

Human Action Recognition for Intelligent Video Surveillance

by

SAMUEL VIWE MQAQA

Dissertation submitted in fulfilment of the requirements for the degree:

Master of Engineering in Electrical Engineering

Department of Electrical, Electronic and Computer Engineering

Faculty of Engineering, Built Environment and Information Technology

Central University of Technology, Free State

Supervisor: Prof. N.J. Luwes

2021

Declaration

I, Samuel Viwe Mqaqa (Student Number:) do hereby declare that this research project which has been submitted to the Central University of Technology for the degree Master Of Engineering: Electrical Engineering, is my own independent work and complies with the Code of Academic Integrity , as well as other relevant policies, procedures, rules and regulations of the Central University of Technology; and has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualification.



Signature

12 January 2021

Date

Acknowledgements

I would like to acknowledge the following individuals and institute, without whom, the completion of this dissertation would not have been possible:

- Firstly, my creator, God Almighty, for giving me the determination, perseverance, and capability to complete this project and dissertation.
- My mother, Olivia Mqaqa, for raising me to be the best I can be, for all the opportunities she provided me, and her unconditional love and support.
- My study leader, Prof. Nicolaas Luwes, for his guidance, endless knowledge and wisdom, and his friendship during my studies.
- The Central University of Technology and RGEMS Research Unit, for granting me the opportunity to undertake this project, for the monetary assistance, and the knowledge and experience gained during the project.
- The research colleagues and friends at RGEMS who were always willing to assist in bouncing off ideas and recording videos.

Abstract

Crime remains a persistent threat in South Africa. This has significant implications for our ability to function as a country. As a result, there is a dire need for crime prevention strategies and measures that seek to reduce the risk of crimes occurring, and their potential harmful effects on individuals and society. Many local businesses, organisations and homes utilise video surveillance as a measure, as it can capture the crime as it is committed, thus identifying the perpetrators, or at least presenting a few suspects. In current video surveillance systems, there is no software that enables security officers to manage the data collected (i.e. automatically describe activities occurring in the video) and make it easily accessible for query and investigation. Access to the data is difficult because of the nature and size of the data. There is a need for efficiently extracting data to automatically detect, track, and recognise objects of interest, including understanding and analysing data through intelligent video surveillance. The aim of the study is to create an intelligent vision system that can identify a range of human actions in surveillance videos. This would offer security officers additional data of activities occurring in the videos, thus enabling them to access specific incidents faster and provide early detections of crimes. To achieve this, a literature study was done in the research area to reveal the prerequisites for such systems, the separate software modules designed and developed and eventually integrated into the intended system. Tests were developed to validate the system and evaluate how all the modules work together. This inevitably confirms the functionality of the fundamental components and the system in its entirety. The results have indicated that each module in the system operates successfully, can effectively extract pose estimation features, generate features for training/ classification and classify the features using a deep neural network. Further results showed that capability of the system can be applied to intelligent surveillance systems and enable security officers' early detection of abnormal behaviour that can lead to crime.

Table of Contents

Declaration	i
Acknowledgements	ii
Abstract	iii
Table of Contents	iv
List of Figures	ix
List of Tables	xii
Chapter 1 : Introduction	13
1.1 Background of the study	13
1.2 Problem statement.....	15
1.3 Hypothesis	15
1.4 Research, aims and objectives	15
1.5 Layout of the dissertation	16
Chapter 2 : Study Design	19
2.1 Introduction	19
2.2 Study design	19
2.3 Conclusion	22
Chapter 3 : Literature Review	23
3.1 Introduction	23
3.2 Typical video surveillance systems	23
3.3 Intelligent video surveillance system.....	25

3.4	Human action recognition	27
3.4.1	Challenges in human action recognition	28
3.4.2	Global representation.....	30
3.4.3	Local representation	32
3.4.4	Depth-based representation.....	34
3.5	Neural networks	36
3.5.1	Classification.....	37
3.5.2	Clustering.....	38
3.5.3	Regression.....	39
3.5.4	Elements of a neural network.....	39
3.5.5	Convolutional neural networks	41
3.6	Human pose estimation	44
3.6.1	DeepPose	45
3.6.2	Efficient object localisation using convolutional networks	47
3.6.3	Convolutional pose machines	48
3.6.4	Human pose estimation with iterative error feedback.....	49
3.6.5	Stacked hourglass networks for human pose estimation	50
3.6.6	Simple baselines for human pose estimation and tracking	51
3.6.7	Deep high-resolution representation learning for human pose estimation	51
3.6.8	OpenPose.....	52
3.7	Datasets.....	53
3.7.1	Joint-annotated human motion data base	53

3.7.2	North-western-UCLA multiview action 3D dataset	54
3.7.3	AAMAZ Human Action Recognition Dataset	55
3.8	Conclusion	55
Chapter 4 : Video Feature Extraction.....		58
4.1	Introduction	58
4.2	Software development environment.....	59
4.3	Convert videos to images.....	60
4.4	Extracting skeleton data using OpenPose	62
4.5	Preprocessing of raw skeleton data	64
4.5.1	Scale joint co-ordinates.....	65
4.5.2	Removal of head joints.....	65
4.5.3	Discard of invalid frames.....	66
4.5.4	Fill in missing joints	66
4.6	Feature generation.....	67
4.6.1	Normalise joint positions	68
4.6.2	Calculate body velocity	69
4.6.3	Calculate joint velocities.....	69
4.6.4	Principal component analysis (PCA).....	70
4.7	Results of feature extraction	70
4.7.1	Convert videos to images.....	70
4.7.2	Extracting skeleton data using OpenPose	72
4.7.3	Preprocessing Raw Skeleton Data	75

4.7.4	Feature generation.....	75
4.8	Discussion	76
Chapter 5 : Development of a Deep Neural Network for Action Classification		77
5.1	Introduction	77
5.2	Software environment for the development of the Deep Neural Network (DNN)	78
5.3	Training and testing datasets	78
5.4	Development of the neural network	79
5.4.1	Learning.....	80
5.4.2	Parameters	82
5.5	Results.....	83
5.5.1	Confusion matrix.....	83
5.5.2	Accuracy report.....	84
5.6	Discussion	86
Chapter 6 : Evaluation of the Human Action Recognition System		88
6.1	Introduction	88
6.2	Validation	89
6.2.1	Test 1.....	90
6.2.2	Test 2.....	95
6.2.3	Test 3.....	101
6.3	Discussion	107
Chapter 7 : Discussion and Conclusion.....		109
7.1	Introduction	109

7.2	Human action recognition system	110
7.3	Concluding remarks and future work	111

List of Figures

Figure 2.1: Logic diagram of the HARS showing the two major components.....	19
Figure 2.2: Flow Diagram of Study Design.....	20
Figure 3.1 Typical video surveillance system.....	24
Figure 3.2 Ring Doorbell[12].....	25
Figure 3.3: Overview of an intelligent video surveillance system[14]	26
Figure 3.4: RGBD Kinect camera with (A) colour images and depth maps, and (B) skeletal information[43].....	35
Figure 3.5: Typical structure of a neural network	37
Figure 3.6: Example of a network node	40
Figure 3.7: Logic Diagram of a CNN[69]	41
Figure 3.8: RGB Image[70].....	42
Figure 3.9: Logical diagram of convolution layer.....	43
Figure 3.10: Max pooling and average pooling[52]	44
Figure 3.11: Detected skeleton using pose estimation[73].....	45
Figure 3.12: Schematic view of the DNN-based pose regression[74]	46
Figure 3.13: Refining regressor is applied on a sub image to refine a prediction from the previous stage[74].....	46
Figure 3.14: Overview of cascaded architecture[76].....	47
Figure 3.15: Architecture and receptive fields of Convolutional Pose Machines[77]	48
Figure 3.16: Implementation of iterative error feedback[78].....	49
Figure 3.17: Schematic view of the hourglass model[79].....	50
Figure 3.18: Architecture of Simple Baseline[80]	51
Figure 3.19: Architecture of the HRNet[82]	52

Figure 3.20: OpenPose Architecture[83].....	53
Figure 3.21: JHMDB dataset: Sample video frames for each of the 21 action categories[84].....	54
Figure 3.22: Snapshots of frames from a UCLA multiview action 3D dataset	55
Figure 3.23: Snapshots of frames from AAMAZ Human Action Recognition Dataset	55
Figure 4.1: Overall study design	58
Figure 4.2: Logical layout of feature extraction	59
Figure 4.3: Flow chart of converting video to images.....	61
Figure 4.4: Logical design for OpenPose algorithm[89]	63
Figure 4.5: Preprocessing raw of skeleton data	65
Figure 4.6: Example of Frames that have been discarded.....	66
Figure 4.7: Logical design for feature generation.....	68
Figure 4.8: Input Videos from Video Dataset	71
Figure 4.9: Converted JPEG Images	71
Figure 4.10: Skeleton Extraction from an image using OpenPose.....	72
Figure 5.1: Overall Study Design	77
Figure 5.2: Scikit-Learn Logo	78
Figure 5.3: One hidden layer MLP[91]	79
Figure 5.4: Plot of ReLu activation function[92].....	80
Figure 5.5: Confusion Matrix	84
Figure 6.1: Overall study design	88
Figure 6.2: Logical diagram of the HARS system	89
Figure 6.3 Test 1: Jump	90
Figure 6.4 Test 1: Kick.....	91
Figure 6.5 Test 1: Run	92
Figure 6.6 Test 1: Sit.....	93

Figure 6.7 Test 1: Stand	94
Figure 6.8 Test 1: Walk.....	95
Figure 6.9 Test 2: Jump.....	96
Figure 6.10 Test 2: Kick.....	97
Figure 6.11 Test 2: Run	98
Figure 6.12 Test 2: Sit.....	99
Figure 6.13 Test 2: Stand	100
Figure 6.14 Test 2: Walk.....	101
Figure 6.15 Test 3: Jump.....	102
Figure 6.16 Test 3: Kick.....	103
Figure 6.17 Test 3: Run	104
Figure 6.18 Test 3: Sit.....	105
Figure 6.19 Test 3: Stand	106
Figure 6.20 Test 3: Walk.....	107

List of Tables

<i>Table 2.1: Summary of different sub-phases in Phases 3 – 5</i>	21
<i>Table 4.1: Skeleton Joints Label output by OpenPose</i>	64
<i>Table 4.2: Detected Skeleton Joints and Score by OpenPose</i>	73
<i>Table 4.3: Number of skeletons detected in the dataset</i>	74
<i>Table 4.4: Number of Skeletons Detected per Class</i>	74
<i>Table 4.5: Number of samples remaining after the reduced dataset</i>	75
<i>Table 4.6: Number of generated feature after PCA was applied</i>	75
<i>Table 5.1: Dataset Train-Test Split</i>	79
<i>Table 5.2: Parameters used to Initialise MLPClassifier</i>	82
<i>Table 5.3: Accuracy Report</i>	85

Chapter 1: Introduction

1.1 Background of the study

South Africa is a country plagued with high levels of crime. The direct and indirect effects of crime to individuals, families, neighbourhoods, businesses, government and the country include: financial loss, increased fear of victimisation, restricted behaviour and movement, a breakdown of trust relationships, an untold short-term and long-term trauma, and potentially lasting physical and psychological consequences [1][2]. According to the Governance's Public Safety and Justice Survey 2018/19, 1,3 million incidences of housebreaking affect 5,8% of households in South Africa [3]. These housebreakings are increasing annually and as a result, gated communities have become increasingly popular within the South African middle class. These gated communities are protected by high perimeter walls topped with electric fencing, by barred doors and windows with security officers who patrol the perimeter [4]. Citizens who do not reside in gated communities, rely on private security companies to protect them and their assets. These security companies make use of video surveillance systems which use strategically placed cameras to capture and store videos surveillance. All these measures are put in place as an attempt to prevent crime.

