1 2 9 0

UNIVERSIDADE D
COIMBRA

Hugo Rafael Mendes Luís

Deep Learning Based Human Activity Recognition:
A Real-Time Perspective

Dissertation submitted in partial fulfillment of the requirements for the Degree of Master of Science in Electrical and Computer Engineering - Specialization in Automation, supervised by Prof. Dr. Urbano José Carreira Nunes and presented to the Department of Electrical and Computer Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

October of 2020

Hugo Rafael Mendes Luís

Deep Learning Based Human Activity Recognition: A Real-Time Perspective

Dissertation submitted in partial fulfillment of the requirements for the Degree of Master of Science in Electrical and Computer Engineering - Specialization in Automation, supervised by Prof. Dr. Urbano José Carreira Nunes and presented to the Department of Electrical and Computer Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

October of 2020

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Ð
COIMBRA

Hugo Rafael Mendes Luís

Deep Learning Based Human Activity Recognition: A Real-Time Perspective

Supervisor:

Prof. Dr. Urbano José Carreira Nunes

Co-Supervisor:

Master Luís Carlos Artur da Silva Garrote

Jury:

Prof. Dr. João Pedro de Almeida Barreto

Prof. Dr. Rui Alexandre de Matos Araújo

Prof. Dr. Urbano José Carreira Nunes

Dissertation submitted in partial fulfillment of the requirements for the Degree of Master of Science in Electrical and Computer Engineering - Specialization in Automation, supervised by Prof. Dr. Urbano José Carreira Nunes and presented to the Department of Electrical and Computer Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

October of 2020

Acknowledgements

The writing process of this dissertation put me to the test in several ways. I deepened theoretical knowledge and tested its practical implications. I also learned to manage frustration even when the results were not as desired. The working method was adjusted in the context of a pandemic, but I still had the incessant support of my advisors in this enriching journey. I start by thanking Professor Doutor Urbano Nunes and Mestre Luís Garrote who, both from a distance and in person taught and guided me. They also encouraged me to participate in the Summer-School on Robotics and Machine Learning, which had great importance in this learning journey.

I also thank the Institute of Systems and Robotics - University of Coimbra (ISR-UC) for the opportunity to work in an adequately equipped place. I also thank the people with whom I shared this journey at ISR-UC, particularly Master Luís Garrote for the patience and help provided throughout the dissertation research work.

I am also grateful to be surrounded by people who supported me outside the academic context: my close friends, my parents and in-laws, my grandparents and my fiancée. Thank you all!

Abstract

Over the last two decades there has been a massive development of robotic systems in the field of service, social and personal robots. Robots are being deployed in scenarios such as: public places, homes, hospitals, elderly care centers or front desk applications. This type of robots will greatly benefit from the integration of a Human Activity Recognition module in their architecture. Having the ability to identify a variety of human activities allows for a more diverse human robot interaction. By identifying humans and their social interactions the robots will also be able to adapt their navigation behavior according to social norms. Empowering social robots with these capabilities will help towards the future deployment of such robotic systems. Their behavior will look more natural, helping humans develop a sense of comfort towards the robot's presence.

The objective of the dissertation research work is to develop, train and evaluate a Human Activity Recognition framework. For such purpose two frameworks, (1) RGB-based framework and (2) Skeleton-based framework were developed, trained and evaluated. A performance comparison between both frameworks is made based on classification accuracy and runtime.

The developed frameworks are primarily validated on a large-scale human activity recognition dataset. After validation, the best performing framework, in terms of both classification accuracy and runtime, is validated in a small-scale dataset collected at the Human-Centered Mobile Robotics Laboratory in the Institute of Systems and Robotics - University of Coimbra.

Tests were conducted in two subsets of classes with the objective of comparing the frameworks' generalization capabilities when trained and tested in a more or less challenging subset of classes. The results obtained met the expectations set for each developed framework. Both frameworks can successfully identify activities on the large-scale dataset. However, the performance on the small-scale dataset needs more research.

Keywords: Human Activity Recognition, Social Robot, Service Robot, Deep Learning, Convolutional Neural Network

Resumo

Nas últimas duas décadas, houve um grande desenvolvimento de sistemas robóticos na área dos robôs de serviço, sociais e pessoais. Robôs têm vindo a ser integrados em cenários como: locais públicos, casas, hospitais, lares de idosos ou como recepcionistas. Este tipo de robôs irá beneficiar com a integração, na sua arquitetura, de um módulo de Reconhecimento de Atividades Humanas. Ter a capacidade de identificar uma variedade de atividades humanas permite uma interação homem-máquina mais diversa. Ao identificar humanos e interações sociais, os robôs serão capazes de adaptar o seu comportamento de navegação de acordo com as normas sociais. Habilitar robôs sociais com estas capacidades irá ajudar na futura integração destes robôs. O seu comportamento irá parecer mais natural, ajudando os humanos a desenvolver uma sensação de conforto em relação à presença do robô.

O objetivo do trabalho de investigação da dissertação é desenvolver, treinar e avaliar uma *framework* de reconhecimento da atividades humanas. Para tal, duas *frameworks*, (1) *framework* baseada em RGB e (2) *framework* baseada em esqueleto foram desenvolvidas, treinadas e avaliadas. É feita uma comparação, com base na precisão da classificação e no tempo de execução, do desempenho de ambas.

As *frameworks* desenvolvidas são validadas primeiramente num *dataset* de reconhecimento de atividades humanas de grande escala. Após esta validação, a *framework* com melhor desempenho, tanto em termos de precisão de classificação como de tempo de execução, é validada num dataset de pequena escala colhido no Laboratório *Human-Centered Mobile Robotics* do Instituto de Sistemas e Robótica - Universidade de Coimbra.

Os testes foram realizados em dois subconjuntos de classes com o objetivo de comparar a capacidade de generalização das *frameworks* quando treinadas e testadas em subconjuntos de classes com diferentes níveis de dificuldade. Os resultados obtidos correspondem às expectativas estabelecidas para cada *framework* desenvolvida. Ambas as *frameworks* conseguem identificar, com êxito, atividades no *dataset* de grande escala. No entanto, o desempenho no *dataset* de pequena escala necessita de investigação adicional.

Palavras Chave: Reconhecimento de Atividades Humanas, Robô Social, Robô de Serviço, Aprendizagem Profunda, Rede Neuronal Convolucional.

Contents

List of Acronyms

2s-AGCN Two-Stream Adaptive Graph Convolutional Network.

3D-CNN Three-Dimensional Convolutional Neural Networks.

AS-GCN Actional-Structural Graph Convolution Network.

BGD Batch Gradient Descent.

C3D Convolutional 3D.

CNN Convolutional Neural Network.

DL Deep Learning.

DNN Deep Neural Network.

FAIR Facebook's Artificial Intelligence Research Lab.

FC Fully Connected.

FFT Fast Fourier Transform.

FLOPS Floating Point Operations Per Second.

FPS Frames Per Second.

GAP Global Average Pooling.

GCN Graph Convolutional Networks.

GPU Graphics Processing Unit.

HAR Human Activity Recognition.

HAR-RGB-bFI HAR-RGB-based Framework I.

HAR-RGB-bFII HAR-RGB-based Framework II.

HAR-S-bF HAR-Skeleton-based Framework.

HMI Human-Machine Interface.

HRI Human-Robot Interaction.

I3D Two-Stream Inflated 3D ConvNet.

IMU Inertial Measurement Unit.

IoT Internet of Things.

IR Infrared.

LSTM	Long Short-Term Memory.
MBGD	Mini-batch Gradient Descent.
ML	Machine Learning.
NN	Neural Network.
NPU	Navigation Processing Unit.
POV	Point of View.
ReLU	Rectified Linear Unit.
RFID	Radio-Frequency Identification.
RGB	Red-Green-Blue.
RGB-D	Red-Green-Blue-Depth.
RNN	Recurrent Neural Network.
ROS	Robot Operating System.
SGD	Stochastic Gradient Descent.
SPS	Sequences Per Second.
YOLO	You Only Look Once.

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the context and motivation for the developed work. The Human Activity Recognition (HAR) problem is defined and examples of its applications are given. The main objective and main contributions of the dissertation are described. Finally, the dissertation's outline is presented.

1.1 Context and Motivation

HAR is a classification problem that deals with the recognition of both human activities and actions. HAR approaches can be broadly divided into two categories: video-based approaches and wearable/ambient sensor based approaches. For the purpose of this dissertation the focus is on video-based methods, more concretely, on methods that use a sequence of frames to classify the activities. Although HAR can be performed on single frames, it is a more challenging task due to the lack of temporal information on individual frames [8–12]. On the other side, a sequence of frames provides both spatial and temporal information that can be used for a more reliable inference given it is used an appropriate architecture that can model both spatial and temporal information.

A simple example on the advantage of having temporal features is the following scenario: if it is shown a picture of a person lying on the floor it is an easy task identifying the pose as ”lying on the floor”. However, identifying the activity/action that led to that particular pose is a more challenging task without having prior information. The person could be either practicing yoga or, in a more extreme situation, it could have just fallen and requires urgent help. Thus, having prior information is crucial for acting accordingly to the situation.

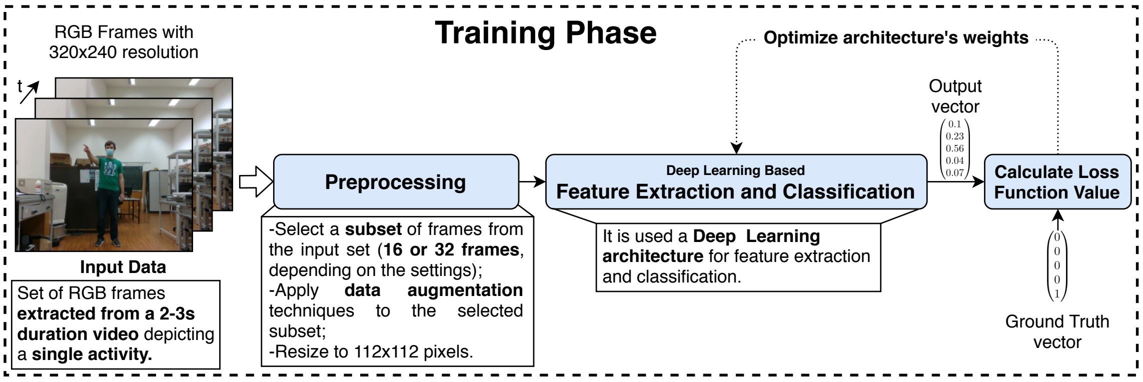
The focus of the dissertation will be on HAR for indoor applications. The development of social and assistive robots, which mainly operate in indoor environments, have received much attention in the last decade [13, 14]. Robots have moved from industrial settings to applications where Human-Robot Interaction (HRI) is heavily present [15]. Robots are being deployed in homes, hospitals, elderly care centers or museums [13, 16–19]. Robots such as Pepper [20] (a commercially available social humanoid robot presented in 2014), SPENCER [21] (a socially aware robot) and SocialRobot [22] (a social platform for interaction with the elderly) have been deployed in a wide variety of scenarios such as educational environments [23], front desk applications [24], museums [25, 26], social interaction settings [27, 28] and welcoming people in public places [29]. SPENCER was deployed in an airport for passenger guidance and SocialRobot was deployed, as a pilot demo, in an elderly care center in Netherlands. This type of robots will greatly benefit from the integration of an HAR module in their architecture. Having the ability to identify a variety of human activities allows for a more diverse human robot interaction. A service robot deployed in an elderly care center will be able to identify emergency situations, motivate the elderly for physical exercise or fetch objects such as medication or a cup of water. On the other hand, a robot deployed in a public building will be able to detect when someone needs directions or requires some service provided by the robot. Gestures such as pointing to something or hand-waving at the robot could be used to initiate the HRI.

Moreover, a critical aspect to take into consideration when deploying robots in the aforementioned scenarios is the robots' navigation behavior. Distinguishing between a human and an ordinary object is a crucial aspect to take into consideration when generating navigation paths. A robot equipped with an HAR module can provide this capability by adapting its navigation behavior when it encounters humans in its path (*Human-aware Robot Navigation*)[30–32] These robots are capable of identifying humans as well as their social interactions and having a navigation conduct that takes into account social norms, e.g., not interrupting a conversation between two or more persons by passing between the humans, navigating on the right of a human and having a smooth navigation path [33]. Empowering a social robot with *Human-aware Robot Navigation* will help towards the future deployment of social, service and personal robots. The robot's navigation behavior will look more natural, helping humans develop a feeling of comfort towards the robot's presence which boosts the HRI confidence [34–36].

1.2 Main Objective

The main objective of the dissertation research work was to develop, train and evaluate an HAR framework. For such purpose two frameworks, (1) HAR-RGB-based framework and (2) HAR-Skeleton-based Framework (HAR-S-bF) were developed, trained and evaluated. A performance comparison between both frameworks is made based on classification accuracy and runtime.

The developed work is divided into two phases: (1) training phase and (2) testing phase. During the training phase the framework, more concretely the feature extraction and classification module, is trained on a labeled training set ($S_{training}$). On the testing phase the HAR framework is tested on a labeled testing set ($S_{testing}$) and its classification accuracy is calculated. Both $S_{training}$ and $S_{testing}$ are subsets of the same dataset $S_{dataset}$ where $S_{training} \cup S_{testing} = S_{dataset}$ and $S_{training} \cap S_{testing} = \emptyset$. Figure 1.1 presents two illustrative diagrams of the training phases of the two HAR frameworks. In Fig. 1.2 it is presented two illustrative diagrams of testing phase of the two HAR frameworks.

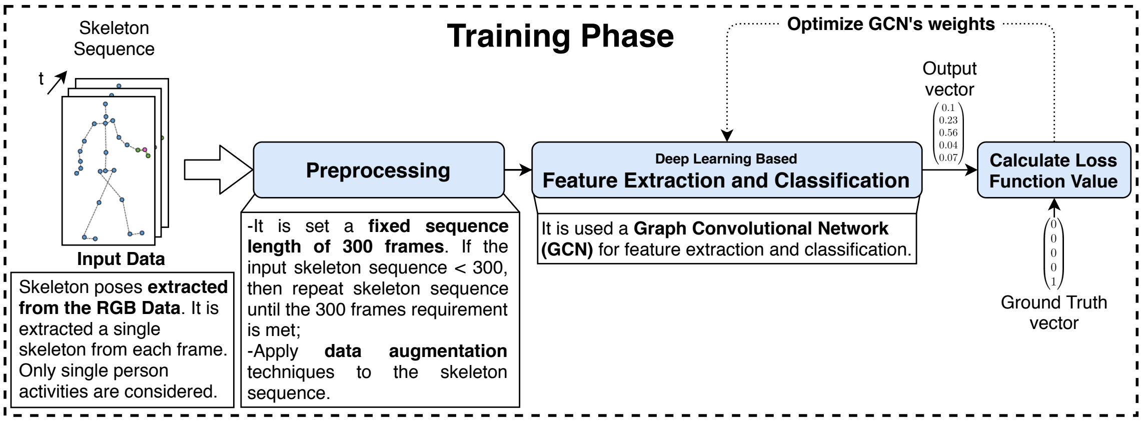

Training Phase
RGB Frames with 320x240 resolution
t
Optimize architecture's weights
Output vector
0.1
0.23
0.56
0.04
0.07
Preprocessing
Deep Learning Based
Feature Extraction and Classification
Calculate Loss Function Value
Input Data
0
0
0
0
1
Ground Truth vector

Set of RGB frames **extracted from a 2-3s duration video** depicting a **single activity.**

-Select a **subset** of frames from the input set (**16 or 32 frames**, depending on the settings);
-Apply **data augmentation** techniques to the selected subset;
-Resize to 112x112 pixels.

It is used a **Deep Learning architecture** for feature extraction and classification.

(a) RGB-based HAR framework during training phase, for a single training example.



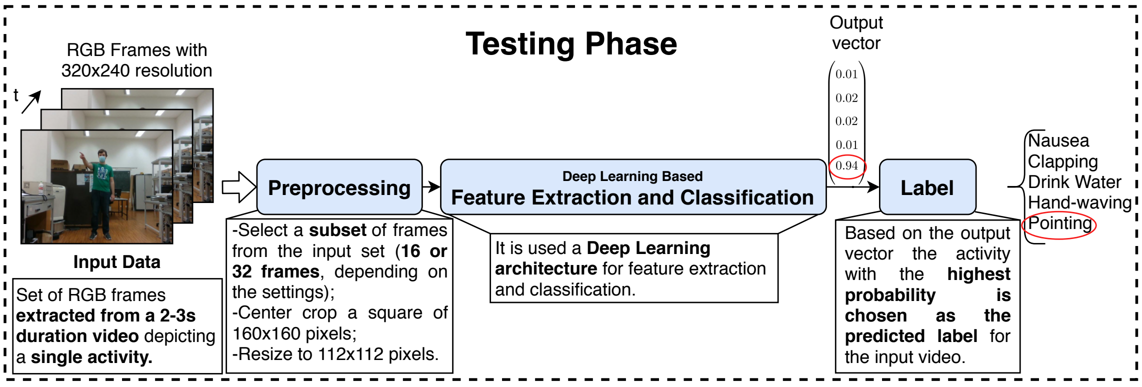
Skeleton poses **extracted from the RGB Data**. It is extracted a single skeleton from each frame. Only single person activities are considered.

-It is set a **fixed sequence length of 300 frames**. If the input skeleton sequence < 300, then repeat skeleton sequence until the 300 frames requirement is met;
-Apply **data augmentation** techniques to the skeleton sequence.

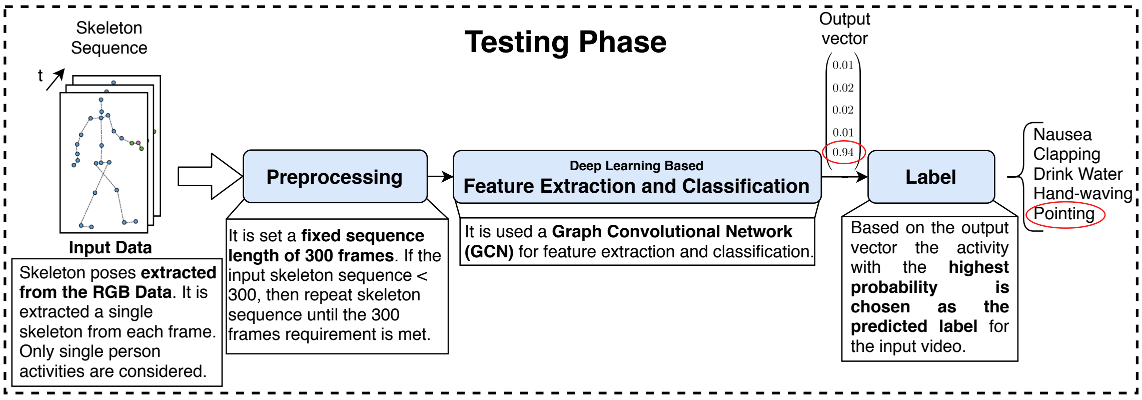
It is used a **Graph Convolutional Network (GCN)** for feature extraction and classification.

(b) HAR-S-bF during training phase, for a single training example.

Figure 1.1: Illustration of the training phase for the frameworks evaluated in this work, for a single training example.


Testing Phase
RGB Frames with 320x240 resolution
t
Input Data
Set of RGB frames extracted from a 2-3s duration video depicting a single activity.
Preprocessing
-Select a subset of frames from the input set (16 or 32 frames, depending on the settings);
-Center crop a square of 160x160 pixels;
-Resize to 112x112 pixels.
Deep Learning Based Feature Extraction and Classification
It is used a Deep Learning architecture for feature extraction and classification.
Output vector
0.01
0.02
0.02
0.01
0.94
Label
Based on the output vector the activity with the highest probability is chosen as the predicted label for the input video.
Nausea
Clapping
Drink Water
Hand-waving
Pointing

(a) RGB-based HAR framework during testing phase, for a single testing example.


Testing Phase
Skeleton Sequence
t
Input Data
Skeleton poses extracted from the RGB Data. It is extracted a single skeleton from each frame. Only single person activities are considered.
Preprocessing
It is set a fixed sequence length of 300 frames. If the input skeleton sequence < 300, then repeat skeleton sequence until the 300 frames requirement is met.
Deep Learning Based Feature Extraction and Classification
It is used a Graph Convolutional Network (GCN) for feature extraction and classification.
Output vector
0.01
0.02
0.02
0.01
0.94
Label
Based on the output vector the activity with the highest probability is chosen as the predicted label for the input video.
Nausea
Clapping
Drink Water
Hand-waving
Pointing

(b) HAR-S-bF during testing phase, for a single testing example.

Figure 1.2: Illustration of the testing phase for the frameworks evaluated in this work, for a single testing example.

1.3 Main Contributions

Typically HAR RGB-based frameworks use the whole image as input for the feature extraction and classification module managing to achieve high classification accuracies. However, it is common for these frameworks to include some sort of attention mechanism [37, 38] that is able to focus on the subjects present in the image, ignoring background clutter and irrelevant regions of the image. For the purpose of this dissertation the methods used do not include any sort of attention mechanism, which initially led to poor classification accuracies. Thus, to address this issue, it is proposed the introduction of a human detection stage to extract, from each frame, a bounding box containing the person performing the activity. The resulting bounding boxes are then used as input data for the RGB-based framework. This approach improves the classification accuracy when compared to a more traditional approach where the whole frame is used as input to the framework.

A subset of the NTU RGB+D 120 Dataset was prepared, more concretely frames were extracted from the original videos and stored as 320x240 pixels images, for training and evaluating seven different Deep Learning (DL) architectures. It was also developed the necessary source code to make the dataset compatible with source code available in the GitHub repository[1] created by [7] and used in the dissertation.

Seven different DL architectures for both feature extraction and classification were trained from scratch and evaluated on the NTU RGB+D 120 Dataset [2]. Pre-trained models on Kinetics-600 [39] of the architectures used in this dissertation were also used to transfer learning by only optimizing their output layer on the NTU RGB+D 120 Dataset, which improved the classification accuracy by reducing the overfit. The DL architectures performance is compared in terms of classification accuracy and runtime.

A small-scale human activity recognition dataset was collected at the Human-Centered Mobile Robotics Laboratory in the Institute of Systems and Robotics - University of Coimbra. The dataset contains 300 videos depicting a single activity with 2-3s each and 10 subjects participated in the dataset recording each activity in 3 different angles. The RGB-based framework was evaluated on this dataset.

1.4 Dissertation Outline

The dissertation is organized as follows:

- **Chapter 2:** State of the art of the HAR problem with a brief historic note;
- **Chapter 3:** Presents the theoretical concepts needed to understand the HAR frameworks developed, trained and evaluated in this dissertation;
- **Chapter 4:** Presents the material and methods used in the dissertation such as the NTU RGB+D 120 Dataset [2], DL architectures used for the feature extraction and classification modules or source codes used to train and evaluate the frameworks;
- **Chapter 5:** Explains the developed work, presenting the proposed solutions to overcome the problems encountered during the course of the work;
- **Chapter 6:** Results obtained during the developed work are shown and discussed;
- **Chapter 7:** A conclusion and suggestions for future work are presented.

[1]https://github.com/okankop/Efficient-3DCNNs

Chapter 2

State of the Art

In this chapter it is presented an overview of the most common methods for solving the HAR problem. The chapter is divided into two parts: (1) Video-based methods and (2) Wearable/Ambient Sensor based methods.

2.1 Human Activity Recognition

A generic HAR framework is presented in Fig. 2.1. The framework's input is either a set of video frames or a temporal sequence of inertial data, depending if the HAR framework is Video-based or Wearable/Ambient sensor-based, respectively. The preprocessing stage generates input data for the feature extraction module. The extracted features, in the form of a feature vector, are then fed to the classifier. After processing the feature vector, the classifier outputs a prediction for the activity depicted in the sequence of frames or temporal sequence of inertial data.

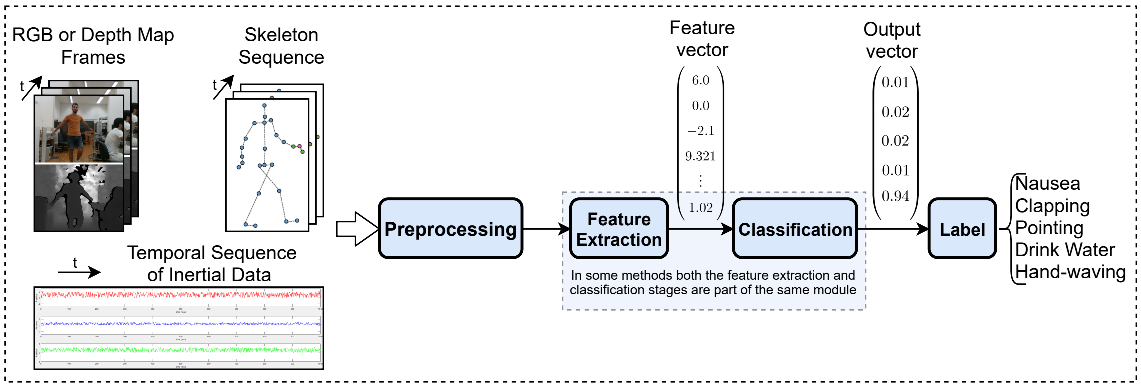


Figure 2.1: Typical HAR framework.

A possible categorization diagram for the HAR frameworks is presented in Fig. 2.2. There are two major aspects that characterize an HAR framework: (1) sensor modality used to generate the input data and (2) the feature extraction and classification method.

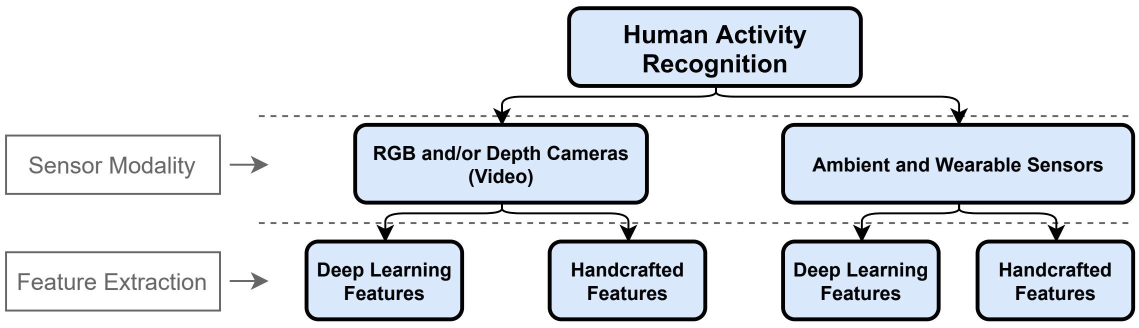


Figure 2.2: Categorization of the HAR frameworks.

Regarding sensor modality there are two major categories: video-based approaches and wearable/ambient sensor-based approaches. Video-based approaches, which are the focus of this work, generally use RGB, RGB-D or depth map videos as inputs. On the other hand, wearable/ambient sensor-based approaches use wearable and/or ambient sensors to collect the input data. Feature extraction is performed by either a deep neural network or an handcrafted method.

For the video-based methods it is important to note that feature extraction and classification can be performed on two possible data modalities: image sequences (RGB and/or depth map images) or human skeleton sequences (extracted from the RGB and/or depth map images).

2.1.1 Video-based Methods

In the last decade very cost-effective RGB-D sensors such as Intel RealSense Cameras, Microsoft Kinetic and Asus Xtion have been released allowing for an easier access to this technology and contributing for significant advances in the research of HAR methods [40–42].

The first methods for video-based HAR were broadly divided into two categories [43]: (1) state-space approaches and (2) template matching approaches. In (1) each pose or activity is encoded as a state and the recognition is performed by a Hidden Markov Model [44–46]. In (2) a template is created for each activity or pose and the recognition is made by matching the extracted features with a set of predefined templates [47–49].

