – 60° Vertical (Figure 2.11)
Audio	16-bit per channel with 48 kHz sampling rate
Skeletal Joints	25 joints tracked (Figure 2.12)
Skeletons Tracked	6 with joints (Figure 2.12)
Vertical Adjustment	Manual, also ±27 degrees of freedom
Latency	~50ms
USB	3.0

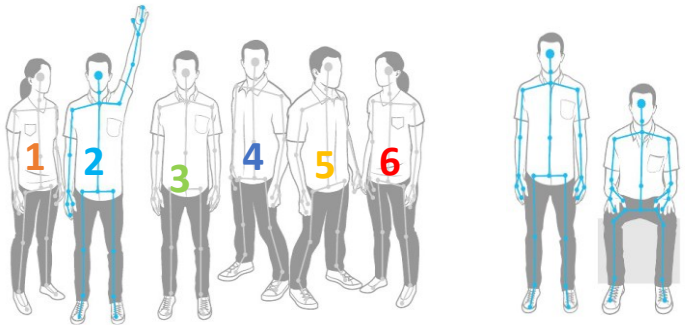

Figure 2.12: Kinect V2 body joints estimation in two different postures (on the right side) and skeleton tracking of a maximum of 6 people (on the left side).

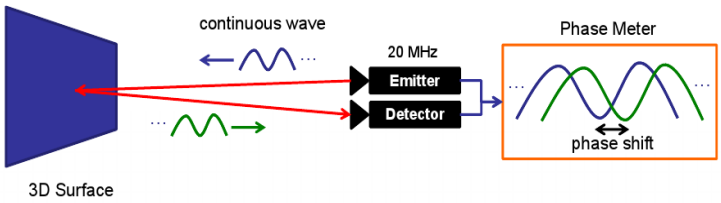

Figure 2.13: calculation of the *phase shift* of a TOF camera with *Continuous Waves Intensity Modulation* method.

Kinect V2 computes the depth image using TOF approach based on *Continuous Waves* (CW) *Intensity Modulation*. The three IR emitters, employ a light Near Infrared periodic light (NIR) to illuminate the framed area and hit the objects in the scene. As consequence, a return wave is produced and captured by the depth sensor (IR detector, Figure 2.13). The *phase shift* ($\phi[s]$) between the emitted and the returned waves is used to calculate the distance (d) of the object from the Kinect V2, through the following formula: $d = \frac{c\phi[s]}{4\pi}$, where c is the speed of light.

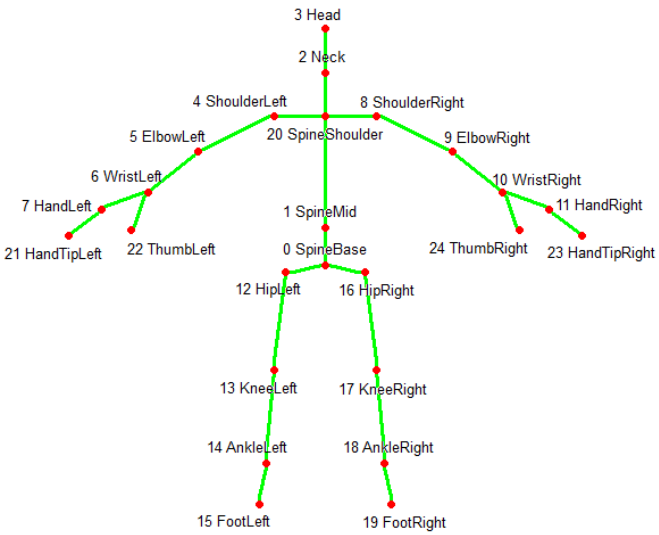

Figure 2.14: The 25 Kinect V2 joints recognized though skeletal tracking algorithm.

Starting from the depth image, Kinect V2, through its Software Development Kit (SDK), estimates 25 skeleton body joint positions in space (Figure 2.14). The joint estimation procedure, described by Shotton et al. [76], is based on three major steps. First, the human body is divided into 31 distinct parts, as shown Figure 2.15 [77]. In order to do that, they acquired almost 100k depth images of skeletons of a subject group, while driving, dancing, running…etc, employing a Motion Capture system. Starting from those depth images, they also generated synthetic data through computer graphic techniques, obtaining over a million training examples [77], [78].

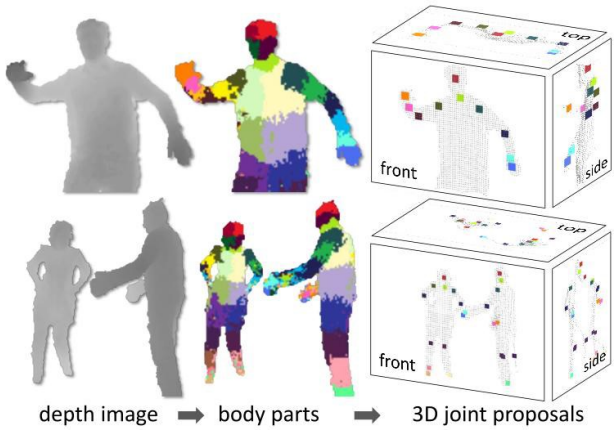

Figure 2.15: steps of skeletal tacking algorithm: starting from a single depth image, input body parts distribution is inferred and then the joints proposal estimated by the algorithm.

Second, a probabilistic per-pixel classification is performed, associating each pixel with a probability value of belonging to a body part. Third a machine learning algorithm (Randomized Decision Forest) is defined to classify with the final joint.

Kinect V2 allows also a more detailed hand and face tracking, for recognizing some specific hand gestures and facial expressions, respectively. Additionally, it is capable of fully tracking the skeletons of six users in real-time standing in the sensor's field of view. (Figure 2.12).

Kinect V2 error sources

Kinect V2 has several error sources due to the type of technology that characterized it [79,80,84]. According to Sarbolandi et al the error sources ca be summarized as follows [79]:

1. dependency on the ambient background light: Kinect V2 data acquisition can be disturbed from the ambient light. When the room is poor lighting or in the dark the acquisition frequency is reduced and the accuracy in the body joint positions reconstruction dramatically decreases;
2. multi-device interference: interference problems can occur when several Kinect V2 are used simultaneously. The IR emitters of one camera can disturb the acquisition of another camera placed in front;
3. temperature drift: as the Kinect V2 camera captures images, its illumination unit and its sensor heat up. With the increase of the temperature the characteristics of the components change, leading to a temperature-dependent drift of distance measurements;
4. systematic distance error: this error source can occur in Kinect V2 depth measurement, and it is caused due to a defective generation of the modulated light. In case of an approximated sinusoidal shape this effect is called 'wiggling' since the deviations from the ideal sinusoidal shape led to a periodically oscillating distance error (Figure 2.16, top left);
5. depth inhomogeneity (also called flying pixels): at object boundaries, a pixel may observe inhomogeneous depth values. These flying pixels lead to wrong distance values (Figure 2.16, top right);
6. multi-path effects: the IR light may not only travel along a direct path from the IR emitters via the object's surface to the depth sensor. When *indirect path* occurs only few photons hit the respective pixels, deteriorating the accuracy of the distance measurement (Figure 2.16, bottom left);
7. intensity-related distance error: when two objects are at the same distance with respect to the camera and one is more reflective than the other, the camera receives dichotomous data, resulting in an inaccurate distance calculation from the two objects (Figure 2.16 bottom right);
8. dynamic scenery: one key assumption for any camera-based device is that every pixel observes a single object factor in the course of the

entire acquisition process. This assumption is violated in case of relocating objects or shifting cameras, ensuing in action artifacts.

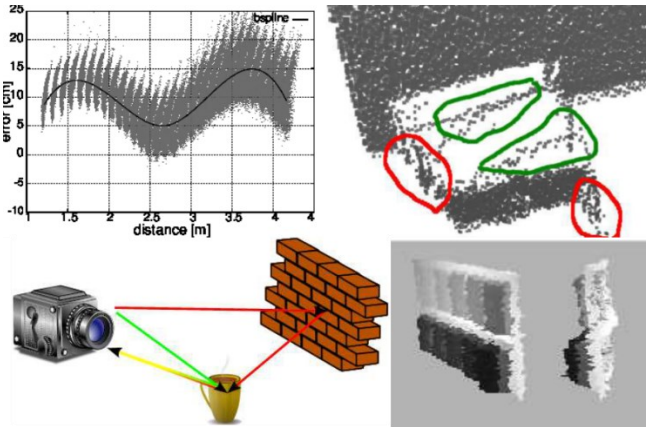

Figure 2.16: example of some error sources of Kinect V2 camera. Top left: systematic distance error. Top right: depth inhomogeneity. Bottom left: multi-path effect. Bottom right: intensity- related distance error [79].

2.2. Data pre-processing

The data pre-processing is characterized by all those operations to be carried out on the raw signal that are useful to process and to prepare the signal for the HAR next phases. The human activity identification is not a simple task as there can be a periodic occurrence of an action or two activities may have similar properties (for instance drinking and brushing teeth). Besides, how an activity is performed varies from person to person, i.e., a simple action such as picking up and object can be executed in several ways: a person can pick up an object by crouching toward the ground, while another might perform the action without bending their legs [34], [80], [81].

The data pre-processing has to be customized according to the type of sensors employed during data acquisition and, accordingly, to the nature of the data collected (i.e., RGB or Depth images, skeletal tracking, temporal or frequency signals and so on).

The most common processes used in the data pre-processing are filtering, data normalization, and segmentation. In addition, these procedures can be implemented individually or in combination with each other.

2.2.1. Noise filtering

The filtering process aims at reducing the noise that is inevitably contained in the data acquired in both *indoor* and *outdoor* settings. A filter consists in a process that, completely or partially, suppresses unwanted components from a signal [82], [83]. According to that, this phase is necessary to make the acquired data less disturbed, without artifacts, with a

reduction of any missing data ranges, ultimately more readable and usable, especially for HAR applications.

Generally, the type of noise on the acquired data particularly depends on the type of sensor employed. For commonly used recognition systems based on wearable sensors (*Sensor-based* systems) the runtime changes in the sensor setup seriously disturb the reliability of the data. These effects could be caused, for example, by relative motion between the body and sensor, wrong sensor body placement or faults related to the battery's calibration [84]–[87]. On the other hand, in case of *Vision-based* sensors the noise can be produced by the environmental light conditions, people clothing, sensor viewpoint and occlusion (especially for the skeletal tracking reconstruction) [88].

Usually, the standard denoising methods employed for the HAR process are low-pass filter, moving mean filter, linear filter, wavelet filter and Kalman Filter, depending on the nature of the signal [36], [46], [89]. However, as shown in Figure 2.17, when choosing one or more filters, it is necessary to be careful about some factors that characterize the quality of the filtered signal. For instance, the presence of noise can mask the target signal, or interfere with its analysis. However, if signal and interference occupy different spectral regions, it may be possible to improve the signal-to-noise ratio (SNR) by applying a filter to the data. Filtering takes advantage of the difference between spectra of noise and target to improve SNR, attenuating the data more in the spectral regions dominated by noise, and less in those dominated by the target. The improvement in SNR offered by the filter is welcome but filtering also affects the target signal in ways that are sometimes surprising. Obviously, any components of the target signal that fall within the stop band of the filter are lost [90]. At the same time filtering a signal introduces a delay (waveform delay). This means that the output signal is shifted in time with respect to the input. This factor is very important especially because if real time filtering is needed. In this case, the filtering of the signal at the same time as its acquisition is important, both in order not to accumulate delay between the filtered and raw signal. Moreover, a filter operates by allowing a specific range of frequencies to pass through, called cutoff frequency. For instance, since the human activity frequency is usually about 0–20 Hz [91], [92], the cut-off frequency usually sets is equal to at least twice the frequency of the signal [35], [93]–[96].

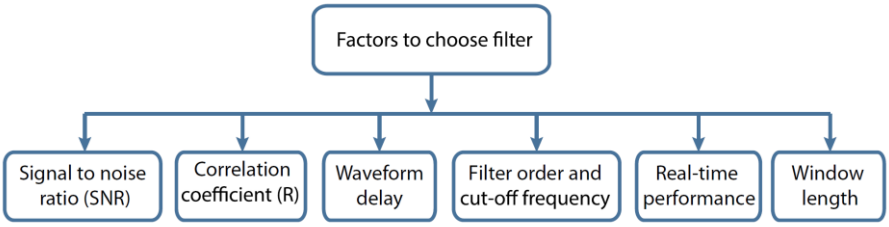

Figure 2.17: the main factors to consider when choosing a filter to apply to the raw data [36].

For instance, Ahad et al., applied a 1st order Butterworth low-pass filter with a cut-off frequency of 10Hz on Kinect V2 skeletal joints [36]. Alternatively, Jaouedi et al. proposed a Kalman filter for denoising the Kinect V2 skeletal joints [97].

2.2.2. Data normalization

Data normalization is one of the typical pre-processing approaches in which the data, with different ranges of variability, is either scaled or transformed to make it comparable. This situation often occurs in the HAR process where data originated from different types of sensors or from people with different anthropometric characteristics. According to the type of data (e.g. RGB or depth data, temporal data, skeleton data) different data normalization methods can be implemented [36], [98]. For RGB and depth images the commonly used normalization methods are [99]: min-max normalization, mean normalization, standardization and scaling to unit length. The first method scales the data in their maximum and minimum range: the minimum value is subtracted from each data point and divided by the data range. In the second one the mean of all data samples is subtracted from the data vector, and the result is divided by the difference between the maximum and minimum samples. In the standardization method instead, the mean value of all data samples is subtracted from the data vector, and the result is divided by the standard deviation value. Finally, the last normalization method scales all the data with respect to the sum of all elements of the data vector [36], [100]–[102].

When considering skeletal tracking data, there are also two other types of normalization methods. The first one is the Bounding-box normalization (referring to the border's coordinates that enclose the subject), in which all skeleton 3D joints coordinates are normalized using the maximum side-length of the bounding box of the skeleton (Figure 2.18) [103].

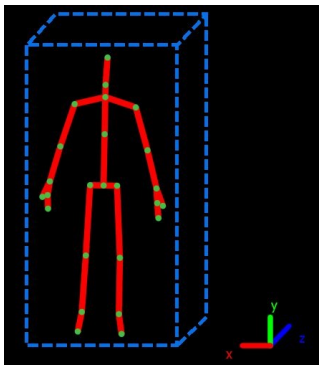

Figure 2.18: example of Bounding-box of the skeleton.

For instance, Liu et al. proposed a bounding-box-based normalization for the raw skeleton data to eliminate the skeletal coordinate differences caused by

different recording environments, individuals, and posture displacements. In the second method data are normalized by dividing the 3D coordinates of the skeleton with respect to the length of a specific the body segment (i.e., head, neck, torso and so on) or of the subject height. For example Cippitelli et al., scale joint position dividing each value by the Euclidean distance between the neck and torso joints [104].

2.2.3. Segmentation

Data segmentation process is strongly related to the type of data. When dealing with temporal data, it consists of partitioning the data into time windows. Otherwise, when RGB or depth images are analyzed, the segmentation involves the separation of the selected target subject in the scene from the background.

The data subdivision in time windows is principally done to overcome the limitations due to the difference between the duration of the action and the sampling rate imposed by the data acquisition device [35], [36] The window size has to be a compromise between data information and resolution, since the optimal window size should be calculated depending on the activity being carried out and the application field of the HAR process [36], [105], [106] For example, when the HAR process is part of a system aiming at monitoring a person in AAL environment, in which the rapidity of the recognition is mandatory, smaller window segmentation is suitable [107]. A smaller window segmentation also reduces the complexity and the computational time of the HAR process.

The segmentation techniques can be categorized into time-driven windows segmentation, event-driven windows segmentation and action-driven windows segmentation (Figure 2.19). Time-driven windows segmentation divides the temporal signal into numerous consecutive windows of fixed-size time intervals. Sometimes with this type of segmentation it could be also necessary to apply a partial overlapping between two consecutive widows. Besides, some studies proposed a fixed percentage of overlap between neighboring windows. For instance, Hammerla et al. introduced a 1s sliding window with a 50% of overlap to build a dataset for the HAR [108]. The overlapping technique can handle the transition between human activities more accurately (i.e., the transitions between sitting and standing postures, or between walking and running) [36], [84], [105], [109].

Regarding event-driven window segmentation, this method is based on the recognition of specific events which characterize the activity to be identified. For instance, in gait event-based segmentation windows are segmented based on the identification of one or more gait events, such as the foot strike or the toe-off [110]. Generally, in this segmentation techniques, the events, may not be uniformly distributed in time. This is why window size does not play an important role [35], [36], [111]. Devanne et al. in their work proposed time-driven windows, in which the entire activity time series

is decomposed into short temporal windows, regardless of the type of action performed (trying different widow sizes: 0.9s, 1.8s, 2.7s, 3.6s, 5.4s). On the other end, they also implemented event-based widows, dividing the continuous time series into motion units by automatically detecting salient motion changes. They concluded that time-driven segmentation tended to be more appropriate to action recognition, since event-driven segmentation seemed not to be so accurate in the event detection [112].

Finally, the action-driven windows segmentation separates the window data where individual activity occurs, in this way each window will have the size equal to the duration of the individualized activity [35], [113].

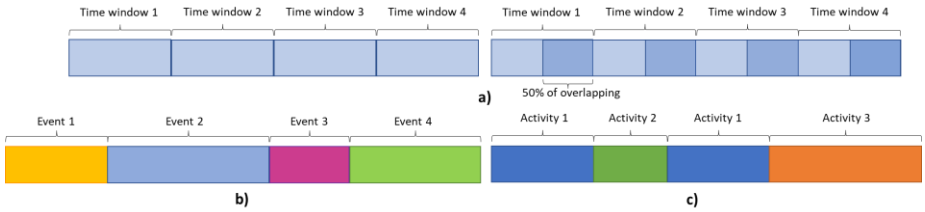

Figure 2.19: application example of different segmentation categories: a) Time-driven window segmentation: on the left sliding windows without overlapping techniques, on the right sliding windows with overlapping method (in this case, 50%); b) Event-driven window segmentation; c) Activity-driven window segmentation.

In case of RGB image, or depth image, the segmentation process is implemented using two different approaches: namely, the background subtraction and the foreground extraction. The first one consists in the extraction of the body silhouette in an image sequence captured from a static camera by comparing each incoming frame with a background model. A crucial step of this technique is to obtain a stable and accurate background model. The second one is recommended when the images are acquired by a moving camera, and it consists in the computation of the difference between consecutive images frames. The foreground extraction is more challenging than the background subtraction because, in addition to the motion of the target object, it also needs to consider the motion of the camera and the change of background [35], [114]–[116].

2.3. Feature extraction

The feature extraction procedure consists in the definition of a set of parameters able to discriminate the actions to be classified. Based on given data nature and characteristics, the features can be divided into several categories: frequency-domain or time-domain features, Kinematic features and global or local features [36], [117]–[119]. The time-domain features are usually defined to describe the data amplitude variation and distribution over

time. This set of features usually includes the main statistical metrics as median, variance, mean value, kurtosis and skewness. On the other hand, the frequency-domain features show the distribution of signal energy and predominantly used to capture repetitive nature of sensor signals The frequency domain features are extracted from data with consideration on frequency band and include Fast Fourier transform, discrete cosine transform, spectral energy, entropy, power spectral density, Fourier coefficient and wavelet features [120]. Generally, time- and frequency-domain features are computed over each segmentation window [36], [105], [121].

When the data referred to the body skeleton, the most privileged discriminative features are the kinematic ones, as the joint spatial coordinates [122]–[124] and the joint angles [125], [126]. Moreover, other features are defined to describe the geometric relations between the body joints (i.e., the size of the 3D bounding box enclosing the body skeleton [127], the distance between two joints, or between a joint and a body segment [128], or between a specific joint and an axis, or between a specific joint and an anatomical plane [125], [129], [130]). Geometric features are synthetic in the sense that they express a single geometric aspect making them particularly robust to spatial variations that are not correlated with the aspect of interest [125].

When RGB and Depth images or videos are used, usually global and local features are computed. Global features describe the image frames as a whole, providing different types of information (spatial, temporal, frequency) [114]. The main global features are the space-time volume and the Discrete Fourier Transform. The space-time volume concatenates consecutive extracted silhouette along the time, capturing the continuity of human action. Whereas the Discrete Fourier transform is widely used to represent information about the geometric structure of the silhouette [131]. Local features extract information around a set of interest points or describe a selected image region. The histogram of oriented gradients is the method more used for extracting local descriptors, consisting in counting the occurrences of the gradient orientation in a localized part of the image [35], [132], [133].

2.4. Feature selection

Working with high dimensional data increases the difficulty of knowledge discovery and human action classification due to the presence of many redundant and irrelevant features. Dimensionality reduction of the problem, achieved by removing redundant and noisy information, allows to reduce or eliminate irrelevant patterns in the dataset, improving the quality of the data and, therefore, making the process of classification more efficient [134]–[136]. Feature selection is one of the techniques used to achieve dimensionality reduction by finding the smallest possible subset of features which efficiently defines the data for the given problem [137]–[140] It can be accomplished using different methods, i.e., filter, wrapper, embedded, and

the more recent hybrid approach [35], [139], [141], [142]. Filter method measure the relevance of features using statistical standards for evaluating a subset. The features are ordered according to the ranking of importance and those below a setting threshold are removed. Among the different algorithms, the most used are: ReliefF, statistical techniques such as Principal Component Analysis, Independent Component Analysis, Neighborhood Component Analysis and Correlation Based filter [143]–[145]. Filter method processes the data before the learning model occurs and it is independent from this latter. Wrapper method selects the optimal features subset evaluating alternative sets by running the classification algorithm on the training data. It employs the classifier estimated accuracy as its metric [141], [146]. The most used iterative algorithms are the Recursive Feature Elimination with SVM, the Sequential Feature Selection algorithm and the Genetic Algorithm (detailed in the next paragraph since it is used in this thesis work) [147]. Compared to filter method, wrapper method achieves better performance and high accuracy, nevertheless it increases computing complexity due to the need to recall the learning algorithm for each feature set considered [138], [141]. In the embedded method, as the name suggests, the selection occurs within the learning algorithm. The most common are the tree algorithms like, for example, the Random Forest and the Decision Tree. Embedded method can be used in multiclass and regression problems and, compared to a wrapper method, it is computationally more effective while retaining similar performance [148]. Finally, the hybrid approach combines filter and wrapper methods to achieve the benefits of both. Commonly, the filter technique is first applied to reduce the search space and then, a wrapper model is used to acquire the best subset [149].

Genetic Algorithm

Genetic Algorithms (GAs) are adaptive heuristic search algorithms that belong to the larger family of evolutionary algorithms. They are based on the ideas of natural selection and genetics, emulating the processes of evolution [150].

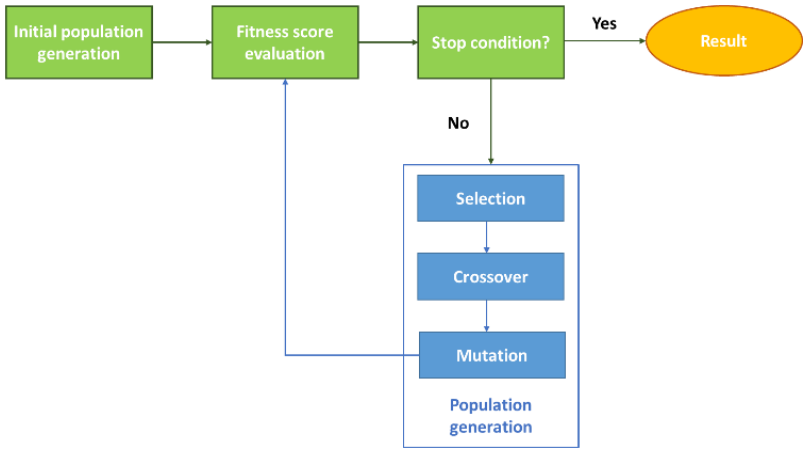

Figure 2.20: flow diagram of a GA process, illustrating the operations involved. The fitness score evaluation, selection, crossover and mutation are repeated until a stop condition is met. This condition can be a specific number of iterations (called generations), or a threshold reached (satisfying result).

The GA optimization process follows the graphical representation illustrated in Figure 2.20. A population contains a defined number of chromosomes. A chromosome is a potential solution and is composed of several characteristics (called genes). An initial population of chromosomes is generated at random, and these are decoded to obtain the corresponding parameters. In the traditional GA approach, the genes are binary quantities (0 or 1 values). An example of chromosome can be appreciated in Figure 2.21 [151].

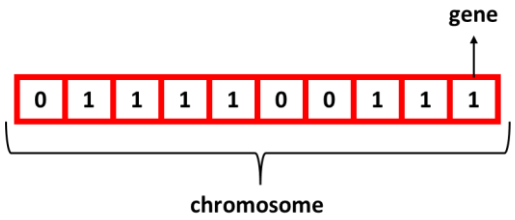

Figure 2.21: example of GA chromosome. In this case, this chromosome is composed by ten genes, each of them could be 0 or 1.

Generally, each function parameter is a gene. So, for a function with 10 parameters, a chromosome has 10 genes (one for each parameter). The following figure (Figure 2.22) is a representation of a four-element population. Each chromosome contains 10 genes

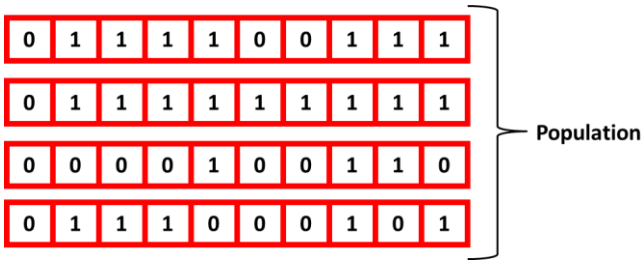

Figure 2.22: representation of four element population.

Each chromosome from the population is evaluated to obtain a *fitness score*. Better is the solution encoded by the chromosome, the better will be its fitness score. The chromosome genes are used as parameters for the function that the algorithm tries to maximize (or minimize). After the fitness evaluation, the *selection* operator chooses the parent chromosomes. They are recombined to create the next chromosome generation (*offspring*). The probability that a chromosome will be selected for reproduction is based on its fitness score. The number of chromosomes selected in each generation is equal to the size of the population. Generally, the most common selection operators are the following ones [151]:

- roulette wheel: is a stochastic selection method, where the probability for selection of an individual is proportional to its fitness. The chromosomes with higher fitness scores are more likely to be selected;
- rank selection: the chromosomes are sorted according to their fitness score; the selection is made based on the position of the chromosome in the ranking;
- tournament selection: involves running several "tournaments" among a few chromosomes, chosen at random from the population. The winner of each tournament is the one with the best fitness.
- uniform selection: this method selects the chromosomes uniformly from a roulette wheel;

The crossover operator is then applied to the chromosomes selected as parents, recombining the selected chromosomes in pairs to generate a new population. The uniform crossover method, for instance, decides whether each of the characteristics (genes) of the offspring comes from one parent or another (Figure 2.23) [151].

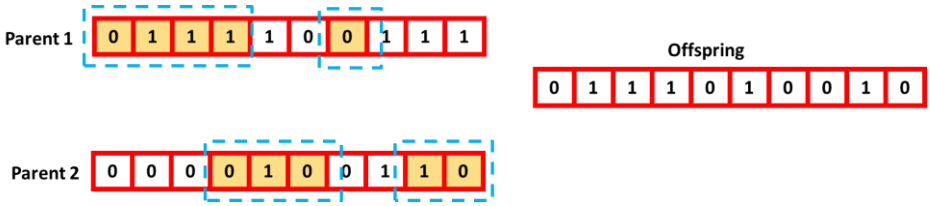

Figure 2.23: example of cross over.

Mutation involves selection, on a random basis, of a certain number of the genes in the current population and random alterations are then made to their values. This provides a random element within the GA search process so that more of the search space is considered (Figure 2.24) [151].

Figure 2.24: an example of possible gene mutation of a chromosome.

Once the chromosomes have been changed to form the new population they have to be evaluated, as were those in the previous generation. The whole procedure is then repeated for a predefined number of iterations (generations) to produce a final solution [151].

2.5. Dataset Construction

Data construction concerns the process that divides the dataset into training, validation and test data. Generally, a set of data is required to train the pattern recognition model and a set of validation data is used to evaluate the performance of the model during training epochs for tuning the hyperparameters and to estimate if the model dose not overfit, i.e., when a statistical model fits exactly against its training data. Finally, the test data, different from those involved in the training test, are used to evaluate the performance of the model [31].

In addition, the training, validation and test data could be described by labels also called classes. In the case of the HAR field, the classes represent the type of activity to be recognized (i.e., walking, sitting, lying down, and so on) [152], [153].

In HAR, three methods have been used to divide the data into training, validation and test set. In the first one, called cross-subject, the subjects are divided in two groups. The data of the first group are used for the training phase, whereas those of the second one are involved in the validation and test phase [105]. The cross-subject method aims at guiding the learning process of the Pattern Recognition model so that it becomes as robust as possible, in order to adapt it to the heterogeneity of the subjects.

The second splitting method is characterized by dividing the whole dataset on a percentage such as 70%-30%, 80%-20%, and so on. The larger portion is fed for training the model where the other portion is kept for validation and test [36]. This is the mostly used splitting criteria in the general problems of pattern recognition algorithms and have been reported in HAR with success. Alternatively, when data are acquired by multiple cameras with different points of view, a cross-view method can be used. In this case the data coming from one or more cameras are used for the training phase and those of the remaining ones for the validation and test phase. According to cross-view and cross-subjects dataset method, Wang et al. compared the performance of 10 different AI algorithms, using six Kinect benchmark datasets. The results showed that the majority of algorithms perform better on data split with cross-subject method [154].

2.6. Pattern Recognition algorithms

This section provides a brief introduction to some of the more common supervised Pattern Recognition models employed in HAR field. However, some of the presented algorithms will be described and illustrated in detail because they are utilized in the development of this thesis work.

The last phase of a HAR process is that of the Pattern Recognition, i.e., the automatic human action identification. This phase consists in the analysis of the acquired preprocessed data through AI algorithms, which may be

either Machine or Deep Learning methods, aiming at classifying human actions. Shapiro et al. defined AI as a field of science and engineering concerned with the computational understanding of what is commonly called intelligent behavior with the creation of artifacts that exhibit such behavior [155]. For this reason, AI has become more popular today thanks to Big Data, advanced algorithms, and computers with improved power and storage. The systems based on AI are becoming an integrated element of digital systems that are generating a profound impact on human decision-making through Machine Learning and Deep Learning (Figure 2.25) [156]. In 1959, Arthur Samuel coined the term Machine Learning and defined it as "the field of study that gives computers the ability to learn without being explicitly programmed". Machine Learning is part of the field of AI and is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on proving confidence intervals around these functions [157]. Along with AI, Machine Learning has emerged as technique that enables computer systems to learn from experience and data, especially for image, speech recognition, natural language processing, robot control, and other applications like HAR. In this context, AI algorithms are supervised namely, during the training phase they take advantage of the information about class present in dataset, to label the output.

2.6.1. Machine Learning

Machine learning is a viable approach for building AI systems that can operate in a complicated real-world environment.

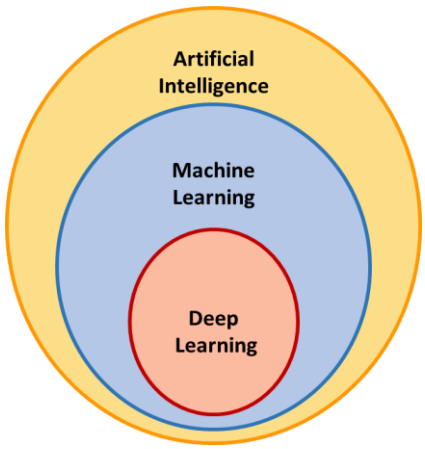

Figure 2.25: a Venn diagram showing how Deep Learning is a kind of Machine Learning, which is used for many but not all approaches to AI.

The most used Machine Learning algorithms in Pattern Recognition for HAR are Support Vector Machine (SVM), k-Nearest Neighbor (kNN),

Naïve-Bayes (NB), Hidden Markov Model (HMM), Decision Tree (DT), Random Forest (RF) and Multi-Layer Perceptron (MLP) [5].

SVM and kNN are instance-based methods: they create decision rules based on the training example data, and the new instances are compared to them using a similarity measure to find the best match and make a prediction. Following this concept, with kNN, which is one of the simplest Machine Learning algorithms, when a new data point arrives, the predictions are made by exploring the full training dataset to identify the k (number) most similar instances (called as neighbors), and by fitting the output variable for those k instances.

The SVM models instead identify a hyperplane or space that best separates the points in the input variables space by their class. All the theorical steps of the SVM algorithm are illustrated in detail in the following section.

Support Vector Machine

The idea behind the SVM classifier is based on the calculation of a hyperplane in an N-dimensional space that divides the data points belonging to different classes. This hyperplane is identified based on the maximum margins which separate the classes. These margins are calculated using data points known as Support Vectors. Support Vectors are those data points that are near to the hyperplane and help in orienting it (Figure 2.26) [158], [159].

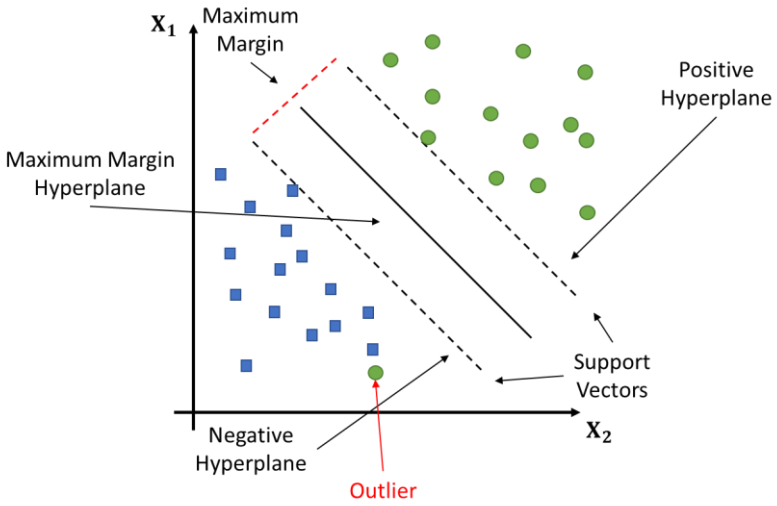

Figure 2.26: Support Vector Machine.

Generally, the calculation of the maximum margin is based on an optimization problem and in the case of the SVM classifier, a loss function known as the hinge loss function h is used and tweaked to find the maximum margin:

$$\min_{w,b} J_\gamma(w,b) = \left(\frac{1}{m}\sum_{i=0}^{m-1} h(y_i, w^T x_i + b)\right) + \lambda\frac{\|w\|^2}{2} \qquad (1)$$

where m is the number of the examples in the training set, x are the features and y are the corresponding classes; w and b are respectively the weights and the biases and λ is a regularization parameter: for $\lambda = 0$ we fall back to unregularized learning, with a small training error and a high risk of overfitting, i.e. the model memorizes the noise and fits too closely to the training set. For $\lambda \to +\infty$ the regularization term becomes dominant, imposing a strong constraint to the parameters of the classifier that improves its generalization, but also increases the training error (underfitting).

In particular, the optimal weights w can be written as a linear combination of the training samples:

$$w = \sum_{i=0}^{m-1} \alpha_i x_i \qquad (2)$$

The optimization of the hinge loss function is obtained through the application of the gradient descend algorithm. To do so, we need an expression for the gradient of the objective function of (5) with respect to the parameters w and b:

$$\nabla_w J_\lambda(w,b) = \left(\frac{1}{m}\sum_{i=0}^{m-1} h'\ (y_i, w^T x_i + b)\, x_i\right) + \lambda w \qquad (3)$$

$$\frac{\partial}{\partial b} J_\lambda(w,b) = \left(\frac{1}{m}\sum_{i=0}^{m-1} h'\ (y_i, w^T x_i + b)\right) \qquad (4)$$

There may be some cases in which the classes are not linear separable, as in Figure 2.27.

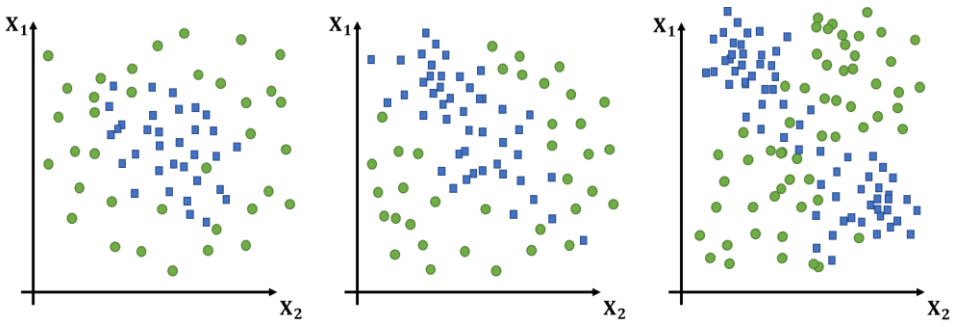

Figure 2.27: some examples of classification problems that cannot be dealt with SVM linear classifer.

A conceptually simple way to deal with classes that cannot be linearly separated is to use a feature map (Φ) to project the features in a space where they are easier to separate. This concept is called kernel trick, which allows to have the best of both worlds: a large space where samples can be easily separated, and a limited use of computational resources [160], [161]. The kernel trick consists in using a kernel function to manipulate vectors mapped

into a high-dimensional space by processing the corresponding vectors in the original low-dimensional space (Figure 2.28). More precisely, a kernel is a binary function $k : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}$ for which it exists a mapping $\Phi : \mathcal{L} \rightarrow \mathcal{H}$ from a low-dimensional space $\mathcal{L}$ into a high-dimensional vector space $\mathcal{H}$, such that:

$$k\left(x, x^{'}\right) = \Phi(\mathrm{x})^{T}\Phi(x)' \quad (5)$$

for every pair $x, x' \in \mathcal{L}$.