Crime prevention comprises strategies and measures that seek to reduce the risk of crimes occurring, and their potential harmful effects on individuals and society. This is achieved by intervening to influence their multiple causes [5]. It therefore is essential to find effective ways of reducing incidents of crime and violence, and to limit the negative effects and the destructive impact thereof. As mentioned previously one of these measures is video surveillance. Video surveillance is an effective measure, as it can capture the crime as it is committed, thus identifying the perpetrators or at least present possible suspects.

Video surveillance systems have played, and continue to play, a vital role in security for our society. Closed-circuit television (CCTV)-based video surveillance systems are the most popular across many enterprises, e.g. local shopping centres, malls, banks, businesses, parking lots, academic institutions and homes. A CCTV system consists of a set of cameras placed in strategic locations that capture and transmit images to a video management system. The video management system allows, inter alia, the viewing and recording of the images [6]. The CCTV systems usually include CCTV cameras, and a video recorder with local and remote monitoring. They record and centrally store hours of footage every day. These systems are usually monitored by human security officers. When there is any incidence there is multiple angles of footage to record the incident, culminating in large data storage. Security officers must scroll through hours and days of footage, and this results in a long and tedious process.

In current video surveillance systems, there is no software that enables security officers to manage the collected data in order to make it easily accessible for query and search. There are currently no methods that automatically define activities occurring in the video. Access to the data is difficult due to the nature of the data and its size. There is a need for efficiently extracting data to automatically detect, track and recognise objects of interest that include understanding and analysing it through intelligent video surveillance [7]. Most surveillance systems usually monitor human actions; a human action recognition system would therefore assist security officers by reporting different actions performed by humans in the videos.

Human action recognition aims at automatically detecting the action of a person, i.e. to identify if someone is walking, dancing or performing other types of activities [8]. The term human action, studied in computer vision research, ranges from limb movement to joint complex movement of multiple limbs and the human body. This process of recognising human actions is dynamic, and is thus usually conveyed in a video lasting a few seconds [9]. For the purpose of security, human action recognition

systems would assist security officers in early detection of abnormal behaviour that can lead to crime, because areas under surveillance usually allow certain human actions, while other actions are not allowed.

1.2 Problem statement

CCTV systems offer innovative solutions to security, as they can capture and store video footage through a video management system. These stored videos can later be viewed by security officers whenever an incident occurs to verify the incidence. This process is usually tedious, as there are days of footage that one needs to go through. Accessing specific videos therefore is difficult. These systems are more of a post measure and mainly used for gathering evidence. This demonstrates the need for intelligent video surveillance that can identify different human actions that can be used as an active measure for preventing crime.

1.3 Hypothesis

Human security officers can be more effective in crime prevention through early detection of abnormal behaviour that can lead to crime by utilising a vision system to intelligently detect human actions in video surveillance.

1.4 Research, aims and objectives

The aim of the study, therefore, is to create an intelligent vision system that can identify a range of human actions within surveillance videos. This would offer security officers additional data of activities occurring in the videos, thus enabling them to access specific incidents faster and provide early detections of crimes.

To reach the aim of the study, the following objectives were devised:

1. Gathering training data from existing video datasets
2. Extracting images from the videos
3. Extracting features from the images to train a neural network
4. Training a neural network to classify human actions
5. Testing neural network from known data
6. Validating neural network from real world data.

1.5 Layout of the dissertation

This dissertation has been prepared in six chapters. Following, is a brief description of the contents of each chapter.

Chapter 1: Introduction

Chapter 1 contains a brief introduction to the study, including the problem statement, as well as the aim and objectives of the study.

Chapter 2: Study design

In Chapter 2 the study design and a description of the different aspects of this research project are presented.

Chapter 3: Literature Review

In Chapter 3 a review of relevant literature is presented on typical video surveillance systems used in our modern day and age. The study further reviews literature on pose estimation, video feature extraction techniques and deep neural networks. Action Video Dataset is also reviewed in this chapter.

Chapter 4: Video Feature Extraction

In Chapter 4 video feature extraction steps implemented as part of the Human Action Recognition System (HARS) are presented. This phase was divided into four sub-phases:

- converting video to images.
- extracting skeleton data using pose estimation.
- pre-processing raw skeleton data; and
- feature extraction.

The methods and results for the video feature extraction process as well as the four sub-phases are presented in this chapter.

Chapter 5: Development of a DEEP NEURAL NETWORK (DNN) for Action Classification

This chapter presents the development of the deep neural network and the steps that were implemented as part of the HARS. This phase is divided into two sub-phases:

- Training; and
- Testing.

The methods and results for development of the DNN for Action Classification process, as well as the two sub-phases are presented in this chapter.

Chapter 6: Evaluation of the Human Action Recognition System

This chapter presents the evaluation of the HARS. This phase has one sub-phases:

- Validation.

The methods and results for the evaluation process, as well as the sub-phase, are presented in this chapter.

Chapter 7: Discussion and Conclusion

In this concluding chapter the key findings of this study are presented. It is shown how these findings are integrated into existing knowledge. It furthermore contains a discussion on the challenges and further development of the HARS.

Chapter 2: Study Design

2.1 Introduction

The purpose of the study is to produce a Human Action Recognition System (HARS) that detects human actions and analyses their activities from video surveillance. There are two main components that were needed to create a HARS. The first component that was created, was a video feature extractor to acquire a video as an input, convert it to images, extract skeleton data, preprocess the raw skeleton data and, lastly, generate features for training and classification. The second component that needed to be created, was a Deep Neural Network (DNN). The DNN is trained using the generated features from the video feature extractor. Figure 2.1 shows the logic design of the HARS and its two main components.

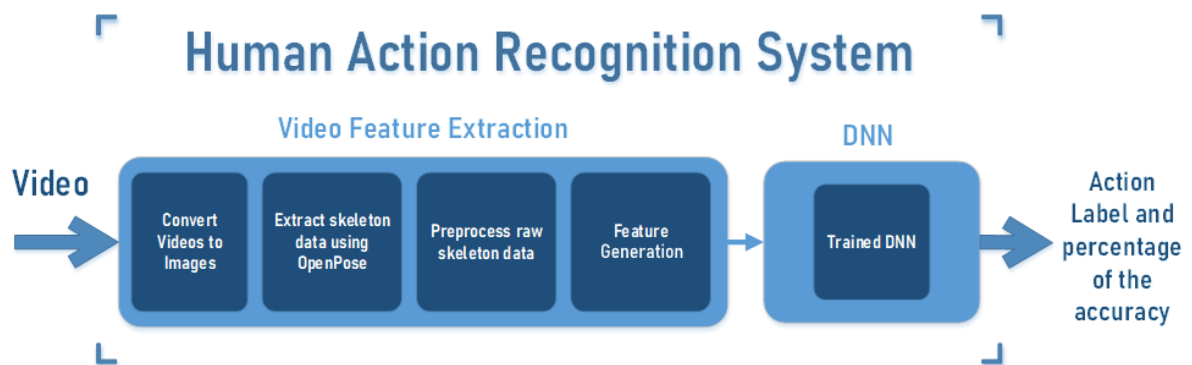


Figure 2.1: Logic diagram of the HARS showing the two major components

2.2 Study design

In order to make the design of the HARS more comprehensible, this research project was broken down into five phases. The first phase comprises of an extensive review of the literature. Literature was obtained on various topics, such as intelligent surveillance, human action recognition, pose estimation, feature extraction and deep neural networks. In the second phase, training data was

collected and reviewed. This phase determines how the neural network performs in terms of floor and ceiling potential. In the third phase features are extracted from the videos collected from various dataset and prepared for training. In Phase 4, the neural network is developed, trained, and tested, using extracted features that were generated in phase three. In the final phase, Phase 5, the completed HARS was tested, using new real-world data that was captured. This was performed to validate the results from the training and testing. The flow diagram of the overall study design is shown in Figure 2.2.

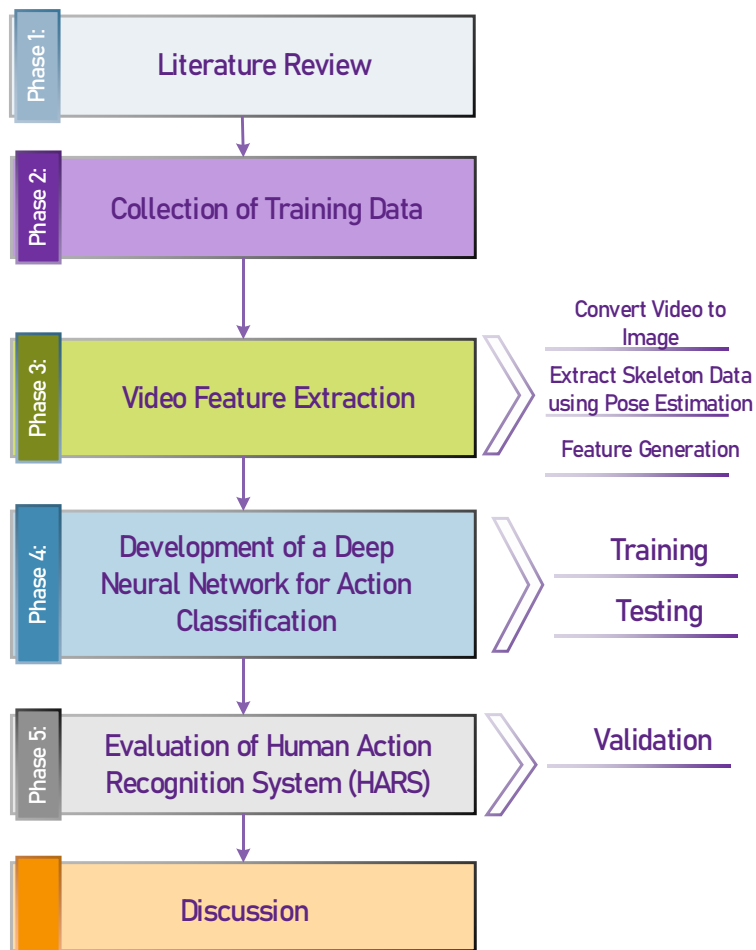


Figure 2.2: Flow Diagram of Study Design

Phases 3, 4 and 5 are divided into six sub-phases. Phase 3 has three sub-phases that prepare input video data for training and testing in Phase 4. Phase 4 is divided into two sub-phases where data extracted in Phase 3, is split, and used for training and for testing the performance of the deep neural

network. In Phase 5 the HARS is evaluated with new data to validate its performance in Phase 4.