Throughout the last decade the DL success has also extended to the HAR problem. Taylor *et al.* [50] used a multi-stage architecture that combined convolutional and fully-connected layers for HAR. First a convolutional gated Restricted Boltzmann Machine extracts features from every successive frame and 3D convolutional layers are used to capture mid-level spatiotemporal cues. The final layer is a fully connected layer whose output is normalized using the softmax function. Ji *et al.* [51] presented a 3D Convolutional Neural Network (CNN) architecture for HAR, the model extracts both temporal and spatial features from the input sequence. Authors reported a classification accuracy of 90.2% on the KTH dataset [52], which is a dataset composed of 6 activities (boxing, handclapping, hand-waving, jogging, running and walking).

Karpathy *et al.* [53] reported the problem of runtime performance when training CNNs, stating that training a CNN could take weeks to train on large-scale datasets (for context, the paper was published in 2014). Thus, it was proposed a two stream CNN architecture: (1) a high-resolution *fovea* stream receives a 89×89 pixels resolution video cropped from the original video center, and (2) a low-resolution context stream receives a downsampled 89×89 pixels resolution video from the original video. Both streams were composed of alternating convolution, normalization and pooling layers, and ending with two fully connected layers. For fusing the two streams, authors evaluated late, early and slow fusion. The method was tested on the large-scale Sports-1M dataset [53] and the model with slow fusion achieved an accuracy of 60.9%. Authors also made experiments on transfer learning on UCF-101 dataset [54] achieving a 3-fold accuracy of 65.4% when fine-tuning the top 3 layers with the slow fusion network pre-trained on the Sports-1M dataset.

Carreira and Zisserman [55] introduced Two-Stream Inflated 3D ConvNet (I3D) for video classification tasks. I3D is based on 2D CNN inflation, i.e., filters and pooling kernels of image classification CNNs are expanded into 3D allowing for spatio-temporal feature extraction. It is composed of two streams, one trained on RGB inputs and another on optical flow inputs, where each stream is trained separately and their predictions are averaged at test time. Authors showed that after pretraining their model on the Kinetics Dataset [55] the classification accuracy improved up to 80.9% on HMDB-51 Dataset [56] and 98.0% on UCF-101 Dataset [54].

In a more recent approach [57] the concept of SlowFast networks was presented for HAR. SlowFast networks are composed of two CNNs (pathways) that operate at different frame rates: (1) a Slow pathway is used to capture spatial semantics at low frame rates and (2) a Fast pathway is used to capture motion at high frame rates. The Slow pathway can be any CNN that operates on a video as a spatio-temporal volume. For this pathway it is applied a temporal stride τ on the input set of frames (only one out of τ frames is extracted from the input set). Typically $\tau = 16$, meaning that for a video recorded at 30 Frames

Per Second (FPS) roughly two frames are extracted per second. The Fast pathway can also be any CNN that operates on a spatio-temporal volume. However, for this pathway the temporal stride is given by τ/α, typically $\alpha = 8$, meaning that this pathway extracts 8 times more frames from the input set than the Slow pathway. Lateral connections are used to fuse the information of both pathways. These connections are unidirectional from the Fast pathway to the Slow pathway. A Global Average Pooling (GAP) is performed on each pathway's output and both output feature vectors are concatenated and are used as the input to a fully-connected classifier layer. The architecture achieved a classification accuracy of 81.8% on Kinetics-600, and also surpassed the state of the art results in two activity detection datasets: Atomic Visual Actions Dataset [58] and Charades Dataset [59].

Girdhar *et al.* [38] proposed an Action Transformer model for recognizing and localizing human actions in videos. The model is able to both detect the persons in the video and classifiy their activities/actions. The model is divided into two main parts, being the first part responsible for generating features and region proposals (bounding boxes) for the persons present in the image while the second part uses the previously generated features to predict the actions and improve the bounding box/boxes limits. The architecture was evaluated on Atomic Visual Actions Dataset [58] achieving state of the art results.

Köpüklü *et al.* [7] reported that in recent years the focus on the action recognition problem has been on creating architectures that can achieve the best classification accuracy. However in real-world applications the hardware available lacks the processing power, memory size or even power capacity to be capable of running the models at decent real-time performance. Thus, the authors converted popular resource efficient 2D CNNs to 3D CNNs and evaluated their classification accuracy on 3 popular datasets (Kinetics-600 [39], Jester [60] and UCF-101 [54]) as well as their real-time performance in terms of runtime. Results showed that the 3D CNN resource efficient architectures are capable of achieving comparable classification accuracies with wider and deeper models while having less parameters and Floating Point Operations Per Second (FLOPS). For example, authors compare 3D-ShuffleNetV1 2.0x with ResNet-18 [61] both achieving similar classification accuracies while ResNet-18 has 7 times more parameters and 14 times more FLOPS.

Another approach for HAR is to extract the human skeleton from each frame and use that information to predict the activity. The human skeleton can be obtained in one of two ways: (1) using a pose estimation algorithm that extracts the 2D or 3D human skeletons from each RGB, RGB-D or Depth Map image [62–64] or (2) using a camera such as Kinect V2 that provides both the 2D or 3D human skeletons present in the frame. Using the human skeleton for HAR has the advantage, over the methods that only use the image for HAR, of being more robust to background clutter, being viewpoint invariant and being computationally efficient due to the smaller data size [65, 66]. Shi *et al.* [1] proposed Two-Stream Adaptive Graph Convolutional Network (2s-AGCN) a two stream graph convolutional network for skeleton-based HAR. The model is composed of two parallel streams: (1) joint stream (J-Stream) and (2) bone stream (B-Stream). J-Stream extracts features from the 3D location of each skeleton joint. B-Stream extracts features from bone information calculated from the 3D locations of the joints. The two streams are fused via decision level fusion. Both streams' output vector is normalized via a softmax layer and then are added together. The resulting vector is used to predict the action depicted in the input video. Li *et al.* [67] proposed Actional-Structural Graph Convolution Network (AS-GCN) where they extended the standard skeleton graph by including structural links that model the dependencies between joints, i.e. the graph links not only include the bones but also "artificial" bones that are connected between highly related joints. Actional links were also proposed, this links are specific to each activity and are learned directly from each activity. The model improved previous classification accuracies on both NTU RGB+D 120 Dataset [2] and Kinetics-600 [39].

2.1.2 Ambient and Wearable Sensors Methods

The Internet of Things (IoT) [68] concept has allowed the proliferation and interconnectivity of large amounts of small devices that are constantly collecting data about people's daily activities. The use of wearable and ambient sensors have the advantage of not suffering from ambient occlusion or brightness changes, which are the two major problems in the video-based approaches. Moreover, sensors like gyroscopes and accelerometers are already embedded in common devices such as smartphones and smartwatches making them very accessible and cheaper than most sensors used in the video-based HAR methods [69–72].

There is a large variety of sensors that can be used to capture the subject's motion and interaction with the environment. For example, an accelerometer can provide valuable information about the subject's motion. Although it will be very difficult to distinguish between "eating" and "drinking" from inertial data alone. In this case a possible solution would be placing a pressure sensor on a cup and fusing information from both sensors to give a better estimate if the subject is in fact ”eating a chocolate bar” or ”drinking a cup of water” [73–75].

The most common used ambient and wearable sensors can be categorized as follows [76–78]:

– Wearable sensors: accelerometer, gyroscope, heart rate sensors or electromyography. Usually these are directly associated with the subject, collecting data about their motion patterns;
– Ambient sensors: RFID, light sensors, pressure sensors, reed switch sensors, Wi-Fi or bluetooth. These sensors provide valuable data about the interaction between the human and their environment. Generally gives information about the location of the subject in his/her environment or if the user interacted with a given object. Having this data can be valuable by providing the context in which a given action/activity is occurring.

Early research works in the HAR area were concerned with medical applications such as analyzing gait patterns and using accelerometer data to calculate energy expenditure [79–83]. Most of the feature extraction and classification was based on handcrafted methods, although some methods started introducing Neural Network (NN)s as classifiers [79, 83]. Application of HAR was then introduced to areas other than medicine, in applications where context awareness was required [84–86].

More recently, in [87] authors developed a CNN for HAR using accelerometer data. The network's input were 3 time series of 64 raw sensor data points, corresponding to the 3D coordinate frame axis (X, Y and Z). The framework was able to outperform the current state of the art methods from 3 public datasets (OPPORTUNITY Activity Recognition Dataset [88], a car manufacturing activity dataset [89] and Actitracker Dataset [90]).

Ordóñez and Roggen [91] presented a DL based framework that was capable of fusing multimodal sensor data. The authors designed a Deep Neural Network (DNN) (DeepConvLSTM) that combined both convolutional and recurrent layers. Both layers had its own purpose, convolutional layers transformed the input values into feature maps acting as feature extractors while recurrent layers modeled the temporal dynamics embedded in the feature maps extracted by the convolutional layers. A sliding window approach was used to select the input data from the raw data provided by each sensor. The method outperformed existing ones in 2 public datasets (OPPORTUNITY Activity Recognition Dataset [88] and a car manufacturing activity dataset [89]). Yang *et al.* [92] also presented a method based on a NN, with the difference that it was only composed of convolutional layers. The framework also achieved state of the art performance on 2 public datasets (OPPORTUNITY Activity Recognition Dataset [88] and Hand Gesture Dataset [93]).

A more recent approach is the work by Qin *et al.* [94], authors used the same technique as [95] and transformed the raw sensor data, from an accelerometer and a gyroscope (acceleration and angular velocity) into Gramian Angular Fields images. The framework contains two main blocks, the first is composed of 2 parallel deep residual networks for both acceleration and angular velocity data. Then a fusion layer fuses both previous networks' outputs and feeds the resulting data to the second block composed of another deep residual network. Classification is performed by 6 fully connected layers and the output vector is then normalized by applying the softmax function.

Recognition of fine-grained hand activities is also possible, in Laput and Chris [96] the authors proposed a method for the recognition of hands activities. The device used was a LG W100 smartwatch from LG which has an embedded Inertial Measurement Unit (IMU) that provides three axis accelerometer data. This data is converted into 3 spectrograms that are then fed to a CNN to perform the feature extraction and classification. An overall accuracy of 90.7% was reported on the 25 class dataset collected and made publicly available by the authors.

Although full DL based frameworks are very popular nowadays there are still works that use a combination of both DL and traditional approaches. An example is the work by Hassan *et al.* [97] where the authors use handcrafted features in conjunction with a classifier based on a deep belief network. A dataset with 12 classes was collected and made publicly available, an overall accuracy of 95.85% was reported outperforming existing methods.

Another possible approach is multimodal sensor fusion where both visual and sensor data are fused and used for inference. Both sensor-based and video-based approaches have their own disadvantages/drawbacks. Sensor-based methods lack the ability to distinguish more complex activities, are location and orientation dependent, multiple sensors are needed to recognize activities that involve partial body movements and wearing single or multiple sensors can be intrusive. On the other hand video-based approaches suffer from background clutter, illumination changes, viewpoint sensitiveness and can be computationally expensive. Both approaches have the potential to complement each other, sensor-based approaches are insensitive to background clutter, illumination and viewpoint changes while video-based approaches can distinguish more complex activities and do not require that the subjects wear any hardware [76, 98, 99]. Some works that follow this approach have been published in recent years. In [100] authors proposed a method that fused, at feature-level, both inertial and RGB-D. First, for both modalities, handcrafted features are extracted from raw data and concatenated into a feature vector. This vector is then fed to a Machine Learning (ML) based classifier, both Support Vector Machine and K-Nearest Neighbors methods were evaluated. Authors reported higher accuracies than existing methods on the UTD-MHAD dataset [101] for both subject-generic and subject-specific modalities.

Imran and Raman [102] also follow a similar approach but instead of a feature-level fusion the authors performed decision-level fusion. An architecture composed of 3 parallel streams analyses data from a gyroscope, RGB video and 3D skeleton joint data. For gyroscope data and RGB video data a CNN is used for feature extraction and classification. For the 3D skeleton data an RNN was used also for both feature extraction and classification. The method achieved an accuracy of 97.91% on the UTD-MHAD dataset [101].

Chapter 3

Background Material

This section presents the various theoretical concepts needed to understand and analyze the frameworks implemented and evaluated in this dissertation.

It is important to note that ML learning methods can be broadly divided into four main paradigms: (1) supervised learning, (2) unsupervised learning, (3) semi-supervised learning and (4) reinforcement learning. In (1) the goal is to learn a mapping function that maps from inputs x to outputs y, given a labeled training set (collection of input-output pairs). In (2) the training set is composed of only inputs and the goal is to find patterns in the training set. In (3) the training set is composed of both labeled and unlabeled examples. In (4) learning an input-output mapping is based on a continuous interaction with the environment where the agent learns by trial and error [103–105]. For the purpose of this dissertation the remainder of this chapter will focus on the supervised learning paradigm.

3.1 Deep Learning

DL is a sub field of ML that introduced the concept of DNNs. A key characteristic of DL architectures is the number of layers that compose a DNN where DL architectures can have more than one hundred layers [61, 106]. DL architectures such as CNNs, RNNs and LSTMs have revolutionized many fields, e.g. natural language processing [107], speech recognition [108], computer vision [53, 109], drug design [110] and genetic mutations classification [111]. In a more general way, DL models can solve tasks such as classification [112], regression [113], transcription [114, 115] or machine translation [116].

3.1.1 Training and testing Process

The main goal of a DL algorithm is to approximate a function f that maps a given input x into an output y, i.e. $y = f(x; \theta)$. For the specific case of HAR, the goal is to approximate a function where y is a discrete value, i.e. denotes a given class such as "Drink Water", "Hand-waving" or "Pointing". The network defines a mapping $y = f(x; \theta)$ and tries to learn the values θ (weights) that result in the best approximation. Assuming a classification problem, training a neural network is performed by feeding it with training examples x with a known label $y = f(x; \theta)$ and then updating the θ values in such a way that the error between the prediction and the ground truth is minimized [103].

To measure how close the network's predictions are to the desired outputs a function has to be defined. This function is named 'cost function' and for a classification problem with multiple labels a possible cost function is the cross-entropy loss which is defined as [105]:

$$J = -\sum_{i=1}^{n} t_i log(p_i) \tag{3.1}$$

where n denotes the number of classes, t_i denotes the ground truth label and p_i denotes the softmax probability for the i^{th} class. Thus, the algorithm's goal is to minimize the loss function. For such purpose, gradients with respect to the loss function are calculated for each learnable parameter θ in the back propagation step. Next, based on the calculated gradients each weight θ is updated based on an optimization algorithm such as gradient descent [117]. For the specific case of Stochastic Gradient Descent (SGD) the equation is the following [103, 117]:

$$\theta_k = \theta_{k-1} - \eta \cdot \nabla_\theta J(\theta_{k-1}; x^{(i)}; y^{(i)}) \tag{3.2}$$

where k denotes the k^{th} optimization step, i denotes the i^{th} training example, θ denotes the weight to be updated, η denotes the learning rate and $\nabla_\theta J(\theta_{k-1}; x^{(i)}; y^{(i)})$ is the gradient of parameter θ with respect to the loss function J. In SGD the weights are updated each training example, contrary to other popular optimization algorithms such as Batch Gradient Descent (BGD) and Mini-batch Gradient Descent (MBGD) where the weights are updated after iterating through all training examples or after iterating through a mini-batch of training examples, respectively. Two important aspects need to be considered when using SGD as the optimization algorithm: (1) the learning rate should slowly decrease during the training phase and (2) the dataset should be reshuffled every epoch to prevent biasing the optimization algorithm [117].

To evaluate the DL algorithm two aspects can be addressed:

- The algorithm's ability to make the training loss function value decrease;
- The algorithm's ability for making the gap between the training and testing loss

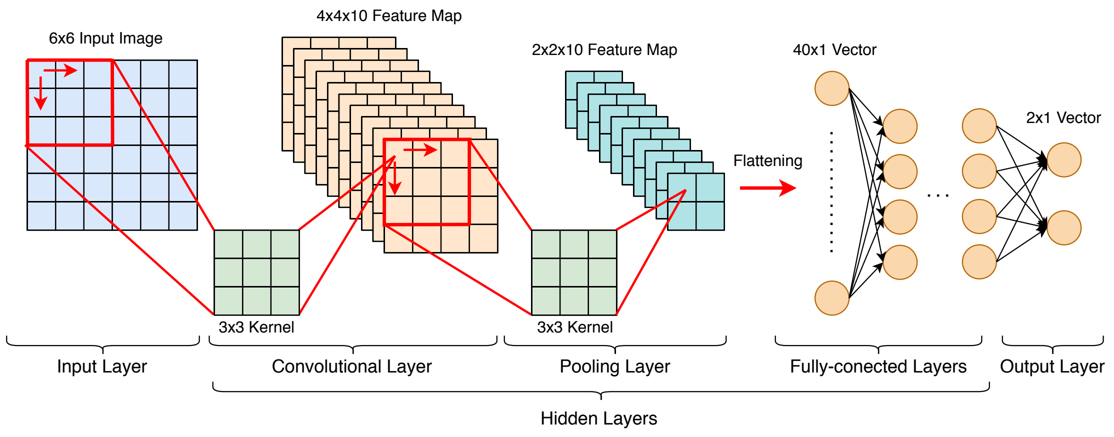


Figure 3.1: Example of a CNN for a binary classification problem with an input image of dimensions 6x6 pixels.

function values small.

Based on these two aspects two problems can be identified: (1) overfitting and (2) underfitting. (1) Occurs when the gap between the testing and training loss function values is too large which means that the function learned has high bias; (2) occurs when the training error is too large meaning that the algorithm failed to approximate an appropriate function, meaning that the function approximated has high variance [103, 118].

3.2 Convolutional Neural Networks

CNNs are commonly used to solve problems that involve data organized in 2D or 3D arrays. The most common applications are related to computer vision problems such as video classification [53], image classification [112] and object detection and recognition [119]. An illustrative example of a CNN architecture is presented in Fig. 3.1. There are mainly 3 types of hidden layers in a CNN: convolutional layers, pooling layers and fully connected layers. The network's input is an image or sequence of images stacked together (video). The ouput, in the case of a classification problem, is a 1 dimensional vector with the scores for each class. This output vector can then be normalized to a probability distribution representing the probabilities for each class using softmax function.

3.2.1 Convolutional Layers

Generally the first layers in a CNN are convolutional layers. They act as feature extractors and are known for their ability to extract high-level features from the input data [112, 120]. Each layer applies the convolution operation (Fig. 3.2) between its input and a given

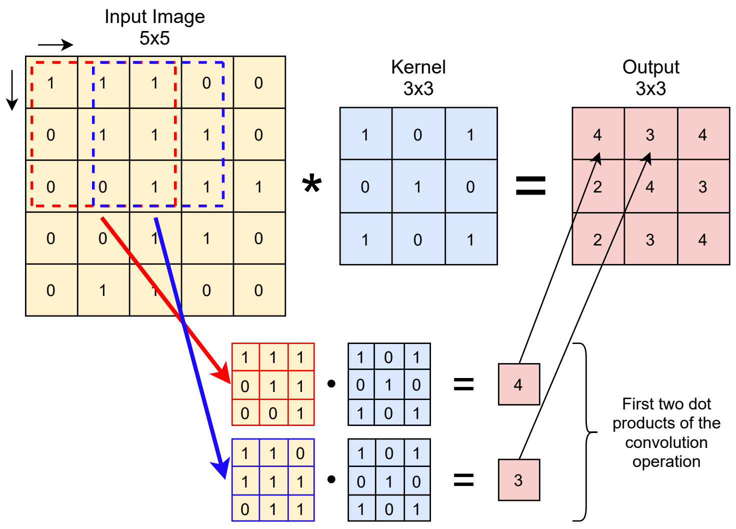


Figure 3.2: Example of a convolution between a 5x5 input and a 3x3 kernel with a stride of 1 in both axis and 0 padding.

kernel. The convolution operation is performed by overlapping the kernel with the input image and performing the dot product between the pixels values and the kernel's weights. In the example presented in Fig 3.2 a total of 9 dot products were performed.

There are some parameters that can be set when performing a convolution:

- Kernel dimensions: height and width of the kernel;
- Stride: number of rows or columns that the kernel slides between each application of the kernel to the data;
- Padding: number of extra rows or columns added to the image boundaries.

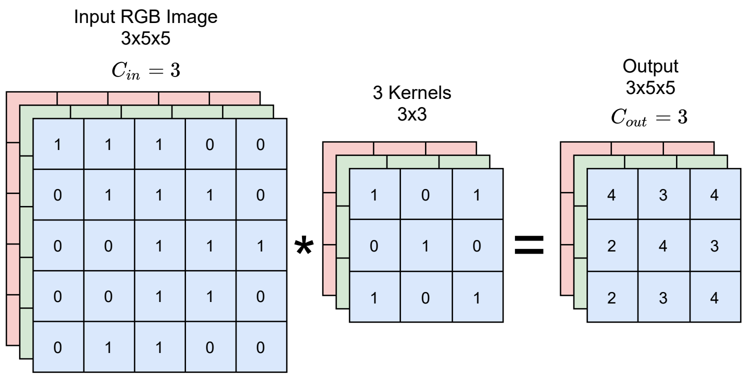
The kernel values represent learnable parameters that are optimized during the training process. Depending on the parameters chosen the size of the output can vary. A general formula to calculate the output feature map dimensions is [121]:

$$n_{o_{HW}} = \left\lfloor \frac{n_{i_{HW}} + 2p - f}{s} \right\rfloor + 1 \tag{3.3}$$

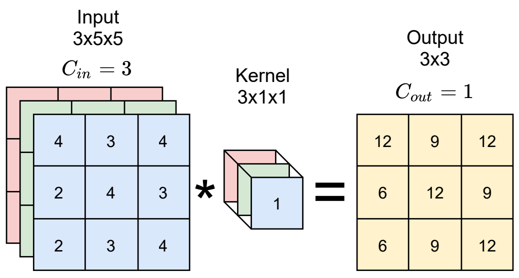
where $n_{o_{HW}}$ is the width/height of the output array, $n_{i_{HW}}$ is the width/height of the input array, p is the padding, s is the stride and f is the dimension of the kernel.

Depthwise Separable Convolution

This convolution factorizes a standard convolution into a depthwise convolution followed by a pointwise convolution. In a depthwise convolution each input channel is convolved with its own kernel, and therefore the number of channels is kept during the operation,



(a) Depthwise Convolution



(b) Pointwise Convolution

Figure 3.3: Depthwise separable convolution illustration, it is applied a depthwise convolution followed by a pointwise convolution. It is assumed that the two channels (Red and Green) behind the front channel (Blue) have the exact same values for calculation purposes.

i.e. $C_{out} = C_{in}$. A pointwise convolution applies a $1{\times}1{\times}C_{in}$ kernel to all input pixels thus producing an output with a single channel while keeping the height and width dimensions of the input tensor. By separating a conventional convolution into two steps both computation time and model dimensions are significantly reduced [5]. MobileNetV1 [5] and MobileNetV2 [6] are examples of architectures, which were evaluated in this work, that use this type of convolution. Both operations are illustrated in Fig. 3.3.

3.2.2 Pooling Layers

Typically pooling layers are used after convolution layers and their purpose is to achieve spatial invariance by reducing the spatial dimensions of the input feature map. Regions of the input feature map are summarized statistically by applying a certain function such as average of the values of that region (average pooling), selecting the highest value (max pooling) or a weighted average based on the distance to the central element [103, 122]. In Fig. 3.4 a max pooling operation is illustrated. A pooling layer does not contain learnable

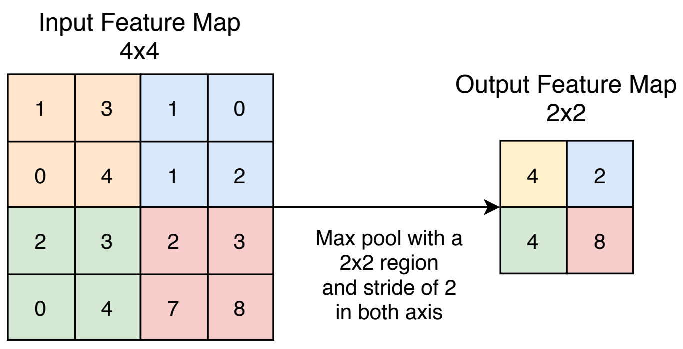
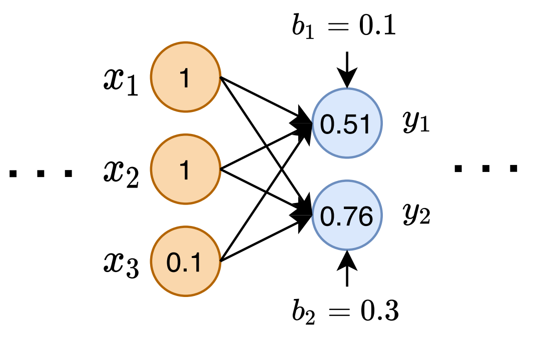


Figure 3.4: Illustration of a max pooling operation. It is used a 2x2 region with a stride of 2 in both axis.

parameters.

3.2.3 Fully Connected Layers

In a Fully Connected (FC) layer all nodes are connected to all output units from the previous layer. The value of each node is calculate by a linear combination of the output units of the previous layer plus an added bias value followed by an activation function [104]. Figure 3.5 illustrates a simple fully connected layer with 3 input units and 2 output units:



$$y'_i = \textstyle\sum_j w_{ij}x_j + b_i \qquad y_i = ReLU(y'_i)$$

$$w_1 = [0.2\ \ 0.2\ \ 0.1]^T \qquad w_2 = [0.3\ \ 0.1\ \ 0.6]^T$$

$$y'_1 = 1 \cdot 0.2 + 1 \cdot 0.2 + 0.1 \cdot 0.1 + 0.1 = 0.51$$
$$y_1 = ReLU(y'_1) = 0.51$$

$$y'_2 = 1 \cdot 0.3 + 1 \cdot 0.1 + 0.1 \cdot 0.6 + 0.3 = 0.76$$
$$y_2 = ReLU(y'_2) = 0.76$$

Figure 3.5: Illustration of a FC layer. The input feature map contains 3 units and the output feature map contains 2 units. b_i denotes the bias values, x_i denotes the input activation units, w_i denotes the weight vectors and y_i denotes the output units. The activation function used in this example is the Rectified Linear Unit (ReLU). On the right side it is presented the calculations for obtaining the output nodes values.

Thus, for a given output unit y_i its value is calculated based on the following equation [104]:

$$y_i = f\left(\sum_j w_{ij}x_j + b_i\right) \tag{3.4}$$

where f denotes the activation function and both w_i and b_i represent the learnable parameters that are optimized during the training process. Generally FC layers are used as the last layers in CNNs architectures. The output feature map of the last convolutional layer is reshaped into a vector and fed to the FC layers. Then a softmax logistic regression layer normalizes the output vector from the last FC layer [112, 123–125].

3.2.4 Global Average Pooling Layers

Lin *et al.* [123] stated that FC layers are prone to overfitting, which lowers the generalization ability of the architecture. As a solution the GAP layer was proposed as a replacement for the FC layers. In a GAP layer, each input feature map is averaged and the resulting values are concatenated into a vector, meaning that the output vector will have has many elements as the number of input feature maps. This layer does not contain any learnable parameters contrary to the FC layers which helps preventing overfitting. Another advantage is the fact that GAP sums out the spatial information which makes it more robust to spatial translations in the input. GoogLeNet [126] and ResNet [61] are example of successful architectures that use GAP layers in the last layers of the architecture. Figure 3.6 illustrates a GAP layer:

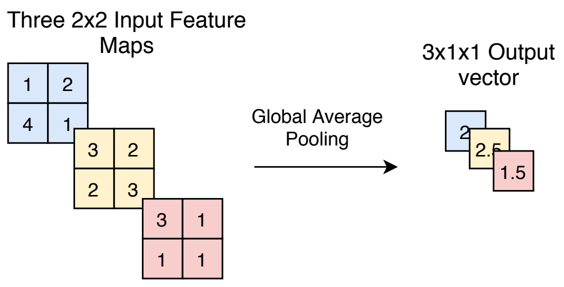
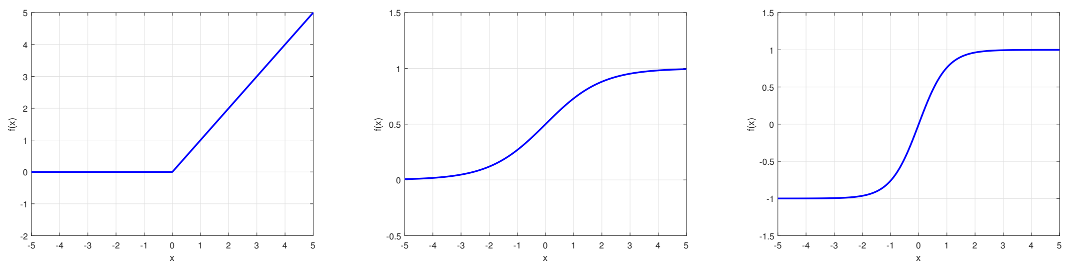


Figure 3.6: Illustration of a GAP layer. The inputs are three 2x2 feature maps.