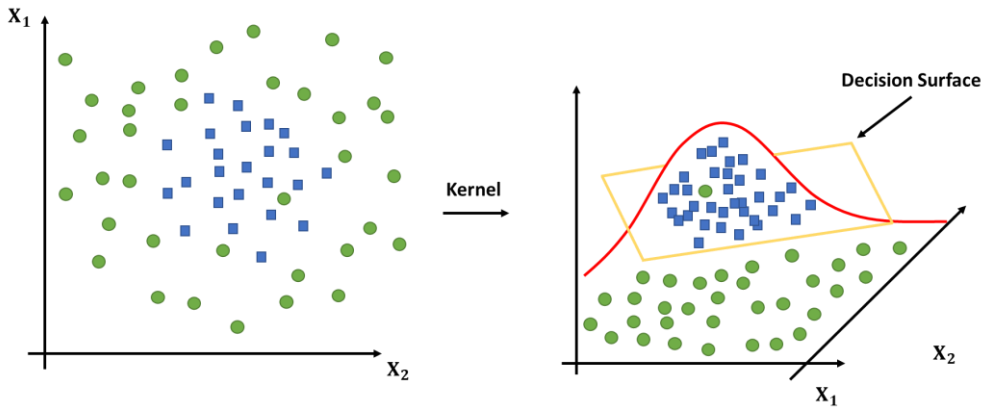

Figure 2.28: example of the application of the kernel trick, mapping the classification problem into high-dimensional space.

Generally, a widely used kernel function is the *polynomial kernel*, defined as:

$$k\,(x, x') = (\,x^{T}x^{'} + 1)^{d} \quad (6)$$

where d is an integer parameter representing the degree of the *polynomial kernel*. Another popular one is the *Gaussian RBF kernel* which is defined as:

$$k\left(x, x^{'}\right) = e^{-\gamma\left\|x-x^{'}\right\|^{2}} \quad (7)$$

Employing the *kernel function*, the hinge loss function h becomes:

$$\underset{w,b}{\mathrm{Min}}\, \mathcal{J}_{\gamma}(w, b) = \left(\frac{1}{m}\sum_{i=0}^{m-1} h(y_i, w^{T}\Phi(x_i) + b)\right) + \lambda\frac{\|w\|^{2}}{2} \quad (8)$$

It can be shown that the optimal vector weights w is guaranteed to be a linear combination of the projected training samples $\Phi(x_i)$:

$$w = \sum_{i=0}^{m-1} \alpha_i \Phi(x_i) \quad (9)$$

In fact, instead of looking directly for the best w it is possible to search for the best coefficients $\alpha = (\alpha_0, \ldots, \alpha_{m-1})$.

The optimization of the new hinge loss function (substituting the equation (12) in the equation (13)), is obtained applying the gradient descent algorithm:

$$\frac{\partial}{\partial \alpha_l} \mathcal{J}_\lambda(\alpha, b) = \left(\frac{1}{m} \sum_{i=0}^{m-1} h' \left(y_i, b + \sum_{j=0}^{m-1} \alpha_j k(x_j x_i) \right) k(x_i x_l) \right) + \lambda \sum_{i=0}^{m-1} \alpha_i k(x_i, x_l)$$

(10)

$$\frac{\partial}{\partial b} \mathcal{J}_\lambda(\alpha, b) = \frac{1}{m} \sum_{i=0}^{m-1} h' \left(y_i, b + \sum_{j=0}^{m-1} \alpha_j k(x_j x_i) \right) \qquad (11)$$

Multi-class SVM

Binary classification models such as SVMs can be adapted to work with more than two classes. One way to achieve this is to break the multi-class problem into multiple binary classification problems. The two most common strategies to do so are called *one versus rest* and *one versus one*.

The *one vs. rest* strategy consists in training one binary classifier for each class. The samples of that class are considered positive and those of all the other classes are considered negative. This strategy requires that the underlying binary classification model is able to output a confidence score instead of just a class label. The combined classifier predicts as output the class for which the highest confidence score has been obtained.

In the *one vs. one* strategy a binary classifier is trained for each pair of classes. For instance, if there are four classes (0, 1, 2, 3) there will be six classifiers (0 vs. 1, 0 vs. 2, 0 vs. 3, 1 vs. 2, 1 vs. 3 and 2 vs. 3). Note that each binary classification problem includes only a fraction of the original training set (only the samples belonging to one of the two classes). At inference time, the combined classifier submits the input samples to the binary classifiers. Each binary classifier votes for one of the two classes it has been trained on, and the class that collects the highest numbers of votes is taken as prediction (ties are broken arbitrarily). As an alternatively, to the instance-based methods there are stochastic-based and tree-based techniques, such as NB, HMM and DT, RF. NB model is a very simple algorithm based on Bayesian procedure: explicitly applies Bayes' Theorem to the training data, requiring the knowledge of a priori and conditional probabilities that are related to the problem under consideration. A Markov model, instead, is a stochastic model that contains states and events, represented by transitions, and is often applied to temporal and sequential data because it can adequately describe the dependencies of current data with

previous data. HMM belong to statistical Markov models, which presumes that the states of the Markov process are unobservable, and each state emits a discrete random output

Tree-based methods, instead, use a series of *if-then* rules to generate predictions from one or more decision trees. DT are easier compared to the RF. A DT combines some decisions, whereas a RF combines several DT. In general, the latter builds up a model, resembling a sort of decision-making diagram, based on identifying some significant information and eventual correlations between the attributes of the dataset. A prediction for a given new record is obtained following the tree structure until a leaf is reached.

Finally, MLP models, are shallow artificial neural networks based on structures inspired by biological neuronal networks. Below, as for the SVM algorithm, the concept and operation of the MLP algorithm will be explained in detail.

Multi-Layer Perceptron

A MLP is a class of fully connected feedforward artificial neural network (ANN). Generally, ANN are modeled as collections of neurons communicating through a network of connections. Each neuron performs a simple computation, and the combination of the processing of multiple neurons produces complex behaviors. A single neuron is modeled as having an activation state depending on neighboring neurons that send their own activation state through the output connection (whose biological counterpart is called synapse) [162]. In one of the first proposals for the workings of an artificial neuron, the sum of the activations of the neighbors, weighted by the strength of the connections, measures the excitation of the neuron, and if that measure exceeds a threshold, the neuron is activated (16).

$$\text{activation} = \begin{cases} 1 & "if\ "\ \Sigma_{j=0}^{n-1} w_j x_j \geq \tau \\ 0 & "otherwise" \end{cases} \qquad (12)$$

where $x_0, \dots, x_{n-1}$ are the activations of the neighbors, $w_0, \dots, w_{n-1}$ are the weights of the corresponding connections and τ is the activation threshold.

This model was proposed by Rosenblatt in 1958 with the name perceptron with the aim of solving image recognition problems [163]. A learning algorithm was designed to tune the parameters, $w_0, \dots, w_{n-1}$ and τ in a supervised way, i.e., learning a function that maps an input to an output based on example input-output pairs. However, the perceptron algorithm (16) works only for linearly separable classes, i.e., if the function to be learned is represented by a linear function. This drawback can be overcome by a real-valued activation level that is the output of a suitable activation function $a(\cdot)$. Furthermore, the threshold τ is replaced by a bias b. The resulting model is represented in Figure 2.29 and its mathematical formulation is the following:

$$\text{activation} = a(z) = a\left(b + \sum_{j=0}^{n-1} x_j w_j\right) \qquad (13)$$

where z is the total input to the neuron.

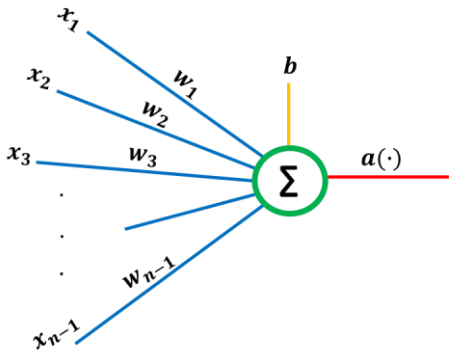

Figure 2.29: model of artificial neuron.

Several different activation functions are considered in the literature [164], [165] (Figure 2.30):

$$a(z) = \max(z, 0) \quad \text{(Rectified Linear Unit - ReLU)}$$

$$a(z) = \frac{1}{1-e^{-z}} \quad \text{(sigmoid)}$$

$$a(z) = \frac{e^{z} + e^{-z}}{e^{z} - e^{-z}} \quad \text{(hyperbolic tangent - tanh)}$$

$$a(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_K}} \; for\; j = 1, \ldots K \quad \text{(softmax)}$$

(14)

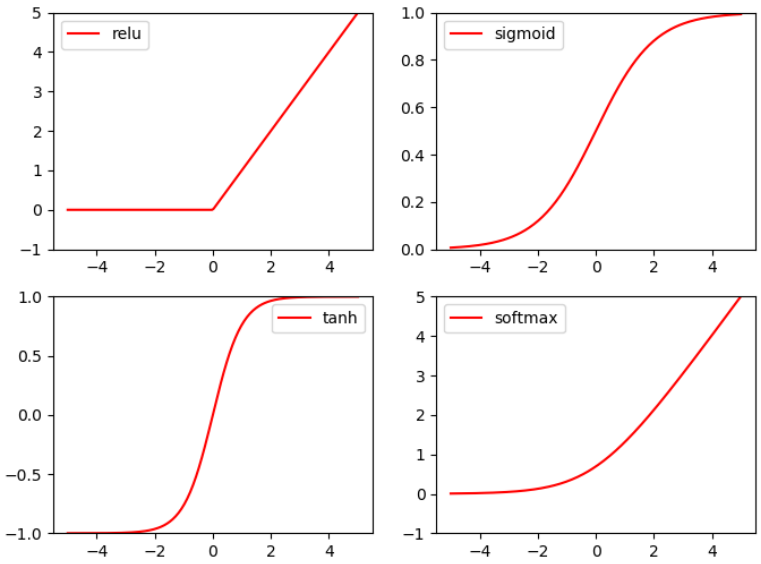

Figure 2.30: example of some activation functions for the neuronal model.

As mentioned, the MLP is a kind of feedforward neural network whose neurons and those of feedforward networks in general may be divided into three groups: input, output and hidden neurons. Input neurons have no incoming projections, and their activation function is taken directly from the input of the network. Output neurons have no projections to other neurons: their activation functions form the output computed by the network. Hidden neurons have both incoming and outgoing projections: they take the information from other neurons, process it, and share the processed information to the other neurons [166].

The total number of the neurons which compose the MLP model depends on the dimension of the problem. For instance, for a classification problem there would be an input neuron for each component of the feature vector, and an output neuron for each class (the class predicted by the network would be that corresponding to the output neuron with the highest activation). The number and the size of the hidden layers determine the complexity of the network (Figure 2.31). Generally, each layer l contains a number n_l of neurons. The activations of the input neurons are the input features. The activation $x_j^{(l)}$ of the j-th neuron of a layer $l > 0$ is calculated as:

$$x_j^{(l)} = a(z_j^{(l)}) \qquad (15)$$

$$z_j^{(l)} = b_j^{(l)} + \sum_{r=0}^{n_l-1} w_{jr}^{(l)} x_r^{(l-1)} \qquad (16)$$

where $z_j^{(l)}$ is the total input to the neuron, activation function $b_j^{(l)}$ is the bias and $w_{jr}^{(l)}$ is the weight of the projection from the r-th neuron of the $l-1$ layer. The activation function a:$\mathbb{R} \rightarrow \mathbb{R}$ here is assumed to be the same for all neurons.

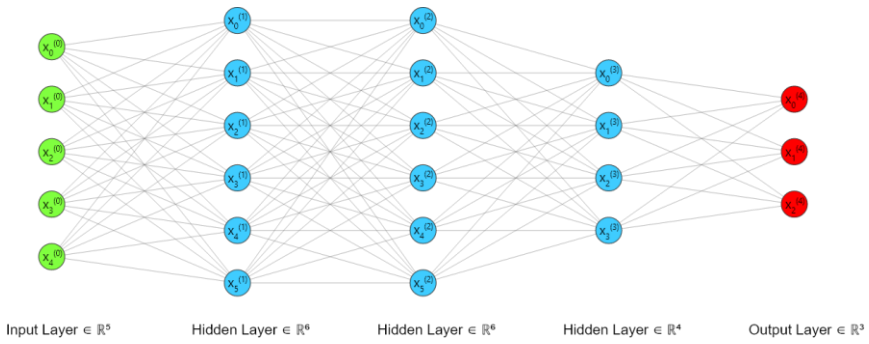

Figure 2.31: example of a MLP with three hidden layers.

Training a MLP consists in finding the weights $W^{(1)}, \ldots . W^{(D)}$ (where D correspond to the *depth* MLP network, i.e., the numbers of l layers), and the biases $b^{(1)}, \ldots, b^{(D)}$ that minimize a suitable loss function $L(\cdot)$.

The backpropagation algorithm focuses on the partial derivatives of the loss with respect to the activations of the neurons:

$$\delta_j^{(l)} = \frac{\partial L}{\partial x_j^{(l)}} \qquad (17)$$

Backpropagation continues by tracing back the operations executed in the forward pass. From each $\delta_j^{(l)}$, on the basis of Equation 19 (19), we can obtain the derivatives with respect to the total inputs to the neurons $z_j^{(l)}$:

$$\frac{\partial L}{\partial z_j^{(l)}} = \frac{\partial L}{\partial x_j^{(l)}} \cdot \frac{\partial x_j^{(l)}}{\partial x_j^{(l)}} = \delta_j^{(l)} \cdot a'\left(z_j^{(l)}\right) \qquad (18)$$

where $a'\left(z_j^{(l)}\right)$is the derivative of the activation function.

The derivatives with respect to the total inputs are used to compute the values of $\delta_r^{(l-1)}$ of the previous layers. To do so the chain rule for the derivative of composite functions is applied to Equation 14 (14):

$$\delta_r^{(l-1)} = \frac{\partial L}{\partial x_r^{(l-1)}} = \sum_{j=0}^{n_l-1} \frac{\partial L}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial x_r^{(l-1)}} = \sum_{j=0}^{n_l-1} (\delta_j^{(l)} \cdot a'(z_j^{(l)})) W_{jr}^{(l)} \qquad (19)$$

The name "backpropagation" comes from the fact that, according to Equation 19 (eq. 19), the derivatives are computed in the backward direction starting from the output layer and moving towards the input (which is the opposite of the "forward" direction used to compute the output of the network). The expression can be written in vector notation as follows:

$$\delta_r^{(l-1)} = (W^{(l)})^T (\delta^{(l)} \odot a'\left(z_j^{(l)}\right)) \qquad (20)$$

where $\odot$ denotes the elementwise product and $a'(\cdot)$ the elementwise application of the derivative of the activation function (Hadamard product).

The vector $\delta^{(l)}$ allows the computation of the partial derivatives of the loss with respect to the parameters $W_{jr}^{(l)}$ and $b_j^{(l)}$:

$$\frac{\partial L}{\partial W_{jr}^{(l)}} = \frac{\partial L}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{jr}^{(l)}} = (\delta_j^{(l)} \cdot a' \ (z_j^{(l)}))x_r^{(l-1)} \qquad (21)$$

$$\frac{\partial L}{\partial b_j^{(l)}} = \frac{\partial L}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} = \delta_j^{(l)} \cdot a'\left(z_j^{(l)}\right) \qquad (22)$$

Putting together the forward and backward pass, the backpropagation algorithm is therefore (Figure 2.32):

1. Forward pass:
   a. set the activation of the input $x^{(0)}$;
   b. for $l = 1,2, \dots D$ do:
      I. compute the input $z^{(l)} = W^{(l)} + x^{(l-1)} + b^{(l)}$
      II. compute the activations $x^{(l)} = a(z^{(l)})$
   a. compute the posterior probabilities: $\hat{p} = softmax\left(x^{(D)}\right)$
2. Backward pass:
   a. compute the output $\delta^{(D)} = \hat{p} - \bar{y}$, where $\bar{y}$ is the true class vector, since MLP is a supervised learning.
   b. for $l = D-1, D-2, \dots 1$ do:
      I. compute $\delta^{(l)} = \left(W^{(l+1)^T}\right)\left(\delta^{(l+1)} \odot a'\left(z^{(l+1)}\right)\right)$
      II. compute the derivatives
      III. $\nabla_{W^{(l)}} L = \left(\delta^{(l)} \odot a'\left(z^{(l)}\right)\right)\left(x^{(l-1)}\right)^T)$
      IV. and $\nabla_{b^{(l)}} L = \left(\delta^{(l)} \odot a'\left(z^{(l)}\right)\right)$

It is possible to use this algorithm to compute the gradients for the parameters corresponding to a single training sample. Given a training set of m samples, the gradient of the average loss is just the average of the single gradients. Finally, the parameters can be updated by gradient descent, with learning rate η:

$$W'^{(l)} \leftarrow W^{(l)} - \eta \nabla_{W^{(l)}} L, \forall l \in \{1,2, \dots, D\} \qquad (23)$$

$$b'^{(l)} \leftarrow b^{(l)} - \eta \nabla_{b^{(l)}} L, \forall l \in \{1,2, \dots, D\} \qquad (24)$$

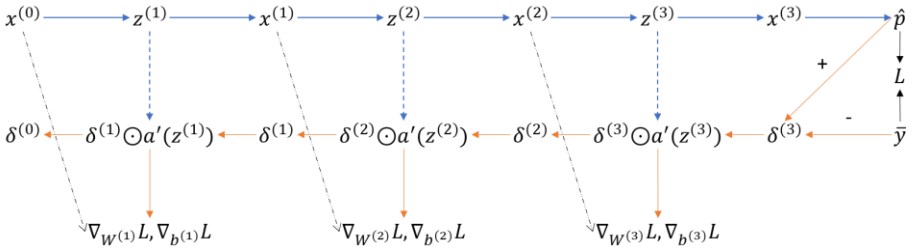

Figure 2.32: schematic view of backpropagation for a multi-layer perceptron

2.6.2. Deep Learning

Deep Learning may be considered as a subset of Machine Learning, involving algorithms able to deal with complicated problems and reach human-level performances (Figure 2.25). Deep learning algorithms work well with large datasets: the greater the number of dataset data, the better performance in the results will be achieved by the algorithm, in fact they can even work with inter-connected and unstructured datasets. Thank to those characteristics, it is possible to use them in more specific and particular contexts [167], developing more efficient and effective approaches for different human-centered Indoor IoT applications, i.e., indoor localization [168], fall detection [169], Human Activity monitoring and recognition [120] and other goals [170].

In general, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) have been confirmed to perform well in various applications of Deep Learning-based HAR research

CNNs are currently the most used deep learning architectures, since they are able to deal with many different tasks, such as image classification, object detection, and text recognition topics. CNNs are based on convolution, a mathematical operation that allows the merging of two sets of information. In the case of CNN, convolution is applied to the input data to filter the information and produce a feature map. This filter is also called a kernel. There are four basic ideas behind CNNs that benefit from the characteristics of natural signals: local connections, shared weights, pooling, and the use of many layers. These four key ideas can be labeled as the Convolution layer, Rectified Linear Unit (ReLU) layer, Pooling, and Fully Connected (FC) Layer, respectively.

RNNs are a subset of deep learning algorithms that contains loops, allowing information to be exchange and stirred within the network. In short, RNNs use the facts learned from previous experiences to inform upcoming events. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are a special kind of RNN capable of learning temporal relationships on a short-term and long-term scale [5], [35], [36], [64], [171]–[175].

All the theoretical steps concerning the RNNs, LSTMs and GRUs neural networks will be discussed in detail below.

Basic RNN model

RNN are a class of networks especially designed to deal with variable-length sequences. They are "recurrent" because they have a backward connection that makes them reprocess as additional input their previous activations. This mechanism forms a feedback loop in the layout of the neurons, that creates a memory of the information processed for previous inputs (Figure 2.33). Because of the loop, these models do not fall in the category of feed forward networks [171], [176], [177].

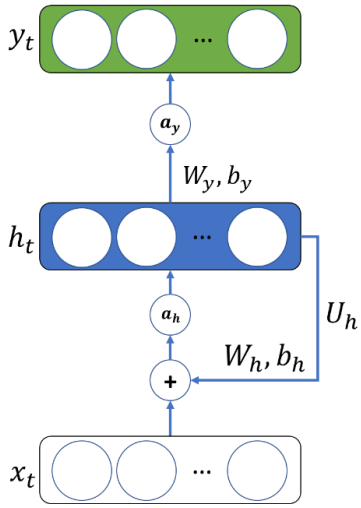

Figure 2.33: graphical representation of a RNN model. All the connections between layers are fully connected.

The process of carrying memory forward is devoted to the hidden state of the network, whose computation can be described with the following mathematical formula:

$$h_t = a_h(W_h x_t + U_h h_{t-1} + b_h) \quad (25)$$

where $b_h$is a n_h −dimensional vector of biases, and $a_h(\cdot)$ is an activation function (hyperbolic tangent and ReLU are common choices in this case, see eq. 14 and Figure 2.30). At the beginning the state is usually set to zero ($h_{-1} = 0$).

The hidden state is also used to compute the output $\hat{y}_t$:

$$\hat{y}_t = a_y(W_y h_t + b_y) \quad (26)$$

where W_y is a $n_y \times n_h$ matrix of weights connecting the hidden to the output layer, b_y is a vector of biases and $a_y(\cdot)$ is the activation function.

For RNN the backpropagation starts from the last time steps and proceeds back to $t = 0$. For this reason it is called Back-Propagation Through Time (BPTT).

The derivation of backpropagation equations for RNN models is similar to that for MLP (Chapter 2, paragraph 2.6.1). Starting from the loss function derivatives are computed by following the operations involved in the network, but in the opposite order with respect to the forward pass. It is useful to simplify the writing of the equations 25 and 26 in the following way:

$$h_t = a_h(z_t) \qquad (27)$$

$$\hat{y}_t = a_y(u_t) \qquad (28)$$

where z_t and u_t are respectively:

$$z_t = W_h x_t + U_h h_{t-1} + b_h \quad (29)$$

$$u_t = W_y h_t + b_y \qquad (30)$$

Therefore, if we consider the softmax as activation function (a_y) (14, Figure 2.30), the loss function is the average cross entropy between the estimates $\hat{y}_t$ and the target values $\bar{y}_t$:

$$L = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=0}^{n_y-1} \bar{y}_{\mathrm{tj}}\, log\hat{y}_{tj} \qquad (31)$$

where T is the time vector $[0, \dots, t]$.

Let δ_t^u be the gradient of L with respect to u_t, it is possible to compute the δ_t^u through the following formula (considering cross-entropy loss and softmax activation function):

$$\delta_t^u = \frac{1}{T}\,\hat{y}_t - \bar{y}_t \qquad (32)$$

Considering the equations 27, 28, 29 and 30 and noting that u_t and y_t influence only the component of the loss at step t and that the hidden state h_t impacts the loss through the output $\hat{y}_t$ and though the next state z_{t+1}. The gradient of the loss L (δ_t^u) is obtained with the sum of two terms:

$$\delta_t^h = W_y^T \delta_t^u + U_h^T \delta_{t+1}^z \qquad (33)$$

where δ_t^h and δ_t^z the gradient of L respect to h_t and z_t.

Finally, δ_t^z is computed with the application of chain rule for the derivative of composite functions (see equations 27 and 30 and the relationship between u_t and h_t):

$$\delta_t^z = a'_h(z_t) \odot \delta_t^h \qquad (34)$$

where a'_h is the element-by-element derivative of the activation function and $\odot$ is Hadamard product. For the last step δ_T^z is set to zero. At this point, it is possible to use the derivative to compute the gradient for all parameters in the RNN. To do so, it is necessary to consider that weights and biases are the same for all the time steps. The result is the following:

$$\nabla_{U_h} L = \sum_{t=0}^{T-1} \delta_{t+1}^z \cdot h_t^T \qquad \nabla_{b_h} L = \sum_{t=0}^{T-1} \delta_t^z \qquad (35)$$

$$\nabla_{W_y} L = \sum_{t=0}^{T-1} \delta_t^u \cdot h_t^T \qquad \nabla_{b_h} L = \sum_{t=0}^{T-1} \delta_t^u \qquad (36)$$

$$\nabla_{W_h} L = \sum_{t=0}^{T-1} \delta_t^z \cdot x_t^T \qquad (37)$$

Long Short-Term Memory

Classic RNN models have a drawback as they are not able to learn long-term dependencies. LSTM is an evolution of the basic RNNs, since thanks to their structure, LSTM networks can model dependencies in the data at a long distance in time. The weakness in the basic RNN comes from the linear operation between states at consecutive time steps. Considering the hidden states h_t and $h_{t+\Delta t}$ at Δt steps of distance and combining the equations eq. 33 and eq. 34, it is possible to observe that the derivative of the loss with respect to h_t depends on a term with U_h^T applied to a multiple derivative with respect to $h_{(t+1)}$. Continuing recursively, it is possible to see that it will depend on a term with $(U_h^{\Delta t})^T$ applied to a multiple derivative with respect to $h_{(t+\Delta t)}$. Even for relatively small numbers of Δt steps, the elements of the matrix $U_h^{\Delta t}$ will tend to vanish to zero, or to explode to $+\infty$. In fact, from the point of view of the backpropagation algorithm, the RNN is like a very deep MLP, only with shared weights. For a MLP 10 layers are enough to make a very deep network, but RNNs are expected to work with sequences that are a lot longer than that.

LSTM introduces a new mechanism to carry information through the computation. In Figure 2.34 it is possible to observe a graphical representation of an LSTM architecture.

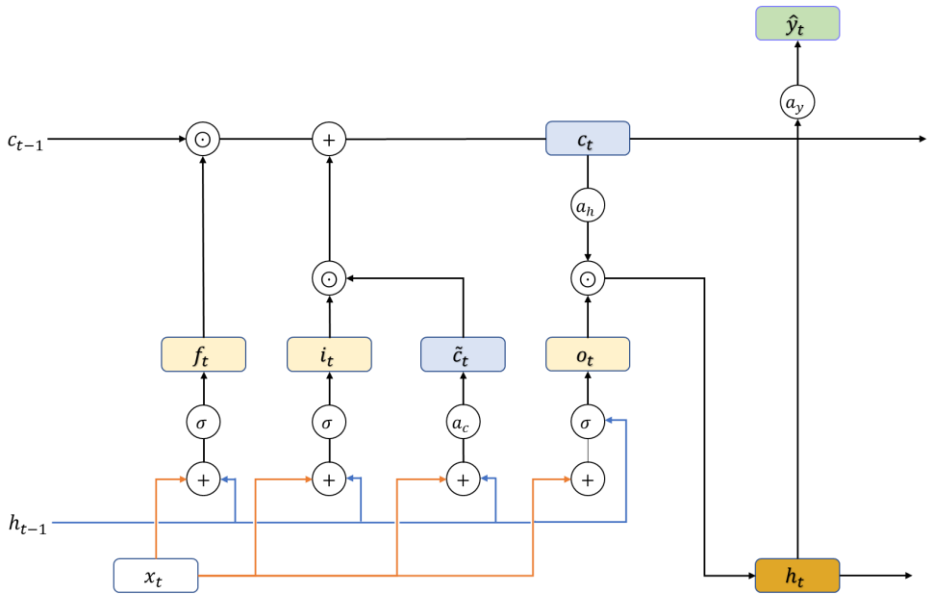

Figure 2.34: graphical representation of the LSTM model. Connections in orange are added to the parameters of the model. Usually, a_c and a_h are the hyperbolic tangent activation function. The function a_y depends on the problem.

The LSTM architecture is composed by memory cells, called *cell states*, each made up of three gates: *forget gate*, *input gate* and *output gate*.

The *forget gate* (f_t) has the purpose to decide which information should be thrown away or kept (forget). The *forget gate* is calculated through the following equation:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \qquad (38)$$

where the information from the previous hidden state (h_{t-1}) and the information from the current input (x_t) is passed through the sigmoid function (see eq. 14 and Figure 2.30). Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

The *input gate* (i_t), similar to the *forget* gate, helps to identify important elements that need to be added to the *cell state* (c_t).

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \qquad (39)$$

The *input* and *forget gate* are used to update the current *cell state* $[\![(c]\!]_t$), through the following formula:

$$c_t = c_{t-1} \odot f_t + \tilde{c}_t \odot i_t \qquad (40)$$

where $\tilde{c}_t$ is called *candidate state* and c_{t-1} is the *cell state* computed at the time step t-1.

The *candidate state* is calculated as the sum between the information that the network received as input (x_t) and the previous hidden state $[\![(h]\!]_{t-1}$):

$$\tilde{c}_t = a_c(W_c x_t + U_c h_{t-1} + b_c) \qquad (41)$$

where $a_c(\cdot)$ is an activation function and typically corresponds to the hyperbolic tangent (see eq. 14 and Figure 2.30). The hyperbolic function ranges the values between [-1, 1].

At this point the *cell state* is used to update the hidden state (h_t). This is done with the *output gate* (o_t), that determine what information is remembered between hidden state:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \qquad (42)$$

$$h_t = a_h(c_t) \odot o_t \qquad (43)$$

where $a_h(\cdot)$ is an activation function and typically corresponds to the hyperbolic tangent (see eq. 14 and Figure 2.30). The hyperbolic function ranges the values between [-1, 1].

As a last operation, the output is computed from the hidden state (this part is the same of the basic RNN):

$$\hat{y}_t = a_y\big(W_y h_t + b_t\big) \qquad (44)$$

LSTM networks are certainly more complicated than the basic RNN, but they are also significantly more powerful, and represent the state of the art in terms of recurrent neural models. Despite their complexity, they can be used as direct replacement of the basic RNN.

Gated Recurrent Unit

Many models have been proposed to simplify LSTM without losing its effectiveness. Among these the GRU is the one that got closer to the objective. The behavior of GRU is similar to that of LSTM, but it uses less parameters and its implementation is a lot simpler. Figure 2.35 summarizes the GRU architecture

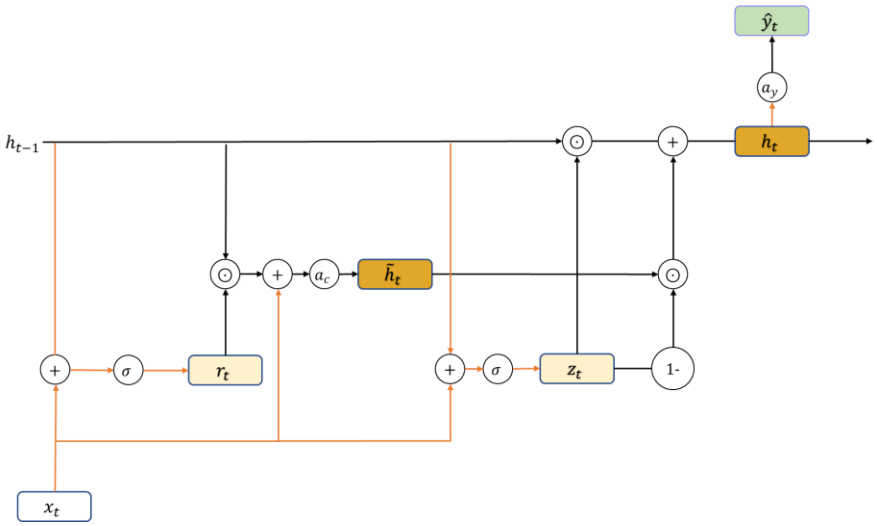

Figure 2.35: graphical representation of the GRU model. Connections in orange are added by the parameters of the model equation. Usually, a_c is the hyperbolic tangent activation function. The function a_y depends on the problem.

In the GRU model there is not a separate *cell state*. The update of the hidden state (h_t) depends on two gates: the *update gate* (z_t) and the *reset gate* (r_t):

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \qquad (45)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \qquad (46)$$

The *reset gate* is used to modulate the previous state in the computation of the new candidate state $\tilde{h}_t$ (how much past information to forget):

$$\tilde{h}_t = a_h(W_h x_h + U_h(r_t \odot h_{t-1}) + b_h) \qquad (47)$$

The *update gate* is similar to the *input and output gates* and is used to mix the previous and the candidate state (decides what information to throw away and what new information to add):

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \qquad (48)$$

Bidirectional RNN

RNNs are asymmetric networks. The output at time t is determined by all the input values with $t' \leq t$, but not by the rest of the input sequence. Sometimes the knowledge of all the inputs allows to achieve significantly

better accuracy. For instance, in optical character recognition knowing both the strokes to the left and to the right may greatly help in recognizing the character at the current location.

A simple but effective way of exploiting the information from both past and future samples is to stack two layers of hidden states. The resulting model is called *bidirectional RNN*. The state $\vec{h}_t$ will be computed for increasing values of t, by combining x_t and $\vec{h}_{t-1}$. The state $\overleftarrow{h}_t$ will be computed for decreasing values of t, by combining x_t and $\overleftarrow{h}_{t+1}$. Both $\vec{h}_t$ and $\overleftarrow{h}_t$ are used to compute the output $\hat{y}_t$ or, are used as input for another bidirectional layer.

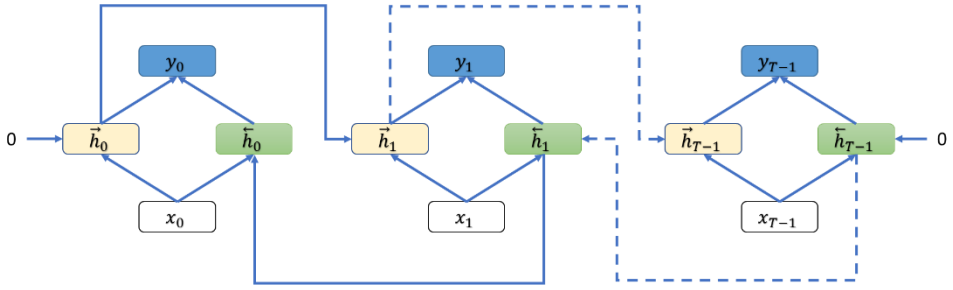

Figure 2.36: diagram of a bidirectional RNN.

This architecture is depicted in Figure 2.36. The basic RNN, LSTM and GRU models can all be used as basic blocks for this architecture.

2.7. AI algorithms performance measure

After training the model to recognize activities based on sensor data, it is essential to investigate the effectiveness of the built model. The consideration of the performance for machine and deep learning methods can be performed using some evaluation matrices. The most common metrics are: Accuracy, Sensitivity, Specificity, F-score, Confusion Matrix and ROC curve. These parameters rely upon the concept of True Positive (TP, the number of outcomes where the model correctly predicts the positive class.), True Negative (TN, indicates the number of outcomes where the model correctly predicts the negative class.), False Positive (FP, denotes the number of outcomes where the model incorrectly predicts the positive class.), and False Negative (FN represents the number of outcomes where the model incorrectly predicts the negative class) [178]–[180].

Accuracy

Accuracy is a metric parameter for evaluating classification models and is defined as the ratio of the number of correct predictions by a model to the

number of input samples in total. In general, for binary classification, accuracy can be calculated as:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad (49)$$

Sensitivity

The Sensitivity (also called Recall) is a metric parameter that measures the proportion of genuinely positive samples that are currently identified as such. It is defined as:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP+FN}} \quad (50)$$

Specificity

The Specificity is the proportion of genuinely negative samples that are currently identified as such. It is defined as:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} \quad (51)$$

F-score

F-score is an overall measure of the model's accuracy that combines precision and recall. Precision is the number of positive results divided by the number of all positive results returned by a classifier. Recall, instead, is the ratio between TP and the number of all samples that should have been identified as positive, which corresponds to the sensitivity parameter.

$$\text{F-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (52)$$

Where:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (53)$$

Confusion Matrix

Confusion matrix is a specific table summarizing the results of the classifier used to visualize the performance of a machine learning and deep learning algorithm. It shows when the model gets confused while predicting the results. Confusion matrix not only specifies the errors made by a classifier but also gives an insight into the type of the errors being made. The term confusion in a confusion matrix or confusion table, determines the

classes that are confused or misclassified as other classes. Generally, the columns of the matrix represent the classifications predicted by the model while the rows represent the instances belonging to each class (see Figure 2.37) [181], [182].

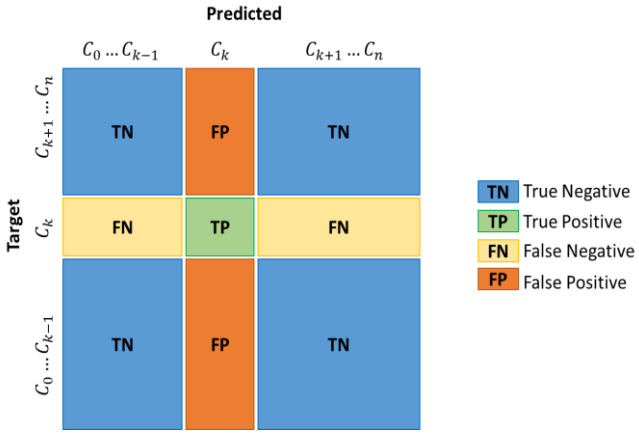

Figure 2.37: Typical representation of a general confusion matrix for multi-class classification. The confusion matrix of a classification with n classes. When considering the class k ($0 \leq k \leq n$), the four different classification results can be obtained: true positive (green), true negative (blue), false positive (orange), and false negative (yellow).