Table 2.1 provides a summary of the sub-phases in Phases 3, 4 and 5.

Table 2.1: Summary of different sub-phases in Phases 3 – 5

Phase 3: Video Feature Extraction	
Sub-Phase	Description
Converting Video to Images	The aim of this sub-phase is to take a video as an input and convert it to image frames, as videos are not usable in their raw format. In order to achieve this, the videos, typically .avi video format, are looped through frame by frame, and each frame is stored as a JPEG image.
Extracting Skeleton Data using Pose Estimation	The aim of this sub-phase is to detect and extract skeleton data from the image frames. This was achieved by using pose estimation. The pose estimation algorithm used, is called OpenPose. OpenPose uses a Convolutional Neural Network to detect human joints in images of people.
Pre-processing Raw Skeleton Data	The aim of this sub-phase is to preprocess the raw skeleton data generated by OpenPose in preparation for feature generation. To meet the aim, the skeleton data is scaled, the head joints are removed, invalid frames are discarded, and missing joints are generated and filled.
Feature Generation	The aim of this sub-phase is to reduce the dimensions and extract prominent features in the data. Preprocessed skeleton features are further processed, creating new features from one or multiple existing features. This is achieved by taking the feature vectors and applying the principal component analysis (PCA) technique.
Phase 4: Development of DNN for Action Classification	
Sub-Phase	Description
Training	The aim of this sub-phase is to train the deep neural network. To achieve 70% of the dataset used for training. The trained neural network is used to classify different human actions in videos.

Testing	The aim of this sub-phase is to test the performance of the trained neural network. To achieve this 30% of the dataset is used to test the performance.
Phase 5: Evaluation of the HARS	
Sub-Phase	Description
Validation	The aim of this sub-phase is to test and verify the performance of the HARS. In order to achieve this, a separate set of data that was not used in the training and testing phase, was utilised.

2.3 Conclusion

The review of relevant literature and the collection and review of training data (Phases 1 and 2) is presented in Chapter 3. The methods and results of the video feature extraction (Phase 3) are discussed in Chapter 4 and the methods and results of the development of the deep neural network (Phase 4) are discussed in Chapter 5. The evaluation of the HARS (Phase 5) are presented and discussed in Chapter 6. Chapter 7 is an overall discussion of all the results presented.

Chapter 3: Literature Review

3.1 Introduction

This chapter consists of a literature study done to acquire preliminary knowledge in the field of surveillance systems, human action recognition, neural networks and pose estimation. The study focuses on describing the most popular and relevant research techniques related to this research project.

3.2 Typical video surveillance systems

Video surveillance systems have played, and continue to play, a vital role in security within our society. A typical surveillance system will be examined with regard to what it consists of and how it operates. Closed-circuit television (CCTV)-based video surveillance systems are very popular across a wide range of enterprises, e.g. local shopping centres, malls, banks, businesses, parking lots, academic institutions, homes, etc. A CCTV system simply consists of a set of cameras placed in strategic places that capture and transmit images to a video management system. The video management system allows, inter alia, the viewing and recording of the images. CCTV systems usually include CCTV cameras, video recorders, and local and remote monitors, as seen in Figure 3.1. They daily record and centrally store hours of footage. These systems are usually observed by human security officers. An analogue CCTV system is set up to send its signal to a digital video recorder (DVR) through a Bayonet Neill–Concelman (BNC) cable. The footage recorded by the DVR is stored in hard disk drives. The DVR also has a video output that connects to a screen which allows for central monitoring, as seen in Figure 3.1. An advanced feature of the DVR is remote video live streaming and playback.



Figure 3.1 Typical video surveillance system

Figure 3.1 shows all the components needed in a typical video surveillance system and how they interconnect. Analogue cameras are rare to find in our day and age, though they still exist. Most systems use internet protocol (IP) security cameras. An IP security camera sends its signal through the network. It uses a Cat 5 or Cat 6 cable to send a signal to a network video recorder (NVR). The NVR has a similar feature with the DVR. IP CCTV Systems have more advanced features compared to analogue CCTV systems. They are able to support higher resolution, produce more advanced video codec, and more secured video and audio transmission. The security systems mentioned above only record and store footage. Some systems use motion detection algorithms to trigger recording to save storage. When there is any incidence or when footage is required for any reason, security personnel must scroll through hours and days of footage. This a long and tedious process, as currently there are no products available to assist in searching through footage visually[10].

Smart doorbells allow homeowners to receive notifications when a visitor arrives at the door, see who the visitor is, and contact with them all from the convenience of their own home. They have a significant impact on people's quality of life and contribute to the creation of smart homes[11]. These systems are use technology like motion detection to trigger recording of objects moving at your front door. They also can alert the owner on their smartphone when someone is at the door.



Figure 3.2 Ring Doorbell[12]

Figure 3.2 shows an example of the doorbell device which was developed by Ring company. There are two types of doorbells: wired doorbells and wireless doorbells, which are differentiated by the need for wall wiring. The former requires a wire to link both the front and back door buttons to a transformer, whilst the later uses telephone technology to transmit the signal wirelessly. Wireless doorbell systems, which use radio technology to signal doorbells and answer doors remotely, are common in modern structures[11].

3.3 Intelligent video surveillance system

The aim of an intelligent video surveillance system is to detect an interesting event efficiently from a large number of videos in order to avoid dangerous situations. Intelligent surveillance systems require much fewer human operators because of the automated services they provide, such as intrusion detection, robbery prevention, people counting, and loitering detection [13]. In order to achieve such

a system, it requires two video processing levels, as illustrated in Figure 3.3. On the first level there are two steps, low-level features and the primitives based on low-level features. The low-level features aim to detect the regions of interest in the scene and extract them. The primitives based on low-level features are then generated to describe the region of interest. On the second level the semantic information about the human action is detected, and it is determined whether the behaviour is normal or not[14].

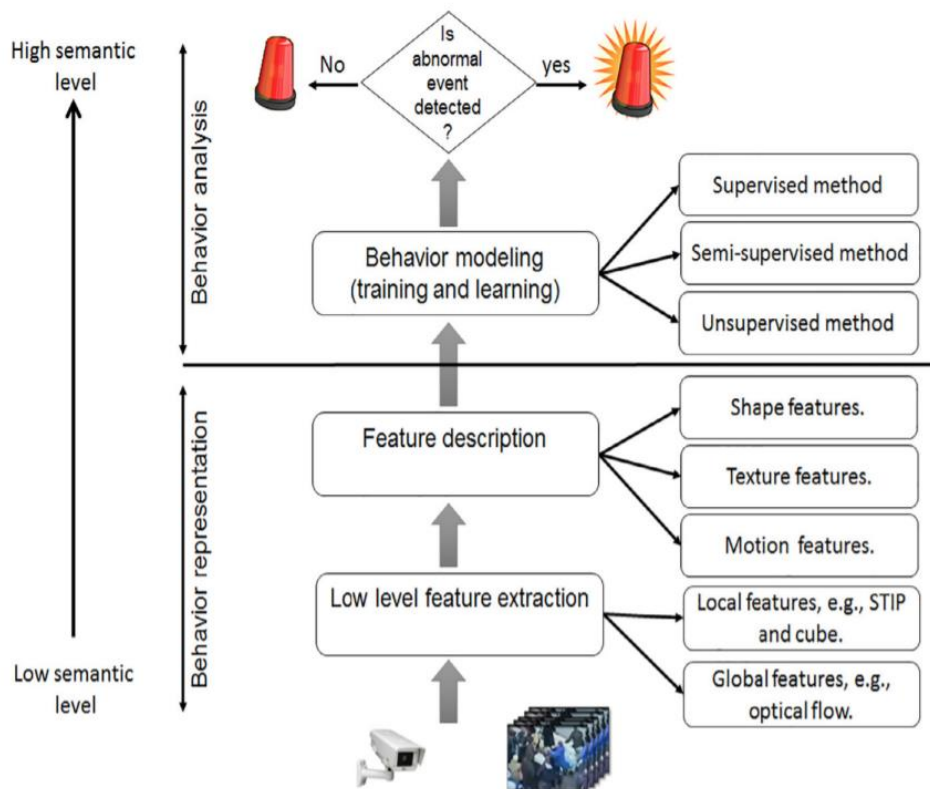


Figure 3.3: Overview of an intelligent video surveillance system[14]

Figure 3.3 shows an overview of an intelligent video surveillance system proposed by Mabrouk and Zagrouba[14], and all the components it comprises. Many have shown concern in using an intelligent surveillance system, because it will replace human security officers. On the contrary, such a system will not replace human security officers but rather equip them with knowledge. For example, instead of a security agent constantly monitoring screens with live video surveillance feeds – a job in

which humans do not display high performance due to the lack of significant events for most of the time – an automated system might filter the videos and indicate that only certain video segments are more likely to contain relevant activities, such as suspicious behaviour[15].

3.4 Human action recognition

Human action recognition (HAR) aims to automatically identify and recognise the state of an action from unknown video sequences. There is a growing demand for automatic interpretation of human behaviour, and HAR has caught the attention in both academia and industry. Analysing and understanding a person's behaviour is fundamentally required for a wide range of applications, such as video indexing, biometrics, surveillance and security[16]. Most often the motions of the different human body parts are part of functional movements without showing intentions and thoughts. Human activities can be grouped in four main categories, depending on body parts engaged in the action and its complexity. The groups are as follows.[17]:

- **Gesture.** It is an elementary movement of a person's body part. It is visible bodily action representing a specific message. It is a type of non-verbal communication that utilises a movement made with the hands, face or other parts of the body, such as okay gestures and thumbs up.
- **Action.** It is a set of physical movements conducted by only one person. Actions can be composed of multiple gestures organised temporally, like walking and running.
- **Interactions.** It is a set of actions that are performed by at most two actors. At least one subject is a person and the other one can be a human or an object (hand-shaking, chatting, etc).
- **Group activities.** It is a mixture of gestures, actions, or interactions. The number of performers is at least two plus one or more interactive objects (playing volleyball, obstacle racing, etc).

3.4.1 Challenges in human action recognition

In this section some of the challenges in HAR are reviewed. The challenges can dramatically downgrade the system performance. In an HAR system extracted features need to generalise several variations that may include human appearance, points of view and backgrounds, to overcome the challenges encountered in action recognition.