3.2.5 Activation Functions

Activation functions are used to compute the output values of the hidden layers in a neural network. Generally the activation function is nonlinear allowing for the network, as a whole, to learn a nonlinear function of its input [103]. The following list presents the most common activation functions and their visualization is shown in Fig. 3.7:

- ReLU: $f(x) = max(0, x)$;
- Sigmoid: $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$;
- Hyperbolic Tangent: $f(x) = 2\sigma(2x) - 1 = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.



(a) ReLU (b) Logistic Sigmoid (c) Hyperbolic Tangent

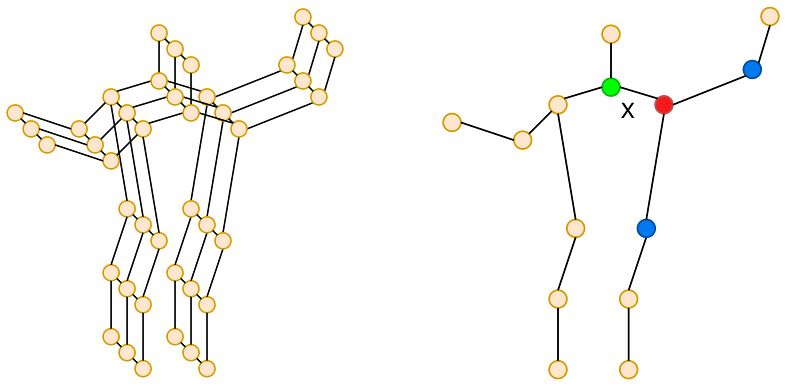
Figure 3.7: ReLU, Logistic Sigmoid and Hyperbolic Tangent activation functions.

3.3 Graph Convolutional Networks

Graph Convolutional Networks (GCN) are specially suited for graph structured data. Data such as citation networks, gene data, social networks or the human skeleton are all examples of data that can be structured as a graph [127]. It is easily perceivable that graphs can have very different sizes, the nodes can be very unordered and each node can have a different number of neighbor nodes. This characteristics impose a difficulty when applying the classical convolution operation to this type of data. To solve this problem, a possible solution is to use GCNs where the classical convolution operation is generalized to graph data [127–129].

GCNs have been successfully applied in problems related to 3D semantic segmentation [130], classification and segmentation of point clouds [131], human-object interaction [132], machine translation [133, 134] or human activity recognition [1, 67, 135, 136].

For the purpose of this dissertation it is of interest the use of GCNs to solve the HAR problem. Thus, in the following paragraphs a possible spatiotemporal graph to model the human skeleton structure and motion will be presented. In Fig. 3.8(a), is presented the spatiotemporal graph that represents a temporal sequence of 3 skeletons. The orange circles represent the vertexes that are defined based on the human skeleton joints. The edges connecting each vertex represent the bones connecting each joint and are named spatial edges. For the temporal dimensions the same vertex is connected, by temporal edges, with its equivalent in adjacent frames. Each vertex attribute is set as the 3D coordinate vector of the corresponding joint.



(a) Spatiotemporal graph. (b) Mapping strategy.

Figure 3.8: Left: Spatiotemporal graph used to model the human skeleton structure and motion. Right: Mapping strategy for the graph convolution. Adapted from [1].

3.3.1 Graph Convolution Definition

Based on the spatiotemporal graph previously defined, the graph convolution operation on a given vertex v_i is defined as:

$$f_{out}(v_i) = \sum_{v_j \in B_i} \frac{1}{Z_{ij}} f_{in}(v_j) \cdot w(l_i(v_j)) \tag{3.5}$$

where f_{out} and f_{in} denote the output and input feature maps. v_i denotes the target joint and v_j denotes the joint j at a distance of 1 edge to the target joint. B_i is the sampling area, similar to the kernel size in the typical convolution operation, w is the weight function. The number of weight vectors is fixed, while the number of joints in the sampling area is variable depending on the target joint. Thus, it is necessary to create a mapping function l_i to assign each joint with its own weight vector. The strategy adopted for the mapping function is illustrated in Fig. 3.8(b). The sampling area (defined as the joints that are adjacent to the target joint in the spatial dimension) encloses the green and blue joints. The red joint represents the target joint. B_i is divided into 3 subsets (S_{ik}): S_{i1} is the target joint; S_{i2} is the centripetal subset and contains the vertexes that are closer to the center of gravity of the skeleton (marked as 'x' in the graph, neck joint), contains the green joint; S_{i3} is the centrifugal subset and contains the vertexes that are farther from the center of gravity of the skeleton, blue vertexes. Z_{ij} is the number of vertexes in the subset that contains the joint v_j.

Thus, the feature map is a C×T×N tensor where C denotes the number of channels (when using 3D coordinates $C = 3$), T denotes the number of frames and N denotes the number of vertexes.

To implement the graph convolution, in the spatial dimension, the following equation is defined:

$$\mathrm{f}_{\mathrm{out}} = \sum_{k}^{K_v} W_k(f_{in}A_k) \cdot M_k \tag{3.6}$$

where K_v denotes the kernel dimension. $A_k = \Lambda_k^{-\frac{1}{2}} \bar{A}_k \Lambda_k^{-\frac{1}{2}}$, where $\bar{A}_k$ is an $N \times N$ matrix, and the element $\bar{A}_k^{ij}$ indicates if the joint v_j is in the subset S_{ik} of joint v_i. Thus, $\bar{A}_k$ maps each joint to its corresponding weight vector. $\Lambda_k^{ii} = \sum_j(\bar{A}_k^{ij}) + \alpha$ is the normalized diagonal matrix. W_k is a $C_{out} \times C_{in} \times 1 \times 1$ weight vector which represents the weighting function $(w(l_i(v_j))$ in Eq. 3.5. M_k is an $N \times N$ learnable matrix that represents the importance of each joint [137].

For the temporal dimension the convolution is similar to a standard convolution. It is defined that each joint is connected to the equivalent vertexes in the consecutive frames. Then a $K_t \times 1$ convolution is applied on the output feature map computed previously, where K_t denotes the kernel size of temporal dimension.

3.4 Data Augmentation Techniques

DNN are heavily dependent on large amounts of training data to avoid overfitting and generalize well to new data. However, in practice the amount of data is limited and it is not always possible to collect a large training set. Data augmentation techniques help alleviate this problem by creating new training data [103, 138]. Data augmentation has been particularly effective in computer vision problems where new training examples can be created from the existing training set. Operations such as translating the image a few pixels in any direction, cropping a region, injecting salt and pepper noise, zoom in, rotating, flipping vertically or color space transformations are all examples of commonly used techniques in computer vision problems [138, 139]. Examples of spatial and temporal augmentation techniques are provided in Chapter 5.

3.5 Transfer Learning

Transfer learning can be defined as transferring knowledge learned in a given *base* dataset (*base* model) to a *target* dataset (*target* model). This method can be a powerful tool to help prevent the overfit problem. It is important that the features learned in the *base* model are general and not specific to the *base* dataset, otherwise they might not be suitable for the *target* dataset [140]. Generally, the first-layers features are general and aplicable to many datasets, these can specialize in identifying edges, corners or other general shapes. On the other hand, the last-layers features tend to be specific to a given dataset and learn complex shapes specific to a given class [103, 141, 142].

Two types of transfer learning can be identified: (1) fine-tuning the whole *base* model or (2) use the *base* model as feature extractor. In (1) the *base* model's weights of every layer are optimized for the *target* dataset, in (2) the *base* model's first layers' weights are frozen and are used as feature extractors while the last layers' weights are optimized for the *target* dataset acting as classifiers [140]. The option of whether or not to fine-tune the whole model or optimize only the last layers depends mainly on the size of the *target* dataset and the number of parameters in the first layers. If the *target* dataset is small and the number of parameters in the first layers is large then the best option is to use the *base* model as feature extractor and optimize the last layers on the *target* dataset, otherwise if it is used a fine-tuning the model could overfit the *target* dataset. On the other hand, if the *target* dataset is of large-scale and the number of parameters is low, and thus the overfitting is not a problem, then fine-tuning the whole *base* model would be more appropriate [140]. It is also important to note that generally, when training a pre-trained model in a new dataset, the model converges faster, and thus requires less time to train than training the model from scratch on the new dataset. During the course of the work it was possible to verify that, in fact, the training process is faster when training a pre-trained model.

For the purpose of the dissertation, due to the high number of parameters in the DL architectures used and the small-scale property of the datasets, the architectures are used as feature extractors and only the last layer's weights are optimized.

Chapter 4

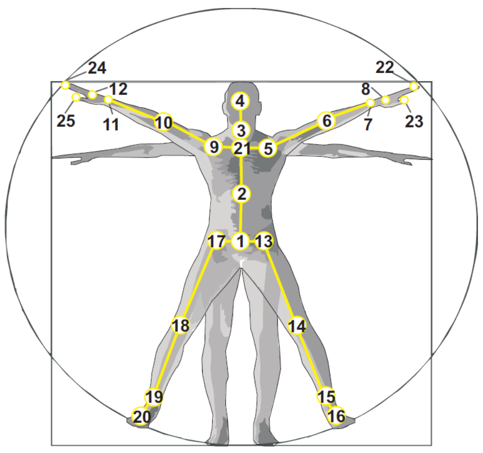
Materials and Methods

In this chapter it is presented the different materials and methods used in the dissertation. The chapter is organized as follows:

1. Dataset used for training the neural networks;
2. Performance metrics for evaluating the frameworks's performance;
3. DL architectures used for feature extraction and classification of the human activities;
4. Object detection framework used for extracting from each image a bounding box containing the subject performing the activity;
5. Brief introduction to the PyTorch package as well as an overview on how to create and train a neural network;
6. Presentation of the source codes, based on the PyTorch package, used to train and evaluate the neural networks;
7. Presentation of the mobile platform used for evaluating the frameworks and of the Depth camera used for collecting the dataset.

4.1 NTU RGB+D 120 Dataset

NTU RGB+D 120 [2] is a dataset for human activity recognition. It features 120 activities including daily actions, medical conditions and mutual actions. It was recorded using 3 concurrent Microsoft Kinect v2 cameras and contains 114480 activity samples (around 954 video samples per class). Each sample has the following data modalities: (i) RGB video; (ii) depth map video; (iii) 3D skeletal data and (iv) Infrared (IR) video. The RGB videos have 1920×1080 resolution and the IR and depth map videos have 512×424 resolution. The 3D skeletal data contains the 3D coordinates of the 25 major body joints in each frame [2].



Dataset Characteristics		
#Action Classes	120	82 daily actions 12 medical conditions 26 mutual actions
#Subjects	106	15 different countries heights: 1.3m to 1.9m ages: 10 to 57 years
#Views	155	32 setups 3 angles: $-45°, 0°, +45°$ different backgrounds different locations

Figure 4.1: From left to right: Configuration of the 25 joints (adapted from [2]) and dataset characteristics.

Each subject performed every action twice, one facing a camera positioned to their left and another facing a camera positioned to their right. To further increase the number of camera views the authors used 32 different camera angles and heights configurations [2].

The authors proposed two methods for evaluating the classification algorithms performance:

- Cross-subject evaluation: it is defined a group of 53 subjects as the training group and the remaining 53 subjects belong to the test group.
- Cross-setup evaluation: a similar procedure was applied, setups were divided into 2 groups, one for training the frameworks and the other for testing their performance.

Although it is of no interest to this dissertation the authors also introduced a one-shot recognition setting. The dataset is divided into two parts, one composed of 100 classes for training. And a smaller group composed of 20 classes to test the one-shot recognition capacity of the frameworks.

Sample images of this dataset, a table with the different camera setups and the dataset classes are shown in the Appendix C.

4.2 Performance Metrics

To evaluate the frameworks' performance, the classification accuracy was calculated for each model. Classification accuracy is defined as the ratio of the number of correct predictions to the total number of predictions made, expressed in percentage:

$$\text{Accuracy (\%)} = \frac{\text{Number of correct predictions}}{\text{Total number of prediction made}} * 100\ (\%) \tag{4.1}$$

A confusion matrix was also constructed for each model. Confusion matrices provide a representation of the classification performance of the model. Each row represents the true label while each column represents the predicted labels for each true label. Table 4.1 provides a generic example of a confusion matrix for a problem with 3 possible classes. In the left table for the true label A, 90 examples were labeled as class A, 1 example was labeled as class B and 9 examples were labeled as class C. This means that class A was labeled correctly in 90% of the examples.

Table 4.1: Left table: confusion matrix. Right table: confusion matrix in percentages.

		Predicted Class		
		A	B	C
True Class	A	90	1	9
	B	1	98	1
	C	5	15	80

		Predicted Class		
		A	B	C
True Class	A	90%	1%	9%
	B	1%	98%	1%
	C	5%	15%	80%

Frameworks' real-time performance was also evaluated, the total time it takes to process a video sample is calculated. Depending on the framework this time is calculated individually for each framework's stage and then all times are summed giving the total time for analyzing a video sample. This is an important metric for the context of this dissertation since one of the objectives is to implement an HAR framework with real-time performance.

4.3 Deep Learning Architectures for HAR

A total of 6 DL architectures were trained and tested in this dissertation, of which five are 3D versions of well known 2D architectures and one is a popular 3D CNN for large-scale supervised training datasets:

– ResNets: He *et al.* [61] address a common issue that arises when training deeper neural networks: increasing the depth of a network leads to an accuracy saturation and then rapid degradation. To overcome this issue the authors introduced a new concept: *residual learning*. This concept is implemented by adding a "shortcut connection", namely *identity shortcut connection*, to neural networks [61]. This type of connection skips one or more convolutional layers by performing an identity operation over the input and adding it to the output of the skipped connections, Fig. 4.2. Results showed that this type of deeper networks (Residual Networks) performed better when compared with their equivalents without "shortcut connections" and achieved state-of-the-art results. In this dissertation 3D versions [3] of 3 ResNets variations were tested, namely, ResNet-18 (Fig. 4.2(a)), ResNet-50 (Fig. 4.2(b)) and ResNeXt 4.2(c)) [3].

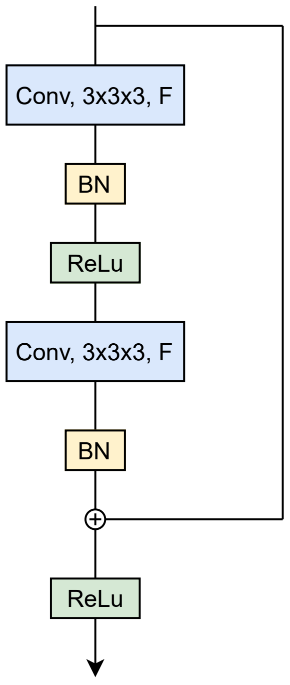
– Convolutional 3D (C3D) [4]: C3D (Fig. 4.3) is a 3D CNN composed of 8 convolutional layers, 5 max-pooling layers and 3 fully connected layers, the output vector is normalized by a softmax function. It has a simple and generic architecture that achieved state of the art results in a variety of video classification benchmarks [4].

– MobileNetV1 [5]: MobileNetV1 introduces a new type of convolution called *depthwise separable convolutions*, presented in Chapter 2. Authors reported that MobileNet achieves performances close to the state of the art in various tasks while having a smaller model and faster computation times when compared to the state of the art models. A 3D version [7] of MobileNetV1 was used in this dissertation. Figure 4.4(a) presents an illustration of the main building block of MobileNet V1.

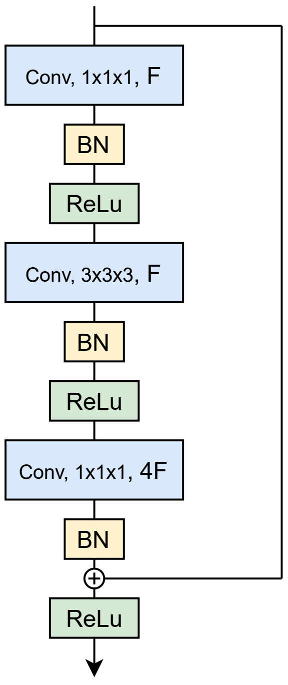
– MobileNetV2 [6]: this second version of MobileNet builds upon the concept of depthwise convolution while adding two new concepts: linear bottlenecks and shortcut connections. Linear bottlenecks allows for reduced memory usage during inference by first expanding the number of channels by applying a 1×1 convolution then applying a depthwise convolution followed by another 1×1 convolution to restore the number of channels. Shortcut connections on the other hand allow for deeper models following the same principle introduced by He *et al.* [61]. A 3D version [7] of MobileNetV2 was used in this dissertation. Figure 4.4(b) presents an illustration of the main building block of MobileNet V2.

4.4 Graph Convolutional Neural Network for HAR

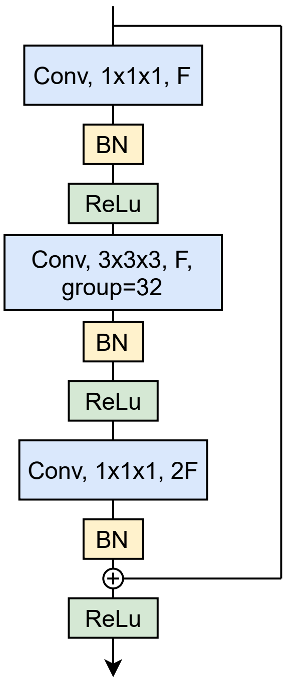
A skeleton-based approach for HAR was evaluated in this work, for such purpose a GCN was used for feature extraction and classification. Shi *et al.* [1] presented 2s-AGCN, a two stream graph convolutional network for HAR. It is a network where the graph structure modeling the human skeleton is adaptative and learnable during the training process contrary to existing methods where a fixed graph is set manually. The model is composed of two parallel streams: (1) joint stream (J-Stream) and (2) bone stream (B-Stream). J-Stream extracts features from the 3D location of each skeleton joint. B-



(a) ResNet-18 main block



(b) ResNet-50 main block



(c) ResNeXt main block

Figure 4.2: Main blocks of 3 Residual Network architectures [3]. BN: batch normalization, ReLu: rectified linear unit, conv: convolution, F: number of feature maps, group: number of groups in group convolution. Adapted from [3].

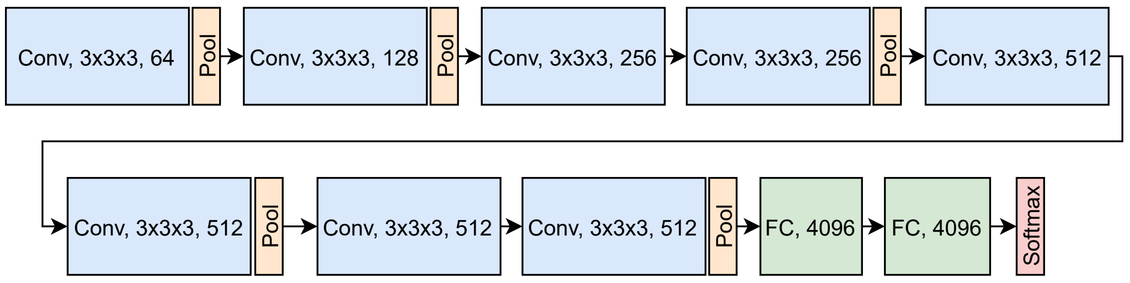
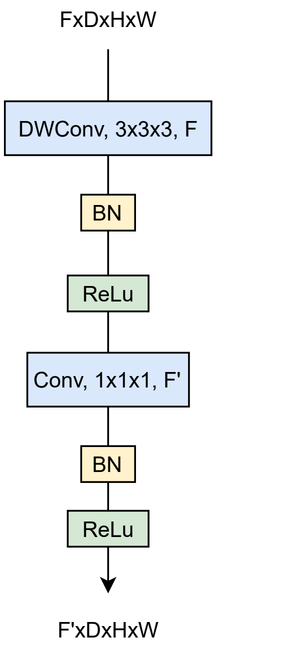
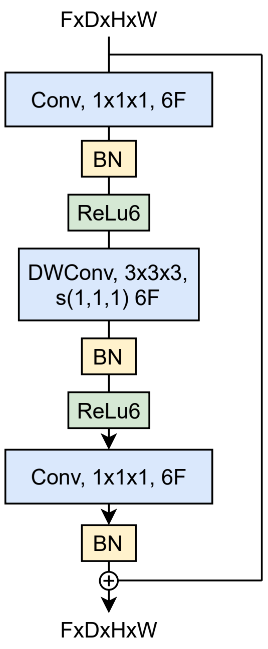


Figure 4.3: C3D [4] architecture. Conv: convolution, Pool: pooling layer, FC: fully connected layer, softmax: softmax output layer. All convolutions have a stride of 1 in both spatial and temporal dimensions. The number of filters for each convolution is the last element in each box. Each fully connected layer has 4096 output units. All pooling kernels are 2×2×2 except for the first one which is 1×2×2. Adapted from [4].



(a) MobileNetV1 main block



(b) MobileNetV2 main block

Figure 4.4: Main blocks of MobileNetV1 [5] and MobileNetV2 [6]. BN: batch normalization, ReLu6: rectified linear unit with maximum value 6, Conv: convolution, F: number of feature maps, DWConv: depthwise convolution, s(1,1,1): stride of 1 in each dimension. Adapted from [7].

Stream extracts features from bone information calculated from the 3D locations of the joints. The two streams are fused via decision level fusion. Both streams' output vector is normalized via a softmax layer and then both are added together. The resulting vector is used to predict the action depicted in the input video.

The authors use the same spatiotemporal graph presented by Yan *et a.l* [137] to represent the skeleton sequences. However, the graph convolution layer (presented in Chapter 3) is improved to an adaptative graph convolution layer where the authors introduce two learnable graphs that are optimized in conjunction with the weight vector (W_k) during the training process:

$$\mathrm{f}_{\mathrm{out}} = \sum_{k}^{K_v} W_k f_{in}(A_k + B_k + C_k) \tag{4.2}$$

the new parameters are A_k, B_k and C_k. A_k is the $N{\times}N$ normalized adjacency matrix of the skeleton graph, set manually and it is fixed. B_k is a $N{\times}N$ learnable adjacency matrix whose values are optimized during the training process. C_k is a $N{\times}N$ learnable matrix which purpose is to learn a graph for each sample (*data-dependent graph*). For such purpose, to determine if two given vertexes are connected and how strong is that connection, it is used the normalized embedded Gaussian function to determine the similarity between the two given vertexes:

$$f_{(v_i,v_j)} = \frac{e^{\theta(v_i)^T \phi(v_j)}}{\sum_{j=1}^{N} e^{\theta(v_i)^T \phi(v_j)}} \tag{4.3}$$

N is the total number of vertexes. Given an input feature map (f_{in}) with dimensions C_{in}×T×N where C_{in} denotes the number of input channels and T denotes the number of frames. f_{in} is embedded into $C_e \times T \times N$ by applying two embedding functions (θ and ϕ) that are defined as a 1×1 convolution layer. The two embedded feature maps are then reshaped into an $N \times C_e$T and a C_eT×N matrix. Both matrices are multiplied to obtain a $N \times N$ matrix, which is normalized to 0 - 1, whose element C_k^{ij} represents the similarity of vertexes v_i and v_j. Finally the C_k similarity matrix is calculated by applying the *softmax* function to the previous matrix. Thus, C_k is calculated based on the following equation:

$$C_k = softmax(f_{in}^T W_{\theta k}^T W_{\phi k} f_{in}) \tag{4.4}$$

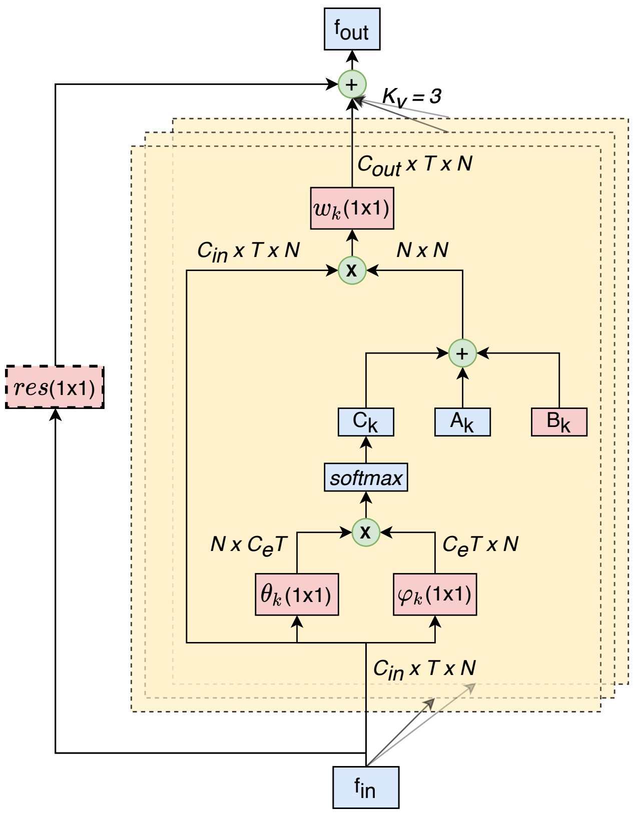
where W_θ and W_ϕ denote the parameters of the embedding function.

Figure 4.5 illustrates both the 2s-AGCN complete architecture as well as its main block that implements the adaptative graph convolution layer defined previously.

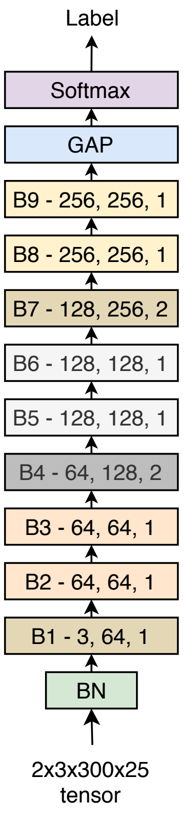
4.5 CNN-based Object Detection

One of the HAR frameworks evaluated in this dissertation requires a human detection module. For such purpose it was used the state of the art object detection system You Only Look Once (YOLO) [143]. Since its original release in 2016 the authors published two additional papers [144, 145] where improvements were made to the original architecture.

In the YOLO system the input image is segmented into an $S \times S$ cells grid. If the object's bounding box center is inside a given cell that cell is responsible for detecting that object. Thus, each cell predicts a given number of bounding boxes and associates with each box its confidence that there is an object inside that box. Moreover, each cell also has a conditional probability for each class, meaning that if a given box contains an object then that object's class is the one with the highest probability. Finally it is performed a non maximum suppression to eliminate duplicate predictions and it is applied a threshold detection. In YOLOv3 [145] the authors used a different backbone CNN, Darknet-53 which is a CNN with 53 convolutional layers. It was also used a new type of bounding



(a) 2s-AGCN main block.



(b) 2s-AGCN architecture.