ROC Curve

Receiver Operating Characteristics (ROC) curve graph shows the performance of a classification model. True positive rate (sensitivity) is plotted against the false positive rate (1-specificity) at different classification thresholds. The area under the ROC curve (AUC) gives an index of the performance of the classifier. Higher values of AUC correspond to a good prediction of the model (Figure 2.38)

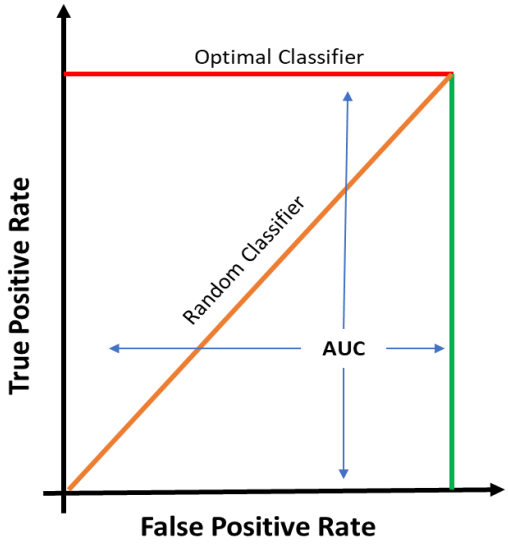

Figure 2.38: basic representation of ROC curve

2.8. Artificial intelligence algorithms for HAR: State of Art

In this section are reported some literature works concerning HAR developed with using *Vision-based* technologies (Chapter 2, paragraph 2.1).
Table 2.2 summarizes all the reported literature works.

Machine Learning

Malekmohamadi et al. compared the results of three different machine learning algorithms (Naïve Bayes (NB), Multi-Layer Perceptron (MLP) and Random Forest (RF)) for identifying 13 possible human daily activities performed in front of a Kinect RGB-D camera using skeletal joints' coordinates. They obtained an average precision value of 84.1 % with NB, 98.7% with MLP and 99.0 % with RF [183]. Alternatively, Akyash et al. proposed a new kernel function based on Dynamic Time Warping for SVM classification of 8 different human postures on two different data sets (TST fall detection dataset and UTD-MHAD dataset). The data of both datasets were collected with the subject positioned in front of the camera. The proposed kernel is applied to each coordinate of every joint. With this method they obtained an overall accuracy classification of 98.8% with the TST fall detection dataset and 98.75% with the UTD-MHAD dataset [184]. Tariq et al. presented an HMM classifier for improving the detection of the assistive activities related to sitting posture. They built a dataset from multiple sensing devices such as Microsoft Kinect and Smartwatches. After collecting the data and labeling them in 10 categories, they analyzed the data through cross-validation of the HMM. They achieved an average accuracy

of 64.88 % [185]. Su et al. suggested a multi-level hierarchical recognition model, using a custom classification algorithm for processing Microsoft Kinect skeletal joints' coordinates. At the first level, they used an SVM classifier and at the second level an HMM algorithm. With this solution they aimed at identifying 20 human actions like bend, hand catch, pick up and throw, etc. using MSRAction3D Dataset, in which the actors are positioned in front of the camera during the acquisitions. They obtained an average recognition rate of 91.41% [186]. In the same vein, Ahad et al. trained a SVM classifier for human activities identification (e.g., walk, sit down, stand up etc.) with kinematics features (3D linear joint positions and angles between bone segments) from a 3D skeletal joints datasets, in which subjects are positioned in front of the camera (UT-Kinect Action 3D, Kinect Activity Recognition Dataset, MSR 3D Action Pairs, Florence 3D and Office Activity Dataset). The number of classes defined varied across 9 to 18, depending on the dataset used. The SVM classifier was trained with a linear kernel function obtaining, for each dataset, the following results in terms of accuracy and precision: 93.91%, 97.51%, 74.78%, 71.58% and 94.92%, respectively [121].

Deep Learning

Liu et al. proposed a model based on a CNN neural network suggesting a unified end-to-end framework called 3D PostureNet. This model was developed by introducing 3D CNN to learn on the Gaussian voxel representation of the skeleton corresponding to a naturally mutual positional relationship of the joints. Three different datasets (MSRA hand gesture dataset, writing posture dataset and Body pose dataset) were used to identify 15 different classes. The overall accuracy was 98.65%, 97.77% and 98.16%, respectively for each dataset[103]. Ahad et al. trained three different deep learning models using temporal statistical features computed through a sliding time window on 3D skeletal joints data from five public datasets and compared their performances with that of the SVM classifier. The first deep model was composed of two LSTM layers, the second one was arranged with one CNN layer followed by a LSTM network and the last model was organized with two CNN networks and a LSTM network for the last layer (ConvRNN). The best model for all the datasets used was the ConvRNN architecture, which obtained accuracies ranging 94.7% and 98.1% [121]. Zhu et al. proposed a new spatial model with end-to-end bidirectional LSTM-CNN (BLSTM-CNN). First, a hierarchical spatial–temporal dependent relational model was used to explore rich spatial–temporal information in the skeleton data. Then, a new framework was implemented to fuse CNN and LSTM. The LSTM was used to extract the temporal features and a standard CNN was used on the output of the LSTM to exploit spatial information. They used two well-known CNN architectures: VGG16 and AlexNET. The proposed models were trained and tested on the NTU RGB+D, SBU Interaction and UTD-MHAD datasets and the number of classification labels

ranged between 8 in the SBU Interaction dataset and 60 for the NTU RGB+D one. In terms of overall accuracy, the BLSTM-CNN implemented with VGG16 provided the best results on the NTU-RGB+D dataset (87.1% and 93.3% in the cross-subject and cross-view benchmarks, respectively) and UTD-MHAD dataset (93.1%), while the AlexNET implementation was the best algorithm on the SBU Interaction dataset with 98.8% [187]. Alternatively, Devanne et al. compared two kinds of temporally hierarchical deep learning models to identifying human activities of daily living through skeletal data captured with a Kinect V2 sensor. The first model was a conventional LSTM architecture with a single LSTM layer, a fully connected layer and a Softmax layer. The second one was similar but used an additional LSTM layer. They decomposed human activity sequences into a set of short temporal segments with the purpose of classifying 21 types of activity (10 human behaviors in a domestic environment and 11 in an office context). They obtained an overall accuracy of 58.9% regarding the domestic environment and 58.5% for the other one [112]. Zhu et al. proposed a deep LSTM network with three bidirectional LSTM layers and two feed-forward layers. The last LSTM layer was a custom-designed LSTM layer including dropout in order to prevent data overfit-ting. They trained and tested the classifier on three different online databases: SBU Kinect Interaction Dataset, HDM05 Dataset and CMU Dataset. Depending on the type of dataset used, they had a total 8, 65 and 45 classes. They obtained an overall accuracy of 90.41%, 97.25% and 81.04%, respectively, for each dataset [188]. Liu et al. proposed a tree-structure based method to explore the kinematic relationship between the skeletal joints. They used these data as input to the first LSTM network layer, whose output was in turn fed to the second LSTM layer and finally a softmax layer. In the two LSTM layers a new gate was added to the LSTM block to handle the noise and occlusion in 3D skeleton data. They trained and tested this model with 5 different online databases (NTU RGB+D Dataset, SBU Inter-action Dataset, UT-Kinect Dataset and MHAD) and obtained an overall accuracy of 77.7%, 93.3%,97.0%, 95.0% and 100%, respectively, for each dataset [189]. On the other hand, Liu et al. proposed a new class of LSTM networks: Global Context-Aware Attention for skeleton-based action recognition, which was capable of selectively focusing on the in-formative Kinect joints in each frame by using a global context memory cell. The model is structured with a first LSTM layer, which encoded the skeleton sequence and generated an initial global context representation for the action sequence, and a second layer that per-formed attention over the inputs by using the global context memory cell. They trained the network on 5 different datasets, i.e, NTU RGB+D, SYSU-3D, UT-Kinect, SBU-Kinect and MHAD and achieved the following results in term of accuracy: 76.1%, 78.6%, 99%, 94.9% and 100%, respectively [190].

Table 2.2. Summary of HAR literature works developed with using *Vision-based* technologies and AI algorithms.

Author	Model	Dataset	Accuracy (%)
Malekmohamadi et al [183].	Naïve Bayes	Custom dataset, data collected by Kinect V2	84.1
	Multi-Layer Perceptron	Custom dataset, data collected by Kinect V2	98.7
	Random Forest	Custom dataset, data collected by Kinect V2	99.0
Akyash et al [184].	Dynamic Time Warping + SVM	Online dataset, data collected by Kinect V2 and IMU system (TST fall detection dataset)	98.8
		Online dataset, data collected by Kinect V1 and IMU system (UTD-MHAD dataset)	98.75
Tariq et al [185].	HMM	Custom dataset data collected by Kinect V2 and Smartwatches	64.88
Su et al [186].	SVM + HMM	Online dataset, data collected by camera depth sensor (MSRAction3D dataset)	91.41
	SVM	Online dataset, data collected by depth sensor (UT-	93.91

Ahad et al [121].		Kinect Action 3D dataset)	
		Online dataset, data collected by Kinect V2 sensor (Kinect Activity Recognition dataset)	97.51
		Online dataset, data collected by depth sensor (MSR 3D Action Pairs dataset)	74.78
		Online dataset, data collected by Kinect V2 sensor (Florence 3D dataset	71.58
		Online dataset, data collected by Kinect V2 sensor (Office Activity dataset)	94.92
Deep Learning			
Author	**Model**	**Dataset**	**Accuracy (%)**
Liu et al [103].	3D PostureNet	Online dataset, data collected by Kinect V2 sensor (MSRA dataset)	98.65
		Custom dataset, data collected from RGB images (Writing posture dataset)	97.77
		Online dataset, data collected by Kinect V2 sensor	98.16

		(Body pose dataset)	
Ahad et al [121].	LSTM	Online dataset, data collected by Kinect V2 (UTKinect dataset)	85.96
	CNN + LSTM		89.47
	ConvRNN		94.73
Ahad et al [121].	LSTM	Online dataset, data collected by Kinect V2 (Kinect Activity Recognition dataset)	96.27
	CNN + LSTM		96.27
	ConvRNN		98.11
Ahad et al [121].	LSTM	Online dataset, data collected by Kinect V2 (MSR 3D Action Pairs dataset)	86.36
	CNN + LSTM		92.42
	ConvRNN		95.45
Ahad et al [121].	LSTM	Online dataset, data collected by Kinect V2 (Florence 3D dataset)	91.66
	ConvRNN		96
Ahad et al [121].	LSTM	Online dataset, data collected by Kinect V2 (Office Activities dataset)	91.25
	CNN + LSTM		97.25
	ConvRNN		88.69
Zhu et al [187].	BLSTM-CNN(VGG16)	Online dataset, data collected by Kinect V2 (NTU RGB+D dataset)	87.1 (cross subject NTU-RGB+D) 93.3 (cross view

			NTU-RGB+D)
Zhu et al [187].	BLSTM-CNN(VGG16)	Online dataset, data collected by Kinect V2 (SBU Interaction dataset)	98.8
Zhu et al [187].	BLSTM-CNN(VGG16)	Online dataset, data collected by Kinect V2 (UTD-MHAD Interaction dataset)	93.1
Zhu et al [187].	BLSTM-CNN(AlexNet)	Online dataset, data collected by Kinect V2 (NTU RGB+D dataset)	87.1 (cross subject NTU-RGB+D) 93.3 (cross view NTU-RGB+D)
Zhu et al [187].	BLSTM-CNN(AlexNet)	Online dataset, data collected by Kinect V2 (SBU Interaction dataset)	98.8
Zhu et al [187].	BLSTM-CNN(AlexNet)	Online dataset, data collected by Kinect V2 (UTD-MHAD Interaction dataset)	93.1
Devanne et al [112].	LSTM	Online dataset, data collected by Kinect V2 (Domestic environment dataset)	58.9

	ConvLSTM	Online dataset, data collected by Kinect V2 (Office environment dataset)	58.5
Zhu et al [188].	LSTM	Online dataset, data collected by Kinect V2 (SBU Kinect Interaction dataset)	90.41
		Online dataset, data collected by Kinect V2 (HDM05 dataset)	97.25
		Online dataset, data collected by Kinect V2 (CMU dataset)	81.04
Liu et al [189].	LSTM	Online dataset, data collected by Kinect V2 (NTU RGB+D dataset)	77.7
		Online dataset, data collected by Kinect V2 (SBU Interaction dataset)	93.3
		Online dataset, data collected by Kinect V2 (UT-Kinect dataset)	95.0 (Half protocol) 97% (Leave one out protocol)
		Online dataset, data collected by Kinect V2 (MHAD dataset)	100

Liu et al [190].	LSTM + Attention	Online dataset, data collected by Kinect V2 (NTU RGB+D dataset)	76.1
		Online dataset, data collected by Kinect V2 (SYSU-3D dataset)	78.6
		Online dataset, data collected by Kinect V2 (UT-Kinect dataset)	99
		Online dataset, data collected by Kinect V2 (SBU-Interaction dataset)	94.9
		Online dataset, data collected by Kinect V2 (MHAD dataset)	100

Chapter 3

Case study: Human Activity Recognition in Ambient Assisted Living

3.1. The Project

Due to the increase in the global aging population (Figure 3.1), its associated age-related challenges, such as reduced walking speed, mobility, falls, fatigue, difficulties in performing daily activities, memory-related and social isolation issues are becoming increasingly prominent for our public health systems [191], [192].

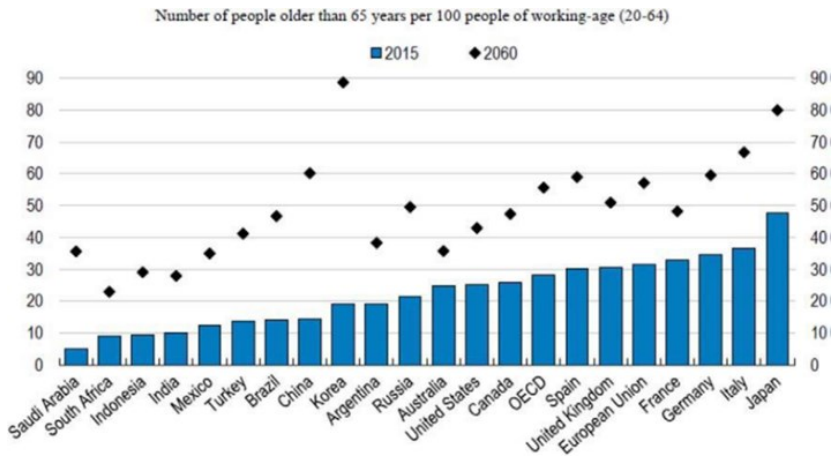

Figure 3.1: elderly people population growth by 2060 [192].

Moreover, the recent COVID-19 pandemic has stressed this situation even further, thus highlighting the need for taking action. AAL technologies come as a viable approach to help facing these challenges, thanks to the high potential they have in enabling remote care and support [4]. AAL systems are designed to provide support in daily life in an unobtrusive and user-friendly manner. Besides, this environment is conceived to be smart, to be able to learn and adapt to the requirements and requests of the assisted

people, and to synchronize with their specific needs. Nevertheless, to ensure the uptake of AAL in society, potential users must be willing to use AAL applications and to integrate them in their daily environments and lives. *Vision-based* AAL applications have several advantages, in terms of unobtrusiveness and information richness. Indeed, cameras are far less obtrusive with respect to the burden other wearable sensors may cause to one's activities [65]. The architecture of a *Vision-based* AAL solution traditionally consists of a set of cameras for data capture connected to a server, processing modules for data analysis, up to an alert/decision module which may or may not include a human operator [4], [65], [193]. Nevertheless, cameras are often perceived as the most intrusive technologies in terms of the privacy of the monitored individuals. The solution to this drawback may be RGB-D cameras, like the Kinect V2, which are able to extract the "skeleton" of the subject from the depth image, i.e., represent the subject as a set of body segments and joints, and bypass the need for using the traditional camera image for AAL purposes. These tools for skeletal tracking increase the person's acceptance towards the assistive technology, since it ensures the privacy preservation [65]. Following this concept, we propose AI solutions developed with the aim to identify specific postures assumed by a person in a home environment, in a daily basis scenario.

Moreover, we decided to focus around the three simple postures most frequently assumed by a person in a room during daily activities: standing, sitting, and lying down. In addition to the listed postures, we added one further posture, labeled dangerous sitting, which grouped all situations of malaise or fainting resulting in a seated person slumped or lying backward. This latter allowed us to perform a first distinction between routine activities and alarm situations, prior to making a decision on whether it is necessary to rise an alarm.

Indeed, the information about the identified posture constitutes one of the inputs to a more complex decision system that integrates and analyzes the data coming from a network of environmental sensors in order to distinguish a scenario of daily life from a potentially dangerous situation (for example, a person lying in bed → probable everyday life situation; person lying on the ground → potentially dangerous situation).

The input data to AI models for HAR are acquired with four Kinect V2 devices, arranged in the room according to a configuration that allows monitoring the largest possible room area. Furthermore, starting from the joint coordinates data, referred to a global reference system common to the four Kinect V2, a set of joint angles and the pitch and roll angles of the head and trunk are calculated.

3.2. Experimental set-up

The current implementation is based on Microsoft's Kinect V2 motion sensing system, previously detail described in Chapter 2, paragraph 2.1.1.

Experimental acquisitions were performed in a prototype room, that was set up in the Bioengineering laboratory of University of Pavia. In this setting, we decided to record each experimental trial using four Kinect V2 devices (**K1**, **K2**, **K3**, and **K4** in Figure 3.2). Two devices were positioned to sense the whole room (**K1** and **K4**), while the remaining two were placed to specifically acquire two areas of the room, such as the bed (**K2**) and the desk (**K3**). This decision was made after several careful eye-inspections of the different shots obtained with several camera configurations, different for devices number, position and orientation. The goal was to ensure the recording of the entire room, minimizing possible blind spots. The data of the four Kinect V2 were acquired at the same time but processed separately. A custom-made C#-based tool with GUI was developed using VisualStudio 2017 to control the Kinect V2 acquisitions.

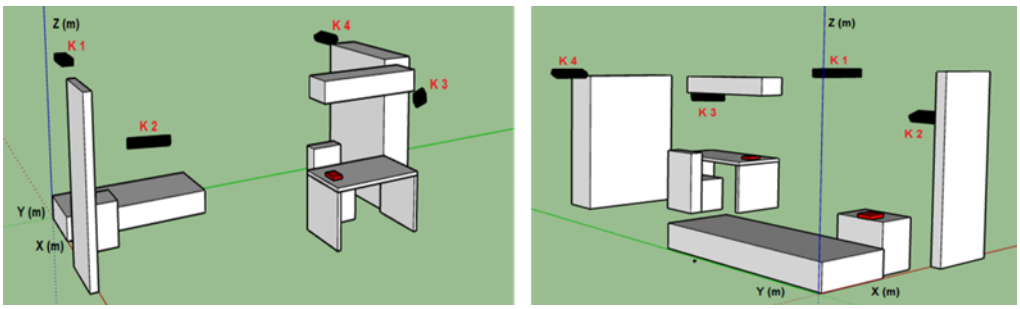

Figure 3.2: positions in the prototype room (**K1**, **K2**, **K3**, and **K4**), reconstructed with a CAD software (SketchUp). Two different fields of view.

3.3. Subjects

In order to build a dataset suitable for training the AI solutions, we performed a set of experimental acquisitions on a group of 12 normal subjects (7 females and 5 males; age ranging 25 and 60 years old; height ranging 1.55 and 1.90 m). All subjects gave written informed consent in accordance with the Declaration of Helsinki.

3.4. Acquisition Protocol

As previously explained, the development of the intended monitoring system required a custom database of ecological skeleton data relative to subjects freely moving in the surveillance room for recognizing, in each data frame, the posture of the subject. No specific orientation of the subjects with respect to the Kinect systems was, therefore, required and held during data acquisition. We chose to separately classify individual frames in order to feed it to the mentioned multifactorial decision system taking into account the position of the subject in the room and relative to the furniture and data from other sensors for deciding whether to trigger an alarm. We recorded a

total of 265 trials of about 13 min each. In each trial, subjects were asked to perform an ordered sequence of postures (standing, sitting, lying, and slumping in a chair with the head leaned forward or backward), transitioning from 1 posture, lasting 10 seconds, to the following one without breaks. In more detail each subject performed the following acquisition protocol consisting of four recording sequences:

- the subject starts to walk from standing position in front of **K1** (Figure 3.2), then grabs a chair near the desk, placing it in front of the camera, and finally sits on it. While sitting, the subject first moves the head backward and then leans the trunk forward, while simultaneously pitching the head as an unconscious person. The subject then returns to the normal sitting position and finally gets up and brings the chair back to its original location. Each posture is maintained for 10 seconds. The sequence was then repeated in front of the other cameras (**K2**, **K3**, **K4** in Figure 3.2);
- the subject starts sitting on the bed, then lies down on the back, turns on the right side, then returns on the back and turns to the left side;
- the subject starts lying on the ground on the back, then turns on the left side;
- the subject starts sitting on the bed, then lies down. The action is repeated three times.

The four Kinect V2 devices were placed in different positions in the experimental room (Figure 3.2). The sequences of postures assumed by the subject are recorded simultaneously by each of the Kinect V2, thus obtaining four acquisitions with a different point of view of the same scene. In this way it is possible to collect more data, yet different quality, from the same recorded scene, avoiding blind spots to the cameras and covering as much of the experimental room as possible. From preliminary experimental trials we observed that the data quality is strongly dependent on the camera viewing angle, the distance and the orientation (frontal and profile view) of the subject respect to the Kinect V2 camera. The four acquisitions are synchronized with each other but processed independently. During the acquisitions, the sequences of postures are timed by the operator experimenting.

3.5. Data Analysis

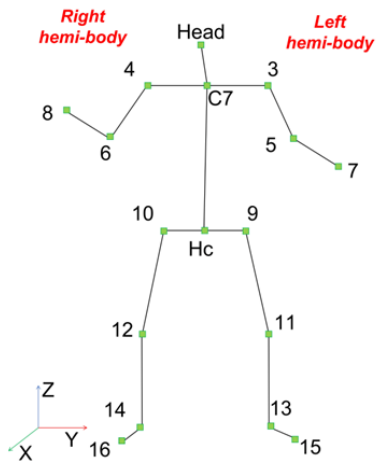

Figure 3.3: skeleton with 17 joints.

Using custom developed software based on the Kinect's SDK we computed the spatial coordinates (x, y, z) of the standardized 25 skeletal joints (Chapter 2, paragraph 2.2.1, Figure 2.14). Moreover, in order to identify the position of the subject in the room, the coordinates of the 25 joints were roto-translated to obtain data referred to an absolute reference system (X, Y, Z) located in one corner of the mock-up room (Figure 3.2).

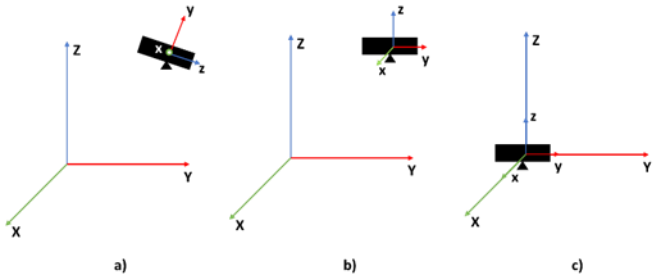

Figure 3.4: Switching from Kinect V2 relative system (x, y, z) to absolute room system (X, Y, Z).

More precisely, the rotation matrix *R* employed to compute the rotation, showed in Figure 3.4 (panel a) was:

$$R = \begin{bmatrix} c_1c_3 + s_1s_2s_3 & c_2s_3 & c_1s_2s_3 - c_3s_1 \\ c_3s_1s_2 - c_1s_3 & c_2c_3 & c_1c_3s_2 + s_1s_3 \\ c_2s_1 & -s_2 & c_1c_2 \end{bmatrix} \quad (54)$$

Where (54 and Figure 3.5):

- α corresponds to the angle between the x axis and the N axis;
- β corresponds to the angle between the z axis and the Z axis;
- Υ corresponds to the angle between the N axis and the x axis;
- $c_1 = \cos(\alpha)$;
- $c_2 = \cos(\beta)$;
- $c_3 = \cos(\Upsilon)$;
- $s_1 = \sin(\alpha)$;
- $s_2 = \sin(\beta)$;
- $s_3 = \sin(\Upsilon)$;

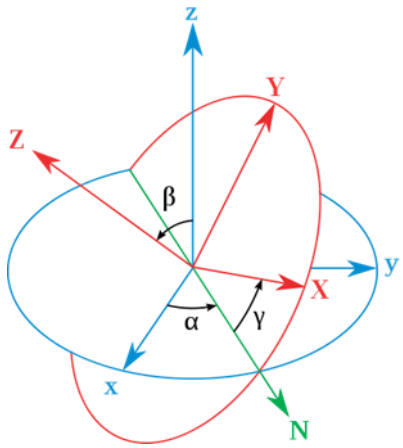

Figure 3.5: rotation applied to the Kinect V2 relative system. The blue lines are the fixed coordinate system (x, y, z), the red lines are the rotated coordinate system (X, Y, Z) and the green line is the line of nodes (N).

Therefore:

$$\begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} = R \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (55)$$

Once the rotation R (55) is performed, the translation $t = [t_x \quad t_y \quad t_z]^T$, as shown in Figure 3.4 (panel b), is calculated. Where t_x, t_y, t_z corresponds to the translation coordinates of the Kinect V2 camera with respect to the reference system $[X \quad Y \quad Z]^T$.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \qquad (56)$$

During the acquisition process we noted that sometimes the Kinect V2 was not able to recognize the subject. For example, transient exits of the subject from the camera sight (Figure 3.2) could cause temporary non-

identifications of all skeletal joints, and the same may occur when the subject assumes a dangerous sitting posture while not facing the camera. This can generate temporal holes between data frames (missing data). For these frames we decided to assign the value '999' to all the selected parameters, in order to maintain temporal consistency among the data of the four Kinect V2 systems. All the preprocessing algorithms were implemented using MATLAB.

3.6. Kinematic attributes definition

Considering the reliability of the detected joints, the aim of this study and the kind of posture we would like to identify, we decided to reduce the number of skeletal joints from 25 to 16. The Kinect V2 joints for the neck, the 'SpineMid' and hands (Hand Left/Right, Hand Tip Left/Right and Thumb Left/Right, (see Chapter 2, Section 2.2.1, Figure 2.4) represented superfluous information in the recognition of standing, sitting, lying and dangerous sitting postures. They were therefore discarded in the analysis of the collected data. Moreover, the 'SpineBase' Kinect V2 joint (Figure 2.4) was substituted with an additional joint, labeled Hc, computed as the midpoint between the two hip joints.

The absolute position in space of each body joint, described by the corresponding (X, Y, Z) triplet, may not be the most convenient description for classifying human postures since: (1) coordinates depend on the relative location of the individual in the room, while the same posture can be taken in different locations within the room; (2) the joint coordinates of two subjects having the same posture in the same room location have different values depending on the size of the subject's body; and (3) posture is independent of where it occurs in space while it is defined by the geometrical relationship between the different body segments. The latter can instead be efficiently captured by articular angles [121], [126], so that we chose to compute the following 16 articular angles defined between two consecutive body segments measured in the plane defined by the segments themselves (Table 3.1). Based on the same line of reasoning we further computed the roll and pitch angles of the head and trunk (Table 3.1) of each subject.

Table 3.1: joint angles between two body segments computed in space and absolute joint angles calculated between a body segment and the horizontal plane passing through the two hips and the two shoulders

Angles	Description
μ_1, μ_2	Angle between head and shoulder segments. Left and right, respectively
ξ	Angle between head and trunk segments
τ_1, τ_2	Angle between trunk and shoulder segments. Left and right, respectively
η_1, η_2	Angle between shoulder and arm segments. Left and right, respectively
θ_1, θ_2	Angle between arm and forearm segments. Left and right, respectively
δ_1, δ_2	Angle between trunk and hip segments. Left and right, respectively
Υ_1, Υ_2	Angle between hip and thigh segments. Left and right, respectively
β_1, β_2	Angle between thigh and leg segments. Left and right, respectively
α_1, α_2	Angle between leg and foot segments. Left and right, respectively
A_{roll}	Head roll angle
A_{pitch}	Head pitch angle
B_{roll}	Trunk roll angle
B_{pitch}	Trunk pitch angle

Finally, with the 51 coordinates of each roto-translated joint, the 16 relative angles and the 4 absolute angles, we obtain a total of 71 kinematic features for describing the collected data.

Chapter 4

Artificial Intelligence solutions

In this chapter, we present the several developed solutions for the HAR skeleton-based system aimed at identifying the human postures and, possibly, dangerous situations. Each one of the AI algorithms employed is characterized by its proper data analysis, feature computation and selection. The aim is to develop an optimized monitoring system, operating in real time.

4.1. First solution proposed: Automatic posture recognition for monitoring dangerous situations in Ambient-Assisted Living

This first solution [194] starts from the data collected with the acquisition protocol described in the previous paragraph (Chapter 3, paragraph 3.4) and proposed a machine learning algorithm, i.e., a MLP neural network, which identified frame-by-frame the posture assumed by the monitored subject.

4.1.1. Data preprocessing

After computing the kinematic attributes (Chapter 3, paragraph 3.6) for the data preprocessing phase, we further considered the 17 vertical coordinates (Z) of the skeletal joints as they are significant for distinguishing the lying down from the standing posture, due to the height difference between the two postures. On the other hand, these are not sufficient for distinguishing sitting and dangerous sitting postures, which are more easily identified through the absolute and relative angles defined between body segments. According to these considerations, we selected a total of 37 attributes (17 vertical joints coordinates, 16 relative angles, 4 absolute angles) for describing the subject in each frame. Moreover, in order to set all the attributes in the same range, they were normalized, scaling them with different values. All angles were normalized dividing them by 180° and all the 17 joints' Z coordinates were scaled on the height of each subject.

Additionally, given the static nature of the chose classification network, all missing data (temporal holes between data frames), filled with the value 999, were not considered as input to the AI model and were temporally deleted from the collected data. Moreover, in this first proposed solution, we also did not consider the frames corresponding to the transitions from a posture to another (i.e., between a sitting posture to lying down posture and vice-versa or between a standing posture and sitting posture and vice-versa), since in this first phase we wanted to concentrate only in the static postures (standing, sitting, lying down and dangerous sitting).

4.1.2. Feature Selection

Following the preprocessing of the 37 features, we applied a filter method approach for the feature selection called ReliefF [195]. The core idea behind ReliefF algorithms is to estimate the quality of attributes on the basis of how well each attribute can distinguish between instances that are near to each other. This process selected a subset of ten features: A_{pitch}, A_{roll}, B_{pitch}, B_{roll}, ξ, $\mu 2$, $\delta 2$, Z_1, Z_{C7}, Z_{Hc} (Table 3.1 and Figure 3.3) as the most relevant ones.

4.1.3. Dataset construction

Since we chose a supervised feed forward neural network, we labeled each acquired frame with the corresponding class, using a custom-made software implemented in LabView. Angles and joints position traces were then visually inspected together with a graphical visualization of the reconstructed skeleton to label each frame with one of the following four postures:

- Class 1: standing posture;
- Class 2: sitting posture;
- Class 3: lying down posture;
- Class 4: dangerous sitting posture.

The final database consisted in a total of 602,530 frames. This was subdivided into training (built using the data from 10 of the 12 subjects) and testing dataset (built using the data of the two remaining subjects), using the cross-subject splitting database method (Chapter 2, paragraph 2.5), (Table 4.1).

Table 4.1: the numerosity of frames in the training and testing dataset

Classes	Dataset	
	Training	**Testing**
Class 1	120,059	21,267
Class 2	188,463	40,591
Class 3	75,028	24,165
Class 4	112,178	20,779
Total	495,728	106,802

4.1.4. Pattern Recognition

The aim of this work was to identify the subject lying on the floor immediately after the fall in order to activate an alarm and intervene with first aid actions. Therefore, in the current implementation we wanted to identify a subject posture at any time, leaving the decision-making process about alarm triggering to a downstream algorithm having access to more data (e.g., subject's position in the room). The posture classification problem is therefore seen as a static mapping problem. For this reason, among a range of possible Machine Learning algorithms, we chose an MLP Neural Network to classify predefined human postures.

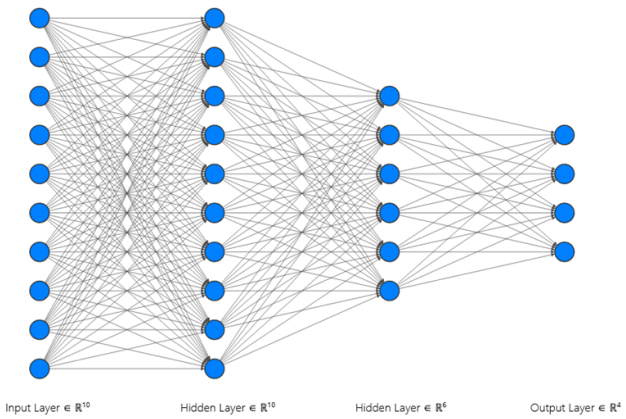

Figure 4.1: MLP proposed model.

The MLP network (Figure 4.1) was implemented in MATLAB using the Neural Network Toolbox. We designed a model consisting of three fully connected layers of neurons, plus an input layer connected to the 10 features describing each frame in the database (Chapter 3, Table 3.1 and Figure 3.3). The first hidden layer had a number of neurons equal to the number of attributes in the database (10), each with a hyperbolic tangent transfer function and a bias. The second hidden layer had a structure similar to the first one but contains a smaller number of neural units (6). The output layer was instead composed by a number of neurons equal to the number of target classes (4) and their transfer function was the softmax function producing, for each input element, the probabilities of belonging to each considered class. The MLP network was trained using the backpropagation algorithm (Chapter 2, paragraph 2.6.1), first with a k-fold cross validation and then using the whole training set. In the cross-validation algorithm the training set is split into k smaller sets (in this case k=10). After the splitting, for each of the k folds follows we implemented the following procedure (Figure 4.2):

1. the MLP model was trained using $k - 1$ of the folds as training data;
2. the resulting model was validated on the remaining part of the data, which was used as a test set to compute performance measures, such as accuracy.

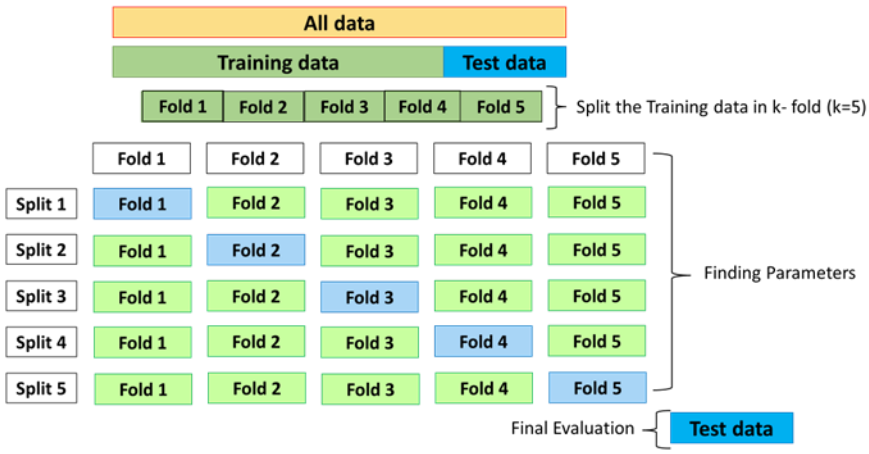

Figure 4.2: example of how cross validation algorithm woks. In this example k is set equal to 5, in our work k=10.

The performance measure returned by k-fold cross-validation was then the average of the values computed in the loop

The MLP learning process was performed over a maximum of 1000 epochs, i.e., 1000 iterations on the training set.

4.1.5. Statistical Analysis

MLP network was trained and tested 50 times to study its classification robustness. Total accuracy (mean over the four classes), class accuracy, F-score, sensitivity, and specificity were calculated for each network simulation.

For each of the five parameters considered, the mean value over the 50 network simulations was then computed. Such averaging was performed only after verifying that the results were normally distributed. Since the number of samples was 50, i.e., the number of network simulations, we used the Shapiro–Wilk test as a hypothesis test. For each test performed the p-value was greater than the chosen alpha level (0.05), therefore the null hypothesis that the data came from a normally distributed population cannot be rejected (IBM SPSS Statistics, IBM). Therefore, for each of the five parameters, the mean and the standard deviation were considered.

We also computed a confusion matrix for each of the 50 network simulations. Then, we calculated a mean confusion matrix, normalizing all the cells of the matrix for the frame cardinality of each class. We further computed, for each class, the ROC curve graph for each of the 50 network simulations. Then, to compute a mean ROC curve, we averaged the ROC curve of the 50 simulations as the mean true positive rate for each value of the false positive rate considered on the abscissa.

4.1.6. Results

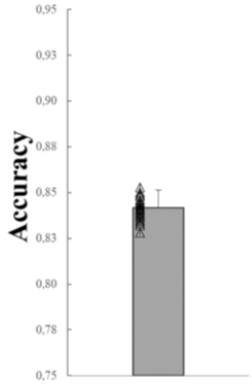

Figure 4.3: Mean, SD of the mean total accuracy obtained over the 50 network simulations (black empty triangles).

In Figure 4.3 the mean value (Mean), the corresponding standard deviation (SD) and the distribution of the 50 mean total accuracy values, each corresponding to one of the 50 network simulations (0.839 ± 0.0073) are shown. Values range 0.852–0.820.