- **Anthropometric variation.** The study of anthropometrics concerns the physical sizes and shapes of humans. A person's age may in many cases influence the body's flexibility or the position in an observed scene, humans exhibit size variabilities. Human movements are quite complex and present infinite variability; the slightest wink and movement can give meaning which is dependent on the context[18]. When designing an HAR system, one needs to factor in the variation in anthropometry.
- **Clutter and camera motion.** Clutter is the main cause background noise. Most existing action features, such as histograms of oriented gradient [19] and interest points [20], also encode background noise which results in reduced action recognition performance. Camera movement is also another factor that should be considered in real-world applications of HAR. If there is significant camera motion, action features will not be accurately extracted. In order to solve this challenge and better extract action features, camera motion should be modelled and compensated [21].
- **Intra-class variability and interclass similarity.** People behave differently for the same actions. For a given semantic meaningful action, for example, 'running', a person can run fast, slow or even jump and run. One action category may contain multiple different styles of human movements. In addition, videos of the same action can be captured from various points. They can be taken in front of the human subject, on the side of the subject, or even on top of the subject, showing appearance variations in different views. People may also show different

poses in executing the same action. The factors mentioned, result in large intra-class appearance and pose variations, which may cause confusion to many existing action recognition algorithms[9].

- **Low-quality videos.** Most videos are acquired in public spaces through poor quality IP cameras and CCTV cameras. These cameras display many disadvantages such as slow frame rates, motion blurring, low resolution, and compression artefacts. This results in poor performing HAR systems.
- **Occlusion.** Occlusion is defined as when two or more objects come too close and seemingly merge or combine with each other. Image processing systems with object tracking often wrongly track the occluded objects[22]. In some cases, after occlusion the system will wrongly identify the initially tracked object as a new object. There are three main types of occlusion, self-occlusion, crowd issue and occlusion created by an object. Self-occlusion occurs when a body part has been obscured by another body part. For example, the action of 'talking' cannot be recognised when a person puts his hand in front of his mouth. Crowd issue occurs when two or more people are hiding each other. Occlusion created by an object occurs when from one point of view some body parts are occluded with an object. In this case, recognising the action 'texting', from the front view, is a difficult task when the subject is behind some object[16].
- **Illumination variation and shadow.** Lighting plays a vital role in HAR and can result in a considerable difference in quality of the human action representation. Different kinds of lighting conditions on the same person may yield different results which may cause the human action to appear differently. This dramatically affects HAR system performance. It is more evident when the light source is blocked, and a dark area appears. This area is called a shadow. The shadow shape is a two-dimensional projection of the person who is blocking the

light. The human action is partially duplicated at different scales according to the position of the light source relative to the person[16].

- **Insufficient data.** In order to design efficient techniques, datasets should contain good scale ranges, occlusion, intra and inter class variations, etc. However, most of human action datasets contain a limited number of labelled videos. Datasets recorded in an unconstrained environment were introduced. These datasets consist of labelled videos collected from web videos or movies[19]. The main issue with HAR datasets is the limited number of labelled training and test sequences. The task of annotating a large dataset is challenging and time consuming. Many solutions to handle the lack of appropriate datasets have been proposed. Researchers have made use of web video search results[23], video subtitles and movie script matching[24].
- **Poor weather conditions.** Poor weather conditions pose many challenges to mechanisms for identifying HAR. For example, darkness, rain, blowing snow and fog affect the visibility and pose a challenge to identification of actions. Many parameters are changed in poor weather conditions, colours are extremely affected and distances are difficult to evaluated[16].

3.4.2 Global representation

Global representations explicitly extract global descriptors from original videos or images and encode them as a whole. The human subject is located and isolated in this representation using methods of background subtraction that form the silhouettes or shapes which become the region of interest (ROI). In other global methods the ROI is encoded as descriptors from which they derive corners, edges or optical flow[25]. Many methods of global representation, based on silhouettes, stack the silhouette image along the time axis to shape the volumes of space-time in three dimensions. Then only the volumes are used for representation. In earlier works, solutions to global representation were mainly suggested and eventually discarded due to the sensitivity to noise, occlusions, and change of

viewpoint[25]. This study looks at three approaches to global representation, namely silhouettes, optical flow and 3D space-time volumes.

- **Silhouettes and shapes.** In order to recognise the human action in videos, an instinctive approach is to isolate the human from the background. This approach is called background subtraction or foreground extraction. The extracted foreground in the HAR is called silhouette, which is the ROI and represented as a whole object in the global representation approach. One essential step is to calculate the background model before having silhouettes extracted. Wren, *et al.* [26] first suggested modelling Gaussian distribution of the background scene. Koller, *et al.* [27] pointed out that some foreground values were unduly updated and thus introduced the selective update strategy for the background. Stauffer and Grimson [28] suggested modelling the values of a single background pixel as a mixture of Gaussians to replace the technique of using just one Gaussian value in the approach before. The Gaussian mixture model (GMM) was commonly used, but the implementation of an expectation-maximization algorithm (EM) increased the cost of computation. In order to reduce the cost, the K-means clustering algorithm was used to replace the EM algorithm with an insignificant loss of precision. This was worth it, seeing that current RGBD cameras make the profile easy to obtain by using the depth data given by the depth sensors.
- **Optical flow.** Optical flow is an efficient way of extracting silhouettes and defining them for a dynamic background. The optical flow may be obtained with the Lucas Kanade-Tomasi (LKT) feature tracker[29]. Lu, *et al.*[30] used a tracker approach based on the LKT feature to track joints in key frames and actual frames. Each activity is represented as a sequence of postures, and each key posture is recorded within a key frame. Specific postures can be recognised in actual frames by finding correspondence between the real frame and the key frame. By mapping body locations, the recognised posture from the actual frame is compared

to the key posture frame, and the corresponding posture sequences are confirmed as the activity.

- **3D space-time volumes (STVs).** A video of an action can be seen as a collection of images containing sequences of action. Concatenating all frames along the time axis, forms the three-dimensional STV, which includes two spatial dimensions X and Y, and one temporal dimension, T. STV-based representations aim to capture the additional complex information that the methods of spatial representation cannot obtain because of the lack of a time dimension. Building STVs for various activities is a method of global representation. But often the STV blends local features to create the final feature sets [25]. The space-time form was first introduced by Blank, *et al.* to represent human actions. Space-time shape is obtained by stacking only the regions of the silhouette within the images. However, conventional 3D structure analysis cannot be extended to space-time activity shapes due to the non-rigidity of the constructed 3D space-time shapes, and the inherent disparity between space and time dimensions. Therefore, the Poisson equation approach is used to derive local space-time saliency and orientation features [31].

3.4.3 Local representation

Local representations process activity videos as a collection of local descriptors rather than extract the silhouette or STV and encoding them. Instead, they focus on specific local patches determined by point of interest detectors or dense sampling[32]. Many of the current local features have been proven to be resilient against noise and partial occlusions, compared to global features. Local features are usually combined with the bag of visual words (BoVW) model to generate a general pipeline of existing state-of-the-art local representation approaches[33]. BoVW-based local representation mainly consists of four steps: extraction of features, generation of codebooks, encoding of features, and pooling and normalisation. The conventional BoVW pipeline is as follows [34]: interest points and

local patches are first obtained by detectors or densely sampled. Local features are then extracted from certain points of interest or patches. A visual dictionary (i.e. a codebook) is then studied by K-means or the Gaussian mixture model (GMM) in the training set. The original high-dimensional descriptors are clustered, and the middle of each cluster is called a visual codeword. Following this, local features will be encoded and pooled. Lastly, the pooled vectors are normalised as a video representation. Two local feature extraction methods, namely Spatiotemporal Interest Point Detector and Local Descriptors, will be reviewed.

- **Spatio-temporal interest point detector.** The basic principle of local representation is to define interest points that provide high-quality information in pictures or videos. Harris and Stephens [35] first suggested a powerful two-dimensional interest point detector, the well-known Harris corner detector which is commonly used in the detection of objects. Laptev and Lindeberg [36] then suggested 3D space-time interest points (STIPs) by expanding on the Harris detector. Spatial points of interest in images are extended to spatial-temporal local structures in videos where image values have significant local variations in both space and time. Spatio-temporal extents of the observed points are calculated by maximising the normalisation of the Laplacian space-temporal operator over spatial and temporal scales. Although these methods have achieved remarkable results in the HAR, a common deficiency is the lack of stable interest points. In addition, it is difficult to regulate the trade-off between the stability of those points and the number of points found. In an effort to detect interest points in an efficient manner, Willems, *et al.* [37] presented a dense, scale-invariant, yet efficient, spatio-temporal interest point detector with minimal effect on computing time.
- **Local descriptors.** Local descriptors are designed to identify areas that have been sampled either densely or at points of interest. Effective descriptors will be considered as discriminative for intended human action incidents in videos, and robust to occlusion, vibration and background noise[25]. Dalal and Triggs [38] proposed a histogram of orientated gradients

(HOG) and proved to be successful in human detection with a linear SVM classifier. Good performance is attributed to the fact that the HOG density distribution of local intensity gradients or edge directions can well describe the local object appearance and shape of the target objects. Lu, *et al.* [39] proposed the PCA-HOG descriptor, which projects the original HOG as a linear subspace through principal component analysis (PCA). The descriptor was used to represent athletes to solve the problem of tracking and action recognition at the same time. Using HOG and histogram of flow (HOF) descriptors. Laptev, *et al.* [40] completes a similar but more difficult activity identification task, as these actions are derived from movies. Klaser, *et al.* [32] applied the HOG descriptor to video sequences and suggested the HOG3D. Integral images are extended to integrated videos for efficient 3D gradient computing. Polyhedrons are used for orientation quantisation as an analogue of 2D space HOG polygons. Optimised parameters for recognition of actions were also explored in their research.

3.4.4 Depth-based representation

Previous HAR research focuses mainly on video sequences captured by traditional RGB cameras. Nevertheless, due to their high cost and complexity of operation, depth cameras have been limited[41]. Over the years the development of low-cost depth sensors, such as Microsoft Kinect, is an affordable and easy way to access depth maps is provided. In addition, Kinect SDK has made it possible to directly obtain skeletal joint positions in real-time[42].

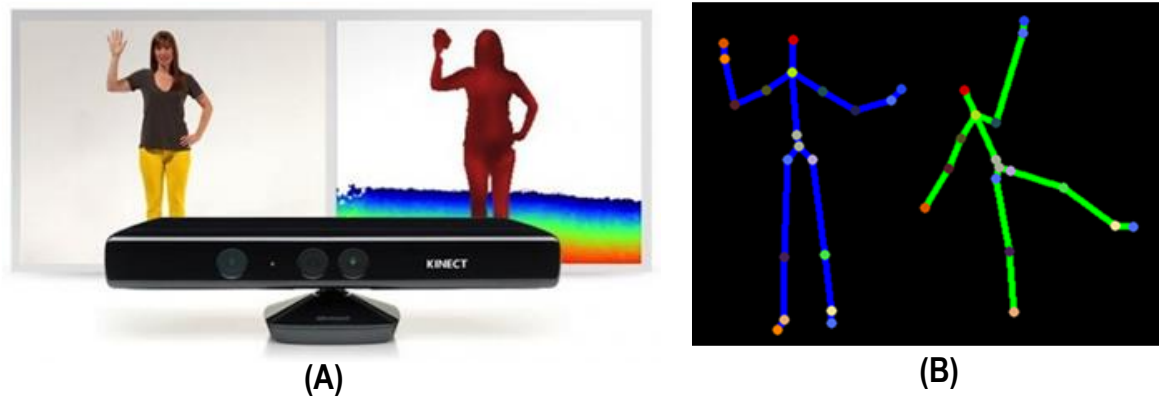


Figure 3.4: RGBD Kinect camera with (A) colour images and depth maps, and (B) skeletal information[43]

In Figure 3.4 is an RGB (red, green and blue) image and a depth image from a Kinect camera. Based on the depth data, it can locate skeleton data on the right. The accessible depth maps and skeletal information contributed actively to the research of computer vision[44]. These two features, and their derivative features, have sparked a wide interest in solving HAR issues using depth-based approaches replacing traditional RGB-based methods, or serving as supplements to improve RGB-based methods. Recent developments of action representations using depth maps or skeletons will be reviewed[45].