Figure 4.5: 2s-AGCN main block and complete architecture. Left diagram: T denotes the temporal length, N denotes the number of vertexes, C_{in} and C_{out} denotes the number of input and output channels respectively. A_k, B_k and C_k are $N \times N$ matrices. f_{in} and f_{out} denote the input and output feature maps respectively. K_v denotes the number of subsets. $\oplus$ denotes elementwise summation and $\otimes$ denotes matrix multiplication. $res(1\times1)$ is a residual layer that is applied when the number of output channels is different than the number of input channels. $\theta_k(1\times1)$ and $\phi_k(1\times1)$ represent the embedding functions. $w_k(1\times1)$ is a 1×1 convolution layer. Red boxes represent learnable parameters. Right diagram: B1-B9 represent the 2s-AGCN main block. BN is a batch normalization layer and GAP is a global average pooling layer. Adapted from [1].

box called anchor boxes. Instead of predicting the size and location for each bounding box the network calculates offsets for predefined boxes (anchor boxes), authors state that it is easier for the network to learn offsets than predicting the coordinates and bounding box's size directly. Another improvement was multiscale prediction, instead of having a $S \times S$ cells grid the authors implemented prediction at 3 different scales: 13×13, 26×26 and 52×52 [146].

4.6 PyTorch Package

PyTorch is an open source scientific computing package. It was first published in 2016 by Facebook's Artificial Intelligence Research Lab (FAIR) and since then has become one of the most popular DL frameworks.

The main focus of the PyTorch library is the DL community, mainly due to the following two features:

- Graphics Processing Unit (GPU) tensor computation: PyTorch allows tensor operations to be performed in the GPU allowing for faster computations;
- Autograd module: it allows automatic differentiation for all operations performed on tensors. In the forward pass these operations are recorded and later the gradients can be calculated automatically and stored into the tensor's `.grad` attribute.

4.6.1 Creating and Training a Neural Network in PyTorch

Creating and training a neural network in PyTorch involves 3 main steps: creating the dataset class, defining the neural network and creating the main loop to iterate over all the training examples.

Generally the main loop is composed of a primary loop that iterates over all the epochs and a secondary loop that iterates over all the training examples:

```
 1 ...
 2 for epoch in range(0, nr_epochs):
 3         for batch, data in enumerate(dataset_loader, 0):
 4                 inputs, labels = data[0].to(device), data[1].to(device)
 5                 outputs = network(inputs)
 6                 loss = criterion(outputs, labels)
 7                 optimizer.zero_grad()
 8                 loss.backward()
 9                 optimizer.step()
10 ...
```

The first step is the forward propagation which outputs the predictions (5th line). Then follows the backward propagation step (8th line) and finally the optimization step where the NN weights are updated (9th line) based on a given optimization method defined previously. It is important to note that in the 6th line the loss is calculated based on a given criterion that was also defined previously. Also an important characteristic of the PyTorch library is that the user does not need to define any of the functions that were used in this main loop, they are all predefined. Next, the parts of the code that need to be built from scratch by the user are presented.

The PyTorch package provides several prebuilt dataset classes for the most popular datasets, however the NTU RGB+D 120 Dataset class is not contemplated in those prebuilt classes. For such purpose it is presented in the Appendix A an example of a dataset class for the NTU RGB+D 120 Dataset. PyTorch documentation states that a dataset class should override the `__len__()` and `__getitem__()` methods. The `__len__()` method returns the number of examples in a dataset and `__getitem__()` allows indexing of the dataset such as `ntu_dataset[i]` which returns the i^{th} example from the dataset. An example of the implementation of these two methods is present, as stated before, in the Appendix A. In this specific example it was implemented a simple augmentation technique (vertical flipping) that has a 50% chance of occurring when calling the `__getitem__()` method. The sample is then resized (height and width in pixels), converted to tensor variable and all the frames are concatenated into a variable (this group of frames represents the video sample). This variable is then returned to the caller function.

Finally, the third step is to define the architecture of the neural network. The PyTorch package also provides several architecture templates for the most commonly used neural networks. For the purpose of illustration it is presented in the Appendix A a template for the C3D architecture [4] with batch normalization [147]. The template of the neural network must implement a `forward()` function that when called computes the outputs given the inputs (forward propagation). For such purpose there are several predefined classes that can be used to define the NN such as `Conv3d()`, `BatchNorm3d()`, `MaxPool3d()`, `Linear()` which applies 3D convolution, batch normalization, 3D max pooling and a linear transformation, respectively, over an input signal. For the sake of brevity only a few of the available predifined classes were presented.

To keep track of the NN's performance a test function should also be implemented. For such purpose it can be created a *for* loop where each test set example is classified by the neural network and the result is compared with a ground truth file. Then the NN's performance can be calculated based on a given criteria.

4.7 Source Codes

For training and testing the DL architectures it was used the publicly available code[1] developed by Köpüklü *et al.* [7]. In their paper the authors converted popular 2D resource efficient CNNs into 3D versions with the purpose of video classification. The new versions' performance was evaluated on three public benchmarks/datasets: Kinetics-600 [55], 20BN-jester Dataset [60] and UCF101 [54].

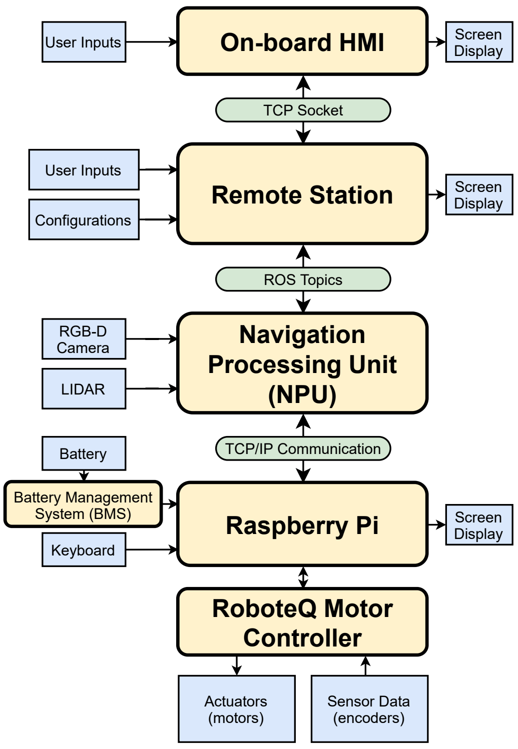
It is important to point out that this code was specifically designed to train and evaluate models on the aforementioned benchmarks. Although, for the purpose of this dissertation a different dataset (NTU RGB+D 120) was used which demanded several adaptations to make the source code compatible with the NTU RGB+D 120 Dataset, details are explained in Chapter 5.

Regarding the skeleton based approach it was also used a public repository[2] developed by Shi *et al.* [1]. Authors proposed 2s-AGCN, a two-stream GCN with decision-level fusion for video classification that was trained and tested on the NTU RGB+D 120 dataset. For this reason the available code was entirely compatible and few changes needed to be made.

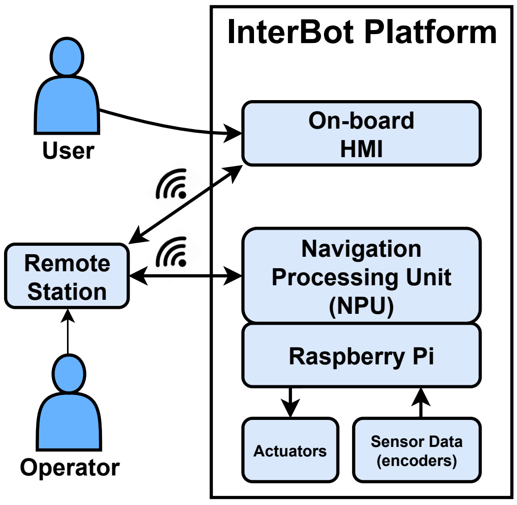
4.8 Validation Platform

The InterBot (Interactive mobile roBot) platform, Fig. 4.6, is an indoor service robot which was developed at ISR-UC [148], it uses a modular software architecture that allows collaborative HRI. The HRI system integrates two Human-Machine Interface (HMI) devices: an on-board portable device and a remote station which is in charge of planning the robot tasks. The software architecture was developed using the Robot Operating System (ROS) [149] framework.

[1]https://github.com/okankop/Efficient-3DCNNs
[2]https://github.com/lshiwjx/2s-AGCN



(a) InterBot architecture: main hardware modules and their inputs and outputs. The arrows show the flow of data (adapted from [148])



(b) Overview of the InterBot hardware architecture and flow of information (adapted from [148])

Figure 4.6: Overview of the InterBot architecture.

4.8.1 InterBot Hardware Architecture

InterBot is a differential drive mobile robot composed of two motorized wheels and a caster wheel. Each motorized wheel is actuated by a DC motor powered by 8-cell lithium batteries which is controlled by a RoboteQ Motor Controller. There is an encoder coupled to each motor axis. The speed commands are sent to the RoboteQ Controller by a Raspberry Pi which also receives information from the wheels' encoders and sends the odometry to the Navigation Processing Unit (NPU). The NPU is an onboard laptop that runs the ROS software nodes. Both a Scanning Laser Rangefinder (Hokuyo UTM-30LX Laser) and an Intel RealSense D435 camera are installed on the platform. Direct or shared control over the InterBot platform is provided via the Remote Station. The on-board HMI is a Nexus 10 tablet, acting as HRI [148]. InterBot can be operated either locally using the on-board HMI Virtual Joystick or remotely from the Remote Station using the same Virtual Joystick but using mouse/keypad events. The InterBot has a set of configuration parameters that are initialized with default values which can be safely changed while the robot is operating [148]. Figure 4.6 illustrates the InterBot main hardware modules and how they are interconnected.

4.9 Intel RealSense Depth Camera D435

Intel RealSense Depth Camera D435 is a stereo vision depth camera. It is able to record videos up to 1920x1080 RGB resolution at 30 FPS. Although not used in this dissertation, the camera is also able to record depth map videos with up to 1280x720 resolution at 30 FPS. It has a range of 0.2m up to 10m, depending on the lighting conditions. The subsystem assembly contains a stereo depth module and a vision processor (Intel RealSense Vision Processor D4). Connection with a host system is done through a USB C to USB 2.0/USB 3.1 conneciton [150].

Interface between the camera and the laptop is done through the official ROS wrapper provided publicly by Intel[3]. A laptopt running Lubuntu 18.04 LTS with ROS Melodic was used to record the images provided by the camera.

[3]ROS Wrapper for Intel RealSense Devices, available at: https://github.com/IntelRealSense/realsense-ros

Chapter 5

Developed Work

This chapter introduces the frameworks proposed in the dissertation for solving the HAR problem.

Figure 5.1 presents the HAR-RGB-based Framework I (HAR-RGB-bFI) and its offline stage. In the offline stage, every video in the dataset is converted to a set of frames which are then saved. During the training phase, each set of frames, obtained on the offline stage, is directly fed to the HAR-RGB-bFI framework. The preprocessing stage is responsible for applying data augmentation techniques and then resizing the images for matching the DL architecture's input layer dimensions. The DL architecture is responsible for extracting features and classifying (feature extraction and classification module) the activity depicted in the input set of frames.

HAR-RGB-based Framework II (HAR-RGB-bFII) is illustrated in Fig. 5.6. In this second framework it is introduced a human detection module during the offline stage. This module substitutes the video processing module, it extracts, from each frame, a bounding box containing the person's whole body and then saves the set of bounding boxes. The introduction of the human detection module aims to solve the problems in the HAR-RGB-bFI, which are discussed later in this chapter. The training phase is similar to HAR-RGB-bFI except that the input is a set of bounding boxes instead of the whole frame.

A third framework, HAR-Skeleton-based Framework (HAR-S-bF), illustrated in Fig. 5.11, was also developed and evaluated in this dissertation. This approach uses the 3D coordinates of each joint of the human skeleton for feature extraction and classification. Feature extraction and classification is performed by 2s-AGCN (presented in Section 4.4).

The subsets of classes of the NTU RGB+D 120 dataset used for training and evaluating the frameworks are listed and their purpose is also explained. Finally, the dataset collected at the Institute of Systems and Robotics - University of Coimbra, which was used for evaluation of the generalization capability of the frameworks, is also presented in this chapter.

5.1 HAR-RGB-based Frameworks

In this section both HAR-RGB-based frameworks developed in this dissertation are presented by explaining their respective offline stages, training phases and testing phases. It is divided into 4 parts:

- HAR-RGB-bFI: explains the offline stage and the preprocessing module of both training and testing phases;
- HAR-RGB-bFII: explains the offline stage and the preprocessing module of both training and testing phases;
- Feature Extraction and Classification Module: explains the feature extraction and classification module for both frameworks. This module is the same for the two frameworks, and thus it is explained in the same section;
- Optimization Step: explains the Optimization Step for both frameworks;
- Data Augmentation: explains and illustrates the data augmentation techniques used for both HAR-RGB-based frameworks.

5.1.1 HAR-RGB-based Framework I

HAR-RGB-bFI is illustrated in Fig. 5.1. It is composed of:

- An **offline stage** where the whole dataset is processed before starting the training phase;
- A **training phase** where the feature extraction and classification module (DL architecture) is trained on the dataset processed by the offline stage;
- A **test phase** where the framework performance is evaluated on the dataset processed by the offline stage.

Offline Stage

The offline stage's (top half of the diagram presented in Fig. 5.1) input is an RGB video with 1920×1080 resolution recorded at 30 FPS. Each video portrays a single subject performing a single activity and lasts for about 2 to 3 seconds, depending on the subject and activity. First, the video is resized to 320×240 resolution and then is converted to a set of frames. The resulting frames are then saved.

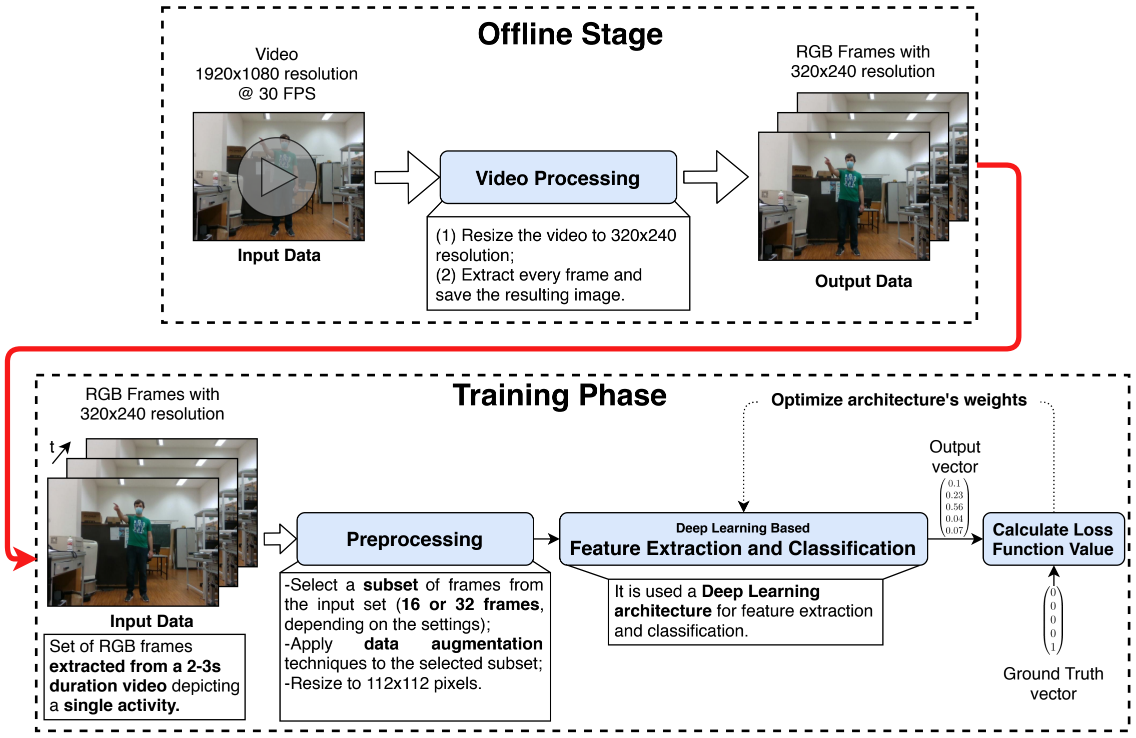

Offline Stage
Video
1920x1080 resolution
@ 30 FPS
Input Data
Video Processing
(1) Resize the video to 320x240 resolution;
(2) Extract every frame and save the resulting image.
RGB Frames with
320x240 resolution
Output Data
Training Phase
RGB Frames with
320x240 resolution
t
Input Data
Set of RGB frames extracted from a 2-3s duration video depicting a single activity.
Preprocessing
-Select a subset of frames from the input set (16 or 32 frames, depending on the settings);
-Apply data augmentation techniques to the selected subset;
-Resize to 112x112 pixels.
Optimize architecture's weights
Deep Learning Based
Feature Extraction and Classification
It is used a Deep Learning architecture for feature extraction and classification.
Output
vector
0.1
0.23
0.56
0.04
0.07
Calculate Loss
Function Value
0
0
0
0
1
Ground Truth
vector

Figure 5.1: HAR-RGB-bFI. The offline stage converts every video in the dataset to a set of frames with 320×240 resolution and then saves them. The training phase is composed of 3 main modules, a preprocessing stage a feature extraction and classification stage and a optimization stage where based on the loss function values the DL architecture's weights are optimized.

Training Phase

During the training phase (bottom half of the diagram presented in Fig. 5.1) the feature extraction and classification module is trained on the dataset, previously processed by the offline stage. One set of frames, representing a video, is fed at a time to the training phase. This set of frames represents a single video and has a single label. Throughout the chapter it is referenced a size of 112×112 pixels, this is the size of the frames that are fed to the feature extraction and classification module.

The preprocessing stage purpose is twofold: converting the input set of frames for matching the DL architecture's input layer dimensions (select 16 or 32 frames from the input set and resizing each frame to 112×112 pixels) and applying data augmentation techniques to the input set of frames. For the purpose of the dissertation, one of the main focus of the developed work was testing and evaluating different cropping methods (spatial augmentation technique, discussed later in this chapter). The cropping method can be interpreted as a data augmentation technique (multiscale random cropping) or, under certain conditions, as a method that extracts the important region of the image for the classification task (center cropping). In the case of the HAR problem, for the center cropping to effectively extract the important region from the image, it must be assumed that the person is centered in the image throughout the whole video (discussed later in this chapter).

The initial approach was to use a multiscale random crop in the input set of frames, depicted in Fig. 5.2. In this cropping technique, a random sized square region is selected and extracted from each frame and then is resized to a 112×112 pixels square. This was a technique implemented, as default, in the source code [7] used for this work. The purpose of such technique is to serve as spatial data augmentation technique. For each epoch, it crops different regions of the set of frames, and thus creating more training examples for the training process. Every frame in a set of frames is cropped at the same position. However, this cropping technique proved to be unsuitable for the dataset used in this dissertation. As illustrated in Fig. 5.2 the cropped regions can be very different from one another. There could be regions extracted containing the subject's whole body, containing part of the subject's body or even containing only background objects. It is clear that this cropping technique is not very suitable for this type of dataset. The datasets (Kinetics-600 dataset [39], UCF-101 dataset [54] and Jester dataset [60]) used by the authors of the source code used in this dissertation are very different from the NTU RGB+D 120 dataset. Jester is a gesture recognition dataset where the subjects are sitting in front of their laptop webcam doing hand gestures. Kinetics-600 and UCF-101 are activity recognition datasets that are a collection of videos downloaded from YouTube, and therefore were recorded with different points of view, resolutions or come from varying sources (personal videos, commercials, TV news or documentaries are all examples of such sources).

A second approach was to resize each frame to a 112×112 pixels square and then applying data augmentation, except for any sort of cropping technique. Essentially, the whole frame is used for feature extraction and classification. This method also proved to be unsuitable for the dataset used in this dissertation. Note that the majority of the visual information in each frame is irrelevant for the activity being performed. Fig. 5.3 illustrates this problem. The DL architectures used in this dissertation for the feature extraction and classification module do not include any sort of attention mechanism that is capable of focusing on the region of interest. An example of such mechanism is the work by Baradel *et al.* where the authors developed a framework that is capable of concentrating on interest points that are relevant to the activities/actions. The authors managed to achieve 86.6% cross-subject

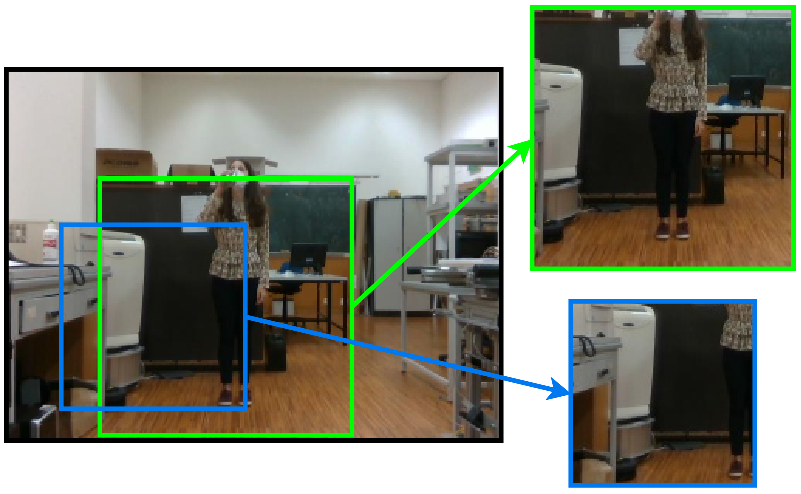


Figure 5.2: Original 320x240 frame and the resulting images after applying two multiscale random crops.

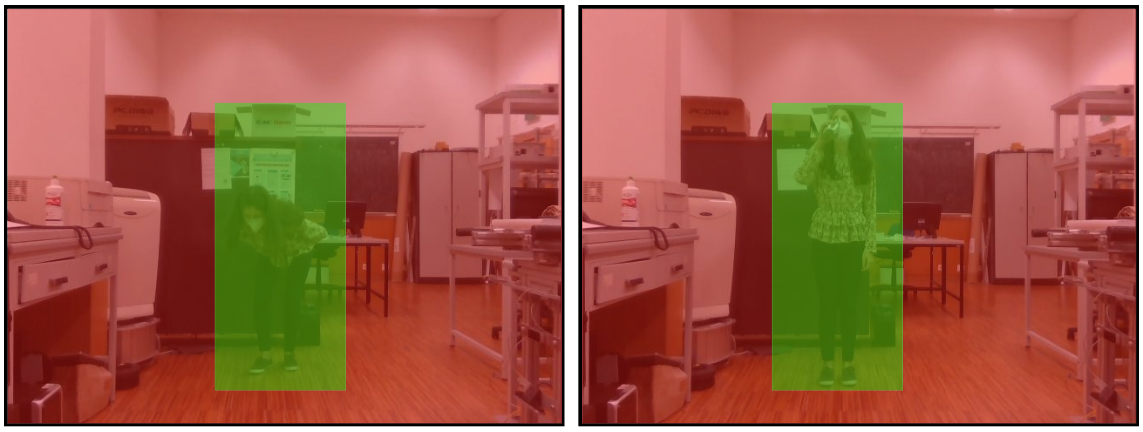


Figure 5.3: Comparisson between two frames depicting different activities, drinking water and vomiting. It shows a red region, which is common to both pictures, that does not contribute with any relevant information to distinguish between the two activities. The green region contains the area where the activity is being performed.

classification accuracy on the NTU RGB+D dataset.

Having into account that the region of interest is located mainly at the center of each frame, a possible strategy, to help the DL architecture concentrate on that region, would be to extract this area and use it as input to the DL architecture. As a first experiment, it was used a 112×112 pixels square region extracted from the frame's center. However, in some cases a 112×112 square is not large enough to include the relevant body parts for some activities (e.g. situations b1) and d1) in Fig. 5.4). Also, in extreme cases, the person's body could be completely left out on the resulting image (e.g. situation c1) in Fig. 5.4).

To try to mitigate the problem of missing body parts in the center crop, one last approach

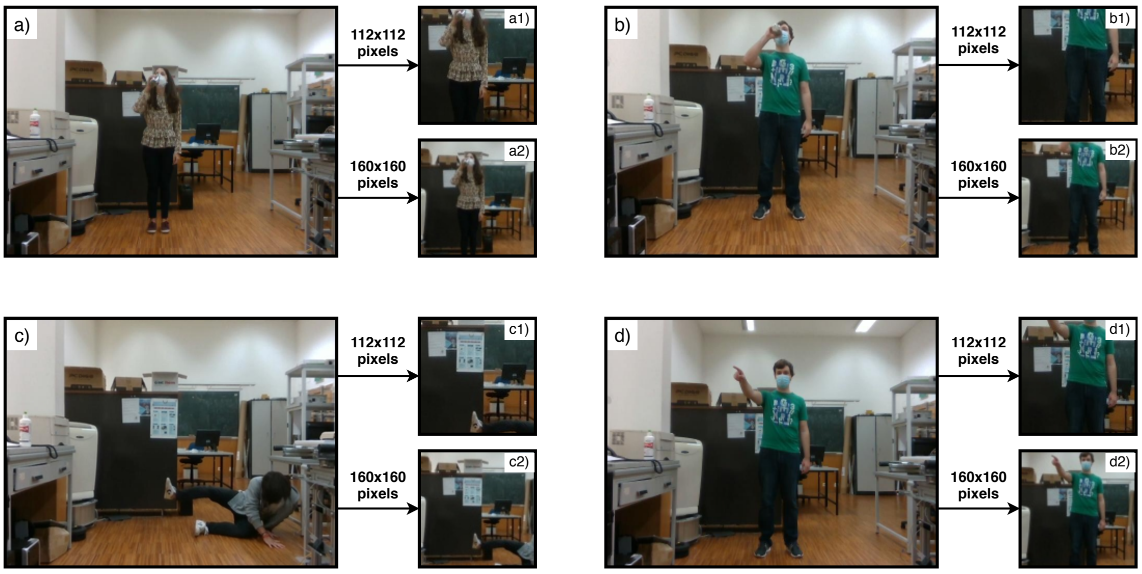


Figure 5.4: Original 320×240 frames and the resulting images after applying two different central crops (112×112 pixels and 160×160 pixels).

was proposed. Extracting first a larger square (160×160 pixels) from the image center, and then resized to 112×112 pixels. This second approach solves the problem for some of the situations (e.g. situation a2) and d2) in Fig. 5.4). However, it is clear that in other cases even with a larger square the person's body is still out of frame in the resulting image (e.g. situation c2) in Fig. 5.4) or a relevant body part is missing (e.g. situation b2) in Fig. 5.4).

Increasing the center crop size would eventually include the person's whole body for every image. However, as the crop size increases, more irrelevant visual information is included resembling the situation where the whole frame is used as input. This problem is can be seen in all of the 160×160 images present in Fig. 5.4, where great part of the image contains irrelevant information.

Overall, HAR-RGB-bFI main problem is the fact that it is assumed the person performing the activity is centered in the image. If the person deviates from the image center this assumption can become a problem since the crop position does not take into account this variations. The person can move out of the image center not only because it moves during the activity but also because the robot can move around during the activity. Thus, the resulting image, after cropping the original, can be a background scene without any person or the person's left or right body side could be left out. Moreover, even if the person remains relatively centered in the image throughout the activity, another difficulty that arises is the fact that the image area occupied by the person varies with the distance

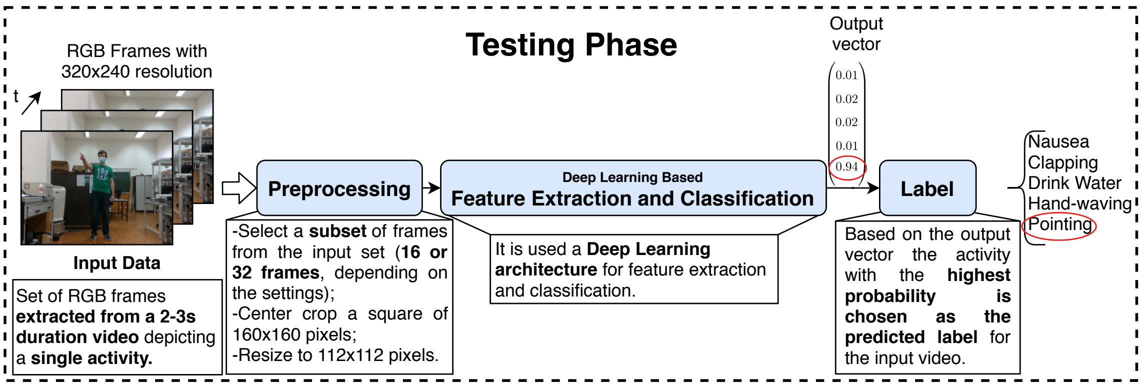

Testing Phase
RGB Frames with 320x240 resolution
t
Input Data
Set of RGB frames extracted from a 2-3s duration video depicting a single activity.
Preprocessing
-Select a subset of frames from the input set (16 or 32 frames, depending on the settings);
-Center crop a square of 160x160 pixels;
-Resize to 112x112 pixels.
Deep Learning Based
Feature Extraction and Classification
It is used a Deep Learning architecture for feature extraction and classification.
Output vector
0.01
0.02
0.02
0.01
0.94
Label
Based on the output vector the activity with the highest probability is chosen as the predicted label for the input video.
Nausea
Clapping
Drink Water
Hand-waving
Pointing

Figure 5.5: HAR-RGB-bFI testing phase.

to the camera. Different crop sizes would be required for different distances between the subject and the camera, i.e. a person closer to the camera occupies more area in the image than a person who is located farther from the camera.