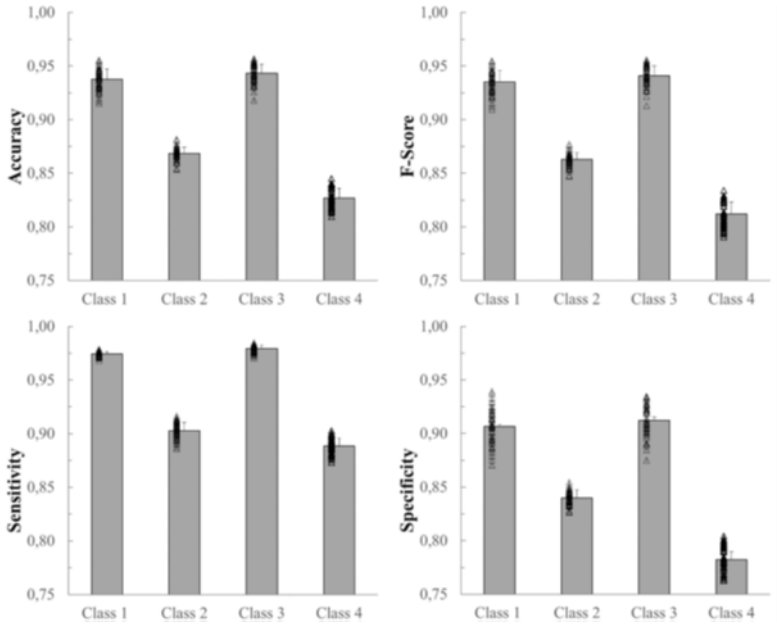

Figure 4.4: Mean, SD and individual results (black empty triangles) of the 50 network simulations. From top left: accuracy, F-score, sensitivity, and specificity for each on the 4 classes.

Figure 4.4 shows mean values, SD, and the distributions of the accuracy, F-score, sensitivity, and specificity of each of the four classes. All four variables represented in Figure 4.4 show a similar trend. Class 3, which corresponds to the lying posture, and Class 1, which corresponds to the standing posture, represent the classes which were best identified by the network. The network, on the other hand, classified Class 2 (sitting posture) and, especially, Class 4 (dangerous sitting posture) with more difficulty, as shown by each of the four variables calculated (Figure 4.4).

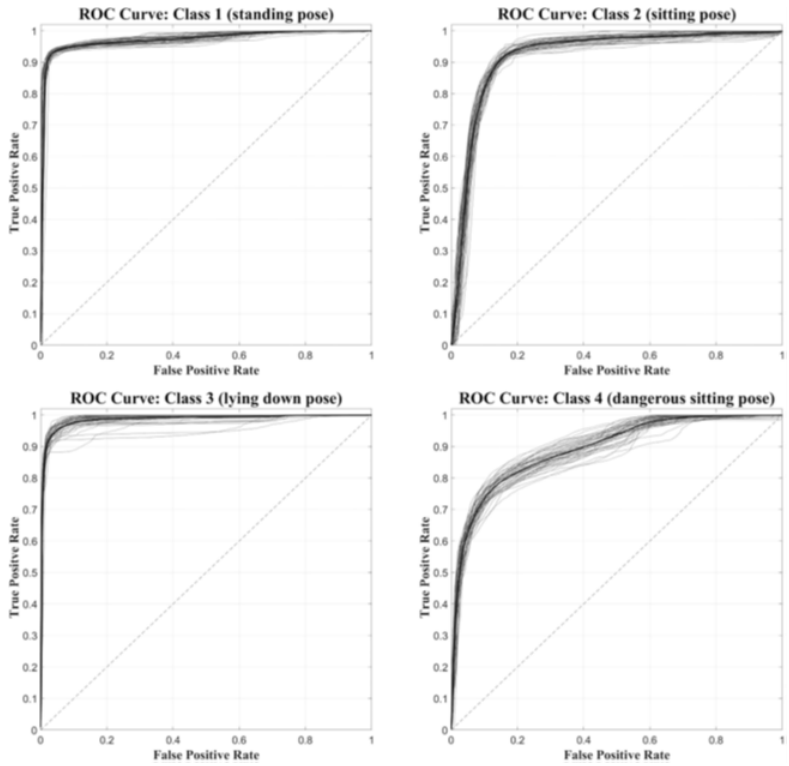

Figure 4.5: Set of 50 ROC curves calculated for each of the four classes (gray traces). The average ROC curve (black trace) was also calculated and superimposed.

Target \ Output	1	2	3	4
1	92.61%	4.83%	0.03%	2.51%
2	5.68%	87.12%	0.05%	7.14%
3	0.05%	1.06%	88.00%	10.87%
4	1.07%	27.06%	5.51%	66.36%

Figure 4.6: Mean confusion matrix obtained from the 50 network simulations. It represents, for each row, the percentage of False Positive (FP) and True Positive (TP) classifications.

Figure 4.5 shows, for each class, the ROC curves calculated on the 50 network simulations. The average ROC curve is also shown for each of the four classes. The average ROC curves confirm the observations made previously, i.e., that Class 1 and Class 3 are better identified by the neural network than Class 2 and Class 4. The same results are confirmed also by computing the AUC values for the average ROC curves of the four classes (97.2 for Class1, 92.1 for Class2, 98.5 for Class3 and 89.2 for Class4). Figure 4.5 also shows the greater variability of the ROC curves relative to Class 4, compared to those obtained with Class 1, Class 2, and Class 3. Figure 4.6 shows the mean confusion matrix computed over the entire set of 50 network simulations performed. It summarizes on each row the average percentage of the False Positives (FP) and True Positives (TP) for the corresponding class.

4.1.7. Discussion

Previous studies had faced similar problems using ML algorithms with good results, although on smaller datasets and asking the subject to maintain the planned postures while facing the camera, i.e., a very favorable condition for the Kinect acquisition, yet unlikely in our project scenario [123], [130]. Our study considered a less constrained dataset in which 12 subjects were recorded in the defined postures both statically (e.g., lying down) and while moving over the entire room area (e.g., the subjects were walking when assuming the standing posture) for a total of 495,728 frames for training and 106,802 frames for testing. As a result, our data was more variable in terms of how each subject interpreted the requested postures, and noisier for the different views recorded by each of the four Kinect One devices, which were necessarily frequently sub-optimal. In spite of these limitations, required to mimic real life conditions, the proposed MLP classifier achieved good results with a total average accuracy of 83.9%. A more detailed inspection of the results relative to the four classes shows that Class 3 and Class 1 are better recognized than the remaining two classes, with average accuracies around 94% (94.3 and 93.8%, respectively). On the other hand, Class 2 and Class 4, both regarding sitting positions yet differing mostly in terms of trunk and head pitch angles, were less accurately recognized (86.9 and 82.7%, respectively), with frames being incorrectly assigned to the two classes (see accuracy values in Figure 4.4). These lower accuracy values are mainly due to the misclassification errors between Class 2 and Class 4 and vice-versa. Indeed the 6.01% of frames labeled as Class 2 were identified as Class 4 and the 2.91% of Class 4 data were classified as Class 2 (see the mean confusion matrix in Figure 4.6). At least two plausible reasons could be considered as contributing to this misclassification error in recognizing these two postures. First and foremost, during sitting some articular joints were covered by other body parts, thereby requiring the Kinect V2 system to reconstruct the positions of the hidden joints and making the resulting data very noisy. Second, despite the careful choice of features as powerful descriptors of body postures while being independent from the physical characteristics of the

subjects who participated to the study, the distinction between two kinematically very similar postures was very difficult. The number of features that could help the classifier to distinguish between them was reduced. Only the upper body features could be discriminative and probably, even among these, the normalized vertical positions of the head and cervical vertebrae (Z_1, Z_{C7}), i.e., the most discriminative joint-related features for the identification of Class 1, 2 and 3, sometimes take comparable values between Class 2 and 4 due to the subjects' individual interpretation of the description of the dangerous sitting posture. Another relatively important misclassification error was between Class 1 and Class 2 and vice-versa (2.24% Class 1 identified as Class 2 and 1.15% Class 2 identified as Class 1).

For the identification of these two postures, the vertical position of the joints (Z_1, $Z_{C7}$and Z_{Hc}) should be more informative for the MLP network. Nevertheless, in our study this was not so evident, probably because some of the data calculated by Kinect V2 devices were particularly noisy, especially when the subject is not exactly in front of the camera [196], [197]. The relative angles and the head and trunk absolute angles did not weight as much in the distinction between the two classes since they assume comparable values. Conversely, lower misclassification error was found for the standing posture (Class 1) and the dangerous sitting posture (Class 4) and vice-versa (0.25% Class 1 identified as Class 4 and 0.62% Class 4 identified as Class 1, respectively). In this case, the relative and absolute angles of head and trunk features in the database were more discriminants. The lowest misclassification error, almost equal to zero, was that between the identification of standing (Class 1) and lying down (Class 3) postures and vice-versa, where the vertical position of the joints is very discriminative.

Considering the assumptions made so far, in order to explain the misclassification errors, we can hypothesize that an appropriate preprocessing of the data could significantly reduce the number of misclassifications. A classification model requires a reliable and valid dataset to efficiently generate the decision-making rules. To reduce classification errors, the quality of the data provided to the classifier is important during both the training and the usage phases, so that data pre-processing techniques removing anatomically implausible body reconstructions resulting in longer than real limbs or in impossible articular angles may be needed.

4.2. Second solution: Skeleton data pre-processing for human posture recognition using Neural Network

In a second proposed solution [198] we defined a Kinect V2 skeleton data preprocessing algorithm to partially overcome the limitations of the device mainly coming from its use in suboptimal conditions of viewing angles. As a matter of fact, the Kinect V2 has some drawbacks in the acquisition and processing of data, such as: 1) variable frame intervals [199] 2) missing data when the subject is at the edge of the camera's calibrated volume or is partially hidden by furniture in the scene (es. desk, bed and chair) [145]; 3) incorrect reconstruction of joints' positions (due to noisy data, room lighting or overlapping of two or more joints) (Figure 4.7, panel a and panel c and Figure 4.8) [196], [200]–[203]; 4) recognition of 'ghost' skeletons caused by moving objects (i.e. chair) [145] (Figure 12, panel b). This preprocessing procedure hopefully can improve the performance of the MLP classifier illustrated in previous section.

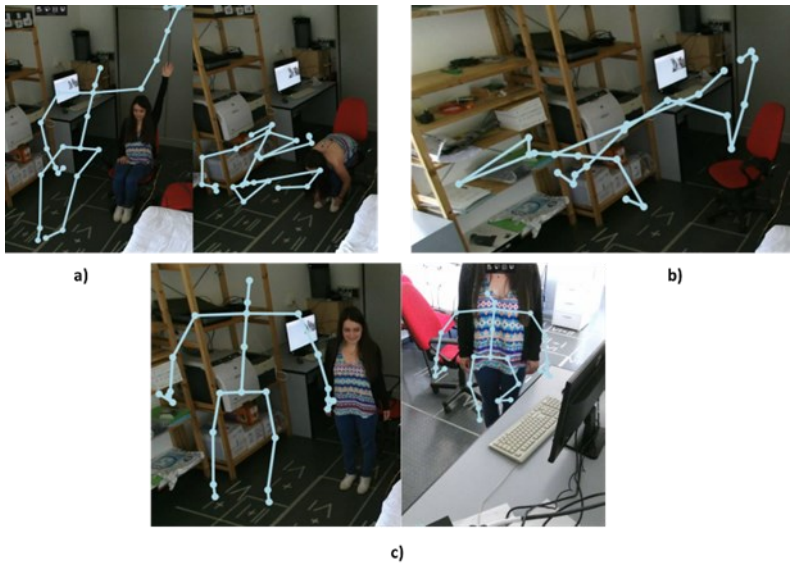

Figure 4.7: some examples of Kinect V2 typical errors. Panel a: example of acquisition of a subject sitting in front of the camera (left) and of a slumped subject (right). Panel b: reconstruction of the skeleton of an inanimate object. Panel c: example of a subject standing in front of the camera (left) and a subject standing covered by the desk (right).

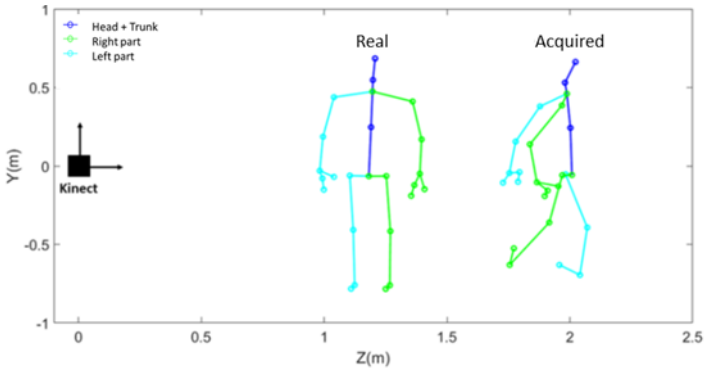

Figure 4.8: projection on the yz (Kinect V2 coordinate system) plane of the skeleton obtained from the data of a Kinect acquisition area (Acquired) and the one obtained from a simulation in which the position of the joints has been correctly estimated (Real).

4.2.1. Data preprocessing

In the first phase of this data preprocessing, we followed the same steps employed in the previous solution (Chapter 4, paragraph 4.1.1). Moreover, we proposed two preprocessing algorithms, both based on several thresholding procedures aimed at removing implausible data, but differing for the addition, in the second one, of: a) a linear fitting method aimed at approximating the missing or over-threshold data; b) an additional data averaging over a temporal sequence of 15 frames (corresponding to 0.5 seconds).

In the first preprocessing algorithm, raw data were averaged over temporal windows of 15 frames and a threshold corresponding to the mean ± 3 Standard Deviations (SD) was applied to detect and remove outliers. All data overcoming such threshold were removed. If a time window contained more than 30% of missing data (999 value), the data of the considered time window were deleted. After such data cleaning procedure, body segments' length was computed for each pair of consecutive joints and then compared with the corresponding anthropometric value for a normal subject [204]. Since the joint positions, calculated by the anthropometric reference model, did not exactly correspond to those processed by Kinect V2, a tolerance threshold of 40% was considered. This, as well as the following controls, was carried out only on the subset of joints considered more accurate [205], namely: head (1), C7 (2), acromion (3 - 4), iliac crest (9 - 10), Hc (17) (Chapter 3, Figure 3.3). In addition, a tolerance of only 30% on the variability of each body segment length between consecutive frames was considered acceptable. Finally, the velocity of joint movements between two successive frames was taken into account. Assuming that the velocity of a subject during a natural walking pace ranges between 4 and 5 km/h, the threshold was set to 6.48 km/h (displacement of 6 cm between two frames). If one of the comparisons described above did not meet the threshold condition, the data coordinates of all joint, referring to that frame, were

removed. Since the purpose of this preprocessing algorithm was to provide reliable data as input to a neural MLP network, the choice of all the threshold values was made to ensure a good compromise between quality and quantity of data.

In the second preprocessing algorithm, to reduce the temporal discontinuity of the raw data, we performed an approximation of the missing data by a linear fitting. The values of the 4 frames preceding and following the missing data were used for the approximation, which was made only if at least five valid values were found around the missing one. Alternatively, no replacement was made. Then, the processed data were analyzed with the same mean ± 3 SD threshold procedure described in the first preprocessing algorithm and the removed samples were fitted again with the linear procedure described above. Finally, data were further handled with an averaging procedure over a time window of 15 frames. This process limits the frequency of the MLP classification at 2 Hz, but we considered that it may be enough to recognize dangerous situations in the AAL domain. The following steps of this preprocessing algorithm replicate the phases of the first algorithm proposed.

4.2.2. Feature selection

In this Feature Selection phase, we maintained the same features selected in the previous section: A_{pitch}, A_{roll}, B_{pitch}, B_{roll}, ξ , $\mu 2, \delta 2$, Z_1, Z_{C7}, Z_{Hc} (Chapter 3, Table 3.1 and Figure 3.3).

4.2.3. Dataset construction

This work was designed and developed as that previously illustrated. The data considered was acquired on 11 subjects.

The data were subdivided and labelled in four classes, following the same principles illustrated in the first solution:

- Class 1: standing posture;
- Class 2: sitting posture;
- Class 3: lying down posture;
- Class 4: dangerous sitting posture.

The final raw database ass composed by a total of 434,264 frames. Also, in this solution the dataset was split following the cross-subject method (Chapter 2, paragraph 2.5). On the other hand, differently from the first proposed solution, the number of the subjects which compose the training and the testing dataset was changed. More precisely the dataset was subdivided into training dataset (built using the data from seven of the 11 subjects) and testing dataset (built using the data of the four remaining

subjects), (Table 4.2). This choice was made to observe whether the performance of the neural network model varied with the change in the number of subjects in training and testing.

Table 4.2 the numerosity of testing and training dataset.

Classes	**Dataset**	
	Training	**Testing**
Class 1	120,059	21,267
Class 2	188,463	40,591
Class 3	75,028	24,165
Class 4	112,178	20,779
Total	282,820	151,444

Starting from the dataset of Table 4.2, we then applied the two data preprocessing algorithms. The two algorithms preprocessed the data in the following ways (Table 4.3):

- The first algorithm preprocessed 77,562 frames;
- The second algorithm preprocessed 24,484 frames.

Table 4.3: Subdivision of training and testing dataset, after the application of the first and/or second Data Preprocessing algorithm.

Case	**Training**	**Testing**	**# Training frames**	**# Testing frames**
Case A	Raw data	Raw data	282,820	151,444
Case B	First Algorithm	First Algorithm	49,530	28,032
Case C	Second Algorithm	Second Algorithm	15,664	8,820
Case D	First Algorithm	Second Algorithm	49,530	8,820

4.2.4. Pattern Recognition

The machine learning algorithm employed in this work was the same MLP model presented for the previous solution, with the same architecture and hyperparameters (Figure 4.1). We trained and tested the model with k-cross validation (k = 10), see Figure 4.2, and then using the whole training set. The learning process was performed over a maximum of 1000 epochs, i.e., 1000 iterations on the training set.

4.2.5. Statistical Analysis

Precision, Sensitivity, Specificity and F-score were calculated for each fold and then the same parameters were computed over the 10 folds (mean values). The trained MLP network was then tested over the whole test database and the accuracy was computed (Table 4.3). Considering these results, we further investigated the data by computing class Precision, Sensitivity, Specificity and F-score in the **Case A** (control scenario) and **Case D** (best accuracy scenario).

4.2.6. Results

Figure 4.9 shows an example of the raw head joint vertical position (upper panel), characterized by noise and temporal holes (missing data), and the effects of the pre-processing algorithms on these data (middle and bottom panel). The first pre-processing erases the noisy data which do not satisfy the criteria defined in the algorithm. All the three coordinates of each joint are analyzed, and it is sufficient that a single data coordinate exceeds the threshold to determine the deletion of the whole frame. For this reason, as shown in Figure 4.9 (middle panel), some raw data that appear stable for head Z coordinate are nonetheless removed. The second pre-processing algorithm (bottom panel), in addition to eliminating noise on the raw data, gives the data temporal regularity thanks to the fitting and the averaging procedures.

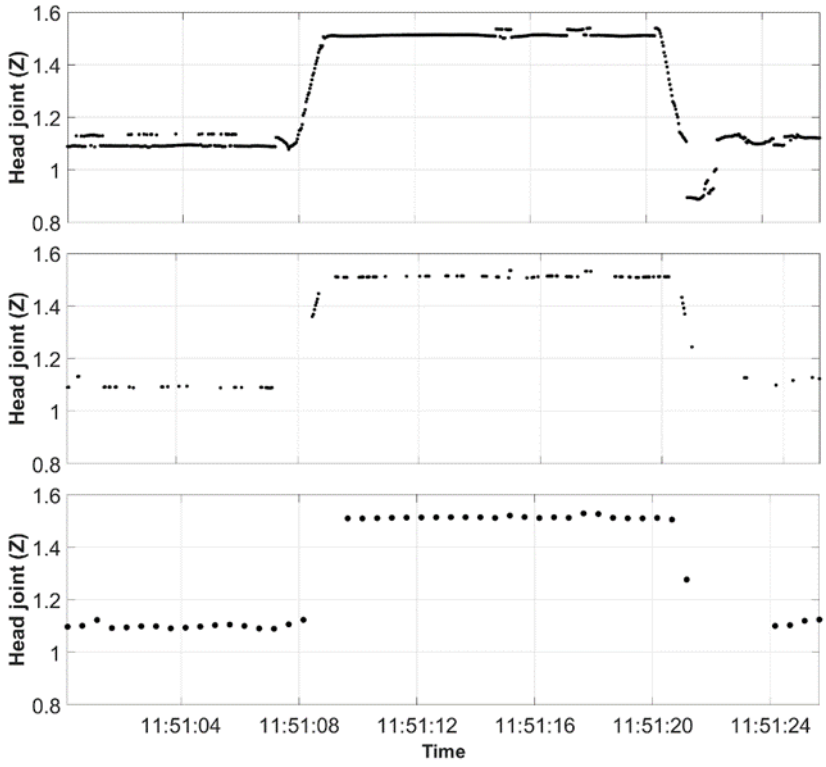

Figure 4.9: example of head vertical position data trace: raw data (top panel); first algorithm pre-processed data (middle panel); second algorithm preprocessed data (bottom panel).

Table 4.4: MLP neural network best accuracy over 30 network simulations. The accuracy is expressed as a percentage.

Best Accuracy (%)			
Case A	**Case B**	**Case C**	**Case D**
86.5	91.6	91.9	94.5

Table 4.5: statistical analysis in terms of Precision, Sensitivity, Specificity and F-score of Case A and Case D. The statistical scores are expressed as a percentage.

	Precision (%)		**Sensitivity (%)**		**Specificity (%)**		**F-score (%)**	
	Case A	**Case D**	**Case A**	**Case D**	**Case A**	**Case D**	**Case A**	**Case D**
Class1	95.4	96.1	92.3	98.1	98.1	98.2	93.8	97.1
Class2	88.6	96.8	90.3	94.0	90.5	97.2	89.5	95.4
Class3	67.5	88.9	75.9	86.6	96.8	99.5	71.4	87.7
Class4	75.8	87.8	72.9	91.7	94.2	97.0	74.3	89.7
Mean value	81.8	92.4	82.8	92.6	94.9	98.0	82.3	92.5

As shown in Table 4.4, all MLP accuracy results on the preprocessed data are higher than on the raw data. The best accuracy was found in **Case D** (first algorithm pre-processing data for training and second algorithm pre-processing data for testing). The statistical results referring to each class for **Case A** (control scenario) and **Case D** (best accuracy scenario) are summarized in Table 4.5. These results confirm the classification improvements obtained when preprocessing the data.

4.2.7. Discussion

In this second solution we defined two preprocessing algorithms, in order to increase the performance of the MLP model obtained in the first solution proposed. The first algorithm removed the data which do not satisfy different threshold criteria based on variability, anthropometric measures and joints velocity. The second one was similar to the first one, but additionally reconstructed the missing data and averages the data on a half-second time window. To verify the effectiveness of the data processing on MLP network classification, we compared the performances of all the possible preprocessing combinations with the results on the raw data (Table 4.4). Therefore, we trained and tested the MLP network with the data preprocessed only with the first algorithm, then only with the second algorithm and finally we evaluated the combination of the two preprocessing algorithms, training

the MLP network with the data preprocessed with the first algorithm and testing it with the data preprocessed with the second algorithm. This last comparison was made to take advantage of both preprocessing algorithms. The idea was to train the MLP network with nosier and larger amount of data to prevent data overfitting and to test it on more stable data, being averaged over a half-second time window and with most of the temporal holes filled. The results of all these preprocessing combinations are summarized in Table 4.4. The accuracy of the MLP network showed that network performance improves in all cases in which the data are pre-processed. The best performance seemed to be obtained by training the network with data processed by the first preprocessing algorithm and to test it with data calculated by the second preprocessing algorithm (Case D). This is probably due to the fact that the MLP model was able to generalize the data in the best possible way, since in Case D the advantages of each of the two procedures were exploited, i.e. the cardinality in the train data and the stability in the test data. Moreover, this method allowed to increase considerably the classification of two relevant classes (lying down and dangerous posture) for automatic dangerous situations recognition systems in AAL. This occurred despite the subject orientation with respect to the camera.

4.3. Third solution: Neural Networks for Automatic Posture Recognition in Ambient-Assisted Living

In third proposed AI solution [206] we performed a customized feature selection, starting from the 10 features selected in the first solution illustrated before, for two different AI architectures, i.e., MLP and LSTM models.

The subset of 10 features, computed from the Kinect skeletal joints coordinates and then chosen using the ReliefF algorithm, was used to train and to test a two hidden layers MLP neural network, obtaining, on the test set, an average posture classification accuracy of 83.9%. This promising result was not, however, satisfying for our purpose since the classifier is the core of a more complex safety system aimed to generate an alarm when dangerous situations occur during every day-life inside a room. Therefore, hoping to increase the MLP performance, in the second solution we proposed two preprocessing algorithms based on velocity threshold and anthropometric constraints. In this case, the overall accuracy reached by the classifier was 92% and it increased to 95% when the test data were also averaged in a timing window of 15 frames (corresponding to 0.5 seconds). This procedure, although it showed increased performances, had nevertheless the weighty drawback of increasing considerably the computational time, making the process not useful for the online demand of the monitoring system. The time required for the preprocessing phase was about 1.031 s, and the frame-by-frame MLP classification was about 0.300 seconds when considering a sequence of 60 frames.

In this third solution we proposed the tuning of the previously implemented MLP classification model and a search for a new model, more appropriate to manage the raw data noise and the constraints of an online safety system. First, we defined a further class as the transition between two consecutive postures (for example between sitting and lying down postures and vice-versa) to enable the system to handle the continuous stream of data from the Kinect device; second, we optimized the previous MLP neural network model (Figure 4.1) selecting a new subset of features using an SVM algorithm, a new set of network hyperparameters and a novel architecture; third, we trained and tested an LSTM sequence network model with a subset of features selected using a genetic algorithm. Both feature selection processes were carried out on the training data and started off from the previously selected set of 10 features. We chose an LSTMs network expecting to take advantage of its ability to produce a frame-by-frame output yet based on a sequence of data instead of only the current input and thereby having the potential to catch a wide range of dependencies among them. Therefore, using this dynamic network we expected to also be able to classify the data referring to the transitions between two successive postures, e.g., when subject passes from the standing to the sitting posture or from the sitting to lying posture, and thereby configuring our system for usage in a more ecological daily life scenario in which the subject freely moves in the room. We analyzed different LSTM sequence architectures to find the one providing the best performance. Each one was configured to separately classify each data frame in the data sequence in order to have results comparable with those of the optimized MLP. The final step of this work was then to compare the performances of the two optimized algorithms.

4.3.1. Data preprocessing

The acquired data were preprocessed with the same techniques described in the previous section in the first solution. Specifically, the skeletal data of 17 joints (Chapter 3, Figure 3.3) acquired by each Kinect V2 were roto-translated in a common, room-fixed, reference system and 16 joint angles plus 4 absolute angles are computed (Chapter 3, Table 3.1). Of the three joint spatial coordinates only the vertical one (Z) was considered for further analysis and then normalized with respect to the height of the subject. All angles were normalized by dividing them by 180. The missing data were filled and identified with the '999' value, but in this solution these values were not deleted from the collected data. Therefore, in this work also frames representing the transition between two consecutive postures (i.e., between a sitting posture to lying down posture and vice-versa or between a standing pose and sitting posture and vice-versa) were maintained and not eliminated from the collected data.

4.3.2. Feature selection

As mentioned in the previous section, in this work we suggested a customized feature selection for each of the two proposed models, starting from the 10 attributed considered in the previous studies: A_{pitch}, A_{roll}, B_{pitch}, B_{roll}, ξ , $\mu 2$, $\delta 2$, Z_1, Z_{C7}, Z_{Hc}.

First proposal for Feature selection

An SVM Classifier with a Gaussian kernel function was used to test all possible combinations of the 10 attributes. Since it was not a binary classification, but a multi-class problem, we decided to employ the *one vs. rest* strategy (Chapter 2, paragraph 2.6). The SVM was first run with all 10 attributes, then with all possible combinations of 9 and so on with 8, 7, 6, 5, 4, 3, 2, 1 feature(s) for a total of 1023 possible combinations tested. For each combination the loss function was computed and the combination of attributes with the lowest loss function was then chosen as the best set for the MLP classifier. All the feature selection processed was carried out on the training set (divided into training-70% and validation set-30%).

Second proposal for Feature selection

A separate feature selection was performed for the LSTM sequence network as this recurrent network uses different principles for classifying the data, i.e., the network output depends on both the current input and on the state of the network, determined by the history of previous inputs. We then considered a standard LSTM sequence architecture similar to the one proposed by Devanne et al. [112] with an input layer followed by a LSTM sequence layer, a dropout layer, a second LSTM sequence layer, another dropout layer, a fully connected layer and a SoftMax layer as our reference architecture. Such network layout, with 100 and 75 neurons in the two LSTM layers and a 30% dropout layer, was used to compute the percentage of correctly classified postures (%CC), which represent the fitness function of the genetic algorithm that we used for the new feature selection. To this goal we built a custom-developed steady state genetic algorithm (in which only a few chromosomes are replaced with each generation) working on a population of 50 binary chromosomes of 10 genes each (one for each feature) (Chapter 2, paragraph 2.5). In the initial population 49 chromosomes were randomly generated while one was forced to have all genes set to 1 in order to test considering all available features. The fitness function built a new LSTM sequence network for evaluating each chromosome, thereby exploring the different combinations of our ten features, and each network was allowed for twenty epochs of training. The genetic algorithm was allowed to evolve for 10 generations. On the first run of the algorithm the genetic operators led to evaluate 390 different combinations of features (out

of the possible 500, the remaining 110 being duplicates). The same 390 networks were then further trained three times for another twenty epochs, in order to test their performance with different initial sets of synaptic weights and biases. The best result in terms of %CC or, in case of comparable %CC results, the set of the best results, identified the set, or the sets, of features which we considered for the optimization of the network architecture (number of neurons in each LSTM layer and dropout percentage). This process of feature selection was computed only on the training set (Table 4.6).

4.3.3. Dataset construction

The full database included a total of 734,339 frames including those containing the 999 values. These latter 35,020 frames were discarded for the training and the testing of the MLP network. In this case, due to the static characteristic of the network, the temporal sequence of the frames was not mandatory and holding 999 values added only noisy data. On the other hand, this removal had the limit of reducing the adherence of the model with the real scenario, corresponding to its final deployment setting, in which a proper reconstruction of the subject skeleton by the Kinect V2 device was not always assured. The handling of the 999 values was, therefore, different for the LSTM sequence model where they were involved in the analysis. Only the sequences entirely composed by frames of 999 values were discarded. The database considered for the MLP network was composed of 699,319 frames, while the LSTM database included 714,330 frames. The latter included 15,011 (2.1%) missing data frames, i.e., 999 frames, whose target class was set to the same class as that of the preceding frame. The class repartition of the two databases is shown in Table 4.6. A training set for the two networks was eventually built using the data from 10 of the 12 subjects, while the test set was built using the data of the remaining 2 subjects, as reported in Table 4.6.

All the collected data were labelled with a new software implemented on MATLAB, easier and more reliable from a point of view of labeling and data synchronization compared to the previous LabView version.

According to that, the following five postures were labelled:

- Class 1: standing posture;
- Class 2: sitting posture;
- Class 3: lying down posture;
- Class 4: dangerous sitting posture;
- Class 5: transition.

Table 4.6: numerosity of frames in the training and testing MLP and LSTM sequence databases, and their repartition in five classes, with the new labelling software

Classes	Database MLP		Database LSTM	
	Training	**Testing**	**Training**	**Testing**
Class 1	121,059	24,267	121,870	25,354
Class 2	189,668	43,990	193,446	45,258
Class 3	76,028	27,165	76,939	27,137
Class 4	113,178	23,779	116,903	24,838
Class 5	66,252	13,933	68,072	14,513
Total	566,185	133,134	577,230	137,100

4.3.4. Pattern Recognition

MLP architecture

Starting from the MLP architecture implemented in the previous study (Figure 4.1), in this work we investigated the effects, on the classification accuracy, of the number of neurons (for each layer we test a number of neurons from 2 to 10) and the type of activation functions (Hyperbolic Tangent, Sigmoidal Tangent, ReLu, Pure Linear and, only for the last layer, SoftMax) in order to achieve a final optimized MLP network.

The training database was then divided into training (70%) and validation set (30%), and the network was trained and validated for 10 times using a k-fold cross validation (k = 10) for each set of considered hyperparameters (see Figure 4.2). After the validation, the MLP was finally trained using the training database, first with a ten-fold cross validation, and then using the whole training database. The learning process was performed over a maximum of 1000 epochs, i.e., 1000 iterations on the training database. Finally, the optimized architecture was tested for 30 simulations for statistical analysis.

LSTM architecture

Since deep learning networks typically have many more parameters to learn than shallow networks and are trained over very large databases, the update of synaptic weights occurs after the network is presented with a subset of the available training data: a mini-batch. The number of samples in mini batches influences learning in the network and represents therefore one of the hyperparameters to be determined. Moreover, when dealing with LSTM sequence networks the input vectors are organized in sequences, typically of the same length. Choosing such length is another important hyperparameter to be considered. Therefore, based on evidence obtained in a previous exploratory analysis, in which we consider different length sequences of either 15, 30 or 60 frames (equal to 0.5, 1 or 2 seconds) and mini-batches of 16, 27, 32 or 64 sequences, the sequence length was set to 60 frames and the mini-batch size was set to 32 sequences each.

The optimization process of the LSTM sequence network architecture was carried out selecting the number and position of dropout layers, the number of hidden LSTM layers and the number of neurons in each of them. We then explored the following architectures (in parentheses the acronyms used to indicate these architectures in the rest of our work): using only one LSTM layer (LSTM), one LSTM layer followed by two fully connected layers (LSTM2FC), two LSTM layers (2LSTM), one bidirectional LSTM layer (BLSTM), two bidirectional LSTM layers (2BLSTM) with two equal dropout layers, two bidirectional LSTM layers with two different dropout layers (2BLSTM2D), two bidirectional LSTM layers with only one dropout layer (2BLSTM1D). The number of LSTM hidden units varied between 75, 100 or 125 and those of the second layer, in the architectures that considered one, were 25 fewer. Each network configuration was trained for 50 epochs and its initialization and training were repeated ten times for statistical analysis.

4.3.5. Statistical Analysis

We use the Shapiro–Wilk test as a hypothesis test. For each test performed the p-value was greater than the chosen alpha level(0.05), therefore the null hypothesis that the data came from a normally distributed population cannot be rejected (IBM SPSS Statistics, IBM). Therefore, for each of the metrics, the mean and the standard deviation were considered.

For what concerns the illustrated deep learning models (investigated in order to identify the best deep learning architecture for our purpose), we observed the distribution of some network hyperparameters (i.e., dropout and shuffling sequences, see Figures 4.13 and 4.15) and the mean accuracy distribution for each of the seven different models (Figure 4.14), through a boxplot representation, to find the best LSTM architecture.

Furthermore, Precision, Sensitivity, Specificity and F-score metrics were computed for each of two proposed architectures, i.e., the MLP network and the best LSTM architecture, such as the 2BLSTM2D model (Figures 4.11 and 4.17). These statistical parameters were calculated for each of the 30 simulations of the two different models. The distribution of these four metrics was studied through the boxplot graphical representation. Moreover, for each of the two models, we also computed a confusion matrix for each of the 30 network simulations. Then, we computed a mean confusion matrix in which the number of frames reported in each cell was the mean, over the 30 confusion matrices of the frames pertaining to that cell (Figures 4.12 and 4.18).

4.3.6. Results

MLP neural network

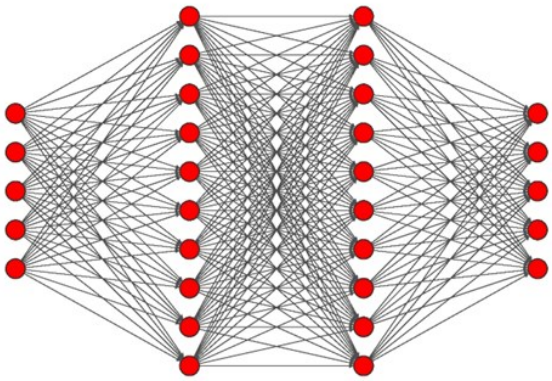

Figure 4.10: Optimized MLP architecture: 5 neurons in inputs and output, 10 neurons in the first, second and third hidden layers, each using the sigmoidal tangent activation function. The output layer was implemented with the softmax activation function.

The best set of attributes identified by the SVM feature selection algorithm was composed by five features: A_{pitch}, B_{roll}, , Z_1, Z_{C7}, and Z_{Hc} which obtained a loss function equal to 0,293. This set of features was used for the training and testing of the MLP network.

As summarized in Figure 4.10, the MLP network architecture that produced the best results, in terms of average accuracy, consists consisted of three hidden layers each including ten neurons with a sigmoidal tangent activation function. Moreover, the output layer, corresponding to the softmax activation function, was composed of five neurons (equal to the number of classes in the database).