- Depth Maps-based representation.** Depth maps contain additional distance co-ordinates relative to colour images which only describe the colour information. Zhao, *et al.* [45] proposed a system for the combining of RGB and depth-map features for HAR, and introduced an optimal scheme. Spatio-temporal interest points are created solely from RGB channels, and HOG and HOF are determined to form RGB-based descriptors. For the depth channel, a Depth Map-based Descriptor called the Local Depth Pattern (LDP) was proposed, which simply calculates the difference in the average depth values between a pair of cells in the surrounding STIP area. Yang, *et al.* [46] proposed the use of HOG on depth maps. Depth maps are projected on three orthogonal planes, and depth motion maps (DMMs) are generated through the

accumulation of global activities throughout entire video sequences. HOG is then computed from a DMM as an action video representation. Jalal, *et al.* [47] considered multi-feature depth images, extracting three-dimensional silhouettes, as well as spatial-temporal joint values for their streamlined and appropriate information for the HAR feature.

- ***Skeleton-based representations.*** Skeletons and joint positions are features typically generated based on depth maps. The Kinect system is common in this representation because of its ability to obtain skeleton and joints. The Kinect SDK software creates 20 joints, while the later versions generate 25 joints, adding 5 joints across the hands and back. There are other methods used to extract skeleton features without the use of a depth camera. This approach uses RGB images (see Section 3.6). The Kinect at times produces inaccurate skeletons, and joints may be completely wrong. Current approaches often solve this, by combining other features that are robust for occlusion, or alleviating the problem of occlusion by dividing the entire skeleton into different parts of the body and handling them independently, since not all parts of the body are occluded[25]. Not all skeleton joints are involved in a particular action, and only a few active joints are relevant and useful for a particular activity[48]. Concentrating on these active joints and abandoning the other inactive parts, will generate more robust features and will be beneficial in dealing with intraclass variations[49]. Finally, as an extracted feature from the depth maps themselves, skeleton-based representation is often merged with the original depth information to provide more insightful and robust representation[49], [50].

3.5 Neural networks

A neural network is a series of algorithms that aim to recognise underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, albeit organic or artificial. Neural networks can adapt to changing input;

the network generates the best possible result without needing to redesign the output criteria. Neural networks enable us cluster and classify data. They can group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on[51].

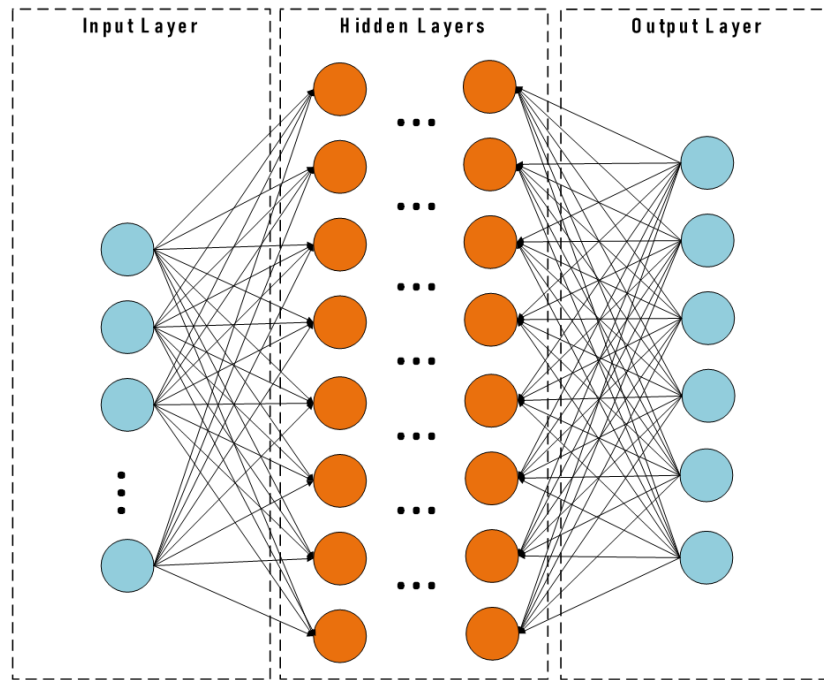


Figure 3.5: Typical structure of a neural network

Figure 3.5 shows the neural network input layers on the left, hidden layers in the middle and output layers on the right. Neural networks also have the ability to extract features that are fed to other algorithms for clustering and classification. Deep neural networks can therefore be thought of as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression[51].

3.5.1 Classification

Classification entails predicting to which class an item belongs. Most classifiers are binary, resulting in a conclusion yes/no, while others are multi-class and can classify an item into one of many categories. All classification tasks rely on labeled data sets; that is, for a neural network to learn the

association between labels and data, humans must pass their information to the dataset[52]. This is called supervised learning. There are several applications of classification algorithms:

- classifying text as spam (in e-mails)[53],[54] or fraudulent (in insurance claims), and recognising sentiment in text (customer feedback);
- detecting faces, identifying people in images, recognising facial expressions (angry, joyful, sad)[55], [56];
- identifying objects in images (stop signs[57], pedestrians[58], [59], lane markers[60], animals[61]);
- recognising gestures in videos[62]; and
- detecting voices, identifying speakers, transcribing speech to text and recognising sentiment in voices[63].

Classification algorithms can be utilised on data with labels that humans can generate. Any outcomes that one cares about, and which correlate with the data, can be used to train a neural network.

3.5.2 Clustering

Clustering is the organisation of unlabelled data into similarity groups, called clusters. A cluster is a collection of data items which are similar to one another and dissimilar to data items in other clusters. Clustering algorithms do not require labels to detect similarities. Learning without labels is called unsupervised learning. Unlabeled data represents most data in the world. One of the laws of machine learning is: the more data an algorithm can train on, the more accurate it will be. Therefore, unsupervised learning has the potential to produce highly accurate models[64]. There are many applications of clustering algorithms namely:

- **Search.** Comparing documents, images, or sounds to surface similar items[65].

- **Anomaly detection.** The opposite of detecting similarities, is detecting anomalies or unusual behaviour. In many cases unusual behaviour correlates highly with things you want to detect and prevent, such as fraud[66].

3.5.3 Regression

A set of statistical techniques for evaluating the associations between an outcome variable and one or more features is known as regression analysis. The most popular type of regression analysis is linear regression, which involves determining which line or more complex linear combination best fits the data according to a set of mathematical criteria. Regression models can also create correlations between present events and future events by the same token that it is exposed to sufficient appropriate data. It can run regression between past and future. In a way the future occurrence will be like the label[67]. Machine learning does not inherently involve time, or the fact that something is still to happen. Given a time series ,deep learning can read a string of numbers and predict the number most likely to occur next[68]. There are many applications of regression algorithms, namely:

- predicting hardware breakdowns (data centres, manufacturing, transport);
- predicting health breakdowns (strokes, heart attacks based on vital stats and data from wearables);
- predicting customer behaviour (the likelihood that a customer will leave, based on web activity and metadata); and
- predicting employee turnover (the likelihood that an employee will leave, based on web activity and metadata).

3.5.4 Elements of a neural network

A deep neural network or deep learning is used for networks that are stacked, i.e. networks composed of multiple layers. These layers consist of nodes, a node being a place where computation takes place.

This can be compared to a neuron in the human brain, which fires when it encounters enough stimuli. A node combines data input with a set of coefficients or weights that either intensify or dampen the input, thereby assigning value to inputs with regard to the function the algorithm is attempting to learn[51].

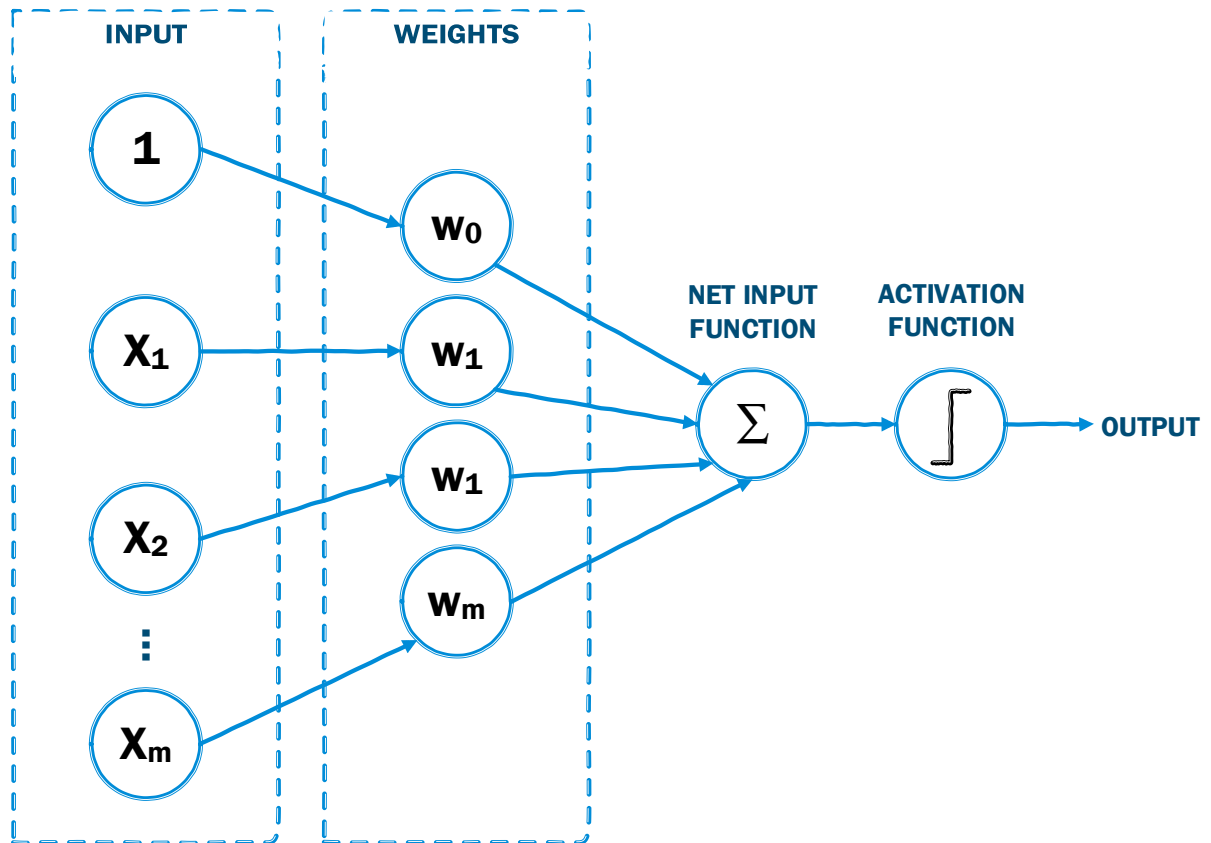


Figure 3.6: Example of a network node

Figure 3.6 shows a network node which consists of the input on the left, the weights, the net function, activation function and, lastly, the output. Such input-weight products are added up and then the total is passed through the so-called activation function of a node, as seen in Figure 3.6, to decide whether and to what degree the signal will move further through the network to influence, e.g. an act of classification, the ultimate result. If the signals pass, the neuron is activated. The net input function calculates the layer's net input by combining its weighted inputs and biases. The activation function

[51]. In a neural network, the activation function describes how the weighted sum of the input is turned into an output from a node or nodes in a layer.