Testing Phase

For the HAR-RGB-bFI testing phase the preprocessing stage is responsible for extracting N frames (16 or 32 depending on the settings) from the original video and applying a central crop of 160×160 pixels square which is then resized to 112×112 pixels. No spatial data augmentation techniques are applied during testing. Finally, the frames are stacked together forming a 3×N×112×112 tensor that is fed to the DL architecture for feature extraction and classification. Figure 5.5 shows a diagram illustrating HAR-RGB-bFI testing phase.

5.1.2 HAR-RGB-based Framework II

Trying to improve the classification accuracy achieved with the first framework, it is proposed in this dissertation a second approach in which a human detection stage substitutes the video processing module in the offline stage. HAR-RGB-bFII is illustrated in Fig. 5.6. It is composed of:

- An **offline stage** where the whole dataset is processed before starting the training phase;
- A **training phase** where the feature extraction and classification module (DL architecture) is trained on the dataset processed by the offline stage;
- A **test phase** where the framework performance is evaluated on the dataset processed by the offline stage.

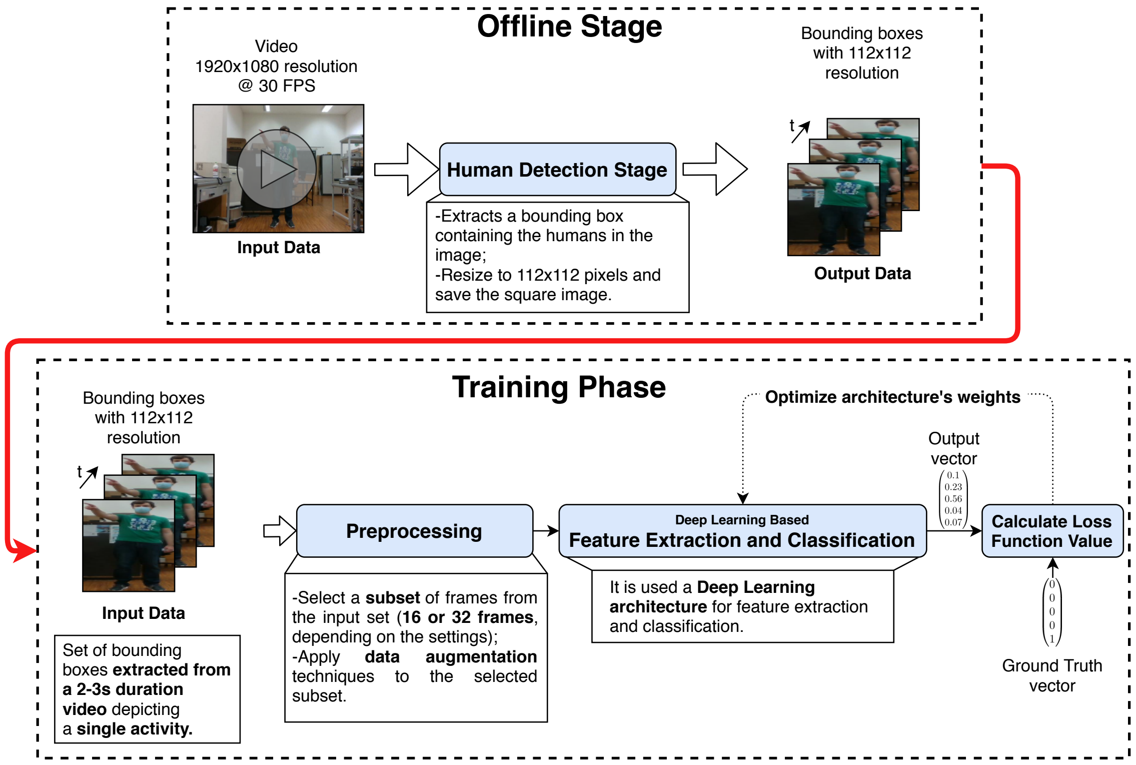

Offline Stage
Video
1920x1080 resolution
@ 30 FPS
Input Data
Human Detection Stage
-Extracts a bounding box containing the humans in the image;
-Resize to 112x112 pixels and save the square image.
Bounding boxes with 112x112 resolution
t
Output Data
Training Phase
Bounding boxes with 112x112 resolution
t
Input Data
Set of bounding boxes extracted from a 2-3s duration video depicting a single activity.
Preprocessing
-Select a subset of frames from the input set (16 or 32 frames, depending on the settings);
-Apply data augmentation techniques to the selected subset.
Optimize architecture's weights
Deep Learning Based
Feature Extraction and Classification
It is used a Deep Learning architecture for feature extraction and classification.
Output vector
0.1
0.23
0.56
0.04
0.07
Calculate Loss Function Value
0
0
0
0
1
Ground Truth vector

Figure 5.6: HAR-RGB-bFII. The offline stage extracts from every video in the dataset a bounding box containing the person's whole body. The resulting bounding boxes are resized to 112×112 pixels and then saved.

Offline Stage

The offline's stage (top half of Fig. 5.6) input is the same as the offline stage of HAR-RGB-bFI. The main difference is the human detection module that substitutes the video processing module. The human detection module extracts, from each frame, a bounding box containing the person's whole body. Independently of the subject's location this bounding box contains its whole body (given it is present in the image), contrary to HAR-RGB-bFI. Naturally, the bounding box dimensions depend on both the distance between the person and the camera and the person's pose (for example, if the person is standing with arms open the bounding box will be wider than a person standing with arms crossed). Thus, after being extracted, the bounding box is resized to a 112×112 pixels square and saved. Therefore, each video in the dataset is converted to a set of bounding boxes containing the person performing the activity (as referenced earlier, each video contains a single person).

Figure 5.7 illustrates the resulting bounding boxes after extraction and resizing for the 4 situations previously presented in Fig. 5.4. It is clear that with the human detection

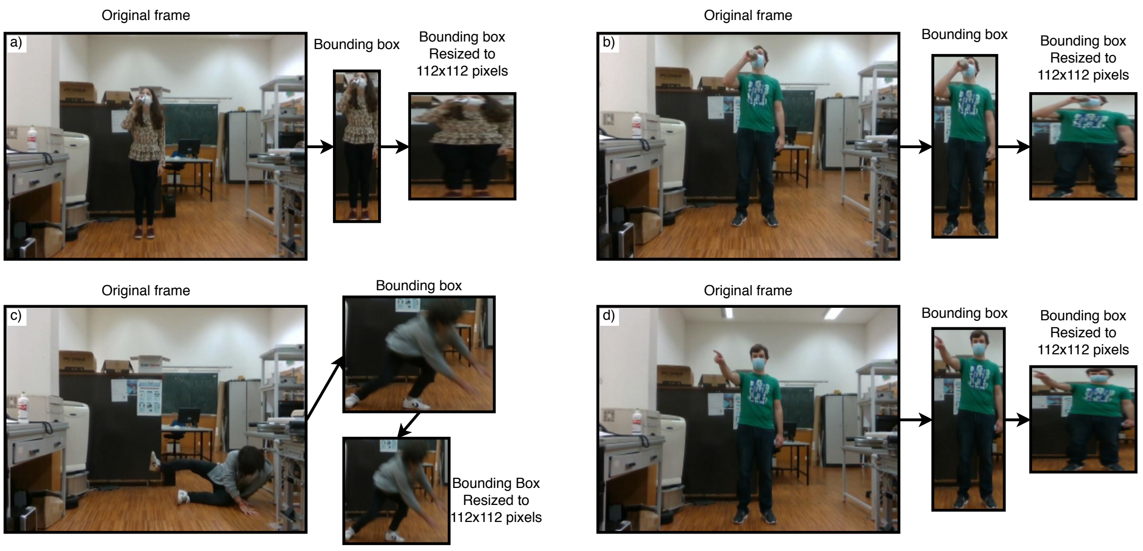

Original frame
a)
Bounding box
Bounding box Resized to 112x112 pixels
Original frame
b)
Bounding box
Bounding box Resized to 112x112 pixels
Original frame
c)
Bounding box
Bounding Box Resized to 112x112 pixels
Original frame
d)
Bounding box
Bounding box Resized to 112x112 pixels

Figure 5.7: Original 320×240 frame and the resulting images after extracting and resizing bounding boxes containing the person's whole body.

module the framework is able to keep track of the person performing the activity, contrary to HAR-RGB-bFI. For the human detection stage it was used YOLOv3-spp [145] (third version of the state of the art object detection system YOLO [143]) with Spatial Pyramid Pooling blocks [151]. A pre-trained model on Common Objects in Context Dataset (COCO Dataset) [152] published by Ultralytics LLC[1] was used. This dataset contains 80 object categories, however for the context of this dissertation only the category 'person' is of interest. Thus, it was necessary to adapt the original code for extracting only the bounding boxes for the category 'person' in each frame. Figure 5.8 illustrates the differences between the original and new outputs of YOLOv3-spp.

Training and Testing Phases

Training and testing phases are very similar to HAR-RGB-bFI. The main difference is the application of cropping techniques, on the HAR-RGB-bFII the offline stage resizes the bounding boxes to 112×112 and then saves them. Thus, the preprocessing stage does not apply any type of cropping technique on the input set of frames. The feature extraction and classifcation module is the same as the HAR-RGB-bFI. Figure 5.9 illustrates the testing phase of HAR-RGB-bFII.

[1]https://github.com/ultralytics/yolov3

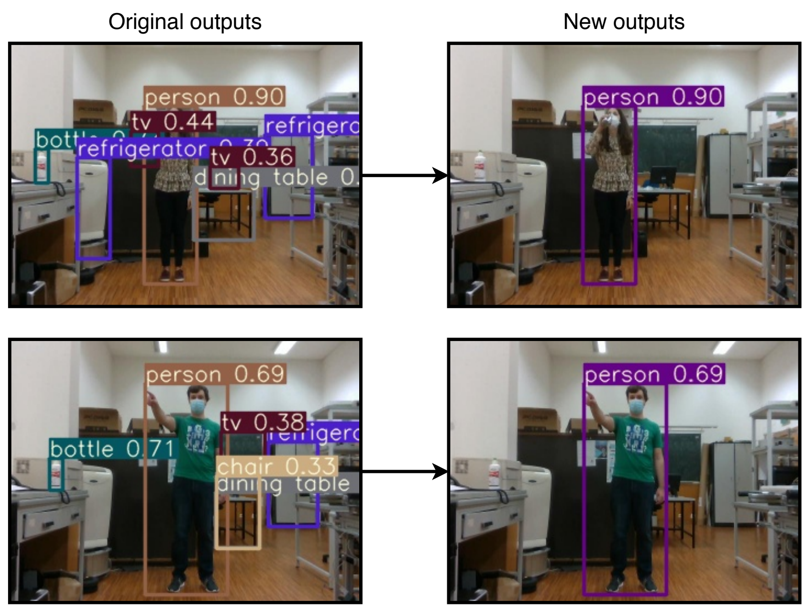


Figure 5.8: Left: YOLOv3-spp original outputs, all detected objects are bounded in the output image. Right: YOLOv3-spp output after code adaptations to the original code for only outputing the bounding boxes for the category 'person' in each frame. All objects are still detected, however the output image only extracts the bounding boxes for the category 'person'.

5.1.3 Feature Extraction and Classification

For feature extraction and classification, for both HAR-RGB-based frameworks, six different DL architectures were tested and evaluated. In Chapter 4 the main building block of each architecture was briefly described and their complete architectures is presented in appendix D, except for C3D which was already presented in Chapter 4. It is important to note that five of the six architectures were originally designed for image recognition. For this work, their versions [3, 7] for video recognition were trained and evaluated.

The input layer for all six architectures has dimensions C×D×H×W where: (1) C

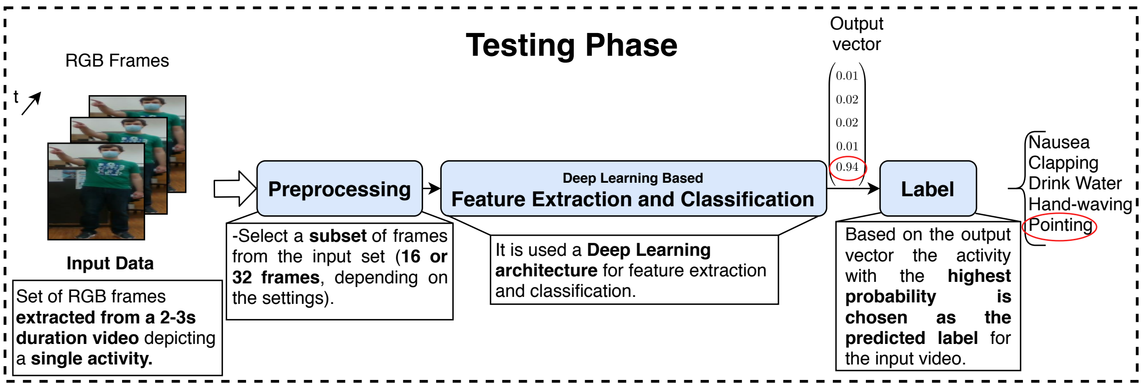

Testing Phase
RGB Frames
t
Input Data
Set of RGB frames extracted from a 2-3s duration video depicting a single activity.
Preprocessing
-Select a subset of frames from the input set (16 or 32 frames, depending on the settings).
Deep Learning Based
Feature Extraction and Classification
It is used a Deep Learning architecture for feature extraction and classification.
Output vector
0.01
0.02
0.02
0.01
0.94
Label
Based on the output vector the activity with the highest probability is chosen as the predicted label for the input video.
Nausea
Clapping
Drink Water
Hand-waving
Pointing

Figure 5.9: HAR-RGB-bFII testing phase.

denotes the number of channels in the input data (since the input data is a set of RGB images this means that $C = 3$, one for each color); (2) D denotes the depth of the input data which represents the number of input images (in this work, depending on the settings, the number of images is either 16 or 32); (3) H denotes the height, measured in pixels, of the input data (in this work $H = 112$); (4) W denotes the width, measured in pixels, of the input data (in this work $W = 112$).

The DL architectures output is a 1 dimensional vector with length equal to the number of classes. For all architectures it is applied the softmax function to the output vector. In the resulting vector each element represents the probability score for each class and all elements add up to 1. The class with the highest probability score is chosen as the label for the input set of frames. Individual performance of each DL architecture is presented in Chapter 6.

5.1.4 Data Augmentation

This section explains both spatial and temporal data augmentation techniques used during the training phase of both HAR-RGB-based frameworks.

Spatial Domain Data Augmentation

Spatial domain data augmentation is commonly used in image classification tasks. However, it can also be used in video classification problems by applying the augmentation techniques to each frame individually. In this work several spatial domain augmentation techniques were used (Fig. 5.10 illustrates the resulting images after applying each data augmentation technique):

– Flipping an image horizontally: each set of frames has a 50% chance of being flipped horizontally;

– Multiplying pixel values by a constant: each image is multiplied by a constant with a value in the interval [0.80, 1.20]. This transformation changes the image's pixels values by a percentage that varies from -20% to 20% (darken or brighten the image). Every frame in the same video sample is multiplied by the same constant, i.e. all frames change brightness equally;

– Applying salt-and-pepper noise: each video sample has a 10% probability of being affected by this transformation. Each pixel's color value has a chance in the interval $[\frac{1}{120}, \frac{1}{30}]$ of being reduced to 0 or increased to 255;

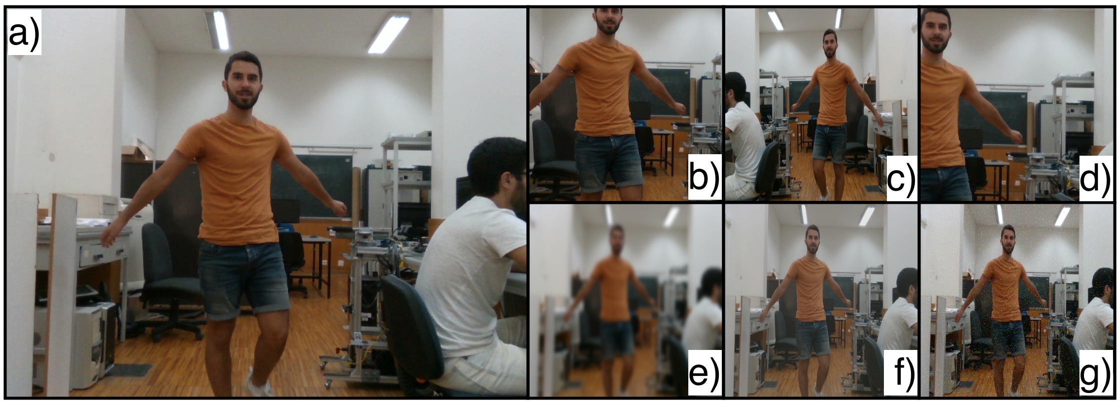


Figure 5.10: Various spatial domain data augmentation techniques and cropping techniques used in this work. From left to right and up to bottom: a) original image, b) center crop, c) horizontal flip, d) multiscale random crop, e) gaussian blur, f) multiply pixel values and g) salt-and-pepper noise.

- Applying a gaussian filter: each video sample has a 20% probability of being affected by this transformation. Convolution with two 1D gaussian kernels, one for each axis, with a dimension of 41×1 and $\sigma = 5$ is applied to the image;
- Multiscale random crop: in this technique the frame is cropped at a random location and the resulting image is used as input for the feature extraction and classification module. In a given set of input frames the crop location and size is the same for all frames. The resulting image is then resized to 112×112 pixels.

It is worth mentioning two other spatial transformations that were tested in this dissertation: (1) Center crop: each frame is cropped at the center. The cropping dimensions, depending on the settings, are a square of 112×112 or 160×160 pixels. In the case of the 160×160 pixels square the resulting image is resized to 112×112 pixels. This transformation is always applied during the testing phase of both HAR-RGB frameworks and (2) Resizing: no cropping is applied, the video sample is only resized to 112×112 pixels. Both transformations are applied to the set of input frames instead of the multiscale random crop. Note that these transformation are not applied with the objective of being used as data augmentation, their purpose was described previously in this chapter.

Time Domain Data Augmentation

Subsampling frames at different frequencies is a type of data augmentation technique that is used in video classification tasks. In this work two variations of subsampling were tested: subsampling frequency and randomizing the starting frame. Both can be used individually or concurrently, for the purpose of this dissertation several combinations of the aforementioned variations were tested (results are shown in Chapter 6).

Subsampling frequency is applied by changing the step between each consecutive frame. For example, given a set of frames $F = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and considering the number of frames to be extracted is 4, two possible subsets are:

- $F_1 = \{1, 3, 5, 7\}$, a subsampling frequency of 2 was used meaning that one frame is selected from the original set every 2 frames;
- $F_2 = \{1, 4, 7, 10\}$, a subsampling frequency of 3 was used meaning that one frame is selected from the original set every 3 frames.

Randomizing the starting frame is applied by picking a random starting frame from the input set of frames. For example, given a set of frames $F = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and the number of frames to be selected is 4, two possible subsets are:

- $F_1 = \{3, 4, 5, 6\}$, the first frame was the 3rd;
- $F_2 = \{6, 7, 8, 9\}$, the first frame was the 6th.

It is important to note that if there are not enough frames to satisfy the desired amount, either because the total number of frames is less than 16 or 32 (depending on the setting) or because the initial frame (selected randomly) was closer to the end of the video, the algorithm loops back to the first frame and continues from there. For example, given a series of frames $F = \{1, 2, 3, 4, 5, 6, 7\}$ and the number of frames to be selected is 5 frames. By applying a sampling frequency of 2 and starting at frame number 2 the selected frames would be:

- $F_1 = \{2, 4, 6, 1, 3\}$.

5.2 HAR-Skeleton-based Framework

HAR-S-bF is illustrated in Fig. 5.11. For this framework few adaptations needed to be made to the publicly available source code[2] by [1]. The framework's input is a sequence of human poses. Each human pose is modeled by a graph where each vertex represent the human joints, a total of 25 joints are used. Each vertex contains a feature vector with 3 elements that correspond to the 3D location of the corresponding joint.

[2]https://github.com/lshiwjx/2s-AGCN

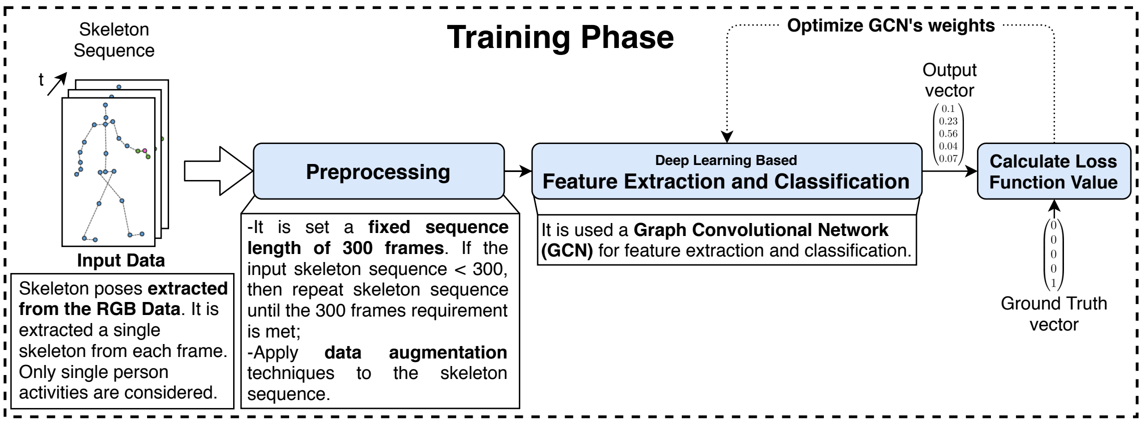

Training Phase
Skeleton Sequence
t
Input Data
Skeleton poses **extracted from the RGB Data**. It is extracted a single skeleton from each frame. Only single person activities are considered.
Preprocessing
-It is set a **fixed sequence length of 300 frames**. If the input skeleton sequence < 300, then repeat skeleton sequence until the 300 frames requirement is met;
-Apply **data augmentation** techniques to the skeleton sequence.
Deep Learning Based
Feature Extraction and Classification
It is used a **Graph Convolutional Network (GCN)** for feature extraction and classification.
Optimize GCN's weights
Output vector
0.1 0.23 0.56 0.04 0.07
Calculate Loss Function Value
0 0 0 0 1
Ground Truth vector

Figure 5.11: HAR-S-bF approach. The subject skeleton is extracted and fed to a GCN.

5.2.1 Preprocessing Stage

Preprocessing stage is responsible for generating the bone and joint data from the input skeleton sequence.

It is assumed that each skeleton sequence has 300 poses (in practical terms, it is assumed that the video samples have a maximum of 300 frames), for skeleton sequences with less than 300 poses null poses are padded until the sequence has a length of 300. The poses are then concatenated into a tensor with dimensions M×C×T×V where:

- M - number of skeletons present in each frame. It is assumed that a maximum of two skeletons are present in each frame. However for the purpose of this dissertation only activities with one subject were evaluated, and thus the second skeleton is composed of null joints;
- C - number of channels. Since each joint is represented as a point in the 3D space the number of channels is equal to 3, one for each axis (X, Y, Z);
- T - number of frames. It is assumed a fixed number of frames equal to 300;
- V - number of joints. Each pose is represented by a set of 25 joints;
- The resulting tensor is of dimensions 2×3×300×25.

To create the bone data it is assumed that a bone connects two joints. The joint closer to the center of gravity of the skeleton is the source joint ($v_1 = (x_1, y_1, z_1)$) and the joint farther away is the target joint ($v_2 = (x_2, y_2, z_2)$). The vertex corresponding to the source joint stores the vector information that points to the target joint from the source joint ($v_2 - v_1$). Because the number of joints is one more than the number of bones it is added a null bone to the central joint (joint number 1, bottom of the spine). Thus the bone data tensor also has the same dimensions as the joint data tensor.

5.2.2 Feature Extraction and Classification

As aforementioned the feature extraction and classification step is performed by a two stream graph convolutional network with decision level fusion. Bone data is fed to the B-stream (bone stream) and joint data is fed to the J-stream (joint stream), the last layer of each stream is normalized by applying a softmax function. The resulting output vectors of each stream are then added and the label is the class with the highest score.

5.3 Dataset Subsets

As mentioned in Chapter 4 the dataset used in this dissertation for training the frameworks was the NTU RGB+D 120 [2]. Two **Subsets** of classes were chosen for training and evaluating the framework: (1) health related classes and (2) interaction + health related classes. **Subset 1** contains health related classes that are useful in applications such as service robots in healthcare [22, 153, 154] or social assistive robots for elderly people [155–157]. **Subset 2** contains both interaction and health related classes, this subset was created with the objective of evaluating the frameworks's performance when trained in a less challenging subset of classes. Also, interaction classes (pointing, clapping and hand-waving) can be useful for detecting when the user intends to interact with the robot.

The following list shows the classes contained in each subset:

- **Subset 1**: drink water, headache, chest pain, sneeze/cough, nausea/vomiting, falling down and staggering;
- **Subset 2**: drink water, nausea/vomiting, pointing, hand-waving and clapping.

5.4 ISR dataset

To evaluate the generalization capability of the implemented frameworks testing was performed on a dataset collected at the Human-Centered Mobile Robotics Laboratory in the Institute of Systems and Robotics - University of Coimbra. Results are presented in Chapter 6.

The dataset was collected with an Intel Realsense D435 Depth Camera. Video was recorded at 30 FPS with 640x480 resolution. A total of 300 RGB videos were recorded with 10 participants performing 10 activities. The subjects were positioned directly in front of the camera at a distance of 3 meters. The camera was positioned on the InterBot

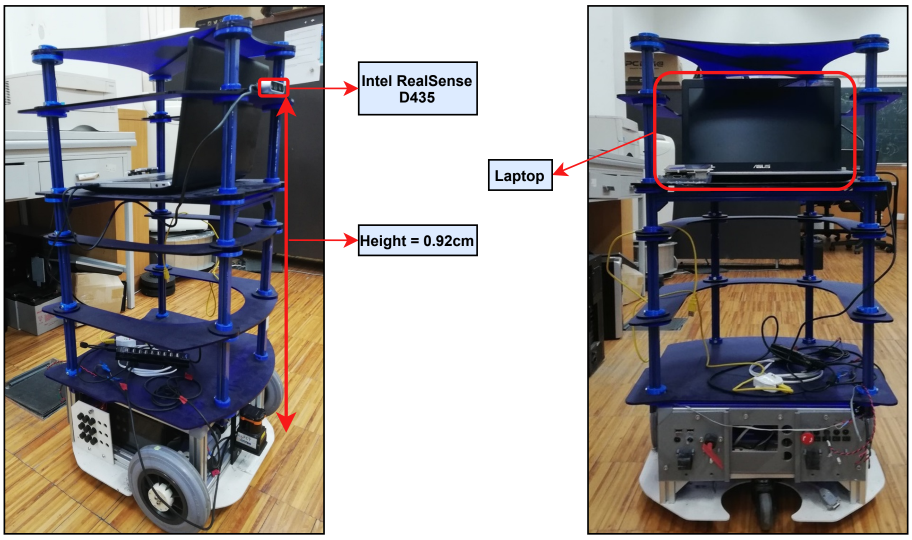


Figure 5.12: InterBot mobile platform equipped with a laptopt and the Intel RealSense D435 Depth Camera. Left: side view of the InterBot mobile platofrm, Right: rear-view of the InterBot mobile platform.