The overall total accuracy reached by 30 simulations of this MLP network architecture varied across 0.78 - 0.79 with a mean value of 0.784 ± 0.003

(Standard deviation (SD)). Figure 4.11 shows the distribution of the results of 30 simulations, separated for each class, for the Specificity, F-score, Sensitivity and Precision statistical parameters. The Specificity results (Panel A, Figure 4.11), compared to that of the other three statistical metrics (Panel B, C and D, respectively, Figure 4.11) were very close to each other and ranged 0.99 – 0.93. The F-score ranged 0.83 – 0.91 in the first three classes showing a decrease in Class 4 (0.73) which became more significant in Class 5 (0.15). A similar behavior characterized the Precision results (panel D, Figure 4.11) where the values for Class 4 and Class 5 were equal to 0.71 and 0.09, respectively. These two values were very different with respect to the other three values varying from 0.90 to 0.95. In the Sensitivity results (panel C, Figure 4.11), the value of the Class 5 was again the lowest (0.58) but much closer to the other four values, ranging between 0.74 and 0.88. To summarize, Class 3, corresponding to the lying posture, was the most precisely identified by the MLP classifier, followed by Class 2 (sitting posture), and Class 1 (standing posture). The network, on the other hand, classified Class 4 (dangerous sitting pose) and, especially, Class 5 (transition between a posture to another) with more difficulty for each of the four statistical parameters calculated (Figure 4.11).

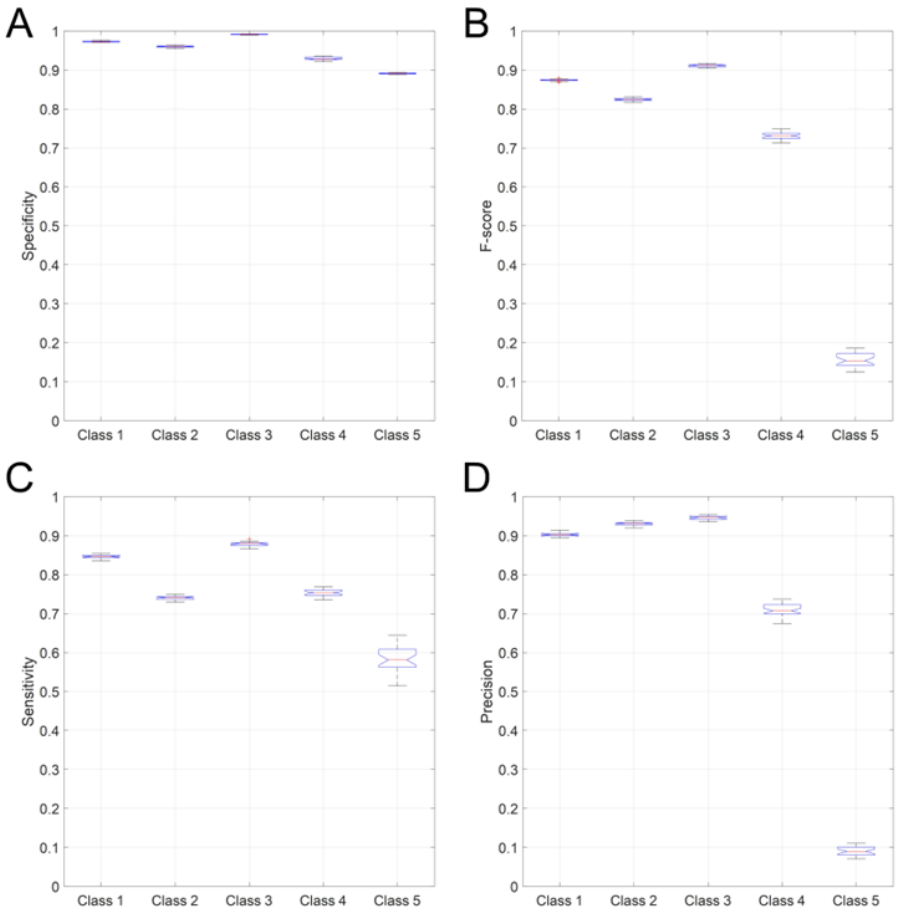

Figure 4.11: Specificity (panel A), F-score (panel B), Sensitivity (panel C) and Precision (panel D) are shown for the 30 MLP optimized network simulations. Each boxplot presents the median, the 25th and 75th percentiles, with the whiskers representing the maximum and minimum non-outlier values.

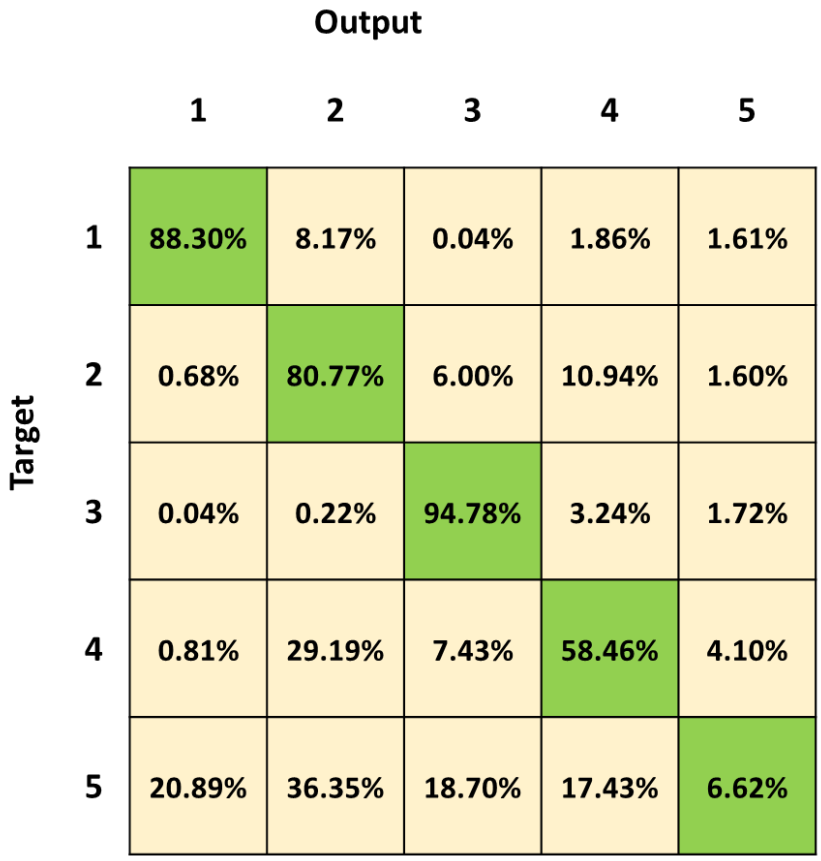

Figure 4.12: Mean confusion matrix obtained from the 30 MLP network simulations

Figure 4.12 shows the mean confusion matrix computed over the 30 network simulations performed. The major misclassifications were between Class 2 and Class 4 and vice versa (10.94% and 29.19 %respectively). Class 1 was mainly misclassified with Class 2 (8.17%) and Class4 with Class 3 (7.43%%) and vice versa (3.24%%). Class 2 was also misclassified with Class 3 (6%). The classifier worse performed in the identification of Class 5 which was confused with all the other four Classes (20.89%, 36.35%, 18.69%, 17.43%). The best identified class was Class 3 (lying down posture), followed by Class 1 (standing posture).

LSTM sequence-to-sequence neural network

As detailed in the methods section we used a two-step process for performing the se-lection of the best set of features for the LSTM network over the LSTM dataset, i.e., including 999 frames and transitions. Each feature set proposed by the genetic algorithm was tested for computing its fitness by allowing the above-mentioned standard two LSTM layers network to train for 20 epochs. For statistical purposes the same network was trained

three more times for further 20 epochs so that we obtained four evaluations for each combination of features selected by a chromosome. The best four feature sets had similar fitness values and were then chosen based on the best performance over such four training trials. We finally performed one run allowing for 100 training epochs, which confirmed the four feature sets as the best performing ones and, at the same time, allowed us to note a decrease in the percentage of correct classifications occurring after about 50 epochs with all feature sets, which was accompanied by an increase in the validation loss function. All four selected sets share the first four features reflecting the roll and pitch angles of the trunk and of the head, two vertical coordinates among the three available, i.e., head, midpoint of the hips and midpoint of the shoulders and two further angles. One set considered seven features, while the other three considered eight. These four sets are:

- Set 1: A_{pitch}, A_{roll}, B_{pitch}, B_{roll}, μ2, Z_1, Z_{C7}
- Set 2: A_{pitch}, A_{roll}, B_{pitch}, B_{roll}, ξ , μ2, Z_{C7}, Z_{Hc}
- Set 3: A_{pitch}, A_{roll}, B_{pitch}, B_{roll}, μ2, $\delta 2$, Z_{C7}, Z_{Hc}.
- Set 4: A_{pitch}, A_{roll}, B_{pitch}, B_{roll}, ξ , $\delta 2$, Z_1, Z_{Hc}.

With such optimized hyperparameters the best performance achieved by the standard LSTM network with the four considered sets of features over the test set ranged 81.9 and 83.5%. The latter was achieved using the Set 3 of features which was the set chosen for the rest of our study.

Further search for the remaining network hyperparameters was then performed with the chosen feature set, exploring the layout of the network, the number of neurons in each LSTM layer and the dropout percentage, which were investigated considering thirty experiments, each one training the network for 50 epochs. All architectures performed best with a first hidden layer of 125 neurons and a second one of 100. The improvement in accuracy compared to the two layers having 100-75 and 125-75 neurons was in the order of 0.5% for both architectures, which was a consistent but not statistically significant finding.

The five dropout percentages from 25 to 45% considered for each tested architecture caused changes in the mean percentage of correct classifications in the order of 1%, which were not statistically significant. One example of the dropout results for the two LSTM layers and the one biLSTM layer networks is shown in Figure 4.13 considering the architectures having highest number of neurons (n=125).

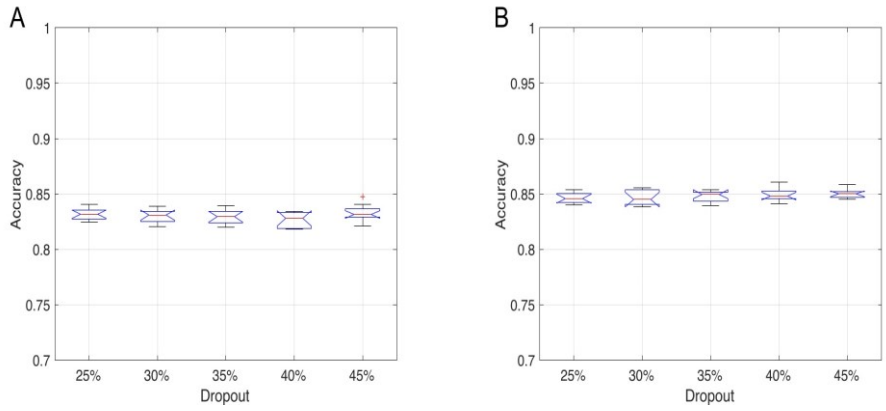

Figure 4.13: overall test set accuracy distributions over 30 trained networks of two LSTM network architectures with different dropout levels. Panel A: Accuracy for a two LSTM layers network having 125 and 100 neurons in the first and second hidden layer, respectively. Panel B: Accuracy distributions for a one hidden bidirectional LSTM layer network with 125 neurons.

A comparison of the accuracy results obtained with the different considered architectures is shown in Figure 4.14 for a chosen dropout level (45%). Bidirectional LSTM architectures achieved almost 2% higher accuracies than traditional LSTM layers even given a comparable number of neurons, while increasing the number of bidirectional LSTM layers did not significantly improve results.

The shuffling of the training sequences at every training epoch significantly improved the network performance compared to the no shuffling procedure, as shown in Figure 4.15 for the 2LSTM, the BLSTM and the 2BLSTM architectures.

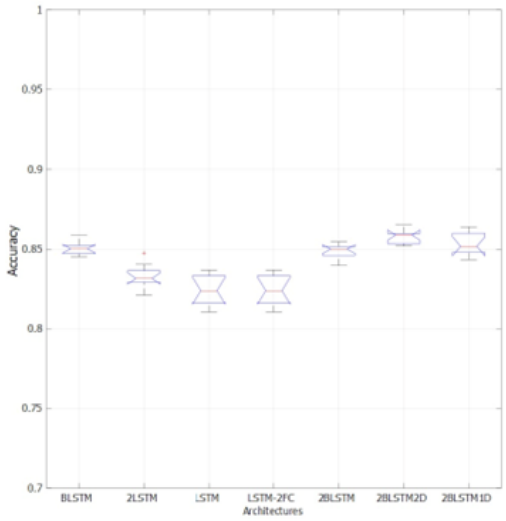

Figure 4.14: accuracy distribution over 30 repetitions of 50-epochs training for different network architectures and dropout fixed at 45%. Tested architectures: one hidden bidirectional LSTM layer (BLSTM), two LSTM layers (2LSTM), one LSTM layer (LSTM), one LSTM layer followed by two fully connected layers (LSTM-2FC), two bidirectional LSTM layers (2BLSTM), two bidirectional LSTM layers with two dropout layers (2BLSTM2D), two bidirectional LSTM layers with one dropout layer (2BLSTM1D).

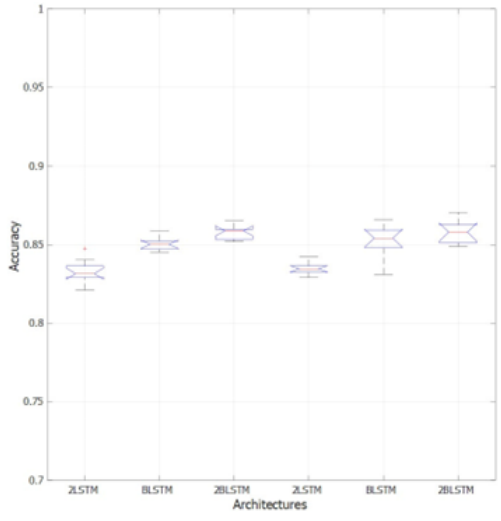

Figure 4.15: effect of shuffling sequences at every epoch over the performance of the two LSTM layers network (2LSTM), the one bidirectional LSTM layer (BLSTM) and two bidirectional LSTM layers network (2BLSTM). The three leftmost boxplots represent the performance of the tested networks without data shuffling, while the three rightmost ones represent the performance of the same networks with data shuffling. represent the performance of the same networks with data shuffling.

Overall, the best performance was therefore that of the 2 bidirectional LSTM layers network as detailed in Figure 4.16, with 125 – 100 neurons in the two hidden layers, two 45% dropout layers, one following each of the LSTM layers, trained with sequence shuffling at every epoch, which obtained a mean accuracy of 0.857±0.008.

The network performance in terms of sensitivity, F-score, specificity and recall for each of the classes is shown in Figure 4.17. The Specificity results (Panel A), show that the network rarely misassigned frames to the standing (Class 1) and sitting classes (Class 2) (mean 0.93 and 0.90, respectively), the lying posture (Class 3) had a mean specificity of 0.86, the dangerous sitting posture (Class 4) attained 0.85 and the transition class (Class 5) 0.72. The Sensibility of the network is its most appealing feature (Panel C) with values ranging 0.91 and 0.98, with the dangerous sitting class specificity reaching 0.95. The F-score (Panel B) ranged 0.90 – 0.95 in the first three classes, showing a decrease in Class 4 (0.88) and worsening for Class 5 (0.76). A similar behavior characterized the Precision results (panel D) where the values for dangerous sitting posture and the transition class were equal to 0.82 and 0.63, respectively, while the other three classes ranged 0.84 to 0.92.

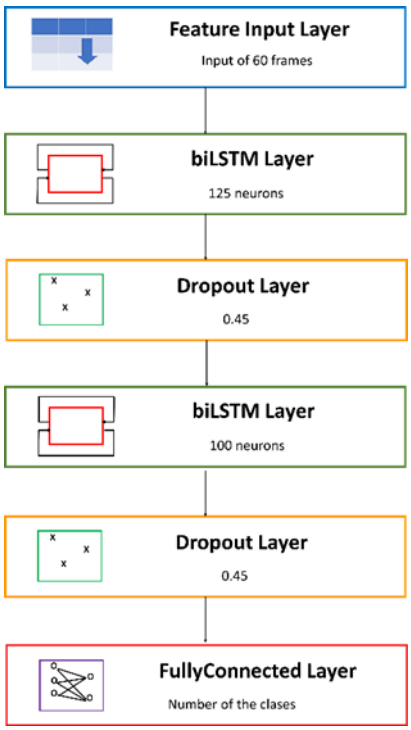

Figure 4.16: definitive architecture of 2BLSTM2D model.

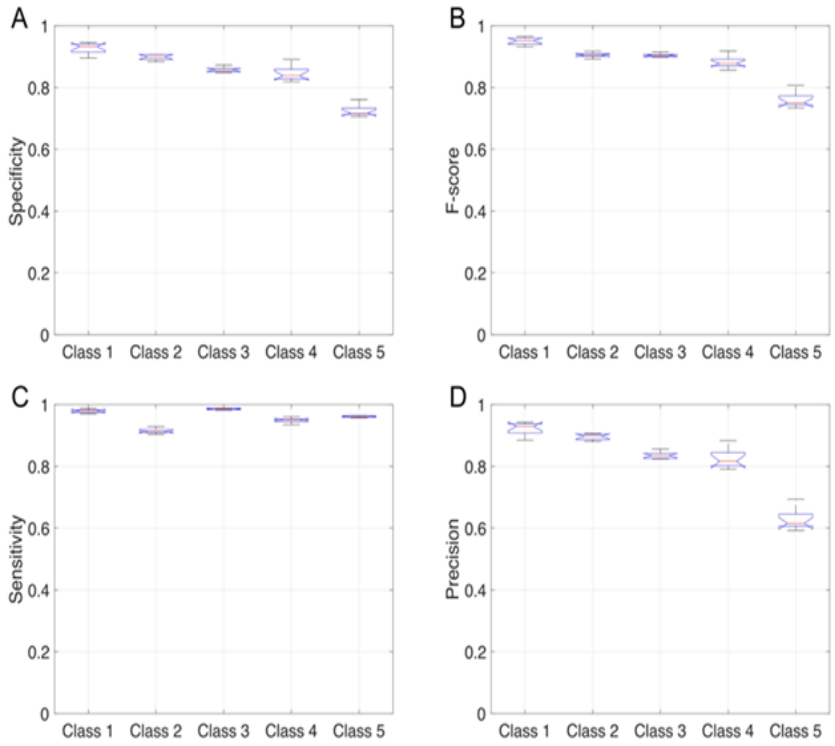

Figure 4.17: the four panels clockwise from the top present the Specificity (Panel A), F-score (Panel B), Sensitivity (Panel C) and Precision (Panel D) achieved for each class with the 2BLSTM2D network.

To summarize, Class 1, corresponding to the standing posture, was the most precisely identified by the LSTM classifier, followed by Class 2 (sitting posture) and Class 3 (lying posture). The network, on the other hand, classified Class 4 (dangerous sitting posture) and, especially, Class 5 (transition between a posture and another) with more difficulty for specificity, F-score and precision (Figure 4.17).

The mean confusion matrix for such network over the 30 training runs is shown in Figure 4.18.

Compared to the MLP, the misclassifications between Class 2 and Class 4 and vice versa decreased to 9.39% and 8.20%, respectively. Class 2 samples were also misclassified as Class 1 (1.16% ± 0.22%), while some Class 3 frames were classified as Class 2 (1.65%) and with Class 3 (6.35%). The frames that were erroneously classified as Class 5 were significantly fewer than with the MLP (14.45%, 11.03%, 9.32% and 11.49% respectively). The best identified class was Class 2 (sitting posture), followed by Class 1 (standing posture).

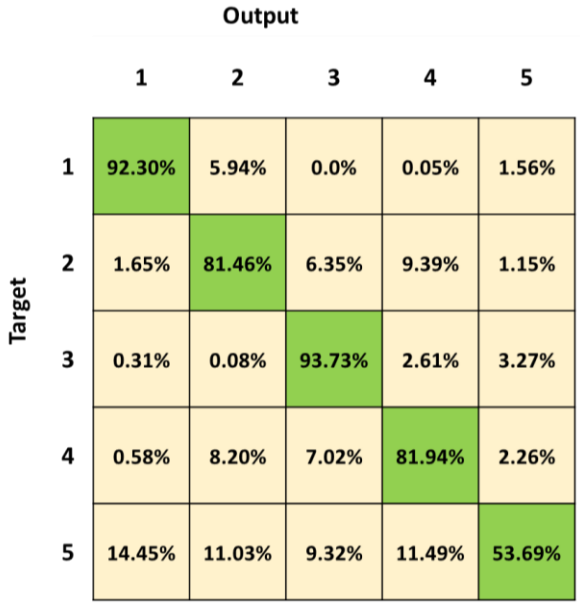

Figure 4.18: mean confusion matrix obtained from the 30 simulations of the 2BLSTM2D network.

4.3.7. Discussion

Developing on our previous work, in which we considered the 'standing', 'sitting', 'lying' and 'dangerous sitting' postures, we have expanded the original dataset by including a fifth class, 'transition', collecting all transitions between two consecutive postures. We consider such a choice necessary in order to be able to provide the network with a continuous stream of data while reducing the risk of incurring in false positives during such transition movements, e.g., while sitting down. With such new database of classified postures, we performed a new feature selection on the 10 parameters that we chose to describe each captured frame, which were derived from the skeleton identified by each Kinect device, in order to optimize the classification ability of an MLP network. The network hyperparameters were then in turn optimized and we trained the MLP with our training set composed of the data relative to 8 of our 10 acquired subjects. The remaining two subjects' data made up the test set, on which the MLP network achieved an average 78.4 %CC.

The same five-classes dataset was considered to train an LSTM sequence network with the addition of 999 frames, i.e., frames in which the Kinect was not able to identify a proper skeleton. Since such noisy frames were present within each acquired subject's data, this has to be considered a more ecological scenario in representing the system's deployment conditions.

A new feature selection exploiting a genetic algorithm aimed at maximizing a fitness function consisting in the %CC computed over the test set using a reference LSTM sequence network was developed. Such network consisted in two LSTM layers, one 25% dropout layer, a fully connected

layer and a softmax layer. This approach led us to use eight of the ten available features, which were presented as sequences of 60 frames in mini-batches of 32 sequences each. As above, the training set was then built using frames from eight subjects and the remaining two subjects' data made up the test set. Several LSTM sequence architectures (one LSTM layer, two LSTM layers, one bidirectional LSTM layer and two bidirectional LSTM layers), using different numbers of neurons in the hidden layer(s) and dropout arrangements were tested in a hyperparameters optimization algorithm.

As expected, the addition of the 'transition' class worsens the classification ability of the MLP network. Indeed, frames that may be very similar to those that are required to be classified in one of the other four classes are now being assigned to the 'transition' class. In other words, a transition between sitting and standing, for example, contains frames that are very similar to those belonging to the 'sitting' class and to those belonging to the 'standing' class. This can be appreciated by examining the last row of the confusion matrix in Figure 4.12, showing how a significant percentage of frames that were classified in class 5 were supposed to be classified in the other classes. Clearly a static network such as the MLP has no means to correctly classify such frames.

The classification accuracy improved when using the LSTM architectures and more so using bidirectional LSTM sequence architectures, in which half of the neurons were presented with the regular sequence of inputs while the other half was presented with the backwards input sequence. Such network architecture, exploiting data shuffling and 45% dropout reached the best results with a mean classification accuracy of over 85%. The effectiveness of this approach was especially evident both in terms of correct classifications for frames belonging to the 'transition' class, in which less than 3% of trials belonging to other classes are now classified (see Figure 4.17), and in terms of correct classifications of '999' frames, for which the misclassified percentage was as low as about 9%.

Comparing the behavior of the two networks throughout the 5 classes shows a higher specificity for the MLP network on all the five classes, yet also a much lower sensitivity than the LSTM on each class. An important contribution to the overall higher %CC obtained by the LSTM is due to the improved ability to correctly classify frames pertaining to the 'transition' class, as mentioned, yet a significant improvement also regards the increased sensitivity, F-score and precision in classifying the 'dangerous sitting' posture, which represents a critical condition for the real usage setting of the application, one requiring rising an alarm for the safety of the monitored user. From this standpoint the LSTM mean sensitivity of 0.95 in detecting true positives for the 'dangerous sitting' posture represents an important improvement from the 0.76 achieved by the MLP. Indeed, a higher chance of false positives (the specificity decreased from 0.93 for the MLP to 0.85 for the LSTM) represents an acceptable cost for being more certain that dangerous situations will not be missed.

Further developments for improving the performance of the system and its customization for the specific purpose of allowing safer autonomous

living conditions in frail individuals may include the classification of each type of transition in a separate class and the use of LSTM 'Last' architectures, which provide only one classification for each sequence of frames. Indeed, for the ultimate patient-safety goal of the proposed system, a classification occurring every one or two seconds would not alter its effectiveness.

4.4. Final proposed classification system

In this last solution, we suggest a more extensive investigation on deep leaning models, proposing a new deep RNN solution, with an architecture based on GRU networks.

Additionally, while still considering the data acquired in the same set of experiments, we chose to build a new dataset with several new characteristics:

1. instead of adopting a sequence-to-sequence RNN model (as developed in the third solution, section 4.3), we chose to employ a sequence-to-last approach, in which, for each input sequence, the model predicts a single output value. This choice was made because possible future developments will include the use of post-processing algorithms for the analysis of classification outputs. For instance, the post-processing algorithms will concern the comparison between the respective outputs coming from the four cameras or correlating the posture of the subject with his position in the room, to distinguish an everyday life scenario from a dangerous situation.
2. inspired by the study developed by Wang et. al, we decided to compute a new set of features, comprising the 3D skeleton joint coordinates (Chapter 3, Figure 3.3). Wang et al. made a comparison of Kinect-based HAR techniques about the grouping of feature types, more precisely between *handcrafted features*, i.e., built using feature engineering approaches, and *deep learning features*, i.e., automatically extracted during the deep model training [154]. This study supports the idea that the *handcrafted features* may not exhaustively consider all the intrinsic characteristics in the data and, therefore, during the learning process it may be important to allow the network to select the most significant features by itself, provided that enough training data is available. According to this idea, in this last solution we decided to build our dataset considering all the 51 joint coordinates discussed in the previous section (Chapter 3 Figure 3.3) as input of the model. We also chose to add one more attribute ($D(C7, K_n)$, where $n = 1, \dots 4$, i.e., the number of the Kinect V2 camera considered), which represents the Euclidean distance between the subject, i.e., the C7 marker, and the position in the room of the active Kinect system.

3. we introduced a new method for labelling the acquired data. We decided to incorporate the transition between two consecutive postures with the posture following the transition. For example, if the subject changes from standing to sitting posture, then this transition will be identified within the sitting posture; vice versa if the person changes from a sitting posture to a standing posture, then this transition will be labeled as standing posture.
4. an exam of the training dataset showed a strong imbalance between classes, especially for Classes 3 and 4 compared to Classes 1 and 2, (see Training, Database LSTM, Table 4.6), which could lead to limitations in the model generalization capacity [207]. We therefore chose to apply a data augmentation method for rebalancing the dataset.

Accordingly, in this work we decided to compare the new model with the old one: testing the new GRU model with the features selected in the previous solution (paragraph 4.3.2, Second Proposal for Feature Selection); testing the previously proposed deep learning model (2BLSTM2D, Figure 4.16) with the new feature combination discussed in this section and testing the new GRU model with the new feature combination.

4.4.1. Deep learning features

Starting from the roto-translated data (Chapter 3, Figure 3.4), acquired from the 12 subjects, we decided to select a new set of features inspired by the work of others [104], [208]–[212]. In these studies, all coordinates of the 3D skeletal joint are transformed from the world coordinate system to the person-centered coordinate system by moving the origin of the coordinate system, for example, to the hips center or to the head or torso location. Similarly, we decided to translate all the coordinates of the joints, roto-translated in the absolute system (X,Y,Z) (Chapter 3, Figure 3.4), to an egocentric reference system centered in the joint C7 (the midpoint between the two shoulders, see Figure 3.3). Additionally, we chose to avoid considering the relative and absolute angles, which we had relied upon in our previous work (Chapter 3, paragraph 3.6), because the information of the orientation of the subject‘s joints is intrinsic to the data remapped in the new egocentric system.

In sum, in this work the input of the model consisted of all 51 coordinates translated in the egocentric system centered in C7. In addition to these attributes, we decided to add another regressor, i.e., the Euclidean distance between the subject’s joint coordinates $[X_{K_n}, Y_{K_n}, Z_{K_n}]$. This distance is defined by the following formula:

$$D(C7, K_n) = \sqrt{\left(X_{C7} - X_{K_n}\right)^2 + \left(Y_{C7} - Y_{K_n}\right)^2 + \left(Z_{C7} - Z_{K_n}\right)^2} \qquad (57)$$

where $n = 1, \dots 4$, i.e., the total number of the Kinect V2 cameras employed in this study.

The introduction of this new attribute was due to the observation of a strong dependency between the subject's position, with respect to the Kinect V2, and the quality of Kinect V2 output data (Figure 4.8).

At the end of the feature computation, we obtained a total of 52 attributes describing the data.

4.4.2. Data preprocessing

As described in the previous solutions, all the missing data were filled and identified with the 999 value, and they were not deleted from the collected data. Also, the transition between two consecutive postures (i.e., from sitting to lying down posture and vice-versa or from standing to sitting and vice-versa) were maintained and not eliminated from the acquired data. Moreover, inspired by [210], which normalized all the coordinates of the skeleton for the Euclidean distance between the head joint and the neck joint , we scaled all the 51 skeletal joint coordinates (translated in the C7 system) with respect to the subject's real height, i.e., the heigh which the person declared during the acquisition. For the $D(C7, K_n)$ feature, instead, it was normalized between 0 and the maximum size of the room, which in this case was 5 meters.

In addition, following the data preprocessing approach proposed in the second solution (Paragraph 4.2.1), a moving mean filter using with a time window of 15 frames (equal to 0.5 seconds) was applied to each feature time series, which provides the calculation of the mean of the values present within the window. If the latter contains only 999 values (missing data), the moving mean is not calculated. Although this procedure leads to a slight decrease in the quality of the input data, it allows to preserve the temporal sequence of the data (Figure 4.19).

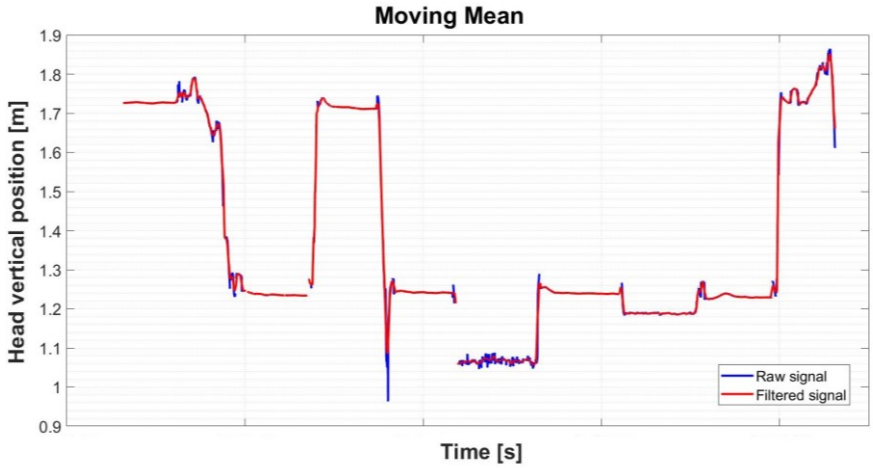

Figure 4.19: Example of head vertical position data: the blue trace corresponds to the raw data and the red one corresponds to the data filtered through the moving mean.

4.4.3. Dataset construction

In order to have more informative data for the training and testing of the new model, in this new solution we decided to carry out a further cleaning of the collected data, eliminating the uninformative acquisitions. In fact, by inspecting the acquired signals, a percentage threshold value on 999 (i.e., missing data) samples was heuristically chosen, below which the signal was discarded, as it was excessively corrupt. The total number of 999 samples within the considered sequence should not exceed the percentage value equal to 35%.

All the collected data were labelled, using the new software implemented in MATLAB, as either:

- Class 1: standing posture;
- Class 2: sitting posture;
- Class 3: lying down posture;
- Class 4: dangerous sitting posture;
- Class 5: transition.

The full database included a total of 629,340 frames comprehending those containing the 999 values. The class subdivision of the collected data is shown in Table 4.7.

Table 4.7: numerosity of frames in the training and testing and their repartition in the five classes.

Classes	**Database 5 classes**	
	Training	**Testing**
Class 1	138,300	32,400
Class 2	190,320	44,880
Class 3	56,760	14,040
Class 4	84,480	19,680
Class 5	40,800	7,680
Total	510,660	118,680

As mentioned above, in developing a new dataset we decided to incorporate the transition between two consecutive postures with the posture following the transition. The class repartition of the new database is shown in Table 4.8.

A training set was built using the data from 10 of the 12 subjects, while the test set included the data of the remaining 2 subjects 1.

Table 4.8: numerosity of frames in the training and testing sets and their repartition in four classes for the new dataset

Classes	**Database 4 classes**	
	Training	**Testing**
Class 1	150,960	34,320
Class 2	208,200	48,960
Class 3	60,960	14,760
Class 4	90,540	20,640
Total	510,660	118,680

The training and testing datasets were then subdivided in sequences composed by 120 frames. To the training sequences an overlapping percentage of 50% is also applied (60 frames).

Given the mentioned imbalance of the classes, in the training database we decided to apply a data augmentation technique. However, the data augmentation technique was only carried out to the sequences described with Class 3 and Class 4, being those with fewer examples (Table 4.8). The data augmentation of these sequences was based on creating a new Class 3 and Class 4 sequences by adding Gaussian noise (calculated on the mean and standard deviation of the sequence under consideration) to the original sequence [213], [214].

4.5. Pattern Recognition

The model proposed in this solution was based on the GRU recurrent network block, more precisely on Bidirectional GRU model.

The developed model (3BGRU3D), shown in Figure 4.20, was composed by a first Masking input layer, allowing the model to ignore the 999 samples while maintaining the time sequence of the data, followed by three bidirectional GRU layers, respectively with 100, 50 and 50 hidden neurons. Each RNN layer was followed by a dropout layer, for preventing overfitting, with a dropout percentage of 40%. The last dropout layer was then followed by two Fully Connected layers, with one interposed 40% dropout layer with 50 and 25 hidden neurons each, and finally the output layer implemented with softmax activation function.

The performance of this new architecture was compared with the network model proposed in the third solution, with which the best mean accuracy results were obtained (2BLSTM2D). The model is composed by 2 bidirectional LSTM layers, with 125 and 100 neurons, respectively, two 45% dropout layers, one following each of the LSTM layers, and a final classification stage with one fully connected layer and a softmax layer (Figure 4.16).

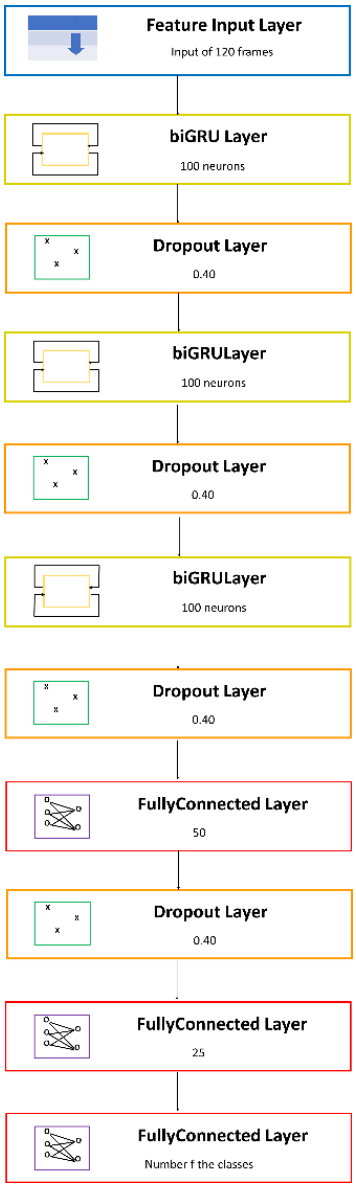

Figure 4.20: the architecture of the new proposed model 3BGRU3D.

4.5.1. Statistical Analysis

The Shapiro–Wilk test was used to test the data normality distribution. For each test performed the p-value was higher than the chosen alpha level (0.05), therefore the null hypothesis that the data came from a normally distributed population is accepted. For the statistical analysis we calculated overall Accuracy, Precision, Sensitivity, Specificity and F-score metrics for each of the proposed architectures and each combination of features and labelling approach. These statistical parameters were calculated for each of the 30 simulations of the different model trials. The distribution of these four metrics was observed through the boxplot graphical representation. Moreover, for each of the 30 trials, we also computed a confusion matrix.

Then, we calculated a mean confusion matrix over the 30 simulations and after that we normalized all the cells of the matrix respect to the frame cardinality of each class. For each model trial, we also computed the mean percentual error. To compare the accuracy values of the different AI solutions investigated, the T-test was performed (alpha level = 0.05).

4.5.2. Results

3BGRU3D trained and tested on 8 features database

Training and Testing datasets

During the training, the model received as input 8,511 sequences, each one composed by 120 frames, with a 50% of overlapping and described by the 8 features (A_{pitch}, A_{roll}, B_{pitch}, B_{roll}, $\mu 2$, $\delta 2$, Z_{C7}, $Z_{Hc.}$) selected in the previous first solution, obtained through the same steps described in before (Chapter 4, paragraph 4.1.2). Each sequence was described by one class (it was a sequence-to-last model) and the total number of classes was 5 (Class 1: standing posture, Class 2: sitting posture, Class 3: lying down posture, Class 4: dangerous sitting posture and Class 5: transition posture). The test sequences were instead 989 in total, built exactly like the training ones, except for the overlap factor which was not present here. The model employed was the 3BGRU3D. (Figure 4.20).

Accuracy

The overall accuracy reached by 30 simulations of this 3BGRU3D network ranged 0.80 and 0.83, with a mean value of 0.82 ± 0.009 (Standard deviation (SD)).