3.5.5 Convolutional neural networks

A convolutional neural network (CNN) in the field of deep learning is an algorithm which takes in an input image, assigns importance (learnable weights and biases) to various aspects/objects in the image and is able to differentiate one from the other. The preprocessing required in a CNN is much lower than other classification algorithms. A CNN can learn these filters/characteristics with enough training, while in primitive methods filters are hand-engineered.

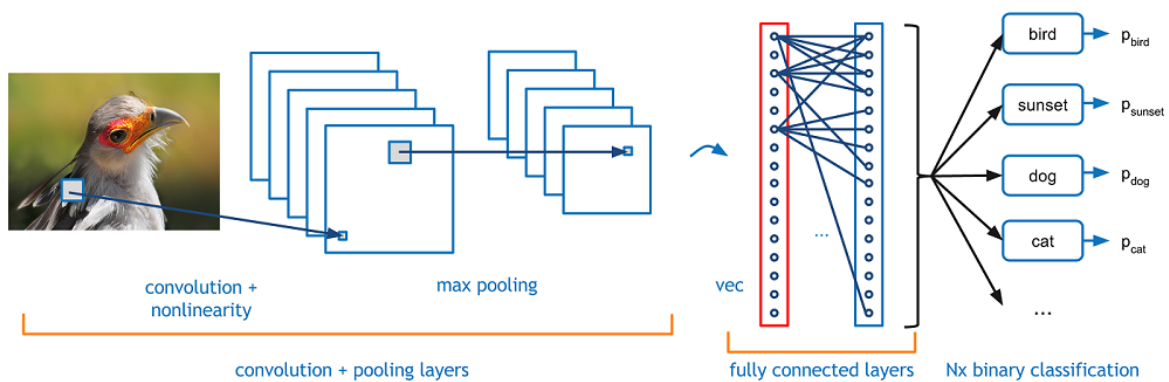


Figure 3.7: Logic Diagram of a CNN[69]

In Figure 3.7 is the logic diagram of a CNN which illustrates the input, convolution layers, pooling layers and the fully connected layer. CNNs can successfully capture the spatial and temporal features in an image through the application of appropriate filters. The architecture performs a better fitting to the image dataset, due to the reduction in the number of parameters involved and reusability of weights. This means the network can be trained to understand the sophistication of the image better. There are many elements that make up a CNN. All these elements play a vital role and affect the performance of the network. The elements are as follows:

- Input Image.** The input image of a CNN is usually an RGB (Red, Green and Blue) image. An RGB image is a colour scheme that has been separated into three colour planes, as seen in Figure 3.8. The input image is a 4x4x3 RGB Image. There are other colour spaces which can be used for an image, e.g. Grayscale, RGB, HSV, CMYK, etc. Processing high-resolution images is computationally intensive.

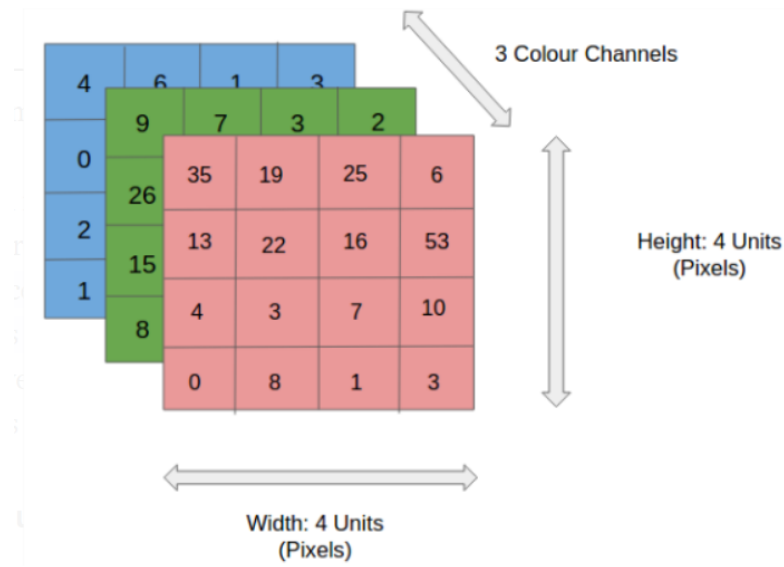


Figure 3.8: RGB Image[70]

Figure 3.8 is an example of an RGB image with its three colour channels and pixel values of each plane. It also shows the resolution height and width.

- Convolution layer.** The convolutional layer is one of the main building blocks of a CNN. The parameters of this layer include a set of learnable kernels (or filters), which have a small receptive field that extends through the full depth of the input volume. Throughout the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the values of the filter and the input, and producing a 2-dimensional feature map of that filter. Therefore, the neural network studies filters that activate when they detect some a specific type of feature at some spatial position in the input. Figure 3.9 shows a logical diagram of the convolution layer[70], [71].



Figure 3.9: Logical diagram of convolution layer

Figure 3.9 shows the input image and filter that will be applied on the left-hand side and on the right-hand side the result of the applied filter on the image.

- Pooling Layer.** The responsibility of the pooling layer is to reduce the spatial size of the feature map. This is necessary in order to reduce the computational power required to process the data through dimensionality reduction. It is also useful for extracting features which are rotational and positional invariant the result of which will maintain the process of effectively training the model. There are two types of pooling, namely max pooling and average pooling, as seen in Figure 3.10. Max pooling returns the maximum value from the portion of the image covered by the filter. Average pooling returns the average of all the values from the portion of the image covered by the filter. Max pooling can also be utilised as a noise suppressant. It can discard the noisy activations and remove noise along with dimensionality reduction. Average pooling performs dimensionality reduction as a noise suppressing mechanism [70], [71].

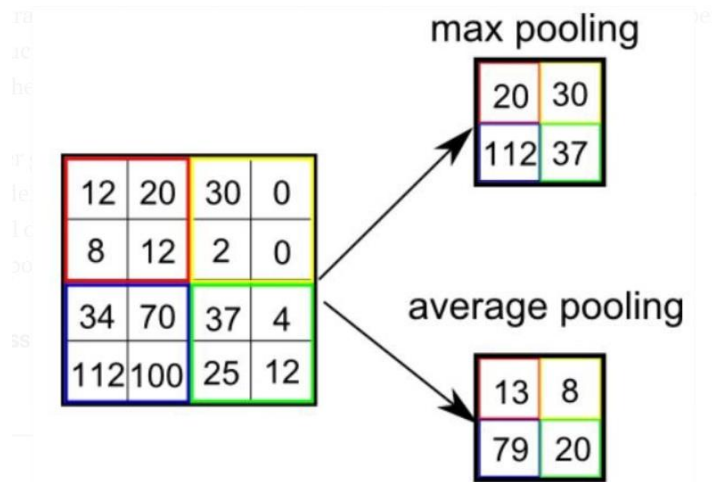


Figure 3.10: Max pooling and average pooling[52]

Figure 3.10 shows the difference between max pooling and average pooling, and the difference each method yields.

- Fully connected layer.** The aim of a fully connected layer is to use the results of the convolution and pooling process to classify the image into a label. The convolution and pooling outputs are flattened into one single value vector, representing a probability that a certain feature belongs to a label. If, for example, the picture is of a dog, features like whiskers or fur should have high probabilities for the 'dog' label. In order to select the most accurate weights, the fully connected part of the CNN goes through its own backpropagation process. All the neurons are given weights that give priority to the label that is most fitting. Finally, the neurons 'vote' on each of the labels; "the winner of that vote is the decision of the network[72].

3.6 Human pose estimation

Human pose estimation is an important problem within the computer vision community. It is a central step towards understanding people in images and videos. Human pose estimation is defined as the localisation of human joints (e.g. wrists, elbows, knees, ankles, etc.) in images and videos. Despite the fact that pose estimation offers innovative solutions to action recognition, its performance may

vary because of strong articulations, small and barely visible joints, occlusions, clothing and change in lighting. Most recent pose estimation systems have universally adopted CNNs as their main building block, largely replacing hand-crafted features and graphic models; this strategy has yielded drastic improvements on standard benchmarks.



Figure 3.11: Detected skeleton using pose estimation[73]

Figure 3.11 is an example of human skeletons detected by pose estimation. In the next section work done in pose estimation and how the field has evolved through the years will be reviewed.

3.6.1 DeepPose

Toshev and Szegedy [74] proposed the first major paper that applied deep learning algorithms to human pose estimation. In their approach a convolutional neural network (CNN)-based regression algorithm was applied. They further cascaded such regressors to refine the pose estimates to ensure more accurate results. The advantage of this holistic approach is it can estimate joints that are hidden in certain situations. Their model was constructed with an AlexNet backend, containing 7 layers, with an additional final layer that outputs 2k joint co-ordinates.