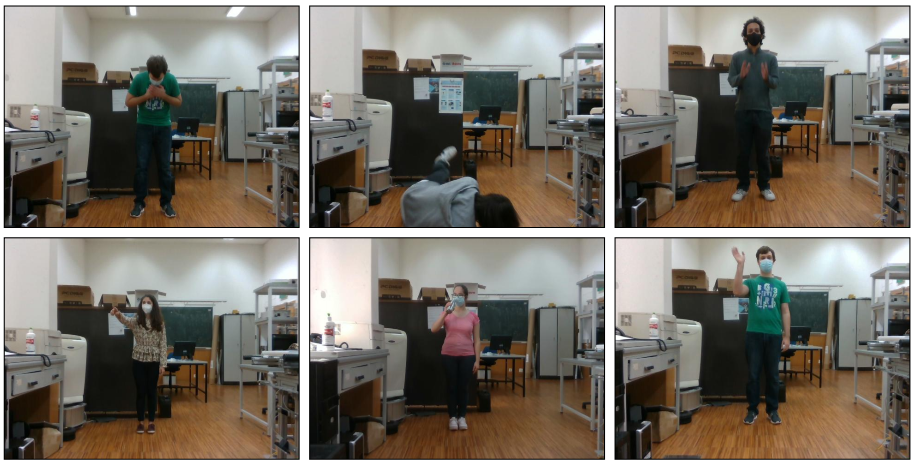


Figure 5.13: Samples taken from the ISR Dataset.

mobile platform at a height of 0.92 meters. Note that the dataset was collected with the robot's Point of View (POV), generally HAR datasets do not provide such POV. For example: Kinetics-600 dataset is a collection of videos downloaded from YouTube with varying sources and POVs and UCF-101 dataset is also composed of videos downloaded from YouTube.

Participants performed each activity 3 times, one facing the camera, one turned to their right 45 degrees and another turned to their left 45 degrees. The 10 activities performed correspond to the 10 different activities presented in section 5.3. Fig. 5.12 shows the InterBot equipped with a laptopt and the Intel RealSense D435 Depth Camera ready for collecting the ISR-dataset. Some samples of the ISR Dataset are shown in Fig. 5.13

Chapter 6

Experimental Results

This chapter presents the experimental results obtained for different training and testing scenarios in terms of classification accuracy on the training and testing sets. It is also calculated the time it takes for each framework to evaluate an input sequence of 16 frames. The chapter is divided into the following sections: (1) Evaluation Protocol, (2) Training Parameters, (3) HAR-RGB-bFI Results, (4) HAR-RGB-bFII Results, (5) HAR-S-bF Results and (6) Frameworks Runtimes.

6.1 Evaluation Protocol

As mentioned in Chapter 5 two **Subsets** of classes were selected from the NTU RGB+D 120 Dataset to train and test the frameworks:

- **Subset 1** is composed of the following 7 classes: (1) Drink Water, (2) Headache, (3) Chest Pain, (4) Sneeze/Cough, (5) Nausea/Vomiting, (6) Falling Down and (7) Staggering;
- **Subset 2** is composed of the following 5 classes: (1) Drink Water, (2) Nausea/Vomiting, (3) Pointing, (4) Hand-waving and (5) Clapping.

On the NTU RGB+D 120 Dataset each class has 948 videos while on the ISR Dataset each class has 30 videos. The classification accuracy was calculated based on cross-subject evaluation, meaning that a part of the subjects is used for training the models and the remaining subjects are used for testing. For the NTU RGB+D 120 Dataset it was used the recommended splits in [2], which means that for **Subset 1** and **Subset 2**, around 70% of the videos are used for training and the remaining 30% are used for testing. For the ISR Dataset, 7 subjects are used for training the models (21 videos per class) and 3 subjects are used for testing the models (9 videos per class).

Table 6.1: Initial learning rate and batch size used for training the DL architectures.

Architecture	Learning Rate	Batch Size
C3D	0.0001	4
MobileNet V1	0.25	32
MobileNet V2	0.25	32
ResNet-18	0.1	32
ResNet-50	0.1	16
ResNeXt	0.1	16

6.2 Training Parameters

This section presents the training parameters used in the training phase for the developed frameworks.

6.2.1 RGB-based Frameworks Training Parameters

To train all six architectures, it was used SGD with a momentum of 0.9 as the optimization algorithm. Categorical cross entropy loss was used as the loss function and the weight decay was set to 0.001. Training was conducted in a machine equipped with 16GB of RAM, a NVIDIA GeForce GTX 1060 6GB and an Intel Core i5-3570K @ 3.40GHz.

The aforementioned hyperparameters and training details are common to all architectures during their training phases. However, both the learning rate and batch size had to be adjusted for each architecture. Table 6.1 summarizes these two parameters for each architecture. It is important to note that learning rate is adjusted during the training process, as explained in Chapter 3. Thus, depending on the architecture the learning rate is divided by a factor of 10 once the loss function value starts converging. The training phase ends when the loss function value converges and further lowering the learning rate does not improve the loss function value.

Training a Pre-trained Model

Another possibility when training a model is to use a pre-trained model as feature extractor and optimize the last layers on the desired dataset (explained in Chapter 3). For HAR, Köpüklü *et al.* [7] published pre-trained models of the MobileNet V1, MobileNet V2, ResNet-18 and ResNet-50 networks [1]. These models were pre-trained on the Kinetics-600 Dataset [39].

[1]https://github.com/okankop/Efficient-3DCNNs

Table 6.2: Initial learning rate used when training the pre-trained models.

Architecture	Learning Rate
MobileNet V1	0.025
MobileNet V2	0.025
ResNet-18	0.01
ResNet-50	0.01

Training parameters are similar to the parameters when training the models from scratch except for the learning rate and the total number of epochs each architecture needs before converging to a solution. Table 6.2 shows the initial learning rates for each architecture. Similar to training from scratch, the training phase ends when the loss function value converges and further lowering the learning rate does not improve the loss function value.

6.2.2 Skeleton-based Framework Training Parameters

For training the 2s-AGCN, it was used SGD with a Nesterov Momentum [158] of 0.9 as the optimization algorithm. Cross-entropy was used as the loss function and the weight decay was set to 0.0001. Due to hardware limitations the batch size was set to 8. Considering the learning rate, it was followed the same approach as the authors of the 2s-AGCN: initial learning rate is set to 0.1 and is divided by a factor of 10 at the epochs 30 and 40 with the training process ending at epoch 50. Following this approach proved to be effective and appropriate for both the **Subsets** used in this dissertation.

6.3 RGB-based Framework I

Six different DL architectures were trained and tested for the feature extraction and classification module of the HAR-RGB-bFI (presented in Section 5.1.1) and the following evaluation scenarios were considered:

1. **MobileNetV1**:
   (a) Different combinations of the ***width multiplier***[2] **value**, **number of extracted frames** and **cropping technique**;
   (b) Different values for the **subsampling frequency** value while using a fixed value

[2]Both MobileNetV1 and MobileNetV2 have a *width multiplier* parameter that allows changing the number of output channels in each convolutional layer by multiplying it by the default number of output channels in each convolutional layer.

for the *width multiplier*, number of extracted frames and cropping technique;

(c) Not applying any cropping technique. The **input set of frames is just resized** to match the DL architecture's input layer dimensions. The remaining parameters (*width multiplier*, subsampling frequency and cropping technique) are fixed.

2. **MobileNetV2**: width multiplier of 0.5, 16 extracted frames, subsampling frequency of 2 and center crop;
3. **ResNet-18, ResNet-50, ResNeXt and C3D**: 16 extracted frames, subsampling frequency of 2 and center crop.

The experiments performed on HAR-RGB-bFI were conducted on **Subset 1**. Due to hardware limitations it was not possible to test all the DL architectures on all scenarios. For such purpose MobileNetV1 was selected as a *testbed* with the goal of testing different combinations of : (1) *width multiplier* value, (2) number of extracted frames from the input set, (3) the type of cropping technique and (4) subsampling frequency. For all the experiments the starting frame is selected randomly from the input set of frames, as explained in Section 5.1.4.

The results for tests 1a), 1b) and 1c) are presented in Table 6.3 and Table 6.4.

Table 6.3: Results obtained for MobileNetV1, on the HAR-RGB-bFI, when trained from scratch on **Subset 1** with different combinations of: *width multiplier*, number of extracted frames and cropping technique parameters.

Width Multiplier	**Nr. of extracted Frames**	**Cropping Technique**	**Train Accuracy**	**Test Accuracy**
0.2	16	center	72.2%	51.6%
0.2	16	random	56.1%	52.8%
0.2	32	center	86.8%	42%
0.2	32	random	70%	46.9%
0.5	16	center	94.3%	53.3%
0.5	16	random	55.4%	48.7%
0.5	32	center	95.8%	45.7%
0.5	32	random	80%	49.5%
0.7	16	center	93.9%	52.1%
0.7	16	random	55%	47.4%
0.7	32	center	94%	35.9%
0.7	32	random	81.2%	32.7%

Using the model highlighted with a darker color (highest test classification accuracy) in Table 6.3 as a reference for comparison, the following observations can be made:

- ***Width multiplier***: increasing or decreasing the *width multiplier* value leads to

Table 6.4: Results obtained for the 1b) and 1c) scenarios. The experiments were conducted with a *width multiplier* value of 0.5, 16 extracted frames and a center crop was applied during training.

Subsampling Frequency	**Cropping Technique**	**Train Accuracy**	**Test Accuracy**
1	center	78.9%	41.5%
2	center	94.3%	53.3%
4	center	99.8%	38.3%
2	none(resizing)	67.6%	36.1%

a lower test classification accuracy. This results can be explained by analyzing how the model complexity changes by increasing or decreasing the *width multiplier* value. With a higher *width multiplier* the network has more learnable parameters, and therefore can be more prone to overfitting the training set [159], and thus the train classification accuracies increase while the test classification accuracy decreases. With a lower *width multiplier* value the network has less learnable parameters meaning that is less capable of learning a mapping function between the input layer and the output layer leading to lower test classification accuracies and underfitting. Despite the lower number of parameters there is still a significant overfit;

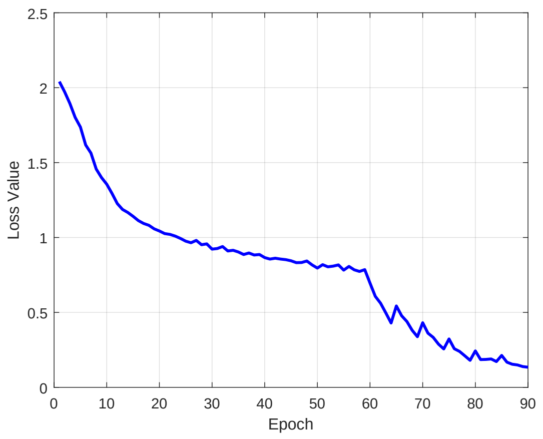
- **Number of extracted frames**: increasing the number of extracted frames to 32 increases the train classification accuracy, however the test classification accuracy is lower. As stated in Chapter 3 the goal of applying augmentation techniques is to help mitigate overfit. One temporal augmentation technique is extracting, in each epoch, a random subset of frames from the same input set. Meaning that in each epoch different temporal information is extracted from the same input set. By increasing the number of extracted frames more information about the input set is being fed to the DL architecture during training. In practice this means that the temporal augmentation technique is not having any effect because in every epoch the temporal information covers roughly the whole activity;
- **Subsampling frequency**: increasing the subsampling frequency leads to results that are similar to the ones obtained by increasing the number of extracted frames. The model is able to classify correctly 99.8% of the training set, however it has a poor generalization capability, achieving a test classification accuracy of 38.3%. Decreasing the subsampling frequency also led to lower test classification accuracies, the extracted frames are not scattered enough through time to represent the activity

being performed. For example, the randomly selected frames could be the beginning of the sequence or the end, which are similar to most of the classes, the person is just standing still;

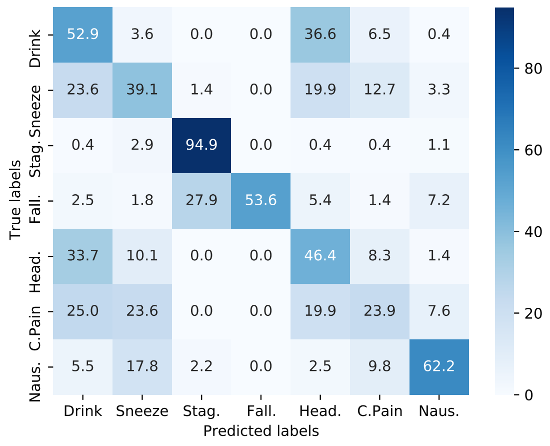
- **Cropping technique**: by applying a center crop to the extracted frames the irrelevant information is excluded and the DL architecture can focus on the important region of the image. The results show that the classification accuracy is greater when compared with the random multiscale cropping or just resizing the frames. However, it is also clear that the difference between the train classification accuracy and the test classification accuracy is greater.

After analyzing the results, it can be concluded that the following combination resulted in the best test classification accuracy: applying a *width multiplier* of 0.5, extracting 16 frames and applying a center crop. Figure 6.1(a) shows the training loss curve for the reference model. The confusion matrix for the reference model is shown in Fig. 6.1(b), it is clear that the model struggles to distinguish activities that share common movements. The activities "Drink Water", "Sneeze/Cough", "Chest Pain" and "Headache" all have in common the hand movement, the person is relatively still and raises one or both hands to the head or upper torso, Fig. 6.2 illustrates a sequence for the four aforementioned activities. The class "Falling Down" is mispredicted with "Staggering" due to the fact that both are related because before falling down there is a period when the person is staggering/losing balance. Extracting only 16 frames from the input set might not be enough to identify if the person is going to fall or is just staggering. Another important observation is the fact that some of the participants of NTU RGB+D 120 Dataset recreated both the "Sneeze/Cough" and "Nausea/Vomiting" with similar movements, they placed one or both hands on their mouth/nose and then bent over.

Finally, after determining a suitable combination of parameters, the remaining 5 DL architectures are trained and tested with the same parameters. Thus, the parameters used are: *width multiplier* of 0.5 (only applies for MobileNetV2), subsampling frequency of 2, 16 extracted frames and a center crop is applied to the extracted frames. The results obtained for the remaining 5 DL architectures (scenarios 2 and 3) are presented in Table 6.5. After analyzing the results, it can be observed that the difference between the train accuracy and test accuracy shows signs of a overfitting problem for the evaluated architectures. Two possible causes could be: (1) the architectures are too complex and overfit the training set or (2) the dataset is too small, meaning that there are not enough



(a) Training loss curve for the reference model on HAR-RGB-bFI. The initial learning rate was set to 0.25 and was divided by 10 on epochs number 60 and 86. The training ended at epoch 135 with a training loss of 0.13.



	Drink	Sneeze	Stag.	Fall.	Head.	C.Pain	Naus.
Drink	52.9	3.6	0.0	0.0	36.6	6.5	0.4
Sneeze	23.6	39.1	1.4	0.0	19.9	12.7	3.3
Stag.	0.4	2.9	94.9	0.0	0.4	0.4	1.1
Fall.	2.5	1.8	27.9	53.6	5.4	1.4	7.2
Head.	33.7	10.1	0.0	0.0	46.4	8.3	1.4
C.Pain	25.0	23.6	0.0	0.0	19.9	23.9	7.6
Naus.	5.5	17.8	2.2	0.0	2.5	9.8	62.2

(b) Confusion matrix for the reference model on HAR-RGB-bFI. Drink - Drink Water, Sneeze - Sneeze/Cough, Stag. - Staggering, Fall. - Falling Down, Head. - Headache, C. Pain - Chest Pain and Naus. - Nausea/Vomiting.

Figure 6.1: Training loss curve and confusion matrix for the reference model on HAR-RGB-bFI.

Table 6.5: Results obtained when training from scratch the six different DL architectures, on HAR-RGB-bFI, on **Subset 1** with the following parameters: *width multiplier* of 0.5 (only applies to MobileNetV2), subsampling frequency of 2 frames, 16 extracted frames and center crop.

Architecture	**Train Accuracy**	**Test Accuracy**
MobileNetV1	94.3%	53.3%
MobileNetV2	88.1%	63.7%
ResNet-18	95.5%	64.1%
ResNet-50	91.9%	59.6%
ResNeXt	93.2%	54.3%
C3D	91.1%	51.9%

training examples to achieve a good generalization. Regarding cause (1), MobileNetV1 has around 41 times less parameters than ResNet-18 and 60 times less parameters than ResNet-50 and overfits the dataset despite being much more narrower and shallower in terms of size. Addressing cause (2), note that typically the DL architectures evaluated in this work are trained from scratch on large scale datasets, contrary to both **Subset 1** and **Subset 2** dimensions. Thus, it is safe to assume that the overfit is mainly related to the two aforementioned causes.

Figure 6.3 shows the training loss curve for the ResNet-50 and its confusion matrix. The confusion matrix for this architecture is similar to MobileNetV1, the classes "Drink Water", "Sneeze/Cough", "Headache" and "Chest Pain" are mispredicted between one

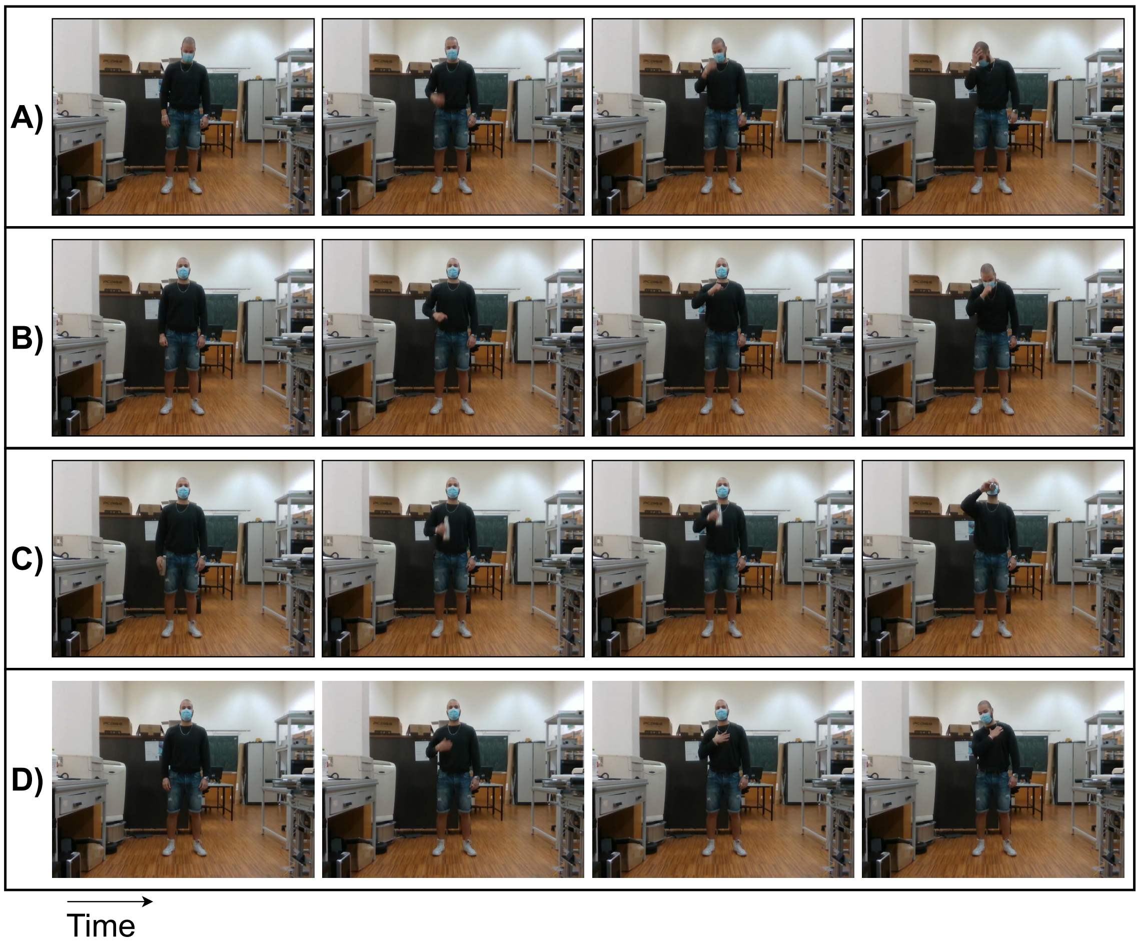


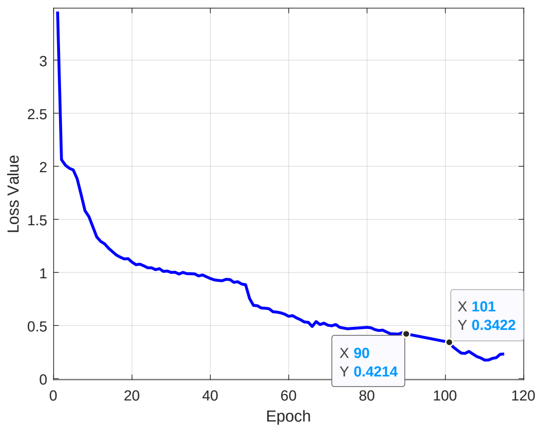
Figure 6.2: Sequences representing the activities: A) Headache, B) Sneeze/Cough, C) Drink Water and D) Chest Pain.

another, however the ResNet-50 is able to predict correctly 83.7% of the ”Drink Water” videos. The classes ”Falling Down” and ”Staggering” are mispredicted with each other as well as ”Nausea Vomiting” and ”Sneeze/Cough”.

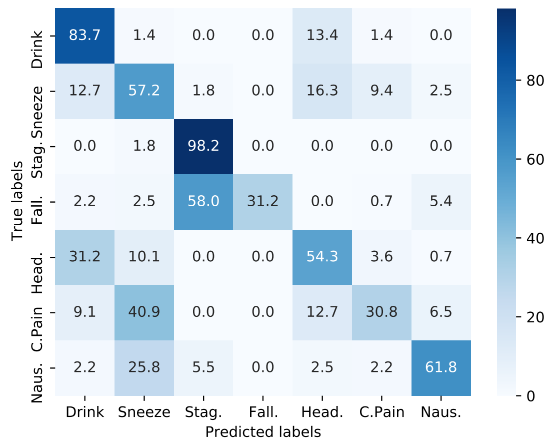
Figure 6.4 presents the training and testing accuracy curves for the reference model. By analyzing both curves it can be seen that after the 30th epoch the model shows signs of overfitting the dataset. The training accuracy continues to improve and the testing accuracy remains stable.

6.4 RGB-based Framework II

Two types of training were performed on the HAR-RGB-bFII: (1) training from scratch and (2) training a pre-trained model. As stated previously, due to hardware limitations it was not possible to test all scenarios on all architectures on both HAR-RGB-based



(a) Training loss curve for the ResNet-50 on HAR-RGB-bFI trained from scratch. The initial learning rate was set to 0.1 and was divided by 10 on epochs number 50, 60, and 100. The training ended at epoch 115 with a training loss of 0.23. Please note that between the epoch 91 and epoch 101 due to an error during the training phase the loss function value was not saved to the *.txt* file.



(b) Confusion matrix for the ResNet-50 on HAR-RGB-bFI trained from scratch. Drink - Drink Water, Sneeze - Sneeze/Cough, Stag. - Staggering, Fall. - Falling Down, Head. - Headache, C. Pain - Chest Pain and Naus. - Nausea/Vomiting.

Figure 6.3: Training loss curve and confusion matrix for the ResNet-50 on HAR-RGB-bFI

Frameworks. Thus, since the HAR-RGB-bFII provided the best results when trained from scratch, it was decided to evaluate the DL architectures when training their pre-trained models only for the HAR-RGB-bFII, since it was more promising than the HAR-RGB-bFI. Therefore, transfer learning was applied to MobileNetV1, MobileNetV2, ResNet-18 and ResNet-50, for ResNeXt and C3D it was not possible to find pre-trained models.

This section is divided into 3 main parts:

1. Training and testing the DL architectures on **Subset 1**;
2. Training and testing the DL architectures on **Subset 2**;
3. Training and testing the DL architectures on the ISR Dataset.

As aforementioned, the remaining experiments will use the same parameters as the reference model of HAR-RGB-bFI except for the cropping technique. In HAR-RGB-bFII no cropping technique is applied to the extracted frames. The extracted frames are the bounding boxes of the subject performing the activity.

6.4.1 Training on Subset 1

The results when training the DL architectures from scratch on **Subset 1** are shown in Table 6.6. As expected, the results have improved, by using a human detection stage the region of interest is extracted from the image and used as input to the feature extraction

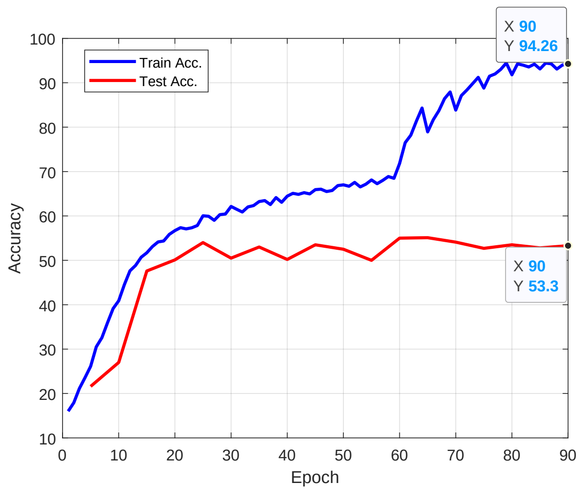


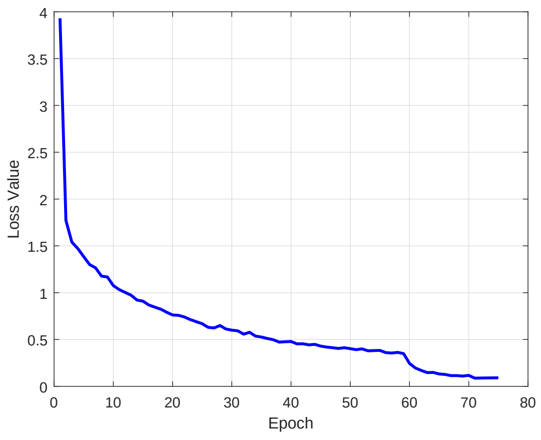
Figure 6.4: Training and testing accuracies for the reference model. The testing accuracy was calculated every 5 epochs.

Table 6.6: Results obtained when training from scratch the six different DL architectures, on HAR-RGB-bFII on **Subset 1** with the following parameters: *width multiplier* of 0.5 (only applies for MobileNetV2), subsampling of 2 frames and 16 extracted frames.

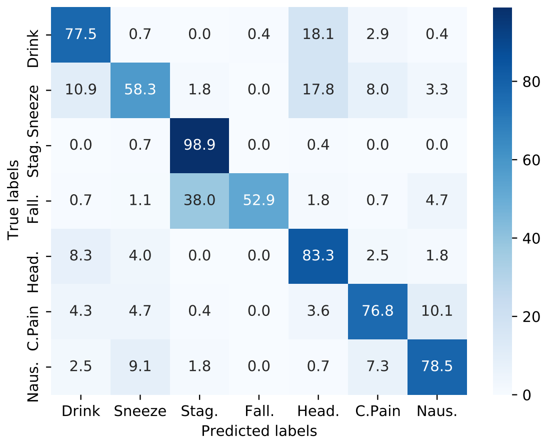
DL Architecture	Train Accuracy	Test Accuracy
MobileNetV1	92.1%	67.5%
MobileNet V2	91.5%	75.2%
ResNet-18	95.1%	74.1%
ResNet-50	96.9%	75.2%
ResNeXt	94.6%	68.8%
C3D	91.8%	74.5%

and classification module. Thus, the DL architecture can focus on this region only. Depending on the model the improvements vary from 10% on ResNet-18 to 22.6% on C3D. A possible explanation for these improvements could be the fact that by removing the majority of the background the DL architecture might not be able to identify patterns present in the data that are not relevant to the recognition task. By concentrating only on the subject performing the activity the DL architecture is able to focus on learning each activity's movements/motions instead of identifying spurious patterns. Many authors identify the property of learning spurious patterns or sampling noise as the explanation why the DL architectures overfit the training sets [103, 159, 160]. By limiting the visual information, to the strictly necessary to identify the activities, the sampling noise is reduced, therefore helping to mitigate the overfit problem.