Mean error ratios

Figure 4.21 shows the mean classification errors, normalized by the total number of frames in each class, over the 30 network simulations. The smallest mean error was found in Class 3 (0.03±0.04), followed by Class 1 (0.12±0.06) and Class 2 (0.18±0.06). Class 4 (0.28±0.09). Class 5 were the classes with the worst mean error ratio (0.38±0.12).

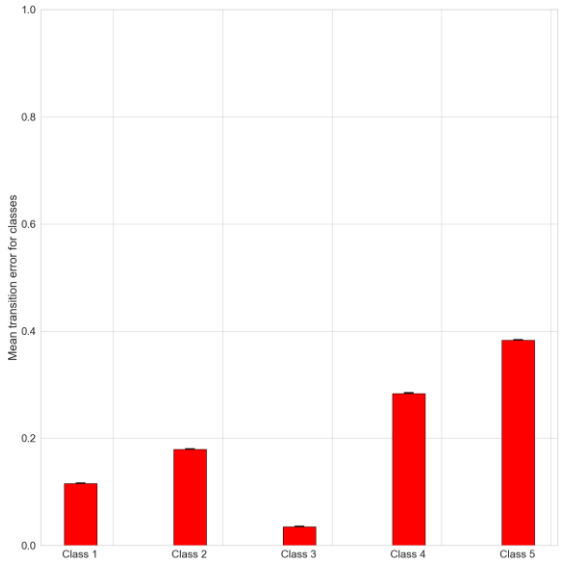

Figure 4.21: mean classification error for each class. The errors for each class were normalized by the number of the total frames for each class.

Specificity, F-score, Sensitivity and Precision

Figure 4.22 shows the distribution of the results of the 30 simulations, separately for each class, for the Specificity, F-score, Sensitivity and Precision statistical parameters. The Specificity results (panel A, Figure 4.22), compared to that of the other three statistical metrics (panel B, C and D, respectively, Figure 4.22) were very close to each other and ranged 0.92 – 0.97. The F-score ranged instead 0.83 – 0.88 in the first three classes showing a decrease in Class 4 (with a mean value equal to 0.75, ranging 0.74 – 0.77), which became more significant in Class 5 (with a mean value equal to 0.66, ranging 0.56 – 0.72). In the Sensitivity parameter (panel C, Figure 4.22), the values for Class 4 and Class 5 ranged 0.62 – 0.78 and 0.45 – 0.71, respectively. These two values were very different with respect to the other three classes: Class 1 ranged 0.84 - 0.82, Class 2 between 0.79 and 0.84 and Class 3 between 0.94 and 0.98. In the Precision results (panel D, Figure 4.22) Class 1 had been proven to be the most precise (ranging 0.80 – 0.91), followed by Class 3, Class 2 and Class 4, which were very close to each other (ranged 0.79 – 0.83). Class 5 presented instead the worst results in term of precision, showing a mean value of 0.72 and ranging 0.65 – 0.78.

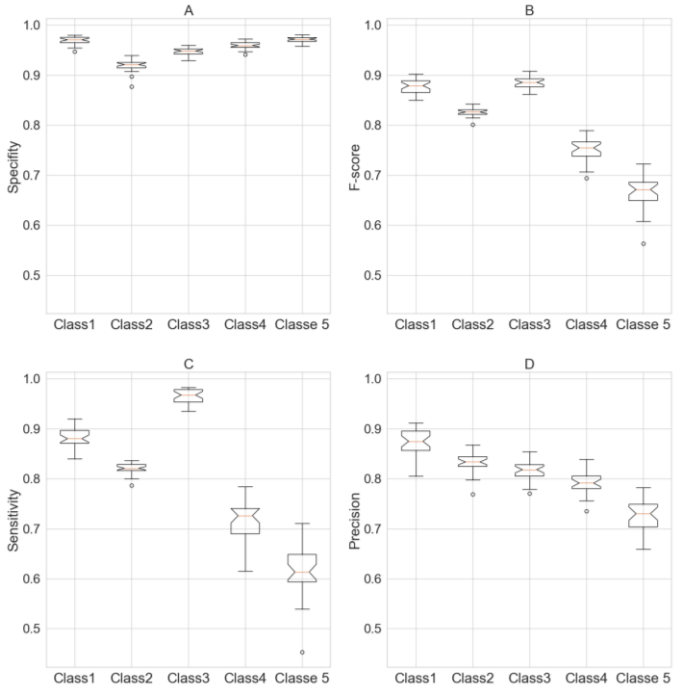

Figure 4.22: Specificity (panel A), F-score (panel B), Sensitivity (panel C) and Precision (panel D) are shown for the 30 3BGRU3D network simulations. Each boxplot presents the median, the 25th and 75th percentiles, with the whiskers representing the maximum and minimum non-outlier values.

Mean Confusion Matrix

Target \ Output	1	2	3	4	5
1	88.78%	6.28%	0.0%	0.89%	4.03%
2	3.65%	82.24%	5.22%	6.52%	2.35%
3	0.0%	0.0%	96.92%	1.31%	1.75%
4	0.97%	15.04%	9.22%	71.84%	2.91%
5	9.52%	13.49%	7.93%	6.34%	62.69%

Figure 4.23: mean confusion matrix obtained from the 30 3BGRU3D network simulations.

Figure 4.23 shows the mean confusion matrix computed over the 30 network simulations performed. Observing the classification results, the major misclassifications were with Class 4 and Class 5, respectively with a percentage of correct classification of 71.84% and 62.69%. Class 2 were mainly misclassified with Class 4 (6.52%) and vice versa (15.04%). The classifier worse performed in the identification of Class 5, which was confused with all the other four Classes (9.52%, 13.49%, 7.93% and 6.94%, respectively Class 1, Class 2, Class 3 and Class 4). The best identified class was Class 3 (lying down posture), followed by Class 1 (standing posture).

3BGRU3D trained and tested on 52 features database described by 5 classes

Training and Testing datasets

During the training, the model received as input 8,511 sequences, each one composed by 120 frames, with a 50% of overlapping and described by the 52 features (the 52 features illustrated in the paragraph 4.4.1). Each sequence was described only by one class (it was a sequence-to-last model) and the total number of classes was 5 (Class 1: standing posture, Class 2: sitting posture, Class 3: lying down posture, Class 4: dangerous sitting posture and Class 5: transition posture). The test sequences were instead 989 in total, built exactly like the training ones, except for the overlap factor which was not present here. The model employed was the 3BGRU3D. (Figure 4.20).

Accuracy

The overall total accuracy reached by 30 simulations of this 3BGRU3D network varied across 0.79 - 0.83 with a mean value of 0.81 ± 0.009(SD).

Mean error ratios

Figure 4.24, shows the mean classification errors, normalized by the total number of frames in each class, over the 30 network simulations. The smallest mean error was found in Class 1 (0.05±0.03), followed by Class 3 (0.13±0.03). Moreover, Class 2 and Class 4 had comparable values to each other, respectively 0.203±0.03 and 0.201±0.05. Class 5 was the class with the worst mean error ratio (0.81±0.1).

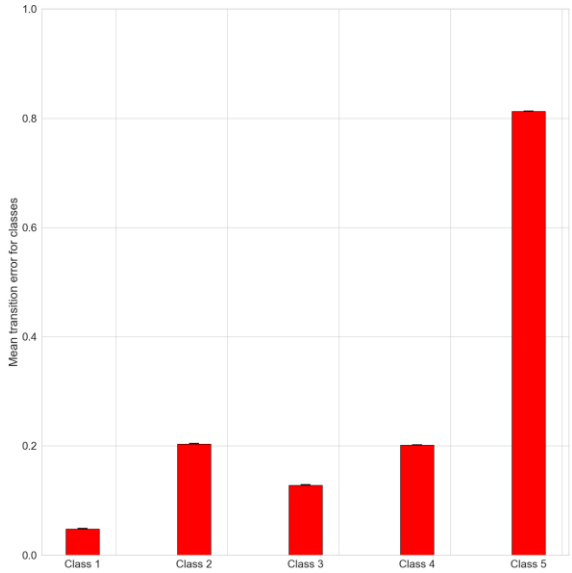

Figure 4.24: mean classification errors for each class. The errors for each class were normalized by the total number of frames for each class.

Specificity, F-score, Sensitivity and Precision

Figure 4.25 shows the distribution of the results of the 30 simulations, separated for each class, for the Specificity, F-score, Sensitivity and Precision statistical parameters. Regarding the Sensitivity metrics (Panel C, Figure 4.25), Class 1 had the highest mean value and was equal to 0.95 (with a range of 0.97-0.94), while Class 2 and Class 4 reached similar values, respectively ranging between 0.78-0.85 whit a mean value of 0.80 and 0.73-0.84, with a mean value of 0.80. On the contrary, the Specificity results (Panel A, Figure 4.25), compared to that of the remaining statistical metrics (Panel B, C and D, respectively, Figure4.25) were very close to each other and ranged 0.86 – 0.99. For what concerned the F-score parameter, it ranged between 0.78 – 0.89 in the first four classes, showing a significant decrease in Class 5 (mean value equal to 0.27 and ranging between 0.077-0.38). By examining the distribution by class of the Precision parameter, we observed that Class 1 and Class 4 exhibit similar behavior to each other (respectively varying between 0.73 – 0.81 with a mean value of 0.78 and 0.71-0.81 with a mean value of 0.77). A similar behavior characterized Class 2 and Class 3 which respectively had a range of variability between 0.84 – 0.90 with a mean value of 0.88 and 0.82-0.91 with a mean value of 0.87. Class 5, on the other hand, had both the worst mean precision value (0.49) and the largest IQR (Interquartile Range) interval (0.25 - 0.63).

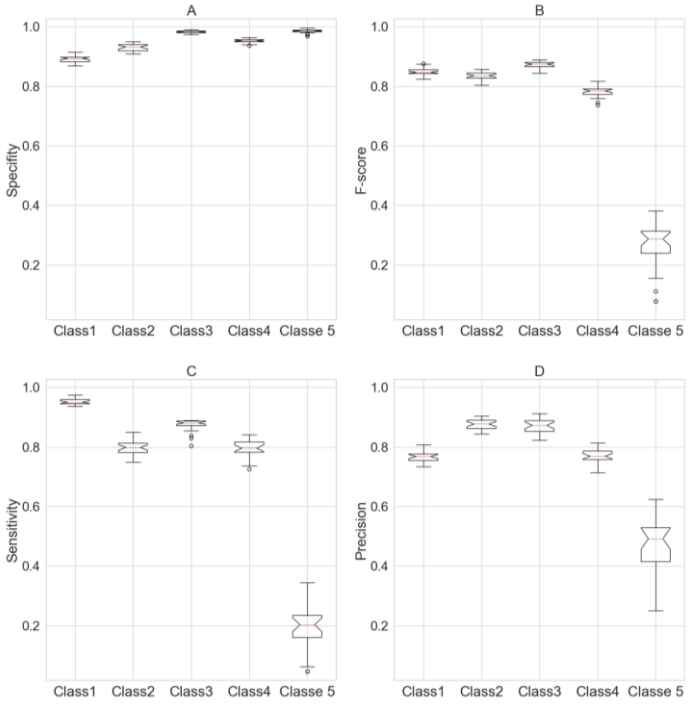

Figure 4.25: Specificity (panel A), F-score (panel B), Sensitivity (panel C) and Precision (panel D) are shown for the 30 2BLSTM2D network simulations. Each boxplot presents the median, the 25th and 75th percentiles, with the whiskers representing the maximum and minimum non-outlier values.

Mean Confusion Matrix

Target \ Output	1	2	3	4	5
1	95.52%	2.61%	0.0%	0.075%	1.11%
2	13.44%	80.10%	0.53%	4.30%	1.61%
3	0.86%	0.0%	87.93%	9.48%	1.72%
4	6.79%	6.17%	5.55%	80.24%	1.23%
5	23.80%	38.09%	4.76%	14.28%	19.05%

Figure 4.26: mean confusion matrix obtained from the 30 3BGRU3D network simulations.

Figure 4.26 shows the mean confusion matrix computed over the 30 network simulations performed. Observing the classification results, the major misclassifications were in Class 5, followed by Class 4 and Class 2, respectively with a percentage of correct classification of 19.05%, 80.24% and 80.10%. Class 2 were mainly misclassified with Class 1 (13.44%) and in Class 4 (4.30%). The classifier worse performs in the identification of Class 5, which was confused with all the other four Classes (23.80%, 38.09%, 4.76% and 14.28%, respectively Class 1, Class 2, Class 3 and Class 4). The best identified class was Class 1 (standing posture), followed by Class 3 (lying down posture).

2BLSTM2D trained and tested on 52 features database described by 4 classes

Training and Testing datasets

During the training, the model received as input 8,511 sequences, each one composed by 120 frames, with a 50% of overlapping and described by the 52 features, illustrated in the paragraph 4.4.1. Each sequence was described only by one class (it was a sequence-to-last model) and the total number of classes was 4 (Class 1: standing posture, Class 2: sitting posture, Class 3: lying down posture and Class 4: dangerous sitting posture). The test sequences were instead 989 in total, built exactly like the training ones, except for the overlap factor which was not present here. The model employed was the 2BLSTM2D (Figure 4.16).

Accuracy

The overall total accuracy reached by 30 simulations of this network varied across 0.80 - 0.87 with a mean value of 0. 85 ± 0.013 (SD).

Mean error ratios

In Figure 4.27, it is possible to observe the mean classification errors, normalized by the total number of frames in each class, over the 30 network simulations. The smallest mean error was found in Class 1 (0.07±0.06), followed by Class 4 (0.16±0.08) and Class 3 (0.19±0.08). Class 2 (0.20±0.12) was the one with the worst mean error. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labeled as transition, which have been classified in that class. They were normalized for the total number of errors of each class. The higher ratio of errors was present in Class 2 (0.03±0.16), while the lower error was in Class 4 (0.012±0.05).

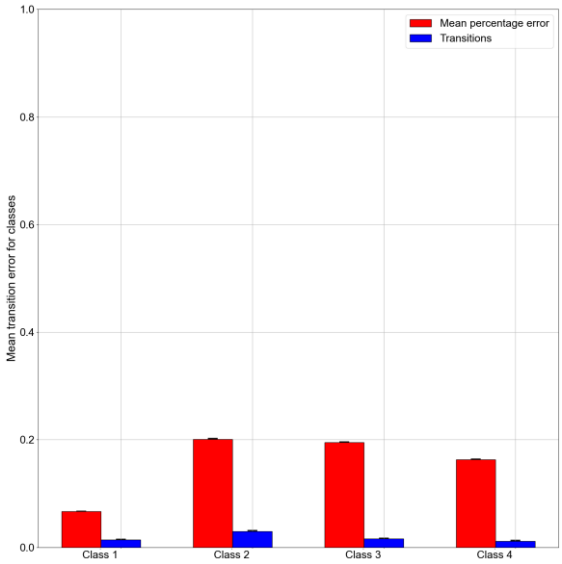

Figure 4.27: mean classification errors for each class. The errors for each class (red bars) were normalized for the number of total frames belonging to each class. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labeled as transition, which have been classified in that class.

Specificity, F-score, Sensitivity and Precision

Figure 4.28 shows the distribution of the results over the 30 simulations, separately for each class, for the Specificity, F-score, Sensitivity and Precision statistical parameters. The Specificity results (Panel A, Figure 4.28) ranged between 0.87 – 0.99, respectively with a mean value equal to 0.92 for Class1, 0.93 for Class 2, 0.98 for Class 3 and 0.95 for Class 4. The best Specificity value was reached with Class 3 (ranging between 0.97-0.99 and with mean value equal to 0.98), vice versa the worst result was with Class 1 (ranging between 0.88-0.95 and with mean value equal to 0.91). For what concerns the F-score parameter (panel B, Figure 4.28), it ranged 0.77 – 0.91 in the first three classes (respectively with a mean value of 0.87 for Class 1, 0.84 for Class 2 and 0.83 for Class 3), showing a decrease in Class 4 (ranging between 0.76-0.86 and a mean value of 0.81). The Sensitivity parameter (panel C, Figure 4.28) reached the greatest value in Class 1, where it ranged 0.89-0.97 and a mean value equal to 0.93. The other three parameters, instead, varied between 0.67 and 0.94, respectively with a mean value equal to 0.80 for Class 2, 0.81 for Class 3 and 0.84 for Class 4. In the Precision results (panel D, Figure 4.28) Class 2 had been proven to be the most precise class (varies between 0.82 and 0.95 with a mean value of 0.89), followed by Class 3 and Class 1 (ranging 0.75 – 0.92, respectively with a mean value of 0.82 for Class 1 and 0.87 for Class 3). Class 4 presented instead the worst results in term of precision, showing a mean value of 0.79, varying between 0.73-0.86.

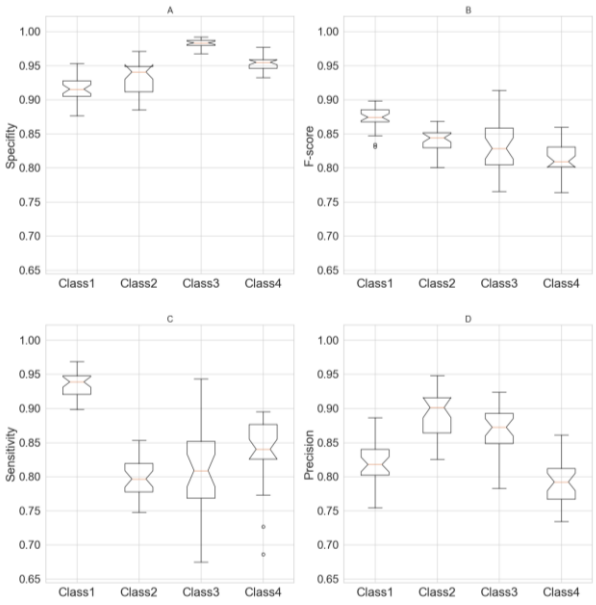

Figure 4.28: Specificity (panel A), F-score (panel B), Sensitivity (panel C) and Precision (panel D) are shown for the 30 2BLSTM2D network simulations. Each boxplot presents the median, the 25th and 75th percentiles, with the whiskers representing the maximum and minimum non-outlier values.

Mean Confusion Matrix

Target \ Output	1	2	3	4
1	93.35%	5.59%	0.0%	1.04%
2	11.73%	79.70%	1.95%	6.60%
3	3.32%	8.94%	80.48%	7.31%
4	4.65%	7.55%	4. 06%	83.72%

Figure 4.29: mean confusion matrix obtained from the 30 2BLSTM2D network simulations.

Figure 4.29 shows the mean confusion matrix computed over the 30 network simulations performed. Observing the classification results, the major misclassifications were in Class 2, Class 3 followed by Class 4, respectively with a percentage of correct classification of 79.70%, 80.48% and 83.79%. Class 2 was mainly misclassified with Class 1 (11.73%) followed by Class 2 with Class 4 vice versa (respectively, 6.60% and 7.55%). The best identified class was Class 1 (standing posture), followed by Class 4 (dangerous sitting posture).

3BGRU3D trained and tested on 52 features database described by 4 classes

Training and Testing datasets

During the training, the model received as input 8,511 sequences, each one composed by 120 frames, with a 50% of overlapping and described by the 52 features illustrated in the paragraph 4.4.1. Each sequence was described only by one class (it was a sequence-to-last model) and the total number of classes is 4 (Class 1: standing posture, Class 2: sitting posture, Class 3: lying down posture and Class 4: dangerous sitting posture). The test sequences were instead 989 in total, built exactly like the training ones, except for the overlap factor which was not present here. The model employed was the 3BGRU3D (Figure 4.20).

Accuracy

The total overall accuracy reached by 30 simulations of this network varied between 0.85 and 0.88 with a mean value of 0. 87 ± 0.008 (SD).

Mean error ratios

Figure 4.30 shows the mean classification errors, normalized by the total number of frames in each class, over the 30 network simulations. The smallest mean error was found in Class 1 (0.09±0.02), followed by Class 3 (0.13). Class 2 and Class 4 had the worst mean error ratio, respectively equal to 0.15±0.09 and 0.17±0.14. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labeled as transition, which have been classified in that class. They were normalized for the total number of errors of each class. The higher ratio was present in Class 2 (0.03±0.12), while the lower was in Class 3 (0.016±0.14) and Class 4 (0.017±0.08).

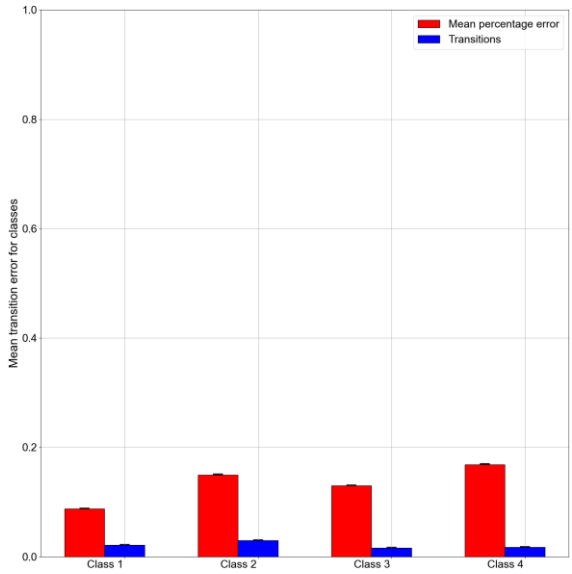

Figure 4.30: mean classification errors for each class. The errors for each class (red bars) were normalized for the number of total frames belonging to each class. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labeled as transition, which have been classified in that class.

Specificity, F-score, Sensitivity and Precision

Figure 4.31 shows the distribution of the results over the 30 simulations, separated for each class, for the Specificity, F-score, Sensitivity and Precision statistical parameters. Observing the distribution of the Specificity results (Panel A, Figure 4.31), it is possible to notice that Class 1 and Class 2 had close values, ranging between 0.94 – 0.97 and respectively with a mean value equal to 0.0.94 and 0.93, while, however, Class 3 and Class 4 had higher values ranging between 0.95 – 0.99 and mean values equal to 0.98 and 0.96. For what concerns the F-score parameter (Panel B, Figure 4.31), it ranged between 0.84 – 0.91 in the first three classes showing a decrease, with respectively a mean value equal to 0.88 for Class 1, 0.87 for Class 2 and 0.87 for Class 3. Class 4 ranged between 0.77 – 0.85, with mean value of 0.82. The Sensitivity parameter (panel C, Figure 4.31) reached the greatest value in Class 1, where it ranged between 0.88-0.96 with a mean value of 0.91. The other three classes, instead varied between 0.89 – 0.80, with a mean value equal to 0.0.85 for Class 2, 0.87 for Class 3 and 0.83 for Class 4. In the Precision results (panel D, Figure 4.31) Class 2 had been proven to be the most precise (varies between 0.85 – 0.95), followed by Class 3 and Class 1, respectively with a mean value equal to 0.88 (varies between 0.83 and 0.93) and 0.86 (varies between 0.80 and 0.91). Class 4 presented instead the worst results in term of precision, showing a mean value of 0.81 and a IQR interval ranging between 0.77-0.86.

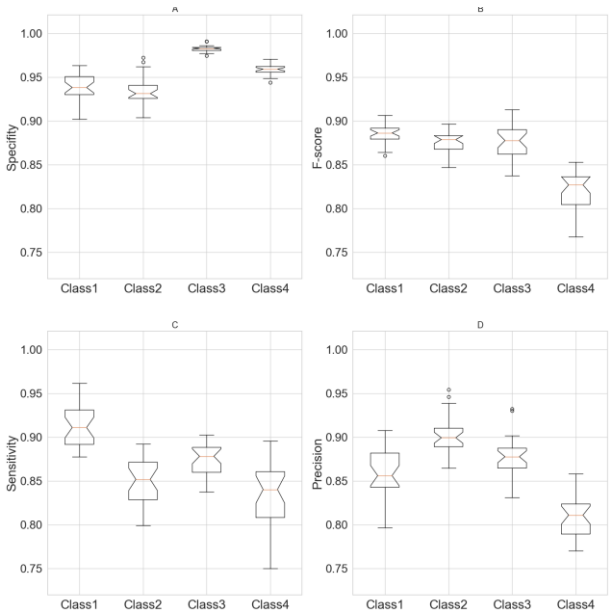

Figure 4.31: Specificity (panel A), F-score (panel B), Sensitivity (panel C) and Precision (panel D) are shown for the 30 3BGRU3D network simulations. Each boxplot presents the median, the 25th and 75th percentiles, with the whiskers representing the maximum and minimum non-outlier values.

Mean confusion matrix

Target \ Output	1	2	3	4
1	91.26%	7.34%	0.0%	1.40%
2	9.06%	85.04%	1.71%	4. 16%
3	0.81%	1.62%	87.80%	9.76%
4	2.90%	8.72%	5.23%	83.13%

Figure 4.32: mean confusion matrix obtained from the 30 3BGRU3D network simulations.

Figure 4.32 shows the mean confusion matrix computed over the 30 network simulations performed. Observing the classification results, the

major misclassifications were in Class 4, Class 2 followed by Class 3, respectively with a percentage of correct classification of 83.13%, 85.04% and 87.80%. Class 2 was mainly misclassified with Class 1 (9.06%) and vice versa (7.34%), followed by Class 2 with Class 4 (4.16%) vice versa (8.72%). Class 4 was also misclassified with Class 3 (5.25%) and vice versa (9.76%). The best identified class was Class 1 (standing posture), followed by Class 3 (lying down posture) and Class 2 (sitting posture).

3BGRU3D trained and tested on 52 features database described by 4 classes, first solution of Data Augmentation

Training and Testing datasets

Observing the imbalance of the classes, especially Class 3 and Class 4, (Figure 4.33), we decided to apply the mentioned data augmentation technique, only to Class 3 and Class 4 (paragraph 4.4.2). In this case, however, in order to feed the network with a homogeneous number of examples for each class, we chose to cut the number of input sequences to 2,032 (red line, Figure 4.33). During the training, the model received as input 8,128 sequences, each one composed by 120 frames, with a 50% of overlapping and described by the 52 features, illustrated in the paragraph 4.4.1. Each sequence was described only by one class (it was a sequence-to-last model) and the total number of classes was 4 (Class 1: standing posture, Class 2: sitting posture, Class 3: lying down posture and Class 4: dangerous sitting posture). The test sequences were instead 989 in total, built exactly like the training ones, except for the overlap factor which was not present here. The model employed was the 3BGRU3D (Figure 4.20).

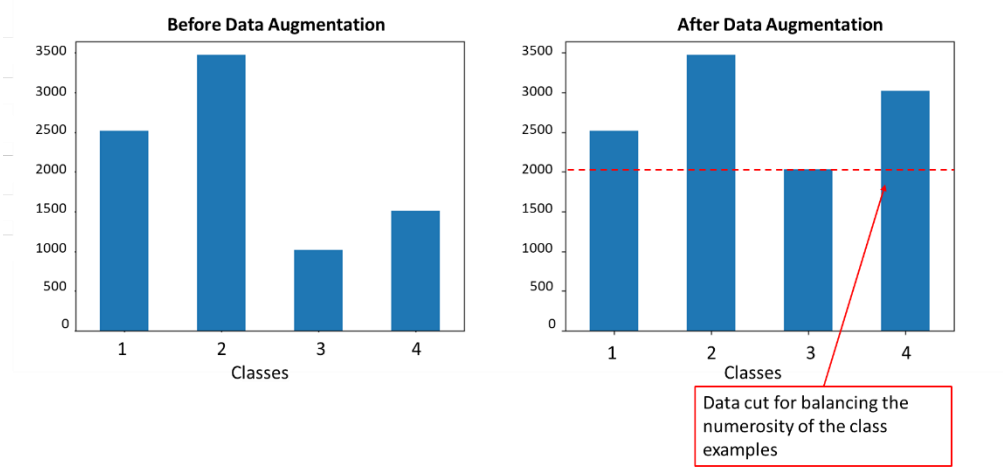

Figure 4.33: the numerosity of the sequences (each composed by 120 frames with a 50% of overlapping) which constitute the training database, before and after a single application of data augmentation to class 3 and 4 sequences. The sequences are cut for dataset class uniformity (red line).

Accuracy

The overall total accuracy reached by 30 simulations of this network varied across 0.78 - 0.83 with a mean value of 0. 81 ± 0.012 (SD).

Mean error ratios

In Figure 4.34, it is possible to observe the mean classification errors, normalized by the total number of frames in each class, over the 30 network simulations. The smallest mean error was found in Class 1 (0.09±0.05), followed by Class 3 (0.17±0.02) and Class 4 (0.20±0.5). Class 2 had the worst mean error ratio equal to 0.27±0.1. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labeled as transition, which have been classified in that class. They were normalized for the total number of errors of each class. The higher ratio was present in Class 2 (0.042±0.1), while the lower was in Class 4 (0.017±0.04).

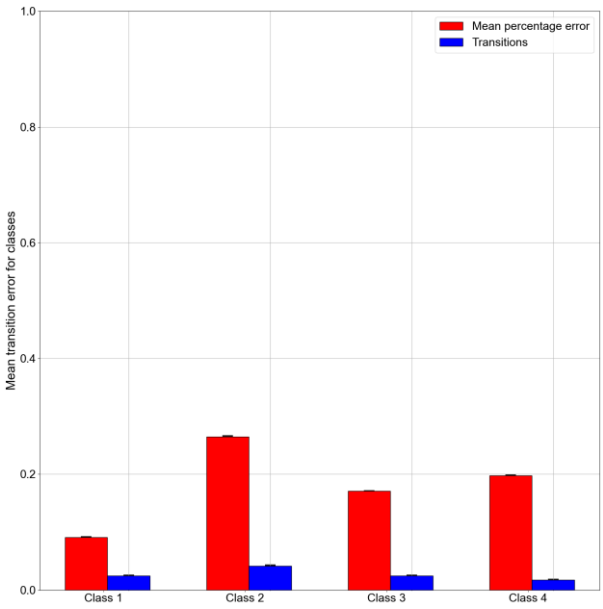

Figure 4.34: mean classification errors for each class. The errors for each class (red bars) were normalized for the number of total frames belonging to each class. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labelled as transition, which have been classified in that class.

Specificity, F-score, Sensitivity and Precision

Figure 4.35 shows the distribution of the results over the 30 simulations, separated for each class, for the Specificity, F-score, Sensitivity and Precision statistical parameters. The results of the Specificity (panel A,

Figure 4.35) values show similar trends between Class 2 and Class 4 (ranging between 0.88 – 0.90, respectively with a mean value equal to 0.93 and 0.92). Class 3, on the other hand, was the parameter that reached the highest mean value (0.99, ranging between 0.98-0.99), while Class 1 the smallest (0.90, ranging between 0.88-0.93). For what concerns the F-score parameter (panel B, Figure 4.35), Class 1, Class 2 and Class 3 varied between 0.89 – 0.76, respectively with a mean value equal to 0.85, 0.80 and 0.86. Class 4, which had the lowest mean value (0.74), varied between 0.69 - 0.78. The Sensitivity parameter (panel C, Figure 17) reached the greatest value in Class 1, where it ranged between 0.87 - 0.95, with a mean value equal to 0.91. The other three parameters, instead varied between 0.70 – 0.88 and respectively with a mean value equal to 0.73 for Class 2, 0.83 for Class 3 and 0.80 for Class 4. Observing the distribution of the Precision results (Panel D, Figure 4.35), it is possible to notice that Class 2 and Class 3 had close values, ranging between 0.84 – 0.94 with a mean value equal to 0.88 for Class 2 and 0.89 for Class 3, while Class 1 had a mean value equal to 0.80 (ranging between 0.76 – 0.85). Class 4 had the smallest equal value (0.68), with a IQR range equal to 0.60 – 0.74.

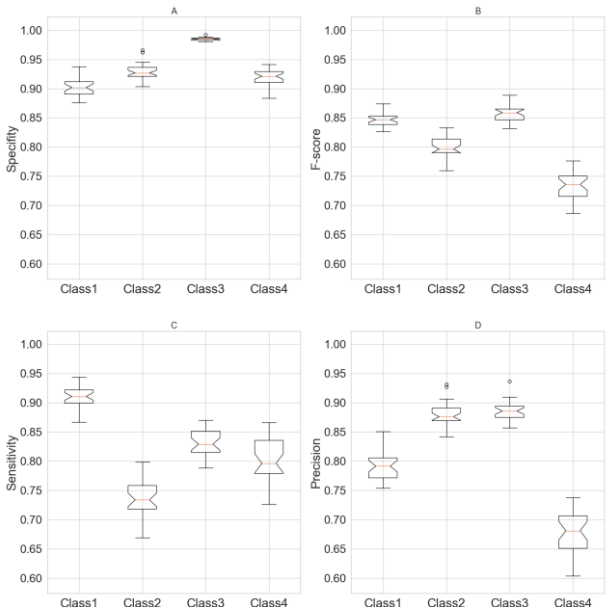

Figure 4.35: Specificity (panel A), F-score (panel B), Sensitivity (panel C) and Precision (panel D) are shown for the 30 3BGRU3D network simulations. Each boxplot presents the median, the 25th and 75th percentiles, with the whiskers representing the maximum and minimum non-outlier values.

Mean confusion matrix

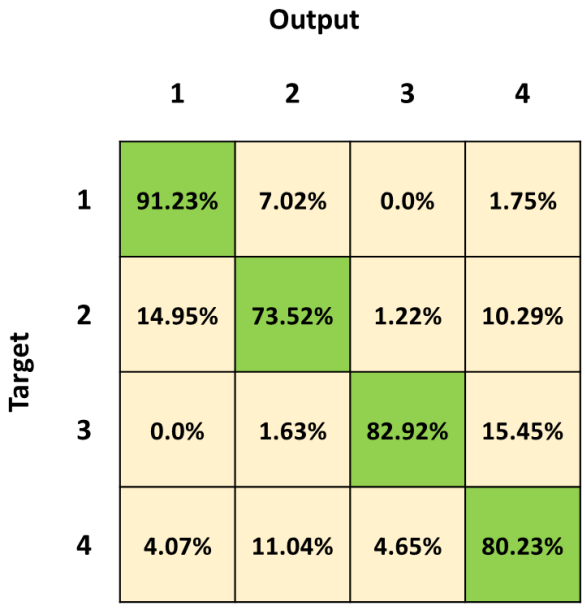

Figure 4.36. mean confusion matrix obtained from the 30 3BGRU3D network simulations.

Figure 4.36 shows the mean confusion matrix computed over the 30 network simulations performed. Observing the classification results, the major misclassifications were in Class 2, Class 4 followed by Class 3, respectively with a percentage of correct classification of 73.52%, 80.23% and 82.92%. Class 2 was mainly misclassified with Class 1 (14.95%) followed by Class 2 with Class 4 (10.28%) vice versa (11.04%). Class 3 was misclassified instead with Class 4 (15.45%), and Class 1 with Class 2 (7.02%). The best identified class was Class 1 (standing posture) and Class 3 (lying down posture), followed by Class 4 (dangerous sitting posture).

3BGRU3D trained and tested on 52 features database described by 4 classes, second solution of Data Augmentation

Training and Testing datasets

We decided to apply again the Data Augmentation technique, only to Class 3 and Class 4. According to that, we cut the number of input sequences to 2,516 (red line, Figure 4.37). During the training, the model received as input 10,064 sequences, each one composed by 120 frames, with a 50% of overlapping and described by the 52 features illustrated in the paragraph (Chapter 2, paragraph 4.4.1). Each sequence was described only by one class

(it was a sequence-to-last model) and the total number of classes is 4 (Class 1: standing posture, Class 2: sitting posture, Class 3: lying down posture and Class 4: dangerous sitting posture). The test sequences were instead 989 in total, built exactly like the training ones, except for the overlap factor which was not present here. The model employed was the 3BGRU3D (Figure 4.20).

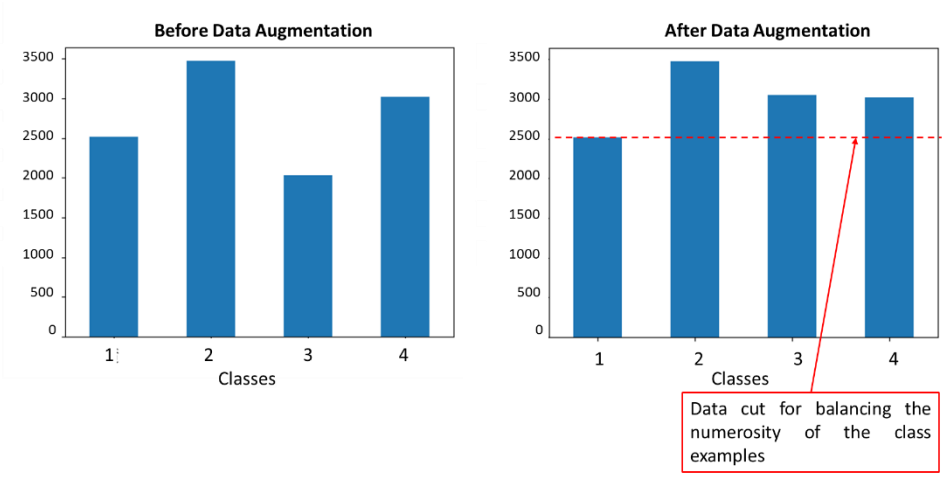

Figure 4.37: the numerosity of the sequences (each composed by 120 frames with a 50% of overlapping) which constitute the training database, before and after a single application of data augmentation to class 4 sequences and a second application to class 3 sequences. The sequences were cut for dataset class uniformity (red line).

Accuracy

The overall total accuracy reached by 30 simulations of this network varied across 0.80 – 0.86 with a mean value of 0. 83 ± 0.017 (SD).