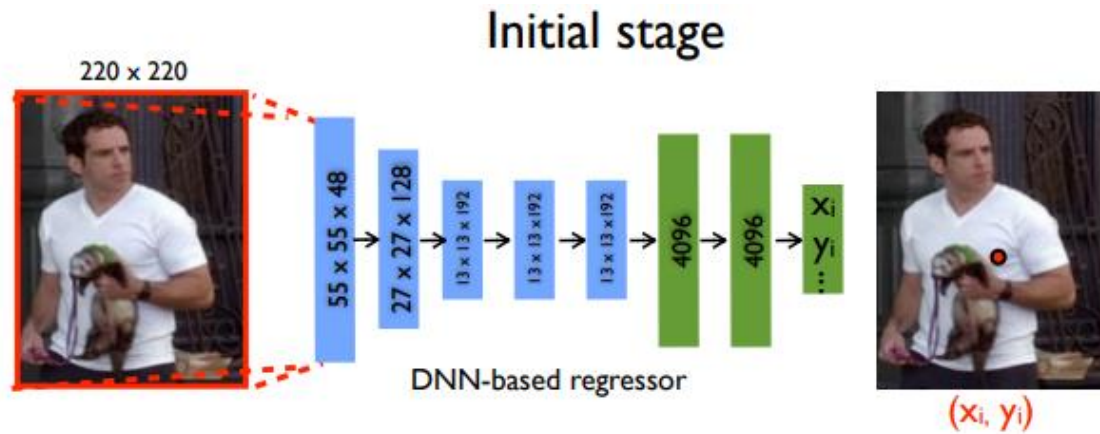


Figure 3.12: Schematic view of the DNN-based pose regression[74]

Figure 3.12 shows the logical layout of the DDN-based regressor. This model consists of an AlexNet[75] backend with 7 layers and an extra final layer that outputs 2k joint co-ordinates – (x_i, y_i) * 2 for $i \in \{1, 2 \dots k\}$ where k is the number joints. The model is trained using L2 loss for regression.

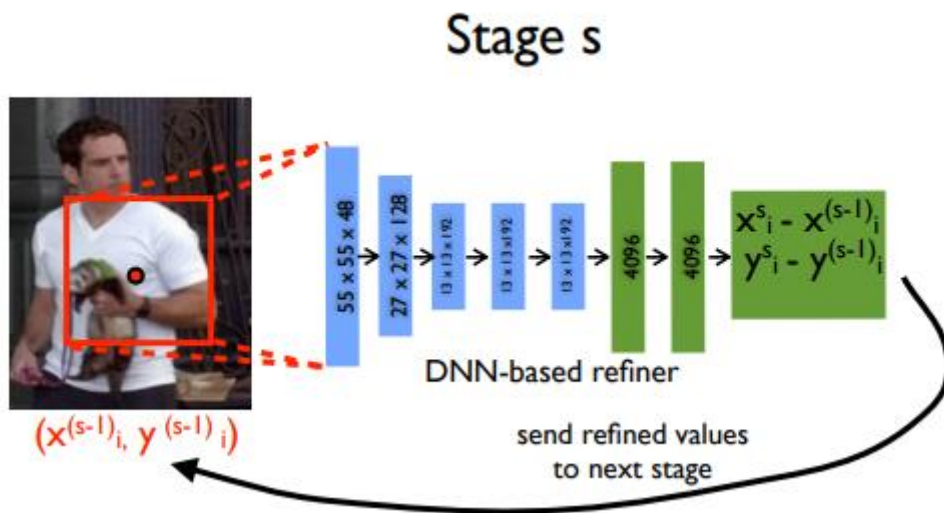


Figure 3.13: Refining regressor is applied on a sub image to refine a prediction from the previous stage[74]

Figure 3.13 illustrates the refining regressor that is applied on a sub-image to refine a prediction from the previous stage. An interesting idea implemented by this model, is refining the predictions by using

cascaded regressors. The initial coarse pose is refined, resulting in a better approximation. Images are cropped around the predicted joint and fed to the next stage, thus allowing the eventual pose regressors to see higher resolution images, thereby learning features for finer scales that eventually contribute to greater precision.

3.6.2 Efficient object localisation using convolutional networks

Tompson, *et al.* [76] introduce a novel architecture which includes an efficient ‘position refinement’ model that is trained to estimate the joint offset location within a small region of the image. It generates heat-maps by running an image through multiple resolution banks parallel to simultaneously capturing features at a variety of scales. The output is a discrete heat-map instead of continuous regression. A heat-map predicts the probability of the joint occurring at each pixel. This approach uses a multi-resolution convolutional neural network (CNN) architecture or coarse heatmap model. The model implements a sliding window detector to produce a coarse heat-map output as seen in Figure 3.14.

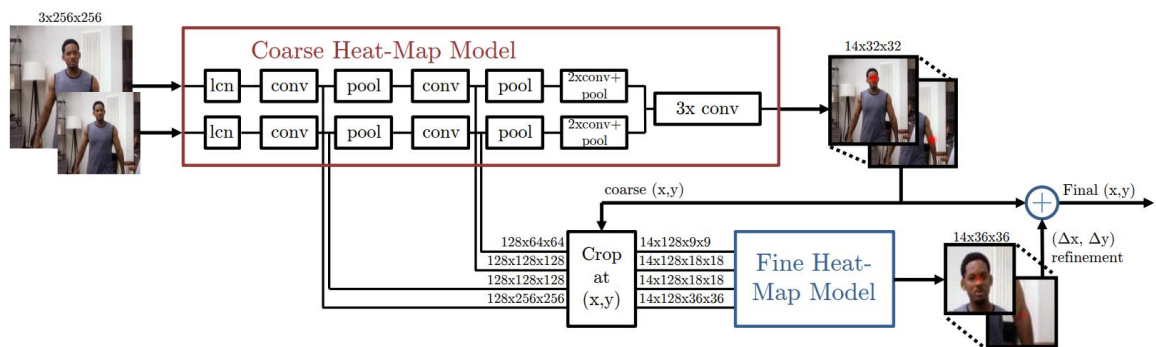


Figure 3.14: Overview of cascaded architecture[76]

Figure 3.14 shows an overview of the cascaded architecture. The model is trained by minimising the mean squared-error (MSE) distance of the predicted heat-map to a target heat-map. The target is a two-dimensional Gaussian of constant variance of $\sigma \approx 1,5$ pixels centered at the ground-truth (x,y) joint location.

3.6.3 Convolutional pose machines

Wei, *et al.* [77] utilise pose machines. Pose machines provide a sequential prediction framework for learning rich implicit spatial models. In their paper they demonstrate a systematic design for how convolutional networks can be incorporated into the pose machine framework for learning image features and image-dependent spatial models for the task of pose estimation.

Figure 3.15 depicts a high-level view of Stage 1 (a) and 2 (b). Stage 1 is the image feature computation module. Stage 2 is the prediction module. g_1 and g_2 predict heat-maps. This is referred to as belief maps in the paper.

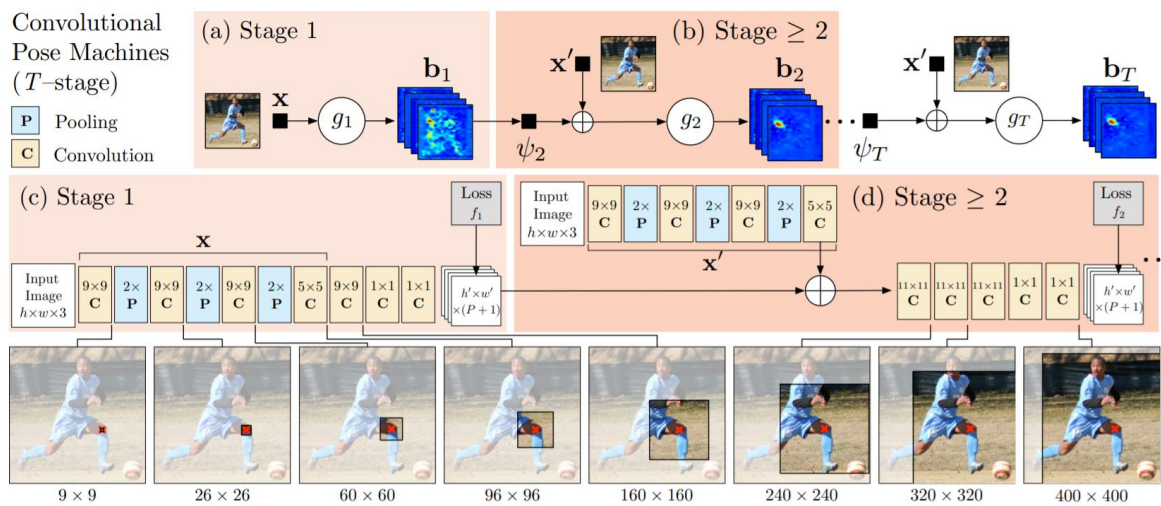


Figure 3.15: Architecture and receptive fields of Convolutional Pose Machines[77]

Figure 3.15 shows the architecture and the receptive fields of a convolutional pose machine and how the stages interact with one another. A convolution pose machine can consist of more than two stages; the number of stages is a hyperparameter. Stage 1 is fixed and stages after the second stage are just repetitions of Stage 2. In Stage 2 heat-maps and image evidence are taken as input. This input heat-maps add spatial context for the next stage. On a high level, the convolution pose machine refines the heatmaps through subsequent stages. In this paper an intermediate supervision after each stage is

utilised to avoid the problem of vanishing gradients, which is a common problem for deep multi-stage networks[77].

3.6.4 Human pose estimation with iterative error feedback

Carreira, *et al.* [78] propose a self-correcting model that progressively changes an initial solution by feeding back error predictions instead of directly predicting the outputs. Feedforward architectures can learn rich representations of the input space, but do not explicitly model dependencies in the output spaces which are structured for tasks such as articulated human pose estimation or object segmentation. They propose a framework that expands the expressive power of hierarchical feature extractors to encompass both input and output spaces by introducing top-down feedback.

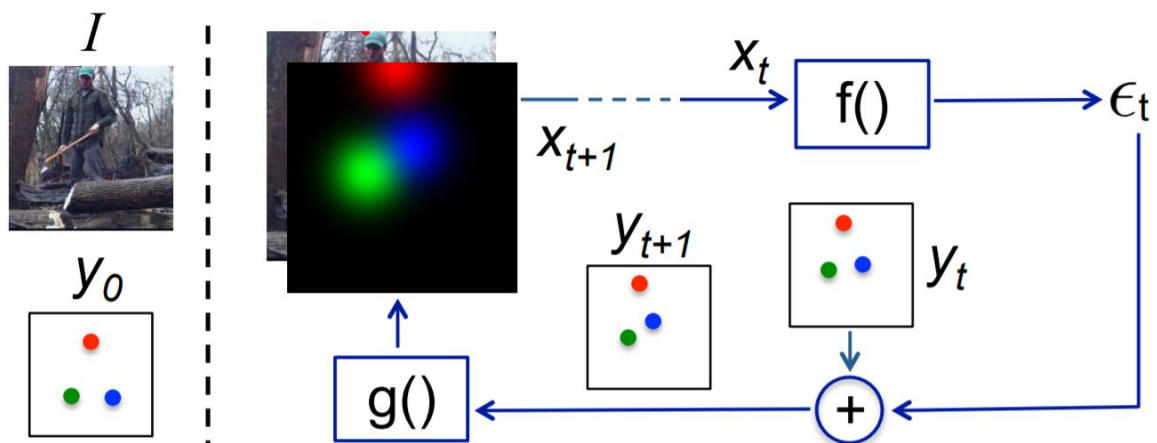


Figure 3.16: Implementation of iterative error feedback[78]

Figure 3.16 shows the model and implementation of the iterative error feedback. The model takes an input image and the previous output. Therefore, the input $x_t = I \oplus g(y_t - 1)$ where I is the image and $(y_t - 1)$ is the previous output. With each iteration the output is more refined. The function $f(x_t)$ outputs the correction ϵ_t which is then added to the current output y_t to create $(y_t + 1)$. The function $g(y_t + 1)$ then converts each key point in $y_t + 1$ into a heat-map and it is stacked to image I to generate the input for the next iteration. This process is then repeated T times until $(y_t + 1)$ is refined and is brought closer to ground truth by addition of ϵ_t .