Figure 6.5 shows both the loss curve and confusion matrix for ResNet-50 on HAR-RGB-bFII. When comparing the results with those obtained for ResNet-50 on HAR-RGB-bFI,



(a) Training loss curve for the ResNet-50 on HAR-RGB-bFII trained from scratch. The initial learning rate was set to 0.1 and was divided by 10 on epochs number 60 and 70. The training ended at epoch 75 with a training loss of 0.09.



	Drink	Sneeze	Stag.	Fall.	Head.	C.Pain	Naus.
Drink	77.5	0.7	0.0	0.4	18.1	2.9	0.4
Sneeze	10.9	58.3	1.8	0.0	17.8	8.0	3.3
Stag.	0.0	0.7	98.9	0.0	0.4	0.0	0.0
Fall.	0.7	1.1	38.0	52.9	1.8	0.7	4.7
Head.	8.3	4.0	0.0	0.0	83.3	2.5	1.8
C.Pain	4.3	4.7	0.4	0.0	3.6	76.8	10.1
Naus.	2.5	9.1	1.8	0.0	0.7	7.3	78.5

(b) Confusion matrix for the ResNet-50 on HAR-RGB-bFII trained from scratch. Drink - Drink Water, Sneeze - Sneeze/Cough, Stag. - Staggering, Fall. - Falling Down, Head. - Headache, C. Pain - Chest Pain and Naus. - Nausea/Vomiting.

Figure 6.5: Training loss curve and confusion matrix for the ResNet-50 trained from scratch on HAR-RGB-bFII.

it is clear that the classification accuracies have improved for all classes except for "Drink Water" which lowered 6.2%. However, the classification accuracy for the "Drink Water" class is still very high and the overall classification accuracy improved by 15.6%. These results demonstrate that by including a human detection stage the classification accuracy of the HAR-RGB framework improves up to 22.6%, as mentioned earlier, depending on the feature extraction and classification module (DL architecture) used.

Another possibility for trying to further improve the results is to apply transfer learning techniques. Typically, when applying transfer learning to an HAR framework the models are pre-trained on the large scale Kinetics dataset [55] and then used as feature extractors while the top layers are trained on a smaller scale dataset [7, 55, 57, 161]. For the purpose of this dissertation, when training the pre-trained models, all the layers but the last are frozen and only the weights from the output layer are updated during the training phase. As aforementioned, it was not possible to find pre-trained models for all architectures. Thus, this technique was applied on MobileNetV1, MobileNetV2, ResNet-18 and ResNet-50 (the models were pre-trained on Kinetics-600 and published online[3] by [7]). Pre-training the models on Kinetics-600 during the dissertation research work was impossible due to both hardware and time limitations.

The results obtained, when training the pre-trained the models, are shown on Table 6.7.

[3]https://github.com/okankop/Efficient-3DCNNs

Table 6.7: Results obtained when training the pre-trained models, on HAR-RGB-bFII on **Subset 1** with the following parameters: *width multiplier* of 0.5 (only applies to MobileNetV2), subsampling of 2 frames and 16 extracted frames.

DL Architecture	**Train Accuracy**	**Test Accuracy**
MobileNetV1	90.7%	78.6%
MobileNetV2	92.1%	81.6%
ResNet-18	95.9%	81%
ResNet-50	96.8%	88.1%

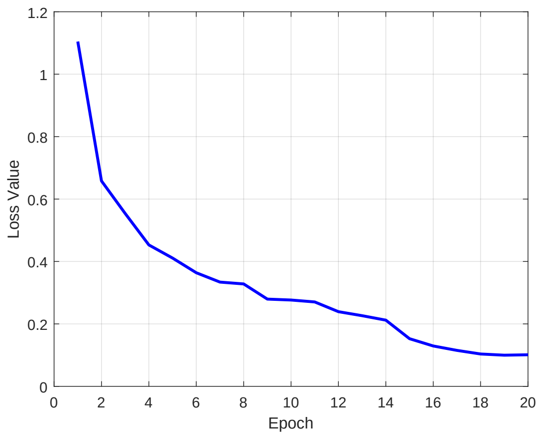
As expected, the test classification accuracy has improved for all models while the train classification accuracies are similar to the previous ones. It was possible to improve the best test classification accuracy obtained previously by 12.9%, up to 88.1%.

Figure 6.6 shows the loss curves and the confusion matrix fro the ResNet-50 on HAR-RGB-bFII when training a pre-trained model. By comparing with the results obtained when training the model from scratch it is clear that all the classes improved their test classification accuracies. It can also be observed that regarding the mispredictions the model follows a similar behavior as before, the classes "Drink Water", "Sneeze/Cough", "Headache" and "Chest Pain" are mispredicted between one another. The class "Staggering" is predicted correctly in 98.9% of the examples and the "Falling Down" class improved its classification accuracy by 31.9% up to 84.8%. It also interesting to note that the class "Nausea/Vomiting" is mispredicted with "Sneeze/Cough" and "Chest Pain" in similar percentages, this might be due to the fact that in some of the "Chest Pain" examples the subjects might be raising one or both hands to the chest area and bending over, similarly to "Nausea/Vomiting" and "Sneeze/Cough".

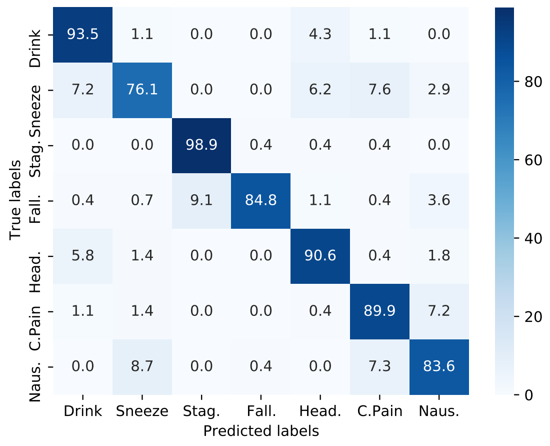
6.4.2 Training on Subset 2

The activities in **Subset 2** were selected with the goal of having a less challenging dataset by chosing a smaller number of activities and activities that are more distinct between one another, and therefore easier to identify. Thus, it is expected that the framework achieves a better performance. Table 6.8 shows the results obtained when training the DL architectures from scratch. As expected, it can be concluded that all architectures perform better. ResNet-18 achieves the highest test classification accuracy with 82.6%.

Following the same method as before, it was used transfer learning. Table 6.9 shows the results obtained when training the pre-trained models on **Subset 2**. ResNet-50 achieves



(a) Training loss curve for the ResNet-50 on HAR-RGB-bFII when training a pre-trained model. The initial learning rate was set to 0.01 and was divided by 10 on epoch number 15. The training ended at epoch 20 with a training loss of 0.1.



True \ Predicted	Drink	Sneeze	Stag.	Fall.	Head.	C.Pain	Naus.
Drink	93.5	1.1	0.0	0.0	4.3	1.1	0.0
Sneeze	7.2	76.1	0.0	0.0	6.2	7.6	2.9
Stag.	0.0	0.0	98.9	0.4	0.4	0.4	0.0
Fall.	0.4	0.7	9.1	84.8	1.1	0.4	3.6
Head.	5.8	1.4	0.0	0.0	90.6	0.4	1.8
C.Pain	1.1	1.4	0.0	0.0	0.4	89.9	7.2
Naus.	0.0	8.7	0.0	0.4	0.0	7.3	83.6

(b) Confusion matrix for the ResNet-50 on HAR-RGB-bFII when training a pre-trained model. Drink - Drink Water, Sneeze - Sneeze/Cough, Stag. - Staggering, Fall. - Falling Down, Head. - Headache, C. Pain - Chest Pain and Naus. - Nausea/Vomiting.

Figure 6.6: Training loss curve and confusion matrix for the ResNet-50 on HAR-RGB-bFII when training a pre-trained model.

Table 6.8: Results obtained when training from scratch the six different DL architectures, on HAR-RGB-bFII, on **Subset 2** with the following parameters: *width multiplier* of 0.5 (only applies to MobileNetV2), subsampling of 2 frames and 16.

DL Architecture	Train Accuracy	Test Accuracy
MobileNetV1	87.6%	75.1%
MobileNetV2	91.1%	80.8%
ResNet-18	96.5%	82.6%
ResNet-50	96.5%	78.8%
ResNeXt	95.7%	71.4%
C3D	93%	82.3%

the best test classification accuracy with 95.1% by improving 16.3% comparatively to the classification accuracy obtained when trained from scratch.

6.4.3 Testing on the ISR Dataset

The experiments made on the ISR Dataset had the objective of evaluating the generalization capability of the HAR-RGB-bFII when trained on the NTU RGB+D 120 Dataset and tested on the ISR Dataset. The results are shown in Table 6.10 (top 4 rows), it can be observed that all the DL architectures have poor generalization capabilities. Despite the high train classification accuracy on the NTU RGB+D 120 Dataset the test classification accuracy on the ISR Dataset is low for all models. The settings in which the ISR Dataset

Table 6.9: Results obtained when training the pre-trained models, on HAR-RGB-bFII on **Subset 2** with the following parameters: *width multiplier* of 0.5 (only applies to MobileNetV2), subsampling of 2 frames and 16.

DL Architecture	**Train Accuracy**	**Test Accuracy**
MobileNetV1	94.7%	83.5%
MobileNetV2	94.8%	87.3%
ResNet-18	97.7%	88.3%
ResNet-50	98.7%	95.1%

was collected were as close as possible to the settings of the NTU RGB+D 120 Dataset (distance to the camera, height of the camera and number of FPS). However, it is clear that the HAR-RGB-bFII struggles to achieve the same results obtained when tested on the NTU RGB+D 120 Dataset.

Trying to find any issues with the ISR Dataset the majority of the 300 collected videos was analyzed, as well as part of the 6636 videos of the NTU RGB+D 120 Dataset. Two observations can be drawn from analyzing the ISR Dataset: (1) the dataset was collected with a dark background (black cabinet and black chalkboard), half of the participants were wearing dark clothes and only 3 participants were wearing short sleeved clothes; (2) the ISR Dataset has an average of 98 frames per video while the NTU RGB+D 120 Dataset has an average of 83 frames per video. In (1) it can be concluded that when taking into account all three factors (dark background, dark clothes and long sleeved clothes) the arms movements are challenging to identify and track, in some of the situations only the hand movement can be identified and tracked effectively. In (2) it can be concluded that a great part of the frames are capturing the person standing still with the arms down, contrary to the NTU RGB+D 120 Dataset where the activity starts sooner and fewer frames are just capturing the person standing still. Also, in the NTU RGB+D 120 Dataset it was observed that it is easier to distinguish the participants from the background and a great majority of the participants were wearing short sleeved clothes, and thus the arms movement is more easily identified and tracked. Taking into account all the aforementioned factors it is safe to assume that, while it was recorded under similar settings, the ISR Dataset is still very different from the NTU RGB+D 120 Dataset.

Finally, two other tests were performed were training the pre-trained models on the ISR Dataset for both HAR-RGB-based Frameworks. The results are shown in Table 6.10, the middle 4 rows present the results obtained for HAR-RGB-bFII and the bottom 4

Table 6.10: Results obtained when training the models on the ISR Dataset.

	DL Architecture	Train Accuracy	Test Accuracy
HAR-RGB-bFII NTU RGB+D 120 Dataset	**MobileNetV1**	90.7%	22.2%
	MobileNetV2	92.1%	17.5%
	ResNet-18	95.9%	25.4%
	ResNet-50	96.8%	26.7%
HAR-RGB-bFII ISR Dataset	**MobileNetV1**	93.2%	47.6%
	MobileNetV2	97.3%	44.4%
	ResNet-18	95.9%	55.5%
	ResNet-50	96.6%	46%
HAR-RGB-bFI ISR Dataset	**MobileNetV1**	97.3%	39.7%
	MobileNetV2	95.9%	41.3%
	ResNet-18	98.6%	41.3%
	ResNet-50	97.6%	35.4%

rows present the results obtained for HAR-RGB-bFI. By training the models on the ISR Dataset it can be seen that the test classification accuracies are higher, although the results are still mediocre. Two conclusion can be drawn: (1) the framework is able to identify the activities during the training phase, meaning that it is possible to use this type of framework for the ISR Dataset; and (2) the dataset is probably too small to effectively train the DL architectures used in this work, even when training a pre-trained model and only optimizing the last layer the DL architecture does not generalize.

6.5 Skeleton-based Framework

As aforementioned in the previous chapters, the HAR-S-bF used in this dissertation was proposed by [1] for both the NTU RGB+D 120 Dataset and the Kinetics-600 Dataset and the source code was made publicly available[4]. Thus, for the purpose of the dissertation few tests and code adaptations needed to be made to the original work.

The HAR-S-bF was trained from scratch on both **Subsets**. The NTU RGB+D 120 Dataset was recorded with a Kinect V2 camera. This camera is capable of extracting, tracking and providing data for up to 6 skeletons in the image. For the purpose of the dissertation, the framework used to extract the skeleton was OpenPose [64], a state of the art pose estimation algorithm. Both OpenPose and Kinect V2 provide a skeleton with 25 joints, however the skeleton configuration is different, some joints are positioned in different

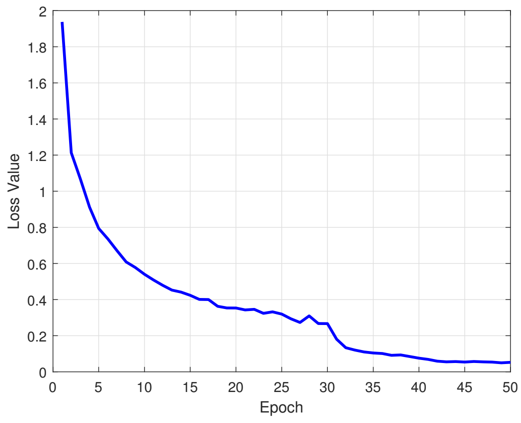
[4]https://github.com/lshiwjx/2s-AGCN

Table 6.11: Results obtained when training the GCN from scratch on both **Subsets**. It was not possible to calculate the train classification accuracy. However, it is shown both validation classification accuracy for the B-Stream and J-Stream. Both streams are run at different moments and then their output vectors are added and the classification is made based on the resulting vector.

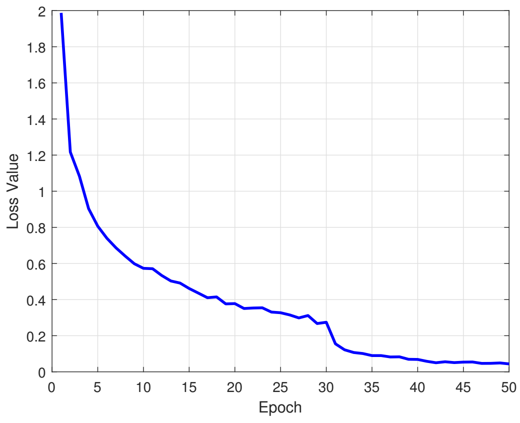
Subset	**Bone-Stream Val. Accuracy**	**Joint-Stream Val. Accuracy**	**Test Accuracy**
Subset 1	86.8%	88%	88.9%
Subset 2	90.3%	92.5%	93%

locations. For example, Kinect V2 provides one joint for the head while OpenPose provides 5 joints (center of the head, eyes and ears). Therefore, the HAR-S-bF is incompatible with the OpenPose skeleton. Some code adaptations can be made to make the code compatible with OpenPose, however, due to time limitations it was not possible to analyze the source code and make the necessary adaptations.

The results obtained for training the HAR-S-bF on both **Subsets** are presented in Table 6.11. Comparing the results with the results obtained by the authors [1] it can be concluded that the framework was trained successfully achieving similar results on both datasets. Figure 6.7 presents the loss curves for the training process of both streams on **Subset 1**.



(a) Loss curve for the J-stream.



(b) Loss curve for the B-stream.

Figure 6.7: Loss curves for both B-stream and J-stream for **Subset 1**.

6.6 Frameworks Runtimes

It is important to take into consideration not only the classification accuracy of the frameworks but also their runtime. The mobile platforms where the evaluated frameworks might

Table 6.12: Runtimes for the different modules and architectures evaluated in this dissertation. For both RGB-based frameworks it is assumed a number of extracted frames equal to 16 and a resolution of 112×112 pixels. For the HAR-S-bF it is assumed that the skeleton sequence length is equal to 300. The runtime is measured in SPS.

Module	Runtime(SPS)
MobileNetV1	25 SPS
MobileNetV2	15 SPS
ResNet-18	47 SPS
ResNet-50	25 SPS
C3D	111 SPS
Resnext	11 SPS
2sAGCN	13 SPS
OpenPose	0.04 SPS
YOLOv3-spp	1.4 SPS

be deployed generally have at most a laptop as the computation unit. Thus, lacking the computational power necessary to have an acceptable real-time performance.

To evaluate the runtime of the frameworks it is necessary to calculate the runtime of all the modules that are part of each framework, summing each one for obtaining the total runtime for each framework. For the HAR-RGB-bFI this means calculating the runtimes for the feature extraction and classification module (the preprocessing stage runtime is negligible when compared to the architectures' runtime). For the HAR-RGB-bFII in addition to the feature extraction and classification module the human detection stage runtime also needs to be calculated. Finally, for the HAR-S-bF the runtime of the pose estimation stage also needs to be calculated in addition to the feature extraction and classification module.

The runtime is measured in terms of Sequences Per Second (SPS). For the RGB-based Frameworks testing phase the input sequence is a set of 16 frames with a resolution of 112×112 pixels. For the HAR-S-bF the input skeleton sequence length is 300, the number of input skeletons is 2 and the number of joints is 25. For both the human detection stage and skeleton extraction stage (corresponds to the offline stage of HAR-RGB-bFII and HAR-S-bF respectively) the input is a set of 16 frames with 640x480 resolution (the same resolution used to record ISR Dataset). Table 6.12 shows the runtimes for each module.

By analyzing the results it is clear that both the human detection stage and pose estimation stages (bottom 2 rows) limit the number of sequences per seconds that both frameworks can compute. At best HAR-RGB-bFII could compute 1.4 SPS while HAR-S-bF could compute 0.04 SPS. On the other hand, because HAR-RGB-bFI do not requires a human detection stage neither a pose estimation stage it is able to compute 111 SPS with the fastest DL architecture and 11 SPS with the slowest DL architecture. Note that for the HAR-S-bF architecture it was assumed that the input sequence contains 300 skeletons. However, in a real scenario the number of input skeletons would be the same as the number of input frames for the RGB frameworks, and thus the number of SPS would be higher than the one calculated.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The main objective of the dissertation was to develop, train and test an HAR framework. For such purpose, both the classification accuracy and, no less important, the runtime for each framework were evaluated.

In terms of classification accuracy, the results show that for both **Subset 1** and **Subset 2** the frameworks are capable of identifying the activities with high accuracies. However, when tested on the ISR Dataset the classification accuracies are significantly lower. This results might be related to the ISR Dataset size, probably has not enough training examples and the models overfit the training set. More research work needs to be done in this topic to fully understand why the results obtained when training on NTU RGB+D 120 dataset could not be replicated in the ISR Dataset. In a future work, the causes that led to a low classification accuracy must be identified and a solution to mitigate them should be developed.

It is also important to note that due to the human detection stage the HAR-RGB-bFII lacks the ability to classify activities with more than one person (its input is a bounding box that contains a single person). Therefore, for applications that require the identification of activities that involve more than one person a solution for this problem must be proposed. Or, in alternative, using the HAR-RGB-bFI or the HAR-S-bF in such scenarios.

The obtained results show that a real-world scenario application for the developed frameworks is possible. Nowadays laptops are equipped with graphics processing units that are capable of running DL frameworks. Despite not having evaluated the frameworks on a laptop, the hardware characteristics of the desktop used are well within the reach of a medium range laptop. Thus, it is safe to assume that similar results would be obtained on a mobile platform, assuming no other programs running in background.

The achieved results for the presented frameworks indicate that they can be successfully implemented in a service robot as to identify health related classes. Or, in the case of

Subset 2, implemented in a social or personal robot which would be able to identify when someone wants to start an interaction with it.

The main objective proposed in the introduction was successfully achieved. Three different frameworks were trained, tested and evaluated, and promising results were achieved.

7.2 Future Work

To improve the current work several topics can be addressed, such as:

- **ISR Dataset**: continue improving the dataset by including more activities and participants. Include different backgrounds and evaluate the effect of background clutter on classification accuracy. Include also activities recorded when the InterBot is moving;
- **Classification accuracy on the ISR Dataset**: identify the reasons behind the low classification accuracy on ISR Dataset and propose solutions for them;
- **HAR-RGB-bFII**: research the problem of detecting social activities;
- **HAR-S-bF**: evaluate this framework on the ISR dataset under two settings: (1) training from scratch and (2) transfer learning from the Kinetics-Skeleton dataset [162] or NTU RGB+D 120 dataset [2].

Bibliography

[1] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[2] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. Kot, "Ntu rgb+d 120: a large-scale benchmark for 3d human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[3] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[4] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: inverted residuals and linear bottlenecks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[7] O. Köpüklü, N. Kose, A. Gunduz, and G. Rigoll, "Resource efficient 3d convolutional neural networks," *IEEE/CVF International Conference on Computer Vision Workshop*, 2019.

[8] G. Guo and A. Lai, "A survey on still image based human action recognition," *Pattern Recognition*, 2014.

[9] Z. Zhao, H. Ma, and S. You, "Single image action recognition using semantic body part actions," *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[10] Z. Zhao, H. Ma, and X. Chen, "Generalized symmetric pair model for action classification in still images," *Pattern Recognition*, 2017.

[11] Y. Zhang, L. Cheng, J. Wu, J. Cai, M. N. Do, and J. Lu, "Action recognition in still images with minimum annotation efforts," *IEEE Transactions on Image Processing*,

2016.

[12] D. Girish, V. Singh, and A. Ralescu, “Understanding action recognition in still images,” *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[13] J. Wirtz, P. G. Patterson, W. H. Kunz, T. Gruber, V. N. Lu, S. Paluch, and A. Martins, “Brave new world: service robots in the frontline,” *Journal of Service Management*, 2018.

[14] M. Mende, M. L. Scott, J. van Doorn, D. Grewal, and I. Shanks, “Service robots rising: How humanoid robots influence service experiences and elicit compensatory consumer responses,” *Journal of Marketing Research*, 2019.

[15] M. Čaić, D. Mahr, and G. Oderkerken-Schröder, “Value of social robots in services: social cognition perspective,” *Journal of Services Marketing*, 2019.

[16] H. Moradi, K. Kawamura, E. Prassler, G. Muscato, P. Fiorini, T. Sato, and R. Rusu, “Service robotics (the rise and bloom of service robots),” *IEEE Robotics & Automation Magazine*, 2013.

[17] A. K. Pandey and R. Gelin, “A mass-produced sociable humanoid robot: pepper: The first machine of its kind,” *IEEE Robotics & Automation Magazine*, 2018.

[18] I. Leite, C. Martinho, and A. Paiva, “Social robots for long-term interaction: a survey,” *International Journal of Social Robotics*, 2013.

[19] J. Broekens, M. Heerink, H. Rosendal, *et al.*, “Assistive social robots in elderly care: a review,” *Gerontechnology*, 2009.

[20] “pepper.” `https://www.softbankrobotics.com/emea/en/pepper`. Accessed: 2020-10-12.

[21] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, *et al.*, “Spencer: a socially aware service robot for passenger guidance and help in busy airports,” *Field and Service Robotics*, 2016.

[22] D. Portugal, P. Alvito, E. Christodoulou, G. Samaras, and J. Dias, “A study on the deployment of a service robot in an elderly care center,” *International Journal of Social Robotics*, 2019.

[23] F. Tanaka, K. Isshiki, F. Takahashi, M. Uekusa, R. Sei, and K. Hayashi, “Pepper learns together with children: development of an educational application,” *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015.

[24] A. Gardecki, M. Podpora, R. Beniak, and B. Klin, “The pepper humanoid robot in

front desk application," *Progress in Applied Electrical Engineering*, 2018.

[25] D. Allegra, F. Alessandro, C. Santoro, and F. Stanco, "Experiences in using the pepper robotic platform for museum assistance applications," *25th IEEE International Conference on Image Processing*, 2018.

[26] G. Castellano, B. De Carolis, N. Macchiarulo, and G. Vessio, "Pepper4museum: towards a human-like museum guide," *Workshop on Advanced Visual Interfaces and Interactions in Cultural Heritage*, 2020.

[27] I. Aaltonen, A. Arvola, P. Heikkilä, and H. Lammi, "Hello pepper, may i tickle you? children's and adults' responses to an entertainment robot at a shopping mall," *Proceedings of the Companion of the ACM/IEEE International Conference on Human-Robot Interaction*, 2017.

[28] M. E. Foster, R. Alami, O. Gestranius, O. Lemon, M. Niemelä, J.-M. Odobez, and A. K. Pandey, "The mummer project: engaging human-robot interaction in real-world public spaces," *International Conference on Social Robotics*, 2016.

[29] E. Saad, M. A. Neerincx, and K. V. Hindriks, "Welcoming robot behaviors for drawing attention," *ACM/IEEE International Conference on Human-Robot Interaction*, 2019.

[30] S. T. Hansen, M. Svenstrup, H. J. Andersen, and T. Bak, "Adaptive human aware navigation based on motion pattern analysis," *IEEE International Symposium on Robot and Human Interactive Communication*, 2009.

[31] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, 2007.

[32] D. Vasquez, P. Stein, J. Rios-Martinez, A. Escobedo, A. Spalanzani, and C. Laugier, "Human aware navigation for assistive robotics," *Experimental Robotics*, 2013.

[33] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: a survey," *Robotics and Autonomous Systems*, 2013.

[34] R. Kirby, *Social robot navigation.* PhD thesis, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2010.

[35] K. Charalampous, I. Kostavelis, and A. Gasteratos, "Recent trends in social aware robot navigation: a survey," *Robotics and Autonomous Systems*, 2017.

[36] A. Vega, L. J. Manso, D. G. Macharet, P. Bustos, and P. Núñez, "Socially aware robot navigation system in human-populated and interactive environments based on an adaptive spatial density function and space affordances," *Pattern Recognition*

Letters, 2019.

[37] F. Baradel, C. Wolf, J. Mille, and G. W. Taylor, “Glimpse clouds: human activity recognition from unstructured feature points,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[38] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, “Video action transformer network,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[39] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, “A short note about kinetics-600,” *arXiv preprint arXiv:1808.01340*, 2018.

[40] P. Wang, W. Li, P. Ogunbona, J. Wan, and S. Escalera, “Rgb-d-based human motion recognition with deep learning: a survey,” *Computer Vision and Image Understanding*, 2018.

[41] Y. Wang, S. Cang, and H. Yu, “A survey on wearable sensor modality centred human activity recognition in health care,” *Expert Systems with Applications*, 2019.

[42] J. Zhang, W. Li, P. O. Ogunbona, P. Wang, and C. Tang, “Rgb-d-based action recognition datasets: a survey,” *Pattern Recognition*, 2016.

[43] J. K. Aggarwal and Q. Cai, “Human motion analysis: a review,” *Computer Vision and Image Understanding*, 1999.

[44] J. Yamato, J. Q. Ohya, and K. Ishii, “Recognizing human action in time-sequential images using hidden markov model,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1992.