Mean error ratios

In Figure 4.38, it is possible to observe the mean classification errors, normalized by the total number of frames in each class, over the 30 network simulations. The smallest mean error was found in Class 1 (0.08±0.04), followed by Class 3 (0.15±0.04) and Class 4 (0.18±0.06). Class 2 had the worst mean error ratio equal to 0.25±0.1. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labeled as transition, which have been classified in that class. They were normalized for the total number of errors of each class. The higher ratio was present in Class 2 (0.04±0.12), while the lower was in Class 1 (0.024±0.03) and Class 3 (0.024±0.07).

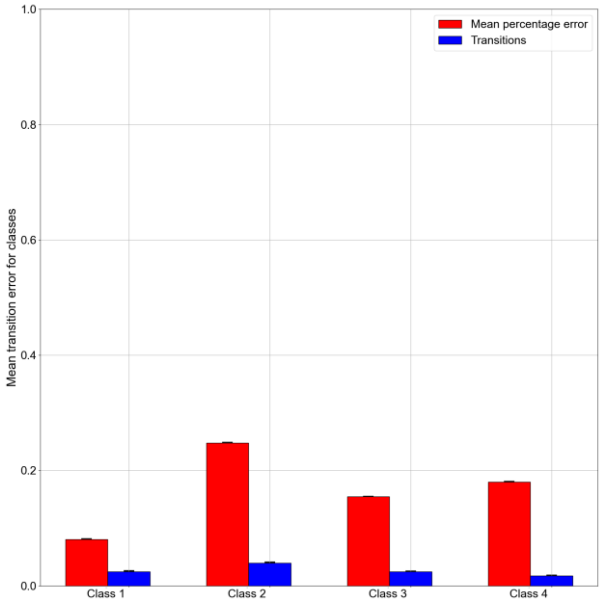

Figure 4.38: mean classification errors for each class. The errors for each class (red bars) were normalized for the number of total frames belonging to each class. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labeled as transition, which have been classified in that class.

Specificity, F-score, Sensitivity and Precision

Figure 4.39 shows the distribution of the results over the 30 simulations, separated for each class, for the Specificity, F-score, Sensitivity and Precision statistical parameters. The Specificity results (panel A, Figure 4.39), compared to that of the other three statistical metrics (panel B, C and D, respectively, Figure 4.39) were very close to each other and ranged 0.88 – 0.99 and respectively with a mean equal to 0.90 for Class 1, 0.94 for Class 2, 0.98 for Class 3 and 0.93 for Class 4. For what concerned the F-score parameter (panel B, Figure 4.39), it ranged 0.78 – 0.91 in the first three classes (respectively with equal to 0.85 for Class 1, 0.82 for Class 2 and 0.86 for class 3) showing a slight decrease in Class 4 (mean value equal to 0.76, ranging between 0.71-0.82). The Sensitivity parameter (panel C, Figure 4.39) reached the greatest value in Class 1 (0.92), where it ranged between 0.89-0.96. Class 3 and Class 4, instead varied between 0.73 – 0.88 with a mean value respectively equal to 0.84 and 0.82. Class 2 ranged between 0.78-0.83, with a mean value equal to 0.70. In Precision (panel D, Figure 4.39) parameters, Class 4 reached the smallest mean value respectively equal to (0.71) and it ranged between 0.67-0.77. Class 2 and Class 3 had close mean value, respectively 0.90 and 0.87, ranging between 0.84-0.94. Regarding Class 1, it ranged between 0.76-0.88 and the mean value was equal to 0.79.

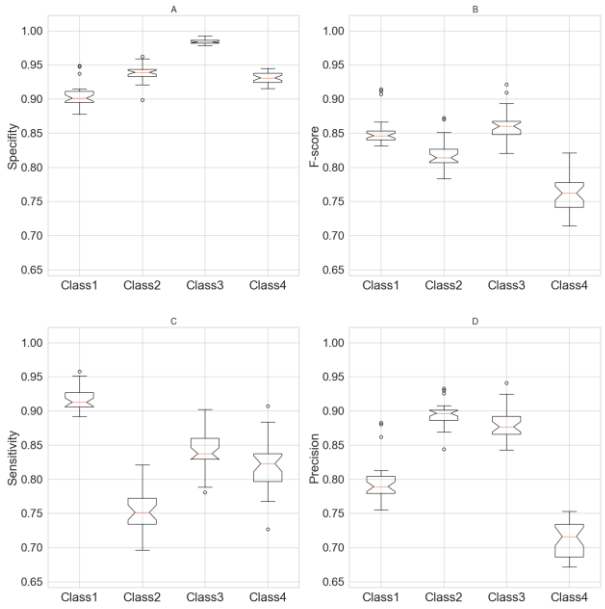

Figure 4.39: Specificity (panel A), F-score (panel B), Sensitivity (panel C) and Precision (panel D) are shown for the 15 3BGRU3D network simulations. Each boxplot presents the median, the 25th and 75th percentiles, with the whiskers representing the maximum and minimum non-outlier values.

Mean confusion matrix

Target \ Output	1	2	3	4
1	91.63%	6.62%	0.0%	1.74%
2	14.46%	75.24%	1.71%	8.57%
3	0.0%	1.62%	84.55%	13.82%
4	4.67%	8.77%	4.09%	82.45%

Figure 4.40: mean confusion matrix obtained from the 30 3BGRU3D network simulations.

Figure 4.40 shows the mean confusion matrix computed over the 30 network simulations performed. Observing the classification results, the

major misclassifications were in Class 2, Class 4, respectively with a percentage of correct classification of 75.24%, and 82.55%. Class 2 was mainly misclassified with Class 1 (14.46%) and vice versa (6.42%), followed by Class 2 with Class 4 (8.57%) vice versa (8.77%). The best identified class was Class 1 (standing posture), followed by Class 3 (lying down posture).

3BGRU3D trained and tested on 52 features database described by 4 classes, third solution of Data Augmentation

Training and Testing datasets

We decided not to cut the sequences but utilized as input the augmented data obtained in the previous paragraph. According to that, during the training, the model received as input 12,052 sequences, each one composed by 120 frames, with a 50% of overlapping and described by the 52 features illustrated in the paragraph (paragraph 4.4.1). Each sequence was described only by one class (it is a sequence-to-last model) and the total number of classes was 4 (Class 1: standing posture, Class 2: sitting posture, Class 3: lying down posture and Class 4: dangerous sitting posture). The test sequences were instead 989 in total, built exactly like the training ones, except for the overlap factor which was not present here. The model employed was the 3BGRU3D (Figure 4.20).

Accuracy

The overall total accuracy reached by 30 simulations of this network varied across 0.85 - 0.92 with a mean value of 0.88 ± 0.012 (SD).

Mean ratios errors

In Figure 4.41, it is possible to observe the mean classification errors, normalized by the total number of frames in each class, over the 30 network simulations. The smallest mean error was found in Class 1 (0.06±0.09), followed by Class 3 (0.11±0.04) and Class 4 (0.13±0.09). Class 2 had the worst mean percentage error equal to 0.15±0.13. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labeled as transition, which have been classified in that class. They were normalized for the total number of errors of each class. The higher ratio was present in Class 2 (0.03±0.14), while the lower was in Class 3 (0.016±0.06)

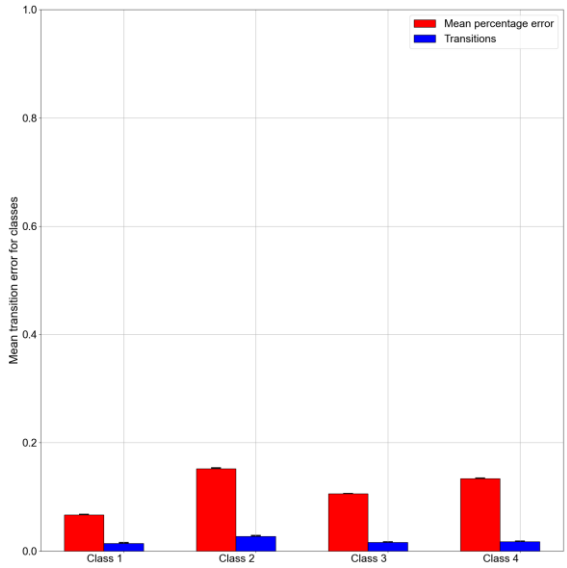

Figure 4.41: mean classification errors for each class. The errors for each class (red bars) were normalized for the number of total frames belonging to each class. The blue bars, for each class, represent the portion of the total error corresponding to the frames previously labelled as transition, which have been classified in that class.

Specificity, F-score, Sensitivity and Precision

Figure 4.42 shows the distribution of the results over the 30 simulations, separated for each class, for the Specificity, F-score, Sensitivity and Precision statistical parameters. The Specificity results (panel A, Figure 4.42), compared to that of the other three statistical metrics (panel B, C and D, respectively, Figure 4.42) were very close to each other and ranged 0.90 – 0.99 and respectively with a mean value equal to 0.95 for Class 1, 0.95 for Class 2, 0.99 for Class 3 and 0.95 for Class 4. For what concerned the F-score parameter (panel B, Figure 4.42), it ranged 0.85– 0.93 in the first three classes (with a mean value equal to 0.90 for Class1, 0.88 for Class 2 and 0.90 for Class 3) showing a small decrease in Class 4 (mean value equal to 0.83, ranging between 0.76-0.88). The Sensitivity parameter (panel C, Figure 4.42) reached the greatest value in Class 1, where it ranged between 0.96-0.88, with a mean value equal to 0.93. Class 3 and Class 4, instead vary between 0.77 – 0.83 respectively with a men value of 0.89 and 0.85. Class 2 ranged between 0.89 – 0.81, with a mean value equal to 0.85. In Precision parameters, Class 4 reached the smallest mean value equal to (0.79) and it ranged between 0.71– 0.84. Class 2 and Class 3 had close mean value, respectively 0.91 and 0.92. Class 3 had an IQR range that varied between 0.88 – 0.95, and Class 2 instead ranged between 0.86 – 0.96. Regarding Class 1, it ranged between 0.83 – 0.94 and the mean value was equal to 0.87.

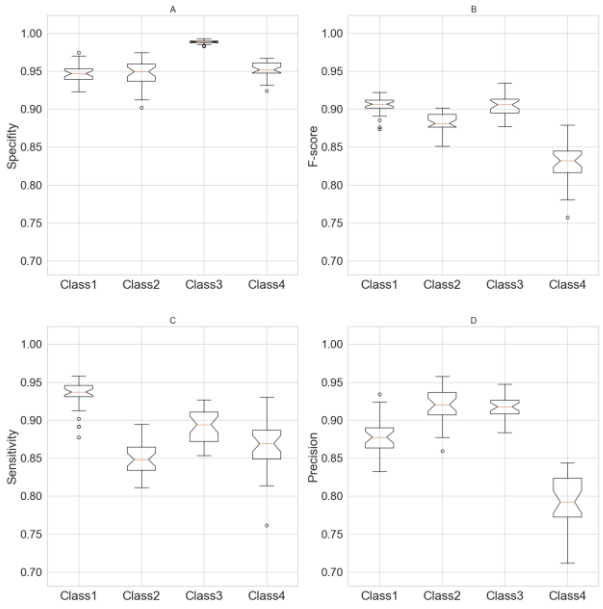

Figure 4.42: Specificity (panel A), F-score (panel B), Sensitivity (panel C) and Precision (panel D) are shown for the 30 3BGRU3D network simulations. Each boxplot presents the median, the 25th and 75th percentiles, with the whiskers representing the maximum and minimum non-outlier values.

Mean confusion matrix

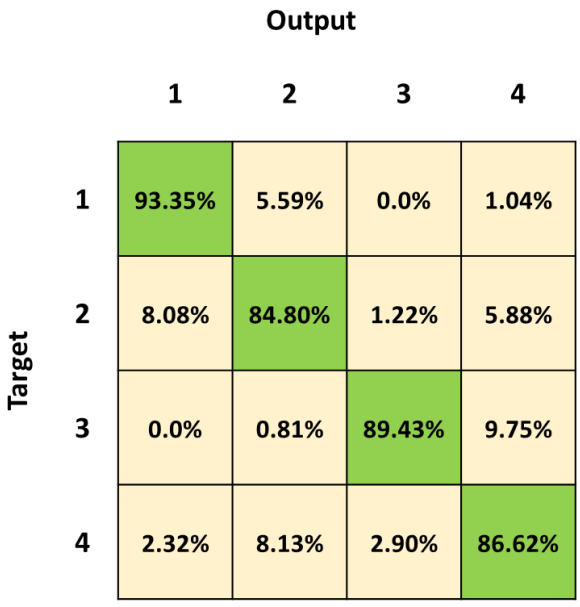

Figure 4.43: mean confusion matrix obtained from the 30 3BGRU3D network simulations.

Figure 4.43 shows the mean confusion matrix computed over the 30 network simulations performed. Observing the classification results, the

major misclassifications were in Class 2, and Class 4, respectively with a percentage of correct classification of 84.80% and 86.62%. Class 2 was mainly misclassified with Class 1 (8.08%), followed by Class 2 with Class 4 (5.88%) vice versa (8.13%). Class 1 was misclassified with Class 2 (5.59%) and Class 3 with Class 4 (9.75%). The best identified class was Class 1 (standing posture), followed by Class 3 (lying down posture).

4.5.3. Discussion

In this solution we proposed a new RNN network model based on bidirectional GRU neural networks, called 3BGRU3D. This new architecture, unlike the previous ones, was a sequence-to-last model, in which each input sequence to the network is identified with a single class.

As first step, in order to compare the performance of the new 3BGRU3D model with that of the 2BLSTM2D, the 3BGRU3D model was trained and tested with the old database considering 8 features (paragraph 4.3.6, LSTM Sequence neural network), and labeled with 5 classes (Class 1: standing posture, Class 2: sitting posture, Class 3: lying down posture, Class 4: dangerous sitting posture and Class 5: transition posture). We trained and tested the 3BGRU3D model for 30 simulations and 60 epochs each, achieving a mean accuracy of 82%, unlike the 2BLSTM2D model with which we obtained a mean accuracy of 85% (paragraph 4.3.6, LSTM Sequence neural network). This decrease in the results was confirmed by the different distribution of the statistical parameters calculated over the 30 simulations (Figures 4.17 and 4.22). The 2BLSTM2D model obtained better results in terms of Specificity, F-score and Sensitivity (Figures 4.40 and 4.22). For example, in the Sensitivity parameter of the new 3BGRU3D model, all 5 classes had lower mean values than with the 2BLSTM2D model. In particular, with the 2BLSTM2D model, the IQR value of the four classes ranged 0.92-0.99, instead, with the new 3BGRU3D model, the IQR interval of the four classes varied in a greater range equal to 0.97 - 0.66. Moreover, observing the confusion matrices of both models (Figures 4.18 and 4.23), we could say that the new 3BGRU3D model compared to the 2BLSTM2D model, performed worst in Class 1 and Class 4, respectively 88.78% and 71.84%, compared to 92.30% and 81.94%, while it performed better in Class 3 and Class 5, with 96.92% and 62.69% compared to 93.73% and 53.69%, respectively. We then proposed a new set of features (the 51 joint ego centric coordinates plus the Euclidean distance of the subject from the camera) and a new data labelling, in which the transition between two consecutive postures was identified with the posture following the transition. We trained and tested the 2BLSTM2D model (Figure 4.16) configured for sequence-to-last classification with the new database just described, for 30 simulations and 60 epochs each, in order to compare the performances obtained with the two different databases, i.e., the old one described by 8 features and with 5 classes and the new one described by 52 features and with 4 classes. The 2BLSTM2D model trained and tested with the new database achieved a mean

classification accuracy equal to 85%, the same mean accuracy obtained with previous 2BLSTM2D model trained and tested with the old database. With the old database, however, the best identified class was Class 3, followed by Class 1 and Class 4. The worst classes were instead the two remaining ones, i.e., Class 4 followed by Class 5. On the other hand, when trained and tested with new database, instead, the best identified classes were Class 1, Class 4 and Class 3 and the worst one was Class 2. Furthermore, observing confusion matrices (Figures 4.18 and 4.36) in the previous 2BLSTM2D model trained with the old database, Class 2 was more misclassified with Class 4 and vice-versa (respectively 9.39% and 8.20%, see Figure 4.18), while in the 2BLSTM2D model trained and tested with new database, it was more misclassified with Class 1 (11.73% percentages of misclassifications, see Figure 4.36). This last misclassification could be partly caused by the new labeling. By examining the blue bars in Figure 4.27 of Class 1 and Class 2 (respectively, 0.017±0.05 and 0.03±0.16), it is possible to say that some frames, during the transition process, had very similar characteristics that led the network model to be confused between the two classes (Figure 4.28). In the previous 2BLSTM2D model trained with the old dataset, Class 2 was also particularly misclassified with Class 3 (9.39%, see Figure 4.18), while with the new database it only misclassified a small percentage of examples (1.95%, see Figure 4.29). This means that the new dataset helped the previous 2BLSTM2D network model to better distinguish the sitting posture from the lying one.

The new 3BGRU3D model was trained and tested on the new database, labeled with the 4 classes, for a total number of 30 simulations and 60 epochs each. In this case, a mean accuracy on the test database of 87% was achieved. This turned out to be the best result achieved so far in terms of accuracy. To confirm this performance, we decided to test the model with the new database labeled with the 5 classes. After 30 simulations and 60 epochs each of the new 3BGRU3D model, we obtained an average accuracy of 82%, therefore a lower average accuracy value than the 2BLSTM2D model. The comparison between Figure 4.24 and Figure 4.30 confirms the lower performances of the new 3BGRU3D model trained with the new database, labeled with 5 classes. Figure 4.24 shows that the values of the mean percentage error were almost all higher than those shown in Figure 4.30 (except for Class 1, respectively 0.05±0.03 in Figure 4.24 and 0.09±0.02 in Figure 4.30). Figure 4.24 also shows that the most misclassified class was Class 5, with a mean error ratio of 0.81±0.1 and a mean precision value of 0.47±0.09, reaching even a minimum value of 0.25 (Figure 4.25, panel D).

On the other hand, comparing the 2BLSTM2D and the new 3BGRU3D models, both trained and tested with the new database labeled with 4 classes, showed that the misclassification error between Class 2 and Class 1 in the new 3BGRU3D model decreased (Figures 4.29 and Figure 4.32). This was confirmed by the mean precision for Class 1, which went from a value of 0.82±0.04 to 0.86±0.03 (Figures 4.29 and Figure 4.32). In addition, the mean precision value of Class 4 also improved over the previous 2BLSTM2D, from 0.79±0.03 to 0.81±0.02 (Figures 4.29 and 4.32). This is a very

important characteristic since Class 4, i.e., dangerous sitting posture, allows an initial distinction between a situation of possible danger, and a situation of normal everyday life. The difference between the accuracy values of the two models was confirmed by the T-test result ($t(58) = -7.334$, $p < 0.001$).

In addition, to better generalize the new 3BGRU3D model, noting that in the training database the sequences described by the classes were highly unbalanced (Figure 4.33), we decided to apply the technique of data augmentation. We implemented a data augmentation method based on adding Gaussian noise only to the training sequences identified with Class 3 and Class 4, i.e., those with fewer instances (Figure 4.33). Once the data augmentation technique was applied, to obtain a balanced number of sequences in the training database, we decided to limit the number of sequences to 2,032 for each class in the database (Figures 4.33), resulting in a total number of sequences of 8,128. Then, we trained and tested the new 3BGRU3D model with the balanced data for 30 simulations and 60 epoch each, thus obtaining an average accuracy of 81%. Comparing the latter result with the previous one reached by the new 3BGRU3D model without data augmentation, we found a decrease at the level of model performance (from 87% to 81%). This is also confirmed by the increase in the mean error (Figures 4.30 and 4.34), especially regarding sitting posture (respectively 0.15±0.09 and 0.27±0.1) Moreover, Figures 4.32 and 4.36 show an increase in the misclassification between Class 2 and Class 4 (from 4.16% to 10.29%) and between Class 2 and Class 1 (from 9.06% to 14.95%). The decrease in performance, despite the data augmentation, could be caused by the decreased number of input sequences. In fact, after this step, the deep learning model receives fewer sequences as input than in the previous trial (from 8,511 to 8,128).

For that reason, we decided to apply the data augmentation technique again only to the sequences identified with Class 3, to be able to input a higher number of sequences for this class. In this case, truncating to balance the number of training sequences, we obtained a total number of 10,064 input sequences. We trained and tested the new 3BGRU3D model for 30 simulations and 60 epoch each, thus achieving a mean accuracy of 83%. In this trial, the mean accuracy value improved in comparison with the previous one (from 81% to 83%). The model especially improved in the misclassification of Class 4 (from 4.07%, 11.04% and 4.65% to 4.67% 8.77% 4.09%, respectively Class 1, Class 2 and Class 3 see Figures 4.36 and 4.40). However, the mean accuracy value compared with the new 3BGRU3D model trained without data augmentation is still lower (from 87% to 83%).

We then chose to train the new 3BGRU3D network with the augmented data, but without limiting the number of training input sequences (in total 12,052), to observe the behavior of the new 3BGRU3D model with a larger amount of input sequences. We trained and tested the model for 30 simulations. This solution achieved a mean accuracy of 88%. Compared with the last illustrated solution, the accuracy performance was remarkably improved, especially in terms of median error rate. Figure 4.41 shows a decrease in the mean error ratio for all classes, especially Class 2 (from

0.25±0.1 to 0.15±0.1, see Figures 4.38 and 4.41) and Class 4 (from 0.16±0.06 to 0.13±0.06, see Figures 4.38 and 4.41). Furthermore, the mean precision values (Figure 4.42, panel D), compared with those observed in Figure 4.39 panel D, changed between the two trials: Class 1 went from a value of 0.80±0.03 to a value of 0.86±0.02; Class 2 went from a value of 0.89±0.02 to a value of 0.92±0.02; Class 3 went from a value of 0.88±0.02 to a value of 0.92±0.01; Class 4 went from a value of 0.71±0.03 to a value of 0.78±0.03. This means that a higher number of input data helped to better generalize on new input provided, never seen by the model.

On the other hand, comparing this solution with the new 3BGRU3D model trained and tested without data augmentation, they achieved a close mean accuracy value, respectively 88% and 87%. Analyzing the results in more detail, the mean precision values of the last illustrated solution improved for Class 2 and Class 3, from a value of 0.90±0.02 to a value of 0.92±0.02 and a value of 0.88±0.02 to a value of 0.91±0.03, respectively (Figures 4.39 and 4.42, panel D). In contrast, in the 3BGRU3D model trained without data augmentation (Figure 4.39), the mean precision value of Class 4 decreased from a value of 0.81±0.02 to a value of 0.79±0.03. Observing, on the other hand, the Sensitivity of the new 3BGRU3D model trained with data augmentation, but with the uncut sequences (Figure 4.42, panel c) relative to the dangerous sitting posture (Class 4), the metric increased from a value of 0.84±0.02 to a value of 0.87±0.03. This was also confirmed by Figures 4.32 and 4.43. In particular the percentage of true positives related to Class 4 increased from a percentage of 83.13% to a percentage of 86.62%, and the percentage of false negatives related to Class 4 decreased from a percentage equal to 2.90%, 8.72% and 5.23% to a percentage of 2.82%, 8.13% and 2.90%, respectively for Class 1, Class 2 and Class 3 (Figures 4.32 and 4.43).

The result of the T-test demonstrated the difference in the accuracy distribution values between the 3BGRU3D model tested with the augmented data and without the augmented data ($t(58) = -4.348$, $p < 0.001$). Moreover, the model trained with the augmented data was better for our purpose since it improved performance on the identification of dangerous sitting posture (Class 4).

Chapter 5

Overall Conclusions and Future Developments

Over the past decade, there has been considerable and growing interest in the development of AAL systems to support independent living, especially for frail individuals. In addition, recent advances in IoT technologies and the reduced cost of sensors have encouraged the development of smart environments, such as smart homes, to improve individuals' quality of life, to enable frail people to live healthier and more independently for longer and to support caregivers and medical personnel. To provide such services, a smart home has to be able to understand the daily activities of its residents. HAR techniques, referred to as the art of using AI for identifying activities from the raw data gathered by utilizing various technologies, respond to such need.

In this thesis we have proposed the implementation of a posture classification system for monitoring frail individuals in their daily living facilities. This was developed using a set of four Kinect V2 devices recording the living space of the individual, and whose data are processed to identify dangerous situations (i.e., the subject has fainted or slipped from the wheel chair, etc.) to trigger an alarm toward third parties, when needed. Here we aimed at classifying the acquired skeleton data provided by the device as one in a set of predefined postures (standing, sitting, lying down and "dangerous sitting"). To achieve such goal, it was necessary to implement AI solutions trained using a large amount of skeleton tracking data referred to real scenarios. Therefore, we built a database consisting of over 600,000 frames in which each posture was described by a set of geometric and joint position features. Different solutions in term of features (type and number), AI algorithms and architectures have been defined and validate. First, a subset of 10 features, derived from the Kinect skeletal joints coordinates with respect to an absolute reference system, and then chosen using the ReliefF algorithm was used for training and testing the AI proposed algorithms (in Table 5.1 it is possible to see a summarization of some proposed AI solutions). The first solution was a three layers MLP neural network (2 hidden layers and a softmax output layer) with which we achieved a mean accuracy of 83.9 %. This result was not, however, satisfactory for our purpose since the classifier was intended as the core of a more complex

safety system aimed to generate an alarm when dangerous situations occur during everyday life inside a room. Therefore, attempting to increase the MLP performance, as a second solution, we proposed a data pre-processing algorithm based on velocity threshold and anthropometric constraints. In this case, the mean accuracy reached by the classifier was 92%, and it increased to 95% when the test data was averaged over 0.5s windows and the rate of classification was reduced to 2 Hz. This procedure, which performed so well, nevertheless had the weighty drawback of increasing the computational time considerably, making the process not useful for the online demand of the monitoring system. For that reason, we decided to propose a third solution which included a tuning of the previous MLP model, selecting a new subset of features using a SVM algorithm, and a LSTM sequence-to-sequence network model working on a subset of features selected using a genetic algorithm. In this third solutions we also decided to include as a possible target classification a fifth class, called "transition", collecting all the transitions between two consecutive postures, in order to be able to provide the network with a continuous stream of data while reducing the risk of incurring in false positives during such transition movements, e.g., while sitting down. For the same reasons, the five-classes dataset considered to train the LSTM sequence network included the frames in which the Kinect was not able to identify a proper skeleton, which were filled with 999 values. On this more ecological, yet more challenging, dataset the best MLP model (3 hidden layers and a softmax output layer) achieved an overall test set accuracy of 78.4%, while the best LSTM architecture, called 2BLSTM2D (2 bidirectional LSTM layers, 2 dropout and a fully connected layer), reached 85.7%.

As a last solution we defined a new set of features based on the joint positions computed with respect to an egocentric reference system plus the Euclidean distance between the Kinect V2 camera and the subject. A new labelling of data, incorporating the transition between two consecutive postures with the posture closely following the transition, was made. Moreover, we implemented, through a more extensive investigation on deep leaning models, a new deep a sequence-to-last RNN solution based on GRU models. We thus obtained a network model for the classification of 4 postures (standing, sitting, lying down and dangerous sitting). Starting from this last solution, we concluded the work implementing a data augmentation method on the training data. This latter was done adding Gaussian noise, to minimize the strong imbalance between classes (especially for increasing the sequences representing the dangerous sitting posture and the lying down posture). Trained with this enlarged set of data the 3BGRU3D (3 bidirectional GRU layers, 3 dropout and 3 fully connected layers), the so far proven best model, reached a mean accuracy of 87%, with a maximum value of 89%. This solution was considered the best because the classification of the dangerous sitting posture produced a higher number of true positives and lower number of false negatives. This result was of particular importance for our monitoring purpose, allowing us an initial distinction between a situation of possible danger and a situation of normal daily life. Nevertheless, when

the other three classes are identified, the output of the model has to be necessarily fed as input to a final classification system taking into account the location of the patient in the room to decide whether or not to trigger an alarm.

As a final consideration, the results reported in the literature and mentioned in Chapter 2, paragraph 2.8, appear to achieve very high accuracy rates, typically around or over 90%. These are generally higher than those reported in our work and are obtained on a larger number of classes, thus a more in-depth understanding of these experiments in comparison to ours is called for. The differences between these studies and the one presented in this work are broad and regard both the acquisition protocol, and hence the resulting database, and the goal of the classification approach. The studies reported in the literature aim at recognizing the daily action carried out in a sequence of data frames rather than facing the problem of recognizing one or more specific postures. Therefore, these works employ databases commonly available in the literature and adapt the set of actions to be classified according to the chosen database, without the need to create ad hoc ones. Our approach is instead quite specific, as it aims at recognizing individual postures during scenarios of everyday life, independently of the action that caused it (e.g., recognizing the lying down posture and not the falling action). In this setting, building our database using different camera points of view helped us to reproduce the data acquired on a subject freely moving in the room as during natural living conditions. However, this realistic approach increases the amount of noise in our data, threatening the accuracy of the classification results. Moreover, most of the approaches are applied and tested in limited contexts, such as in labs or selected use cases. Accordingly, the reported results are based on different experimental data, therefore, the superiority of one methodology rather than another cannot be argued.

Further developments for improving the performance of the system and its customization, for the specific purpose of allowing safer autonomous living conditions in frail individuals, may include the developing of post-processing algorithms. In order to reduce false positives, the outputs obtained separately for the data from the four cameras could be compared based on the softmax value associated with each classification to produce the output classification.

Finally, as a last step, we will develop the above-mentioned algorithm correlating the subject's position in the room with the identified posture to distinguish a condition corresponding to a safe everyday life scenario, from a possible alarm situation.

Table 5.1. Summary of the proposed AI algorithms with the best solution.

Solution	Best AI algorithm	Number of features	Type of classification	Number of classes	Accuracy (%)
First solution	MLP without 999 values	10	Frame by frame	4 classes	83.9
Second solution	Pre-processing algorithm + MLP without 999 values	10	Frame by frame	4 classes	94.5
Third solution	Bidirectional LSTM with 999 values	8	Sequence-to-sequence	5 classes	85
Fourth solution	Bidirectional GRU + Masking Layer for 999 values	8	Sequence-to-last	5 classes	82
Fourth solution	Bidirectional GRU + Masking Layer for 999 values	52	Sequence-to-last	5 classes	81
Fourth solution	Bidirectional LSTM + Masking Layer for 999 values	52	Sequence-to-last	4 classes	85
Fourth solution	Bidirectional GRU + Masking Layer for 999 values	52	Sequence-to-last	4 classes	87
Fourth solution	Bidirectional GRU + Masking Layer for 999 values + one	52	Sequence-to-last	4 classes	81

	application of data augmentation + class-balanced sequences				
Fourth solution	Bidirectional GRU + Masking Layer for 999 values + two application of data augmentation + class-balanced sequences	52	Sequence-to-last	4 classes	83
Fourth solution	Bidirectional GRU + Masking Layer for 999 values + two application of data augmentation + non-class-balanced sequences	52	Sequence-to-last	4 classes	88

References

[1] Blackman, S., Matlo, C., Bobrovitskiy, C., Waldoch, A., Fang, M.L., Jackson, P., Mihailidis, A., Nygård, L., Astell, A. and Sixsmith, A. "Ambient assisted living technologies for aging well: a scoping review", *Journal of Intelligent System,* vol. 25 no. 61, pp. 55-69, 2016.

[2] Calvaresi, D., Cesarini, D., Sernani, P., Marinoni, M., Dragoni, A. F., and Sturm, A "Exploring the ambient assisted living domain: a systematic review", *Journal of Ambient Intelligence and Humanized Computing,* vol 8, no. 2, pp 239-257 2017.

[3] Stodczyk, R. and Uhp, F. H., "Ambient Assisted Living an Overview of Current Applications, End-Users and Acceptance", *Biomedical Journal of Scientific & Technical Research*, vol. 30, no. 3, pp. 23374-23384, 2020.

[4] Aleksic, S., Atanasov, M., Agius, J. C., Camilleri, K., Cartolovni, A., Climent-Peerez, P., ... and Zgank, A., "State of the art of audio-and video-based solutions for AAL", 2022

[5] Cicirelli, G., Marani, R., Petitti, A., Milella, A., and D'Orazio, T., "Ambient assisted living: A review of technologies, methodologies and future perspectives for healthy aging of population" *Sensors*, vol. 21, no. 10, pp. 3549, 2021.

[6] Al-Shaqi, R., Mourshed, M., and Rezgui, Y, "Progress in ambient assisted systems for independent living by the elderly" *SpringerPlus*, vol. 5, no. 1, pp. 1-20, 2016

[7] Salvalavita Beghelli: dispositivi di telesoccorso Available online: https://www.beghelli.it/salvalavita (accessed on Sep 16, 2022).

[8] LIFE ALERT official website - I've fallen and I can't get up!® Available online: http://www.lifealert.com/ (accessed on Sep 16, 2022).

[9] Sixsmith, A. J., "An evaluation of an intelligent home monitoring system", *Journal of telemedicine and telecare*, vol. 6, no. 2, pp. 63-72, 2000.

[10] Mainetti, L., Manco, L., Patrono, L., Secco, A., Sergi, I., and Vergallo, R, "An ambient assisted living system for elderly assistance applications", In *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, IEEE, pp. 1-6,2016.

[11] Siriwardhana, C., Madhuranga, D., Madushan, R., and Gunasekera, K., "Classification of Activities of Daily Living Based on Depth Sequences and Audio", In *2019 14th Conference on Industrial and Information Systems (ICIIS),* IEEE, pp. 278-283, 2019.

[12] Bansal, D., Khanna, K., Chhikara, R., Dua, R. K., and Malhotra, R, "A systematic literature review of deep learning for detecting dementia", In *Proceedings of the Second International Conference on Information Management and Machine Intelligence*, Springer, Singapore, pp. 61-68, 2012

[13] Warunsin, K., and Phairoh, T, "Wristband Fall Detection System Using Deep Learning", In *2022 7th International Conference on Computer and Communication Systems (ICCCS),* IEEE, pp. 223-227, 2022.

[14] Sophia, S., Sridevi, U. K., Boselin, P. S., and Thamaraiselvi, P, "Ambient‐Assisted Living of Disabled Elderly in an Intelligent Home Using Behavior Prediction−A Reliable Deep Learning Prediction System", *Computational Analysis and Deep Learning for Medical Care: Principles Methods, and Applications,* pp. 329-342, 2021.

[15] Rupasinghe, I. D. M. S., and Maduranga, M. W. P, "Towards Ambient Assisted Living (AAL): Design of an IoT-based Elderly Activity Monitoring System", 2022.

[16] Gulati, N., & Kaur, P. D, "FriendCare-AAL: a robust social IoT based alert generation system for ambient assisted living", *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 4, pp. 1735-1762, 2022.

[17] Fernandes, C. D., Depari, A., Sisinni, E., Ferrari, P., Flammini, A., Rinaldi, S., and Pasetti, M, "Hybrid indoor and outdoor localization for elderly care applications with LoRaWAN", In *2020 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, IEEE pp. 1-6, 2020.

[18] Lee, S., "Falls associated with indoor and outdoor environmental hazards among community-dwelling older adults between men and women", *BMC geriatrics*, vol. 21, no. 1, pp. 1-12, 2021.

[19] Carter, S. E., Campbell, E. M., Sanson-Fisher, R. W., Redman, S., and Gillespie, W. J, "Environmental hazards in the homes of older people", *Age and ageing*, vol. 26, no. 3, pp. 195-202, 1997.

[20] Nastac, D. I., Arsene, O., Dragoi, M., Stanciu, I. D., & Mocanu, I., "An AAL scenario involving automatic data collection and robotic manipulation", In *3rd IET International Conference on Technologies for Active and Assisted Living (TechAAL 2019),* IET, pp. 1-6, 2019.

[21] Sokullu, R., Balcı, A., and Demir, E. , "The role of drones in ambient assisted living systems for the elderly", In *Enhanced Living Environments*, Springer, Cham, pp. 295-321, 2019.

[22] Kenfack Ngankam, H., Pigot, H., Lorrain, D., Viens, I., and Giroux, S., "Context awareness architecture for ambient-assisted living applications: Case study of nighttime wandering" *Journal of Rehabilitation and Assistive Technologies Engineering*, vol. 7, 2020.

[23] Babangida, L., Perumal, T., Mustapha, N., & Yaakob, R., "Internet of Things (IoT) Based Activity Recognition Strategies in Smart Homes: A Review", *IEEE Sensors Journal,* 2022.

[24] Tazari, M. R., Wichert, R., and Norgall, T., "Towards a unified ambient assisted living and personal health environment", In *Ambient Assisted Living*, Springer, Berlin, Heidelberg pp. 141-155, , 2011

[25] Lloret, J., Canovas, A., Sendra, S., and Parra, L., "A smart communication architecture for ambient assisted living", *IEEE Communications Magazine*, vol. 53, no.1, pp. 26-33, 2015.

[26] Zouba, N., Bremond, F., Thonnat, M., and Vu, V. T., "Multi-sensors analysis for everyday activity monitoring", *Proc. of SETIT*, pp. 25-29,2007.

[27] Plentz, P. D., and De Pieri, E. R., "An Overview on Real-Time Constraints for Ambient Intelligence (AmI)", In *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)* IEEE, pp. 1-7, 2018.

[28] Zdravevski, E., Lameski, P., Trajkovik, V., Kulakov, A., Chorbev, I., Goleva, R., ... and Garcia, N, "Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering", *Ieee Access*, vol. 5, pp. 5262-5280, 2017.

[29] Al Machot, F., Mosa, A. H., Ali, M., and Kyamakya, K. "Activity recognition in sensor data streams for active and assisted living environments", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 1, pp. 2933-2945, 2017.

[30] Al Machot, F., Ranasinghe, S., Plattner, J., and Jnoub, N., "Human activity recognition based on real life scenarios", In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, IEEE , pp. 3-8, 2018.

[31] Bouchabou, D., Nguyen, S. M., Lohr, C., LeDuc, B., and Kanellos, I., "A survey of human activity recognition in smart homes based on IoT sensors algorithms: Taxonomies, challenges, and opportunities with deep learning", *Sensors*, vol. 21, no. 18, 6037, 2021.

[32] Aggarwal, J. K., and Ryoo, M. S., Human activity analysis: A review. *Acm Computing Surveys (Csur)*, vol. 43, no. 3, pp. 1-43, 2011.

[33] Wang, J., Chen, Y., Hao, S., Peng, X., and Hu, L., "Deep learning for sensor-based activity recognition: A survey", *Pattern recognition letters*, vol. 119, pp. 3-11, 2019.

[34] Jegham, I., Khalifa, A. B., Alouani, I., and Mahjoub, M. A., "Vision-based human action recognition: An overview and real world challenges", *Forensic Science International: Digital Investigation*, vol. 32, pp. 200901, 2020.

[35] Dang, L. M., Min, K., Wang, H., Piran, M. J., Lee, C. H., and Moon, H. "Sensor-based and vision-based human activity recognition: A comprehensive survey", *Pattern Recognition*, vol. 108, 107561, 2020.

[36] Ahad, M. A. R., Antar, A. D., and Ahmed, M., "IoT sensor-based activity recognition", In *IoT Sensor-based Activity Recognition*. Springer, 2020.

[37] Mohd Noor, M. H., Tan, S. Y., & Ab Wahab, M. N., "Deep Temporal Conv-LSTM for Activity Recognition", *Neural Processing Letters*, 1-23, 2022.

[38] Roberge, A., Bouchard, B., Maître, J., & Gaboury, S., "Hand Gestures Identification for Fine-Grained Human Activity Recognition in Smart Homes", *Procedia Computer Science*, vol. 201, pp. 32-39, 2022.

[39] Chen, L., and Nugent, C. D., "Human activity recognition and behaviour analysis*", Springer International Publishing*, 2019.

[40] Ruan, W., Sheng, Q. Z., Yao, L., Li, X., Falkner, N. J., and Yang, L., "Device-free human localization and tracking with UHF passive RFID tags: A data-driven approach", *Journal of Network and Computer Applications*, vol. 104, pp. 78-96, 2018.

[41] Hao, J., Bouzouane, A., and Gaboury, S., "Recognizing multi-resident activities in non-intrusive sensor-based smart homes by formal concept analysis". *Neurocomputing*, vol. 318, pp. 75-89, 2018.

[42] Amendola, S., Bianchi, L., and Marrocco, G., "Movement Detection of Human Body Segments: Passive radio-frequency identification and machine-learning technologies", *IEEE Antennas and Propagation Magazine*, vol. 57, no. 3, pp. 23-37, 2015.

[43] Ozgit, D., Butler, T., Oluwasanya, P. W., Occhipinti, L. G., and Hiralal, P., "Wear and Forget" patch for ambient assisted living", In *2019 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*, pp. 1-3, IEEE, 2019.

[44] Paolini, G., Masotti, D., Antoniazzi, F., Cinotti, T. S., and Costanzo, A., "Fall detection and 3-D indoor localization by a custom RFID reader embedded in a smart e-health platform", *IEEE Transactions on Microwave Theory and Techniques*, vol 67, no. 12, 5329-5339. 2019.

[45] Wang, Y., Cang, S., and Yu, H., "A survey on wearable sensor modality centred human activity recognition in health care", *Expert Systems with Applications*, vol. 137, pp. 167-190, 2019.

[46] Beddiar, D. R., Nini, B., Sabokrou, M., and Hadid, A, "Vision-based human activity recognition: a survey", *Multimedia Tools and Applications,* vol. 79, no. 41, pp. 30509-30555, 2020.

[47] Qiu, S., Zhao, H., Jiang, N., Wang, Z., Liu, L., An, Y., ... and Fortino, G., "Multi-sensor information fusion based on machine learning for real applications in human activity recognition: State-of-the-art and research challenges", *Information Fusion*, vol. 80, pp. 241-265, 2022.

[48] Bassoli, M., Bianchi, V., De Munari, I., and Ciampolini, P., "An IoT approach for an AAL Wi-Fi-based monitoring system", *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 12, 3200-3209, 2017.

[49] Fan, L., Li, T., Yuan, Y., and Katabi, D., "In-home daily-life captioning using radio signals", In *European Conference on Computer Vision* Springer Cham, pp. 105-123, 2020.

[50] Rafferty, J., Nugent, C. D., Liu, J., and Chen, L., "From activity recognition to intention recognition for assisted living within smart homes", *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 3, pp. 368-379, 2017.

[51] Chaccour, K., Darazi, R., el Hassans, A. H., and Andres, E., "Smart carpet using differential piezoresistive pressure sensors for elderly fall detection", In *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE ,pp. 225-229, 2015.

[52] Singh, A., Rehman, S. U., Yongchareon, S., and Chong, P. H. J., "Sensor technologies for fall detection systems: A review", *IEEE Sensors Journal*, vol. 20, no. 13, pp. 6889-6919, 2020.

[53] Bianchi, V., Ciampolini, P., and De Munari, I. (2018), "RSSI-based indoor localization and identification for ZigBee wireless sensor networks in smart homes", *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 2, pp. 566-575, 2018.

[54] Keum, S. S., Lee, C. H., and Kang, S. J., "Device to Device Collaboration Architecture for Real-Time Identification of User and Abnormal Activities in Home", In *2019 29th International Telecommunication Networks and Applications Conference (ITNAC),* IEEE, pp. 1-3, 2019.

[55] Roland, L., Lidauer, L., Sattlecker, G., Kickinger, F., Auer, W., Sturm, V., ... and Iwersen, M., "Monitoring drinking behavior in bucket-fed dairy calves using an ear-attached tri-axial accelerometer: A pilot study", *Computers and Electronics in Agriculture*, vol. 145, pp. 298-301, 2018.

[56] Geman, O., and Costin, H. "Automatic assessing of tremor severity using nonlinear dynamics, artificial neural networks and neuro-fuzzy classifier", *Advances in Electrical and Computer Engineering*, vol. 14, no. 1, pp. 133-139, 2014.

[57] Cebanov, E., Dobre, C., Gradinaru, A., Ciobanu, R. I., and Stanciu, V. D., "Activity recognition for ambient assisted living using off-the-shelf motion sensing input devices", In *2019 Global IoT Summit (GIoTS),* IEEE, pp. 1-6, 2019.

[58] Ryselis, K., Petkus, T., Blažauskas, T., Maskeliūnas, R., and Damaševičius, R., "Multiple Kinect based system to monitor and analyze key performance indicators of physical training", *Human-Centric Computing and Information Sciences*, vol. 10, no. 1, pp. 1-22, 2020

[59] Raeis, H., Kazemi, M., and Shirmohammadi, S., "Human activity recognition with device-free sensors for well-being assessment in smart homes", *IEEE Instrumentation & Measurement Magazine*, vol. 24, no. 6, pp. 46-57, 2021.

[60] Chen, J., Huang, X., Jiang, H., and Miao, X. "Low-cost and device-free human activity recognition based on hierarchical learning model", *Sensors*, vol. 21, no. 7, pp. 2359, 2021.

[61] Zhang, H. B., Zhang, Y. X., Zhong, B., Lei, Q., Yang, L., Du, J. X., and Chen, D. S., "A comprehensive survey of vision-based human action recognition methods", *Sensors*, vol. 19, no. 5, pp. 1005, 2019.

[62] Duque Domingo, J., Cerrada, C., Valero, E., and Cerrada, J. A., "An improved indoor positioning system using RGB-D cameras and wireless networks for use in complex environments", *Sensors*, vol. 17, no. 10, pp. 2391, 2017.

[63] Zerrouki, N., Harrou, F., Sun, Y., and Houacine, A., "Vision-based human action classification using adaptive boosting algorithm", *IEEE Sensors Journal*, vol. 18, no. 12, pp. 5115-5121, 2018.

[64] Shaikh, M. B., and Chai, D., "RGB-D data-based action recognition: a review", *Sensors*, vol. 21, no. 12, pp. 4246, 2021.

[65] Gasparrini, S., Cippitelli, E., Spinsante, S., and Gambi, E., "Depth cameras in AAL environments: technology and real-world applications", In *Gamification: Concepts, Methodologies, Tools, and Applications*, IGI Global, pp. 1056-1075, 2015.

[66] Lopes, A., Souza, R., and Pedrini, H., "A Survey on RGB-D Datasets", *Computer Vision and Image Understading*, vol. 222, pp. 103489, 2022.

[67] Horaud, R., Hansard, M., Evangelidis, G., and Ménier, C., "An overview of depth cameras and range scanners based on time-of-flight technologies", *Machine vision and applications*, vol. 27, no. 7, pp. 1005-1020, 2016.

[68] Giancola, S., Valenti, M., and Sala, R., "A survey on 3D cameras: Metrological comparison of time-of-flight, structured-light and active stereoscopy technologies", *Springer Nature*, 2018.

[69] Kolb and E, A., Barth, R. K., and Larsen, R., "Timeof-flight cameras in computer graphics", In *Computer Graphics Forum*, vol. 29, no. 1, pp. 141-159, 2010.

[70] Choi, J., "Range Sensors: Ultrasonic Sensors, Kinect, and LiDAR. Humanoid Robotics: A Reference", pp. 1-19, 2016.

[71] Srimath, S., Ye, Y., Sarker, K., Sunderraman, R., and Ji, S., "Human Activity Recognition from RGB Video Streams Using 1D-CNNs", In *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI),* IEEE, pp. 295-302, 2021.

[72] Tu, J., Liu, M., and Liu, H., "Skeleton-based human action recognition using spatial temporal 3D convolutional neural networks", In *2018 IEEE International Conference on Multimedia and Expo (ICME),* IEEE, pp. 1-6, 2018.

[73] Li, M., and Sun, Q., "3D Skeletal Human Action Recognition Using a CNN Fusion Model", *Mathematical Problems in Engineering*, pp. 1024-123, 2021.

[74] Taiwo, O., and Ezugwu, A. E., "Internet of things-based intelligent smart home control system", *Security and Communication Networks*,pp. 1939-0114, 2021.

[75] Webb, J., and Ashley, J., "Beginning kinect programming with the microsoft kinect SDK", Apress, 2012

[76] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., ... and Blake, A., "Real-time human pose recognition in parts from single depth images", In *CVPR 2011*, IEEE, pp. 1297-1304, 2011.

[77] Fergus, R., Perona, P., and Zisserman, A., "Object class recognition by unsupervised scale-invariant learning", In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings, v*ol. 2, pp. II-II, IEEE, 2003.

[78] Winn, J., and Shotton, J., "The layout consistent random field for recognizing and segmenting partially occluded objects", In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06,* vol. 1, pp. 37-44, IEEE. 2006.

[79] Sarbolandi, H., Lefloch, D., and Kolb, A., "Kinect range sensing: Structured-light versus Time-of-Flight Kinect", *Computer vision and image understanding*, vol. 139, pp. 1-20. 2015.

[80] Sarkar, A., Banerjee, A., Singh, P. K., and Sarkar, R., "3D Human Action Recognition: Through the eyes of researchers", *Expert Systems with Applications*, pp. 116424, 2022.

[81] Al-Faris, M., Chiverton, J., Ndzi, D., and Ahmed, A. I., "A review on computer vision-based methods for human action recognition", *Journal of imaging*, vol. 6, no. 6, pp. 46, 2020.

[82] Antar, A. D., Ahmed, M., and Ahad, M. A. R., "Challenges in sensor-based human activity recognition and a comparative analysis of benchmark datasets: a review", In *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR),* IEEE, pp. 134-139, 2019.

[83] Gu, F., Chung, M. H., Chignell, M., Valaee, S., Zhou, B., and Liu, X. "A survey on deep learning for human activity recognition", *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1-34, 2021.

[84] Baños Legrán, O., Damas Hermoso, M., Pomares Cintas, H. E., and Rojas Ruiz, I., "On the Use of Sensor Fusion to Reduce the Impact of Rotational and Additive Noise in Human Activity Recognition", 2012.

[85] Kunze, K., Lukowicz, P., Junker, H., and Tröster, G., "Where am I: recognizing on-body positions of wearable sensors", *Locationand Context-Awareness*, vol. 3479 of Lecture Notes in Computer Science, pp. 264-275, 2005.

[86] Kunze, K., and Lukowicz, P., "Dealing with sensor displacement in motion-based onbody activity recognition systems", In *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 20-29, 2008.

[87] Chavarriaga, R., Bayati, H., and Millán, J. D. R., "Unsupervised adaptation for acceleration-based activity recognition: robustness to sensor displacement and rotation", *Personal and Ubiquitous Computing*, vol. 17, no. 3, pp. 479-490, 2013.

[88] Hussain, Z., Sheng, Q. Z., and Zhang, W. E., "A review and categorization of techniques on device-free human activity recognition", *Journal of Network and Computer Applications*, vol. 167, pp. 102738, 2020.

[89] Eck, N. J. V., and Waltman, L., "VOS: A new method for visualizing similarities between objects", In *Advances in data analysis,* Springer, Berlin, Heidelberg, pp. 299-306, 2007.

[90] de Cheveigné, A., and Nelken, I., "Filters: when, why, and how (not) to use them", *Neuron*, vol. 102, no. 2, pp. 280-293, 2019.

[91] Antonsson, E. K., and Mann, R. W., "The frequency content of gait", *Journal of biomechanics*, vol. 18, no. 1,pp. 39-47, 1985

[92] Grossman, G. E., Leigh, R. J., Abel, L. A., Lanska, D. J., and Thurston, S. E., "Frequency and velocity of rotational head perturbations during locomotion", *Experimental brain research*, vol. 70, no. 3, pp. 470-476, 1988.

[93] Johnson, D. H., "Signal-to-noise ratio", *Scholarpedia*, vol. 1, no. 12, pp. 2088, 2006.

[94] Castro, H. F., Correia, V., Sowade, E., Mitra, K. Y., Rocha, J. G., Baumann, R. R., and Lanceros-Méndez, S., "All-inkjet-printed low-pass filters with adjustable cutoff frequency consisting of resistors, inductors and transistors for sensor applications", *Organic Electronics*, vol. 38, pp. 205-212, 2016.

[95] Wang, S., Zhang, X., Liu, X., Zhang, J., Ma, S., and Gao, W., "Utility-driven adaptive preprocessing for screen content video compression", *IEEE Transactions on Multimedia*, vol. 19, no. 3, pp. 660-667. 2016.

[96] Ma, C., Wang, A., Chen, G., and Xu, C., "Hand joints-based gesture recognition for noisy dataset using nested interval unscented Kalman filter with LSTM network", *The visual computer*, vol. 34, no. 6, pp. 1053-1063, 2018.

[97] Jaouedi, N., Boujnah, N., and Bouhlel, M. S., "A new hybrid deep learning model for human action recognition", *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 4, pp. 447-453, 2020.

[98] Pires, I. M., Hussain, F., Garcia, N. M., Lameski, P., and Zdravevski, E. "Homogeneous data normalization and deep learning: A case study in human activity classification", *Future Internet*, vol. 12, no. 11, pp. 194, 2020.

[99] Singh, D., and Singh, B., "Investigating the impact of data normalization on classification performance", *Applied Soft Computing*, vol. 97, pp. 105524, 2020

[100] Patro, S. G. K. S., and Sahu, K. K., "Normalization: a preprocessing stage", *arXiv*, vol. 19. 2015

[101] Mistry, J., and Inden, B., "An approach to sign language translation using the intel realsense camera", In *2018 10th Computer Science and Electronic Engineering (CEEC),* IEEE, pp. 219-224, 2018.

[102] Narkhede, A. H., "Human Activity Recognition Based on Multimodal Body Sensing", 2019.

[103] Liu, J., Wang, Y., Liu, Y., Xiang, S., and Pan, C., "3D PostureNet: A unified framework for skeleton-based posture recognition", *Pattern Recognition Letters*, vol. 140, pp. 143-149, 2020.

[104] Cippitelli, E., Gasparrini, S., Gambi, E., and Spinsante, S., "A human activity recognition system using skeleton data from RGBD sensors", *Computational intelligence and neuroscience*, 2016.

[105] Khan, N. S., and Ghani, M. S., "A survey of deep learning based models for human activity recognition", *Wireless Personal Communications*, vol. 120, no. 2, pp. 1593-1635, 2021.

[106] Adama, D. A., Lotfi, A., and Ranson, R., "Adaptive segmentation and sequence learning of human activities from skeleton data", *Expert Systems with Applications*, vol. 164, pp. 113836, 2021.

[107] Buzzelli, M., Albé, A., and Ciocca, G., "A vision-based system for monitoring elderly people at home", *Applied Sciences*, vol. 10, no. 1, pp. 374, 2020.

[108] Hammerla, N. Y., Halloran, S., and Plötz, T., "Deep, convolutional, and recurrent models for human activity recognition using wearables", *arXiv,* 2016.

[109] Janidarmian, M., Radecka, K., and Zilic, Z., "Automated diagnosis of knee pathology using sensory data", In *2014 4th international conference on wireless mobile communication and healthcare-transforming healthcare through innovations in mobile and wireless technologies (mobihealth)*, IEEE, pp. 95-98, 2014.

[110] Fida, B., Bernabucci, I., Bibbo, D., Conforto, S., and Schmid, M., "Pre-processing effect on the accuracy of event-based activity segmentation and classification through inertial sensors", *Sensors*, vol. 15, no. 9, pp. 23095-23109, 2015.

[111] Patterson, T., Khan, N., McClean, S., Nugent, C., Zhang, S., Cleland, I., and Ni, Q., "Sensor-based change detection for timely solicitation of user engagement", *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2889-2900, 2016.

[112] Devanne, M., and Papadakis, P., "Recognition of activities of daily living via hierarchical long-short term memory networks", In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC),* IEEE, pp. 3318-3324, 2019.

[113] Noor, M. H. M., Salcic, Z., Kevin, I., and Wang, K., "Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer", *Pervasive and Mobile Computing*, vol. 38, pp. 41-59, 2017.

[114] Ke, S. R., Thuc, H. L. U., Lee, Y. J., Hwang, J. N., Yoo, J. H., and Choi, K. H., "A review on video-based human activity recognition", *Computers*, vol. 2, no. 2, pp. 88-131, 2013.

[115] Babaee, M., Dinh, D. T., and Rigoll, G., "A deep convolutional neural network for background subtraction", *arXiv,* 2017.

[116] Mliki, H., Bouhlel, F., and Hammami, M., "Human activity recognition from UAV-captured video sequences", *Pattern Recognition*, 100, pp. 107140, 2020.

[117] Dhiman, C., and Vishwakarma, D. K., "A review of state-of-the-art techniques for abnormal human activity recognition", *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 21-45, 2019.

[118] Ahad, M. A. R., "Vision and sensor-based human activity recognition: challenges ahead", In *Advancements in Instrumentation and Control in Applied System Applications,* IGI Global, pp. 17-35, 2020.

[119] Sousa Lima, W., Souto, E., El-Khatib, K., Jalali, R., and Gama, J., "Human activity recognition using inertial sensors in a smartphone: An overview", *Sensors*, vol. 19, no. 14, pp. 3213. 2019.

[120] Nweke, H. F., Teh, Y. W., Mujtaba, G., and Al-Garadi, M. A., "Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions", *Information Fusion*, vol. 46, pp. 147-170. 2019.

[121] Ahad, M. A. R., Ahmed, M., Antar, A. D., Makihara, Y., and Yagi, Y., "Action recognition using kinematics posture feature on 3D skeleton joint locations", *Pattern Recognition Letters*, vol. 145, pp. 216-224, 2021.

[122] Khan, I. U., Afzal, S., and Lee, J. W., "Human activity recognition via hybrid deep learning based model", *Sensors*, vol. 22, no. 1, pp. 323, 2022.

[123] Patsadu, O., Nukoolkit, C., and Watanapa, B., "Human gesture recognition using Kinect camera", In *2012 ninth international conference on computer science and software engineering (JCSSE),* IEEE, pp. 28-32, 2012.

[124] Choubik, Y., and Mahmoudi, A., "Machine learning for real time poses classification using kinect skeleton data", In *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV),* IEEE, pp. 307-311, 2016.

[125] Müller, M., Röder, T., and Clausen, M., "Efficient content-based retrieval of motion capture data", In *ACM SIGGRAPH 2005 Papers,* pp. 677-685, 2005.

[126] Yadav, S. K., Tiwari, K., Pandey, H. M., and Akbar, S. A., "Skeleton-based human activity recognition using ConvLSTM and guided feature learning", *Soft Computing*, vol. 26, no. 2, pp. 877-890, 2022.

[127] Bevilacqua, V., Nuzzolese, N., Barone, D., Pantaleo, M., Suma, M., D'Ambruoso, D., ... and Stroppa, F., "Fall detection in indoor environment with kinect sensor", In *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings,* IEEE, pp. 319-324, 2014.

[128] Yang, X., and Tian, Y., "Effective 3d action recognition using eigenjoints", *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 2-11, 2014.

[129] Visutarrom, T., Mongkolnam, P., and Chan, J. H., "Postural classification using Kinect", In *2014 International Computer Science and Engineering Conference (ICSEC),* IEEE, pp. 403-408, 2014.

[130] Visutarrom, T., Mongkolnam, P., and Chan, J. H., "Multiple-stage classification of human poses while watching television", In *2014 2nd International Symposium on Computational and Business Intelligence,* IEEE, pp. 10-16, 2014.

[131] Kim, K., Jalal, A., and Mahmood, M., "Vision-based human activity recognition system using depth silhouettes: A smart home system for monitoring the residents", *Journal of Electrical Engineering & Technology*, vol. 14, no. 6, pp. 2567-2573, 2019.

[132] Patel, C. I., Labana, D., Pandya, S., Modi, K., Ghayvat, H., and Awais, M., "Histogram of oriented gradient-based fusion of features for human action recognition in action video sequences", *Sensors*, vol. 20, no. 24, pp. 7299, 2020.

[133] Aly, S., and Sayed, A., "Human action recognition using bag of global and local Zernike moment features", *Multimedia Tools and Applications*, vol. 78, no. 17, pp. 24923-24953, 2019.

[134] Li, M., Wang, H., Yang, L., Liang, Y., Shang, Z., and Wan, H., "Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction", *Expert Systems with Applications*, vol. 150, pp. 113277, 2020.

[135] Sharma, N., and Saroha, K., "Study of dimension reduction methodologies in data mining", In *International Conference on Computing, Communication & Automation,* IEEE, pp. 133-137, 2015.

[136] Abd-Alsabour, N., "On the Role of Dimensionality Reduction", *Journal Computer*, vol. 13, no. 5, pp. 571-579, 2018.

[137] Ayesha, S., Hanif, M. K., and Talib, R., "Overview and comparative study of dimensionality reduction techniques for high dimensional data", *Information Fusion*, vol. 59, pp. 44-58, 2020.

[138] Jindal, P., and Kumar, D., "A review on dimensionality reduction techniques". *Int. J. Comput. Appl*, vol. 173, no. 2, pp. 42-46, 2017.

[139] Blum, A. L., and Langley, P., "Selection of relevant features and examples in machine learning", *Artificial intelligence*, vol. 97, no.1-2, pp. 245-271, 1997.

[140] Chandrashekar, G., and Sahin, F., "A survey on feature selection methods", *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16-28, 2014.

[141] Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D., and Saeed, J., "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction", *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 56-70, 2020.

[142] Sanjograj Singh Ahuja, S.A, "Using Feature Selection Technique for Data Mining: A Review", Irjet, vol. 8, pp. 3536–3540, 2021.

[143] Siddiqi, M. H., and Alsirhani, A., "An Efficient Feature Selection Method for Video-Based Activity Recognition Systems", *Mathematical Problems in Engineering*, 2022.

[144] Suto, J., Oniga, S., and Sitar, P. P., "Comparison of wrapper and filter feature selection algorithms on human activity recognition", In *2016 6th international conference on computers communications and control (ICCCC),* IEEE, pp. 124-129, 2016.

[145] Alzahrani, M. S., Jarraya, S. K., Ben-Abdallah, H., and Ali, M. S., "Comprehensive evaluation of skeleton features-based fall detection from Microsoft Kinect v2", *Signal, Image and Video Processing*, vol. 13, no. 7, pp. 1431-1439, 2019.

[146] Bhavan, A., and Aggarwal, S., "Stacked generalization with wrapper-based feature selection for human activity recognition", In *2018 IEEE Symposium Series on Computational Intelligence (SSCI),* IEEE, pp. 1064-1068, 2018.

[147] Liu, L., and Shao, L., "Learning discriminative representations from RGB-D video data", In *Twenty-third international joint conference on artificial intelligence,* 2013.

[148] Negin, F., Özdemir, F., Akgül, C. B., Yüksel, K. A., and Erçil, A., "A decision forest based feature selection framework for action recognition from rgb-depth cameras", In *International conference image analysis and recognition,* Springer Berlin Heidelberg, pp. 648-657, 2013.

[149] Peng, Y., Wu, Z., and Jiang, J., "A novel feature selection approach for biomedical data classification", *Journal of Biomedical Informatics*, vol. 43, no. 1, pp. 15-23, 2010.

[150] Forrest, S., Genetic algorithms, *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 77-80, 1996.

[151] Jh, H., "Adaptation in natural and artificial systems", *Ann Arbor*, 1975.

[152] Sindhu Meena, K., and Suriya, S., "A survey on supervised and unsupervised learning techniques" In *International Conference on Artificial Intelligence, Smart Grid and Smart City Applications,* Springer Cham, pp. 627-644, 2019.

[153] Sathya, R., & Abraham, A., "Comparison of supervised and unsupervised learning algorithms for pattern classification", *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, pp. 34-38, 2013.

[154] Wang, L., Huynh, D. Q., and Koniusz, P., "A comparative review of recent kinect-based action recognition algorithms", *IEEE Transactions on Image Processing*, vol. 29, pp. 15-28, 2019.

[155] Shapiro, S. C., "Encyclopedia of artificial intelligence second edition", *New Jersey: A Wiley Interscience Publication*, 1992.

[156] Ariza-Colpas, P. P., Vicario, E., Oviedo-Carrascal, A. I., Butt Aziz, S., Piñeres-Melo, M. A., Quintero-Linero, A., and Patara, F. "Human Activity Recognition Data Analysis: History, Evolutions, and New Trends", *Sensors*, vol. 22, no. 9, pp. 3401, 2022.

[157] Samuel, A. L., "Some studies in machine learning using the game of checkers. II—recent progress", *Computer Games I*, pp. 366-400, 1988.

[158] Hastie, T., Tibshirani, R., and Friedman, J., "The elements of statistical learnin", *Cited on*, vol. 33, 2009.

[159] Cristianini, N., and Shawe-Taylor, J., "An introduction to support vector machines and other kernel-based learning methods", *Cambridge university press*, 2000.

[160] O'Shaughnessy, D., "Addison-Wesley series in electrical engineering: digital signal processing", 1987.

[161] Fan, R. E., Chen, P. H., Lin, C. J., and Joachims, T., "Working set selection using second order information for training support vector machines", *Journal of machine learning research*, vol. 6, no. 12, 2005.

[162] Bishop, C. M., "Neural networks for pattern recognition", *Oxford university press,* 1995.

[163] Rosenblatt, F., "The perceptron: a probabilistic model for information storage and organization in the brain", *Psychological review*, vol. 65, no. 6, pp. 386, 1958.

[164] Sharma, S., Sharma, S., and Athaiya, A., "Activation functions in neural networks", *towards data science*, vol. 6, no. 12, pp. 310-316, 2017.

[165] Agostinelli, F., Hoffman, M., Sadowski, P., and Baldi, P., "Learning activation functions to improve deep neural networks", *arXiv,* 2014.

[166] Murtagh, F., "Multilayer perceptrons for classification and regression", *Neurocomputing*, vol. 2, no. 5-6, pp. 183-197, 1991.

[167] Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E., " Deep learning for computer vision: A brief review", *Computational intelligence and neuroscience*, 2018.

[168] Alam, F., Faulkner, N., and Parr, B., "Device-free localization: A review of non-RF techniques for unobtrusive indoor positioning", *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4228-4249, 2020.

[169] Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L. Y., and Kot, A. C., "Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding", *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2684-2701,2019.

[170] Abdel-Basset, M., Chang, V., Hawash, H., Chakrabortty, R. K., and Ryan, M., "Deep learning approaches for human-centered IoT applications in smart indoor environments: a contemporary survey", *Annals of Operations Research*, pp. 1-49, 2021.

[171] Medsker, L., and Jain, L. C., "Recurrent neural networks: design and applications", *CRC press,* 1999.

[172] Sherstinsky, A., "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network", *Physica D: Nonlinear Phenomena*, vol. 404, pp. 132306, 2020.

[173] Lipton, Z. C., Kale, D. C., Elkan, C., and Wetzel, R., "Learning to diagnose with LSTM recurrent neural networks", *arXiv,* 2015.

[174] Dey, R., and Salem, F. M., "Gate-variants of gated recurrent unit (GRU) neural networks", In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS),* IEEE, pp. 1597-1600, 2017.

[175] Hochreiter, S.; Urgen Schmidhuber, J., "Long Shortterm Memory", *Neural Computers*, vol. 9, pp. 17351780, 1997.

[176] Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., and Jenssen, R., "Recurrent neural networks for short-term load forecasting: an overview and comparative analysis", 2017.

[177] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S., "Recurrent neural network based language model", In *Interspeech,* vol. 2, no. 3, pp. 1045-1048, 2010.

[178] Sokolova, M., and Lapalme, G., "Classification of opinions with non-affective adverbs and adjectives", In *Proceedings of the International Conference RANLP-2009*, pp. 421-427, 2009.

[179] Demšar, J., "Statistical comparisons of classifiers over multiple data sets", *The Journal of Machine learning research*, vol. 7, pp. 1-30, 2006.

[180] Midgley, A. R., Niswender, G. D., and Rebar, R. W., " Principles for the assessment of the reliability of radioimmunoassay methods (precision, accuracy, sensitivity, specificity)", *European Journal of Endocrinology*, vol. 62, no. (1_Suppl), pp. S163-S184, 1969.

[181] Fawcett, T., "An introduction to ROC analysis", *Pattern recognition letters*, vol. 27, no. 8, pp. 861-874, 2006.

[182] Hand, D. J., & Till, R. J., "A simple generalisation of the area under the ROC curve for multiple class classification problems", *Machine learning*, vol. 45, no. 2, pp. 171-186, 2001.

[183] Malekmohamadi, H., Moemeni, A., Orun, A., and Purohit, J. K., "Low-cost automatic ambient assisted living system", In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops),* IEEE, pp. 693-697, 2018.

[184] Akyash, M., Mohammadzade, H., and Behroozi, H., "A Dynamic Time Warping Based Kernel for 3D Action Recognition Using Kinect Depth Sensor", In *2020 28th Iranian Conference on Electrical Engineering (ICEE),* IEEE, pp. 1-5, 2020.

[185] Tariq, M., Majeed, H., Beg, M. O., Khan, F. A., and Derhab, A., "Accurate detection of sitting posture activities in a secure IoT based assisted living environment", *Future Generation Computer Systems*, vol. 92, pp. 745-757, 2019.

[186] Su, B., Wu, H., & Sheng, M., "Human action recognition method based on hierarchical framework via Kinect skeleton data", In *2017 International Conference on Machine Learning and Cybernetics (ICMLC),* IEEE, vol. 1, pp. 83-90, 2017.

[187] Zhu, A., Wu, Q., Cui, R., Wang, T., Hang, W., Hua, G., and Snoussi, H., "Exploring a rich spatial–temporal dependent relational model for skeleton-based action recognition by bidirectional LSTM-CNN", *Neurocomputing*, vol. 414, pp. 90-100, 2020.

[188] Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., and Xie, X., "Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks", In *Proceedings of the AAAI conference on artificial intelligence,* vol. 30, no. 1, 2016.

[189] Liu, J., Shahroudy, A., Xu, D., and Wang, G., "Spatio-temporal lstm with trust gates for 3d human action recognition", In *European conference on computer vision,* Springer, Cham, pp. 816-833, 2016.

[190] Blum, A. L., "Langley,"Selection of Relevant Features and Examples", *Machine Learning, Artificial Intelligence*, vol. 97, pp. 245-271, 1997.

[191] Billari, F. C., Muttarak, R., and Spiess, C. K., "Demographic change and growing population diversity in Europe", 2022.

[192] Alsaeeedi, A., Jabeen, S., & Kolivand, H., "Ambient assisted living framework for elderly care using Internet of medical things, smart sensors, and GRU deep learning techniques", *Journal of Ambient Intelligence and Smart Environments*, pp. 1-19, 2022.

[193] Colantonio, S., Coppini, G., Giorgi, D., Morales, M. A., and Pascali, M. A., "Computer vision for ambient assisted living: Monitoring systems for personalized healthcare and wellness that are robust in the real world and accepted by users, carers, and society", In *Computer Vision for Assistive Healthcare,* pp. 147-182, 2018.

[194] Guerra, B. M. V., Ramat, S., Beltrami, G., and Schmid, M., "Automatic pose recognition for monitoring dangerous situations in Ambient-Assisted Living", *Frontiers in Bioengineering and Biotechnology*, vol. 8, pp. 415, 2020.

[195] Urbanowicz, R. J., Meeker, M., La Cava, W., Olson, R. S., and Moore, J. H., "Relief-based feature selection: Introduction and review", *Journal of biomedical informatics*, vol. 85, pp. 189-203, 2018.

[196] Li, B., Han, C., and Bai, B., "Hybrid approach for human posture recognition using anthropometry and BP neural network based on Kinect V2", *EURASIP Journal on Image and Video Processing*, vol. 1, pp. 1-15, 2019.

[197] Rougier, C., Auvinet, E., Rousseau, J., Mignotte, M., and Meunier, J., "Fall detection from depth map video sequences. In *International conference on smart homes and health telematics*", Springer Berlin Heidelberg, pp. 121-128, 2011.

[198] Guerra, B. M., Ramat, S., Gandolfi, R., Beltrami, G., and Schmid, M., "Skeleton data pre-processing for human pose recognition using Neural Network", In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC),* IEEE, pp. 4265-4268, 2020.

[199] Han, J., Shao, L., & Xu, D. (2013). lShatton,". Enhanced computer vision with Microsoft Kinect sensor: a review," in IEEE Transactions on Cybernetics, 43(5), 1318-1334.

[200] Ali, S. M., Augusto, J. C., and Windridge, D., "A survey of user-centred approaches for smart home transfer learning and new user home automation adaptation", *Applied Artificial Intelligence*, vol. 33, no. 8, pp. 747-774, 2019.

[201] Sahak, R., Zakaria, N. K., Tahir, N. M., Yassin, A. I. M., and Jailani, R., "Review on current methods of gait analysis and recognition using kinect", In *2019 IEEE 15th International Colloquium on Signal Processing & Its Applications (CSPA)*, IEEE, pp. 229-234, 2019.

[202] Bian, Z. P., Hou, J., Chau, L. P., and Magnenat-Thalmann, N., "Fall detection based on body part tracking using a depth camera", *IEEE journal of biomedical and health informatics*, vol. 19, no. 2, pp. 430-439, 2014.

[203] Li, R., Si, W., Weinmann, M., and Klein, R., "Constraint-based optimized human skeleton extraction from single-depth camera" *Sensors*, vol. 19, no. 11, pp. 2604, 2019.

[204] Winter, D. A., "Biomechanics and motor control of human movement", *John Wiley & Sons*, 2009.

[205] Wochatz, M., Tilgner, N., Mueller, S., Rabe, S., Eichler, S., John, M., ... and Mayer, F. "Reliability and validity of the Kinect V2 for the assessment of lower extremity rehabilitation exercises", *Gait & posture*, vol. 70, pp. 330-335.210, pp. 2019

[206] Guerra, B. M. V., Schmid, M., Beltrami, G., and Ramat, S., "Neural Networks for Automatic Posture Recognition in Ambient-Assisted Living", *Sensors*, vol. 22, no. 7, pp. 260, 2022.

[207] Johnson, J. M., and Khoshgoftaar, T. M., "Survey on deep learning with class imbalance", *Journal of Big Data*, Nunez, J. C., Cabido, R., Pantrigo, J. J., Montemayor, A. S., & Velez, J. F. (2018). Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. Pattern Recognition, 76, 80-94. vol. 6, no. 1, pp. 1-54, 2019.

[208] Vemulapalli, R., Arrate, F., and Chellappa, R. "Human action recognition by representing 3d skeletons as points in a lie group", In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 588-595, 2014.

[209] Wu, D., and Shao, L., "Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition", In *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 724-731,2014.

[210] Chaudhry, R., Ofli, F., Kurillo, G., Bajcsy, R., and Vidal, R.. "Bio-inspired dynamic 3d discriminative skeletal features for human action recognition", In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops,* pp. 471-478, 2013.

[211] Wang, C., Wang, Y., and Yuille, A. L., "An approach to pose-based action recognition", In *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 915-922, 2013.

[212] Gaglio, S., Re, G. L., and Morana, M., "Human activity recognition process using 3-D posture data", *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 5, pp. 586-597, 2014.

[213] Meng, F., Liu, H., Liang, Y., Tu, J., and Liu, M., "Sample fusion network: An end-to-end data augmentation network for skeleton-based human action recognition", *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5281-5295, 2019.

[214] Nunez, J. C., Cabido, R., Pantrigo, J. J., Montemayor, A. S., and Velez, J. F., "Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition", *Pattern Recognition*, vol. 76, pp. 80-94. 2018.