[45] C. Bregler, “Learning and recognizing human dynamics in video sequences,” *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.

[46] C. Achard, X. Qu, A. Mokhber, and M. Milgram, “A novel approach for recognition of human actions with semi-global features,” *Machine Vision and Applications*, 2008.

[47] R. Polana and R. Nelson, “Low level recognition of human motion (or how to get your man without finding his body parts),” *Proceedings of the IEEE Workshop on Motion of Non-rigid and Articulated Objects*, 1994.

[48] A. F. Bobick and J. W. Davis, “The recognition of human movement using temporal templates,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[49] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, “Actions as space-time shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[50] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," *European Conference on Computer Vision*, 2010.

[51] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[52] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," *Proceedings of the 17th International Conference on Pattern Recognition*, 2004.

[53] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[54] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: a dataset of 101 human actions classes from videos in the wild," *CRCV-TR-12-01*, 2012.

[55] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[56] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," *Proceedings of the International Conference on Computer Vision*, 2011.

[57] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[58] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, *et al.*, "Ava: a video dataset of spatio-temporally localized atomic visual actions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[59] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in homes: crowdsourcing data collection for activity understanding," *European Conference on Computer Vision*, 2016.

[60] J. Materzynska, G. Berger, I. Bax, and R. Memisevic, "The jester dataset: a large-scale video dataset of human gestures," *IEEE/CVF International Conference on Computer Vision Workshop*, 2019.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*,

2016.

[62] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, “Pose flow: rfficient online pose tracking,” *British Machine Vision Conference*, 2018.

[63] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, G. Pons-Moll, and C. Theobalt, “Single-shot multi-person 3d pose estimation from monocular rgb,” *International Conference on 3D Vision*, 2018.

[64] Z. Cao, G. H. Martinez, T. Simon, S. Wei, and Y. A. Sheikh, “Openpose: realtime multi-person 2d pose estimation using part affinity fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[65] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, “View adaptive neural networks for high performance skeleton-based human action recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[66] C. Caetano, F. Brémond, and W. R. Schwartz, “Skeleton image representation for 3d action recognition based on tree structure and reference joints,” *32nd SIBGRAPI Conference on Graphics, Patterns and Images*, 2019.

[67] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, “Actional-structural graph convolutional networks for skeleton-based action recognition,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[68] A. Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier, “The internet of things for ambient assisted living,” *International Conference on Information Technology: New Generations*, 2010.

[69] T. Brezmes, J.-L. Gorricho, and J. Cotrina, “Activity recognition from accelerometer data on a mobile phone,” *International Work-Conference on Artificial Neural Networks*, 2009.

[70] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity recognition using cell phone accelerometers,” *ACM SigKDD Explorations Newsletter*, 2011.

[71] N. Győrbíró, Á. Fábián, and G. Hományi, “An activity recognition system for mobile phones,” *Mobile Networks and Applications*, 2009.

[72] A. Henpraserttae, S. Thiemjarus, and S. Marukatat, “Accurate activity recognition using a mobile phone regardless of device orientation and location,” *International Conference on Body Sensor Networks*, 2011.

[73] E. M. Tapia, S. S. Intille, and K. Larson, “Activity recognition in the home using simple and ubiquitous sensors,” *International Conference on Pervasive Computing*,

2004.

[74] A. Arcelus, M. H. Jones, R. Goubran, and F. Knoefel, "Integration of smart home technologies in a health monitoring system for the elderly," *21st International Conference on Advanced Information Networking and Applications Workshops*, 2007.

[75] C.-H. Lu and L.-C. Fu, "Robust location-aware activity recognition using wireless sensor network in an attentive home," *IEEE Transactions on Automation Science and Engineering*, 2009.

[76] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, "Deep learning for sensor-based human activity recognition: overview, challenges and opportunities," *arXiv preprint arXiv:2001.07416*, 2020.

[77] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: a survey," *Pattern Recognition Letters*, 2019.

[78] Y. Wang, S. Cang, and H. Yu, "A survey on wearable sensor modality centred human activity recognition in health care," *Expert Systems with Applications*, 2019.

[79] K. Aminian, P. Robert, E. Jequier, and Y. Schutz, "Estimation of speed and incline of walking using neural network," *IEEE Transactions on Instrumentation and Measurement*, 1995.

[80] J. Morris, "Accelerometry — a technique for the measurement of human body movements," *Journal of Biomechanics*, 1973.

[81] T. C. Wong, J. G. Webster, H. J. Montoye, and R. Washburn, "Portable accelerometer device for measuring human energy expenditure," *IEEE Transactions on Biomedical Engineering*, 1981.

[82] G. A. Meijer, K. R. Westerterp, H. Koper, and F. ten Hoor, "Assessment of energy expenditure by recording heart rate and body acceleration," *Medicine and Science in Sports and Exercise*, 1989.

[83] K. Aminian, P. Robert, E. Jequier, and Y. Schutz, "Level, downhill and uphill walking identification using neural networks," *Electronics Letters*, 1993.

[84] C. Randell and H. Muller, "Context awareness by analysing accelerometer data," *Digest of Papers. Fourth International Symposium on Wearable Computers*, 2000.

[85] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *International Conference on Pervasive Computing*, 2004.

[86] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," *International*

Workshop on Wearable and Implantable Body Sensor Networks, 2006.

[87] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," *6th International Conference on Mobile Computing, Applications and Services*, 2014.

[88] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen, "The opportunity challenge: a benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, 2013.

[89] T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and Tröster, "Wearable activity tracking in car manufacturing," *IEEE Pervasive Computing*, 2008.

[90] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter*, 2011.

[91] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, 2016.

[92] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," *International Joint Conference on Artificial Intelligence*, 2015.

[93] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys*, 2014.

[94] Z. Qin, Y. Zhang, S. Meng, Z. Qin, and K.-K. R. Choo, "Imaging and fusing time series for wearable sensor-based human activity recognition," *Information Fusion*, 2020.

[95] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015.

[96] G. Laput and C. Harrison, "Sensing fine-grained hand activity with smartwatches," *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2019.

[97] M. M. Hassan, M. Z. Uddin, A. Mohamed, and A. Almogren, "A robust human activity recognition system using smartphone sensors and deep learning," *Future Generation Computer Systems*, 2018.

[98] C. Chen, R. Jafari, and N. Kehtarnavaz, "A survey of depth and inertial sensor fusion for human action recognition," *Multimedia Tools and Applications*, 2017.

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

[100] M. Ehatisham-Ul-Haq, A. Javed, M. A. Azam, H. M. Malik, A. Irtaza, I. H. Lee, and M. T. Mahmood, "Robust human activity recognition using multimodal feature-level fusion," *IEEE Access*, 2019.

[101] C. Chen, R. Jafari, and N. Kehtarnavaz, "Utd-mhad: a multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor," *2015 IEEE International conference on image processing (ICIP)*, 2015.

[102] J. Imran and B. Raman, "Evaluating fusion of rgb-d and inertial sensors for multimodal human action recognition," *Journal of Ambient Intelligence and Humanized Computing*, 2020.

[103] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* Cambridge, MA, USA: MIT press, 2016.

[104] S. Haykin, *Neural networks and learning machines.* Hamilton, Canada: Prentice Hall, 2008.

[105] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT press, 2012.

[106] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[107] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, 2018.

[108] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[109] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: a review," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[110] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure–activity relationships," *Journal of Chemical Information and Modeling*, 2015.

[111] P. Chang, J. Grinband, B. Weinberg, M. Bardis, M. Khy, G. Cadena, M.-Y. Su, S. Cha, C. Filippi, D. Bota, *et al.*, "Deep-learning convolutional neural networks accurately classify genetic mutations in gliomas," *American Journal of Neuroradiology*, 2018.

[112] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep

convolutional neural networks," *Advances in Neural Information Processing Systems*, 2012.

[113] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[114] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," *IEEE Workshop on Automatic Speech Recognition & Understanding*, 2011.

[115] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *International Conference on Learning Representations*, 2013.

[116] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *International Conference on Learning Representations*, 2015.

[117] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[118] A. Ng, *Machine learning yearning - technical strategy for ai rngineers, in the rra of deep learning.* deeplearning.ai, 2018.

[119] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[120] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *European Conference on Computer Vision*, 2014.

[121] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[122] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," *International Conference on Artificial Neural Networks*, 2010.

[123] M. Lin, Q. Chen, and S. Yan, "Network in network," *International Conference on Learning Representations*, 2014.

[124] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, 2014.

[125] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the in-

ception architecture for computer vision," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[126] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[127] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[128] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, 2016.

[129] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[130] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun, "3d graph neural networks for rgbd semantic segmentation," *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[131] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics*, 2019.

[132] S. Qi, W. Wang, B. Jia, J. Shen, and S.-C. Zhu, "Learning human-object interactions by graph parsing neural networks," *Proceedings of the European Conference on Computer Vision*, 2018.

[133] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2017.

[134] D. Marcheggiani, J. Bastings, and I. Titov, "Exploiting semantics in neural machine translation with graph convolutional networks," *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[135] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-based action recognition with directed graph neural networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[136] Y.-H. Wen, L. Gao, H. Fu, F.-L. Zhang, and S. Xia, "Graph cnns with motif and

variable temporal block for skeleton-based action recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[137] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *32nd AAAI Conference on Artificial Intelligence*, 2018.

[138] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, 2019.

[139] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[140] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in Meural Information Processing Systems*, 2014.

[141] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *European conference on computer vision*, 2014.

[142] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: the all convolutional net," *International Conference on Learning Representations*, 2015.

[143] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[144] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[145] J. Redmon and A. Farhadi, "Yolov3: an incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[146] Q.-C. Mao, H.-M. Sun, Y.-B. Liu, and R.-S. Jia, "Mini-yolov3: real-time object detector for embedded applications," *IEEE Access*, 2019.

[147] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[148] R. Cruz, L. Garrote, A. Lopes, and U. J. Nunes, "Modular software architecture for human-robot interaction applied to the interbot mobile robot," *IEEE International Conference on Autonomous Robot Systems and Competitions*, 2018.

[149] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," *IEEE International Confer-*

ence on Robotics and Automation Workshop on Open Source Software, 2009.

[150] Intel, *Intel RealSense D400 Series Product Family - Datasheet*, January 2019. Revision 005.

[151] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang, "Dc-spp-yolo: dense connection and spatial pyramid pooling based yolo for object detection," *Information Sciences*, 2020.

[152] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: common objects in context," *European Conference on Computer Vision*, 2014.

[153] D. Hebesberger, T. Koertner, C. Gisinger, and J. Pripfl, "A long-term autonomous robot at a care hospital: a mixed methods study on social acceptance and experiences of staff and older adults," *International Journal of Social Robotics*, 2017.

[154] A. Meghdari, A. Shariati, M. Alemi, G. R. Vossoughi, A. Eydi, E. Ahmadi, B. Mozafari, A. Amoozandeh Nobaveh, and R. Tahami, "Arash: a social robot buddy to support children with cancer in a hospital environment," *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 2018.

[155] H. Eftring and S. Frennert, "Designing a social and assistive robot for seniors," *Zeitschrift für Gerontologie und Geriatrie*, 2016.

[156] E. Martinez-Martin and M. Cazorla, "A socially assistive robot for elderly exercise promotion," *IEEE Access*, 2019.

[157] S. Coşar, M. Fernandez-Carmona, R. Agrigoroaie, J. Pages, F. Ferland, F. Zhao, S. Yue, N. Bellotto, and A. Tapus, "Enrichme: perception and interaction of an assistive robot for the elderly at home," *International Journal of Social Robotics*, 2020.

[158] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," *International Conference on Machine Learning*, 2013.

[159] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, 2014.

[160] K. P. Burnham and D. R. Anderson, *Model Selection and Multimodel Inference.* Fort Collins, CO, USA: Springer, 2002.

[161] Z. Qiu, T. Yao, C.-W. Ngo, X. Tian, and T. Mei, "Learning spatio-temporal repre-

sentation with local and global diffusion," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[162] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

Appendix A

NTU RGB+D 120 Dataset PyTorch Class

```
import os
import torch
import pandas as pd
import numpy as np
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms, utils
from skimage import io, transform
import json
import time
import random

def label_to_target(classInd_path):
    actions = pd.read_csv(classInd_path, delim_whitespace=True, header=None)
    d = {}
    for i in range(0, len(actions)):
        d[actions[1][i]] = i
    return(d)

def get_dataset(dataset_path, annotation_path, classInd_path, subset):
    annotation_file = json.load(open(annotation_path))
    class_idx = label_to_target(classInd_path)
    dataset = []
    i = 0
    print("Preparing␣dataset␣(paths␣to␣each␣video␣and␣number␣of␣frames).")
    for video_id, value in annotation_file['database'].items():
        if value['subset'] == subset:
            video_path = dataset_path + "/" + value['annotations']['label']
                + "/" + video_id
            f = open(dataset_path + "/" + value['annotations']['label'] +
                "/" + video_id + '/n_frames', "r")
            nr_frames = f.read()
            f.close()
            sample = {
                'video': video_path,
                'nr_frames': nr_frames,
                'label': class_idx[value['annotations']['label']],
            }
            dataset.append(sample)
        i = i + 1
    return(dataset)


class NTU_RGB_Dataset(Dataset):
    def __init__(self, dataset_path, annotation_path, classInd_path, subset,
                    sample_duration, clip_step, pv, sample_size):
```

```
        """
        Args:
            dataset_path (string): path to the directory with all the
                                  classes folders containing each video.
            annotation_path (string): path to the json annotation file.
            classInd_path (string): path to classInd.txt file.
            subset (string): train or test.
            sample_duration (int): number of frames to extract
                                   from each video.
            horizontal_flip (bool): do horizontal flip on the images or not.
            clip_step (int): a frame is extracted each 'clip_step' steps.
            pv: Probability that a vertical flip is made to the clip.
            sample_size: Size of each frame.
        """
        self.dataset_path = dataset_path
        self.annotation_path = annotation_path
        self.classInd_path = classInd_path
        self.subset = subset
        self.sample_duration = sample_duration
        self.clip_step = clip_step
        self.pv = pv
        self.sample_size = sample_size
        self.dataset = get_dataset(self.dataset_path, self.annotation_path,
                                   self.classInd_path, self.subset)

    def __len__(self):
        return(len(self.dataset))

    def __getitem__(self, idx):
        clip = []
        nr_frames = int(self.dataset[idx]['nr_frames'])
        #Sample starts at a random frame
        current_frame = random.randint(1, nr_frames)
        #Randomize vertical flip
        flag_vertical_flip = random.random() > self.pv

        for i in range(1, self.sample_duration + 1):
            if (current_frame) > (nr_frames):
                current_frame = 1
            frame = io.imread(self.dataset[idx]['video'] + '/image_' +
                    str(current_frame).zfill(5) + '.jpg')
            #Vertical Flip
            if  flag_vertical_flip:
                frame = np.fliplr(frame)
            #Resize (Image: H * W * C)
            frame = transform.resize(frame,
                                     [self.sample_size, self.sample_size])
            #To Tensor (Tensor: C * H * W)
            frame = frame.transpose((2, 0, 1))
            frame = torch.from_numpy(frame).float()

            clip.append(frame)
            current_frame = current_frame + self.clip_step
        #Concatenates all frames
        clip = torch.stack(clip, 0)
        clip = clip.permute((1, 0, 2, 3))
        label = self.dataset[idx]['label']
        return clip, label
```

Appendix B

C3D Neural Network PyTorch Definition

```
import torch.nn as nn
import torch.nn.functional as F
import time

class Net(nn.Module):
    def __init__(self, num_classes):
        super(Net, self).__init__()

        self.conv1 = nn.Conv3d(3, 64, kernel_size=(3, 3, 3),
                               padding=(1, 1, 1))
        self.bn1 = nn.BatchNorm3d(64)
        self.pool1 = nn.MaxPool3d(kernel_size=(2, 2, 2), stride=(1, 2, 2))

        self.conv2 = nn.Conv3d(64, 128, kernel_size=(3, 3, 3),
                               padding=(1, 1, 1))
        self.bn2 = nn.BatchNorm3d(128)
        self.pool2 = nn.MaxPool3d(kernel_size=(2, 2, 2), stride=(2, 2, 2))

        self.conv3a = nn.Conv3d(128, 256, kernel_size=(3, 3, 3),
                                padding=(1, 1, 1))
        self.bn3a = nn.BatchNorm3d(256)
        self.conv3b = nn.Conv3d(256, 256, kernel_size=(3, 3, 3),
                                padding=(1, 1, 1))
        self.bn3b = nn.BatchNorm3d(256)
        self.pool3 = nn.MaxPool3d(kernel_size=(2, 2, 2), stride=(2, 2, 2))

        self.conv4a = nn.Conv3d(256, 512, kernel_size=(3, 3, 3),
                                padding=(1, 1, 1))
        self.bn4a = nn.BatchNorm3d(512)
        self.conv4b = nn.Conv3d(512, 512, kernel_size=(3, 3, 3),
                                padding=(1, 1, 1))
        self.bn4b = nn.BatchNorm3d(512)
        self.pool4 = nn.MaxPool3d(kernel_size=(2, 2, 2), stride=(2, 2, 2))

        self.conv5a = nn.Conv3d(512, 512, kernel_size=(3, 3, 3),
                                padding=(1, 1, 1))
        self.bn5a = nn.BatchNorm3d(512)
        self.conv5b = nn.Conv3d(512, 512, kernel_size=(3, 3, 3),
                                padding=(1, 1, 1))
        self.bn5b = nn.BatchNorm3d(512)
        self.pool5 = nn.MaxPool3d(kernel_size=(1, 2, 2), stride=(2, 2, 2),
                                  padding=(0, 1, 1))

        self.fc6 = nn.Linear(8192, 4096)
        self.fc7 = nn.Linear(4096, 4096)
        self.fc8 = nn.Linear(4096, num_classes)

        self.dropout = nn.Dropout(p=0.5)


    def forward(self, x):
        x = self.pool1(F.relu(self.bn1(self.conv1(x))))

        x = self.pool2(F.relu(self.bn2(self.conv2(x))))

        x = F.relu(self.bn3a(self.conv3a(x)))
        x = self.pool3(F.relu(self.bn3b(self.conv3b(x))))

        x = F.relu(self.bn4a(self.conv4a(x)))
        x = self.pool4(F.relu(self.bn4b(self.conv4b(x))))

        x = F.relu(self.bn5a(self.conv5a(x)))
        x = self.pool5(F.relu(self.bn5b(self.conv5b(x))))

        x = x.view(1, 8192)
        x = F.relu(self.fc6(x))
        x = self.dropout(x)
        x = F.relu(self.fc7(x))
        x = self.dropout(x)

        x = self.fc8(x)

        return x
```

Appendix C

NTU RGB+120 Dataset

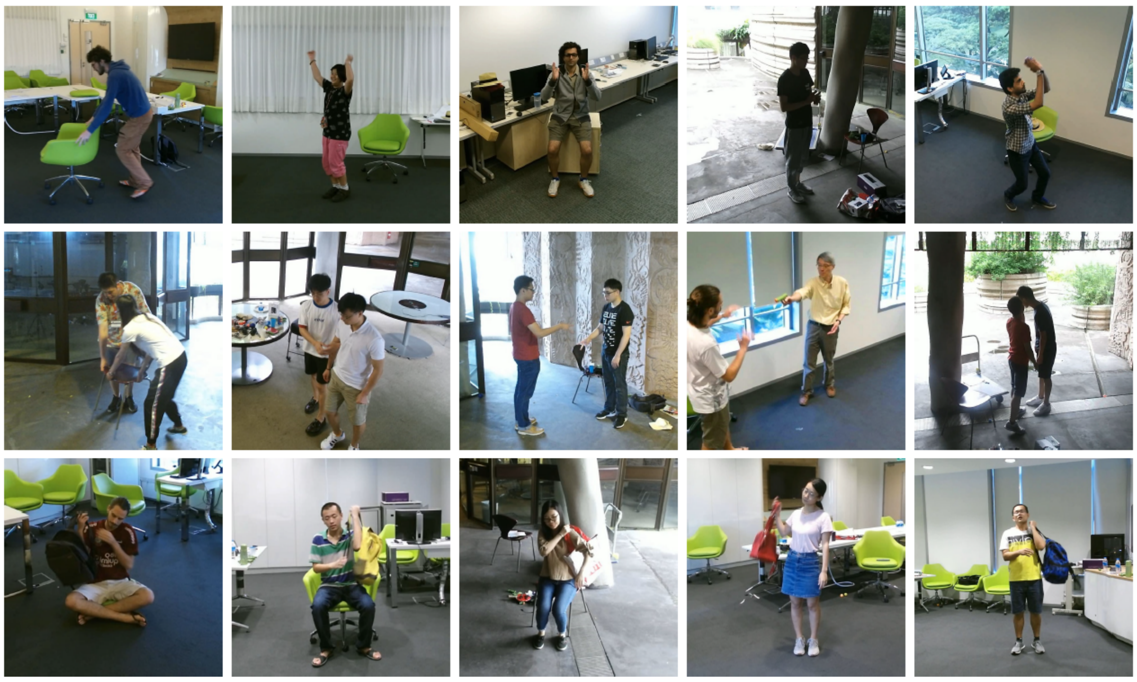


Figure C.1: Samples from the dataset, adapted from [2].

Table C.1: Different camera setups tested. Adapted from [2].

Setup	Height (m)	Distance (m)	Setup	Height (m)	Distance (m)
1	1.7	3.5	17	2.5	3.0
2	1.7	2.5	18	1.8	3.3
3	1.4	2.5	19	1.6	3.5
4	1.2	3.0	20	1.4	4.0
5	1.2	3.0	21	1.7	3.2
6	0.8	3.5	22	1.9	3.4
7	0.5	4.5	23	2.0	3.2
8	1.4	3.5	24	2.4	3.3
9	0.8	2.0	25	2.5	3.3
10	1.8	3.0	26	1.5	2.7
11	1.9	3.0	27	1.3	3.5
12	2.0	3.0	28	1.1	2.9
13	2.1	3.0	29	2.5	2.8
14	2.2	3.0	30	2.4	2.7
15	2.3	3.5	31	1.6	3.0
16	2.7	3.5	32	2.3	3.0

Table C.2: Dataset classes. Adapted from [2].

Daily Actions (82)			
A1: drink water	A2: eat meal	A3: brush teeth	A4: brush hair
A5: drop	A6: pick up	A7: throw	A8: sit down
A9: stand up	A10: clapping	A11: reading	A12: writing
A13: tear up paper	A14: put on jacket	A15: take off jacket	A16: put on a shoe
A17: take off a shoe	A18: put on glasses	A19: take off glasses	A20: put on a hat/cap
A21: take off a hat/cap	A22: cheer up	A23: hand-waving	A24: kicking something
A25: reach into pocket	A26: hopping	A27: jump up	A28: phone call
A29: play with phone/tablet	A30: type on a keyboard	A31: point to something	A32: taking a selfie
A33: check time (from watch)	A34: rub two hands	A35: nod head/bow	A36: shake head
A37: wipe face	A38: salute	A39: put palms together	A40: cross hands in front
A61: put on headphone	A62: take off headphone	A63: shoot at basket	A64: bounce ball
A65: tennis bat swing	A66: juggle table tennis ball	A67: hush	A68: flick hair
A69: thumb up	A70: thumb down	A71: make OK sign	A72: make victory sign
A73: staple book	A74: counting money	A75: cutting nails	A76: cutting paper
A77: snap fingers	A78: open bottle	A79: sniff/smell	A80: squat down
A81: toss a coin	A82: fold paper	A83: ball up paper	A84: play magic cube
A85: apply cream on face	A86: apply cream on hand	A87: put on bag	A88: take off bag
A89: put object into bag	A90: take object out of bag	A91: open a box	A92: move heavy objects
A93: shake fist	A94: throw up cap/hat	A95: capitulate	A96: cross arms
A97: arm circles	A98: arm swings	A99: run on the spot	A100: butt kicks
A101: cross toe touch	A102: side kick	-	-
Medical Conditions (12)			
A41: sneeze/cough	A42: staggering	A43: falling down	A44: headache
A45: chest pain	A46: back pain	A47: neck pain	A48: nausea/vomiting
A49: fan self	A103: yawn	A104: stretch oneself	A105: blow nose
Mutual Actions / Two Person Interactions (26)			
A50: punch/slap	A51: kicking	A52: pushing	A53: pat on back
A54: point finger	A55: hugging	A56: giving object	A57: touch pocket
A58: shaking hands	A59: walking towards	A60: walking apart	A106: hit with object
A107: wield knife	A108: knock over	A109: grab stuff	A110: shoot with gun
A111: step on foot	A112: high-five	A113: cheers and drink	A114: carry object
A115: take a photo	A116: follow	A117: whisper	A118: exchange things
A119: support somebody	A120: rock-paper-scissors	-	-

Appendix D

Deep Learning Networks Architectures

Table D.1: 3D-MobileNetV1 architecture, adapted from [7].

Layer/Stride	**Nr. Blocks**	**Output Size**
Input Layer	-	3x16x112x112
Conv(3x3x3)/s(1,2,2)	1	32x16x56x56
Block/s(2x2x2)	1	64x8x28x28
Block/s(2x2x2)	1	128x4x14x14
Block/s(1x1x1)	1	128x4x14x14
Block/s(2x2x2)	1	256x2x7x7
Block/s(1x1x1)	1	256x2x7x7
Block/s(2x2x2)	1	512x1x4x4
Block/s(1x1x1)	5	512x1x4x4
Block/s(1x1x1)	1	1024x1x4x4
Block/s(1x1x1)	1	1024x1x4x4
AvgPool(1x4x4)/s(1x1x1)	1	1024x1x1x1
Linear(1024xNumCls)	1	NumCls

Table D.2: 3D-MobileNetV2 architecture, adapted from [7].

Layer/Stride	**Nr. Blocks**	**Output Size**
Input Layer	-	3x16x112x112
Conv(3x3x3)/s(1,2,2)	1	32x16x56x56
Block/s(1,1,1)	1	16x16x56x56
Block/s(2,2,2)	2	24x8x28x28
Block/s(2,2,2)	3	32x4x14x14
Block/s(2,2,2)	4	64x2x7x7
Block/s(1,1,1)	3	96x2x7x7
Block/s(2,2,2)	3	160x1x4x4
Block/s(1,1,1)	1	320x1x4x4
Conv(1x1x1)/s(1,1,1)	1	1280x1x4x4
AvgPool/s(1,1,1)	1	1280x1x1x1
Linear(1280xNumCls)	1	NumCls

Table D.3: ResNet-18, ResNet-50 and ResNeXt-101 full architectures. For ResNet-18 the building block is the Basic Block, for ResNet-50 is the Bottleneck Block and for ResNeXt-101 is the ResNeXt Block. All building blocks are presented in Chapter 4. F denotes the number of output channels in each layer. Conv1 layer is common to all three architectures, it is a convolutional layer with a kernel of size (7x7x7) and stride of (1,2,2) followed by a (3x3x3) max-pooling layer with stride of (2,2,2). The first block in Conv3_x, Conv4_x and Conv5_x performs a down-sampling with a stride of (2,2,2). The last layer is composed of a GAP layer followed by a FC layer with as many input as output channels of Conv5_x and a number of output units equal to the number of classes. The FC layer output vector is normalized by a softmax function. Table adapted from [3].

Layers \ **Models**	**ResNet-18**	**ResNet-50**	**ResNeXt-101**
Input Layer	3x16x112x112		
Conv1	Conv(7x7x7)/s(1,2,2), F = 64		
Conv2_x	2 Blocks, F = 64	3 Blocks, F = 256	3 Blocks, F = 256
Conv3_x	2 Blocks, F = 128	4 Blocks, F = 512	24 Blocks, F = 512
Conv4_x	2 Blocks, F = 256	6 Blocks, F = 1024	36 Blocks, F = 1024
Conv5_x	2 Blocks, F = 512	3 Blocks, F = 2048	3 Blocks, F = 2048
Output Layer	GAP − > FC layer − > softmax		