3.6.5 Stacked hourglass networks for human pose estimation

Newell, *et al.* [79] introduces a novel and intuitive architecture that performs better than all previously mentioned methods. The method is known as a stacked hourglass network, since it consists of steps of pooling and up-sampling layers which resemble an hourglass. These are then stacked together. This design of the hourglass was encouraged by the need to capture information at every scale. While local evidence is essential for identifying features like faces and hands, a final pose estimate requires global context. The person's orientation, the arrangement of their limbs, and the relationships of adjacent joints are among the many cues that are best recognised at different scales in the image.

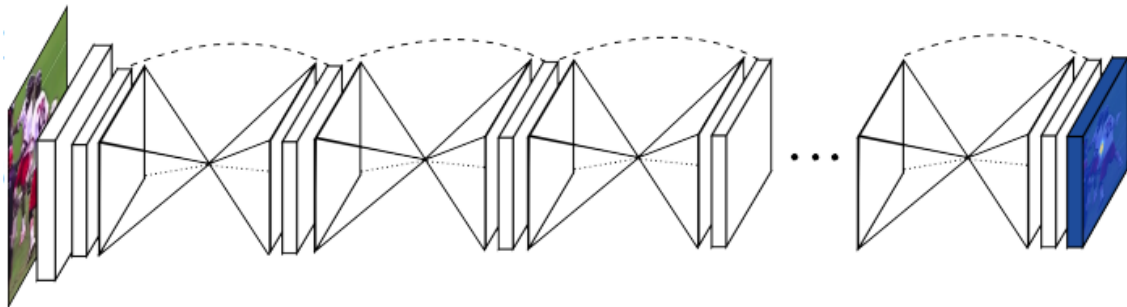


Figure 3.17: Schematic view of the hourglass model[79]

Figure 3.17 is a schematic depiction of the hourglass model. This network performs repeated bottom-up, top-down processing with intermediate supervision. Bottom-up processing is from a high resolution to a low resolution, and top-down processing is from low resolutions to high resolutions. Intermediate supervision is then applied to the predictions of each hourglass stage; thus the predictions of each hourglass in the stack is supervised and not only the final hourglass predictions. The network also uses skip connections to preserve spatial information at each resolution and passes it along for up sampling further down the hourglass[79].

3.6.6 Simple baselines for human pose estimation and tracking

Previous approaches work very well but have very complex architectures. Xiao, *et al.* [80] ask the question of how good a simple method could be? This paper provides baseline methods for both pose estimation and tracking. The network structure is quite simple and consists of a ResNet[81] and deconvolutional layers at the end.

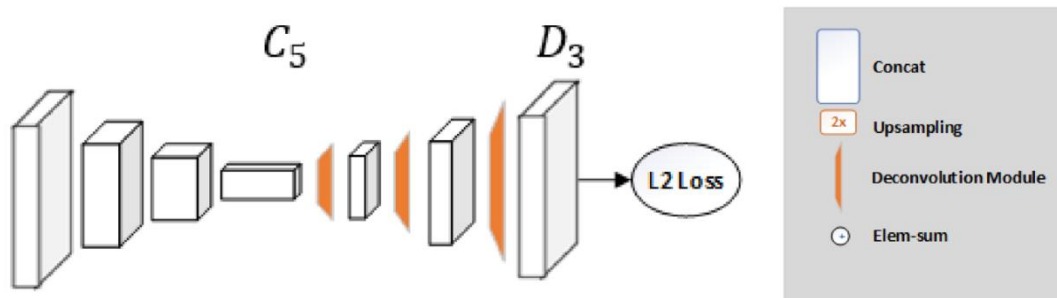


Figure 3.18: Architecture of Simple Baseline[80]

Figure 3.18 shows the architecture of the simple baseline model with its multiple layers.

3.6.7 Deep high-resolution representation learning for human pose estimation

Sun, *et al.* [82] propose the high-resolution network (HRNet) model. Most of the previous papers went from a high to low to high-resolution representation. HRNet maintains a high-resolution representation throughout the entire process. A high-resolution subnetwork as the first stage, gradually adds high-to-low resolution subnetworks one by one to form more stages and connect the multi-resolution subnetworks in parallel. A repeated multi-scale fusion, with each of the high-to-low resolution representations repeatedly receiving information from other parallel representations, leads to rich high-resolution representations.

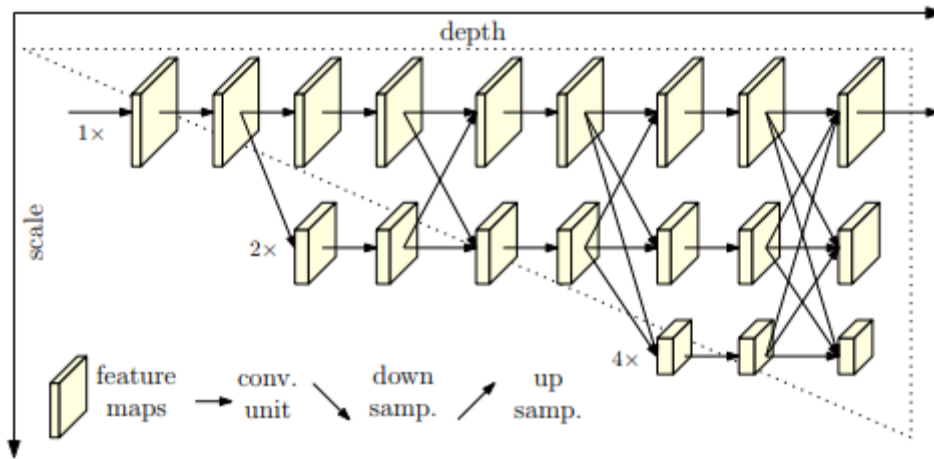


Figure 3.19: Architecture of the HRNet[82]

Figure 3.19 presents the architecture of the proposed HRNet. The architecture consists of parallel high-to-low resolution subnetworks with repeated information exchange across multi-resolution subnetworks. The horizontal and vertical directions correspond with the depth of the network and the scale of the feature maps[82].

3.6.8 OpenPose

In this study the authors [83] present a real-time approach to detect the two-dimensional pose of multiple people in an image. This method uses a non-parametric representation, which is referred to as ‘part affinity fields’, to learn to associate body parts with individuals in the image. This bottom-up approach achieves high accuracy and real-time performance, regardless the number of people in the image. This method is mostly known as open pose.

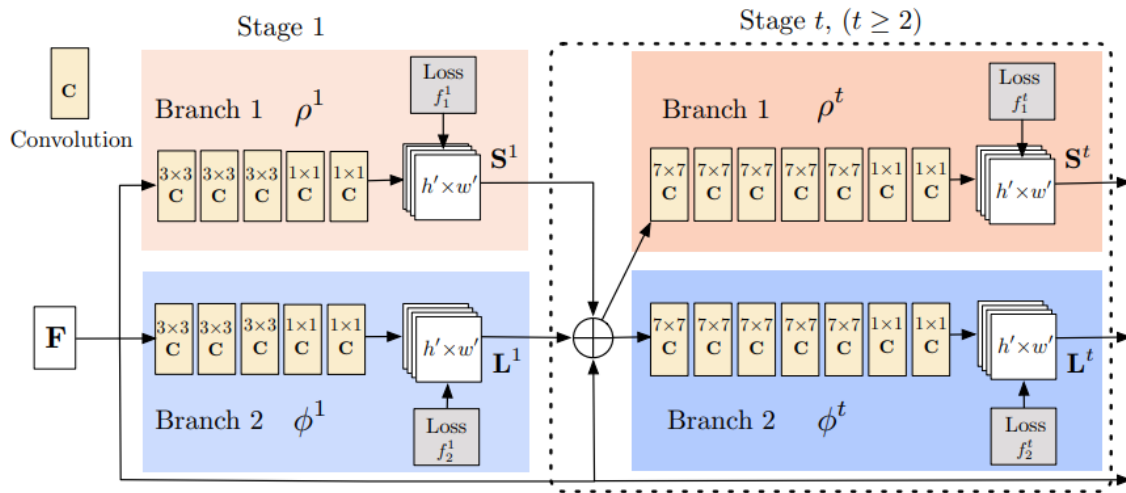


Figure 3.20: OpenPose Architecture[83]

In Figure 3.20 illustrates the architecture of OpenPose which consists of a two branches and multiple stages. The first branch consists of stages that predict confidence maps (S^t). The second branch consists of stages that predicts Part Affinity Fields (L^t). Following each stage, the predictions made from each branch and the image feature are concatenated for the next stage. Every branch utilizes an iterative prediction approach which refines the predictions every iterative with supervision on each stage. Feature Maps (F) are the input to the first stage of the of each stage. These sets of feature maps are generated by a convolutional network that analyses the image.

3.7 Datasets

In this section three action recognition datasets are introduced. The presented datasets are used throughout this study and will be used as training data.

3.7.1 Joint-annotated human motion data base

The large human motion database (HMDB51)[84] contains more than 5 100 video clips of 51 different human actions collected from movies, YouTube and other sources on the internet. The annotation is that this entire dataset is impractical, but the joint-annotated human motion data base (JHMDB)[85] is a subset with fewer categories. The authors exclude categories that

contain mainly facial expressions, interactions with others, and actions that can only be executed in a specific way, such as a cartwheel.



Figure 3.21: JHMDB dataset: Sample video frames for each of the 21 action categories[84]

This resulted in a dataset that contains 21 categories, involving only a single person in action. Figure 3.21 shows an annotation of the selected actions. There are a total of 31,838 annotated frames.

3.7.2 North-western-UCLA multiview action 3D dataset

The Multiview 3D event dataset was captured by Jiang Wang and Xiaohan Nie at the University of California, Los Angeles (UCLA)[86]. The dataset contains RGB, depth and human skeleton data captured simultaneously by three Kinect cameras. This dataset includes ten action categories: picking up with one hand, picking up with two hands, dropping trash, walking around, sitting down, standing up, donning, doffing, throwing, carrying. Each action is performed by ten actors. The dataset contains data taken from a variety of viewpoints.



Figure 3.22: Snapshots of frames from a UCLA multiview action 3D dataset

3.7.3 AAMAZ Human Action Recognition Dataset

The AAMAZ human action recognition dataset is a video dataset, created by Ismaili Hossain[87]. The dataset consists of 11 action categories. The categories are: crawling, boxing left, boxing right, hand clapping, jogging, jumping, one hand waving left, one hand waving right, running, two-hand waving, and walking. The dataset is recorded under very poor lighting conditions.

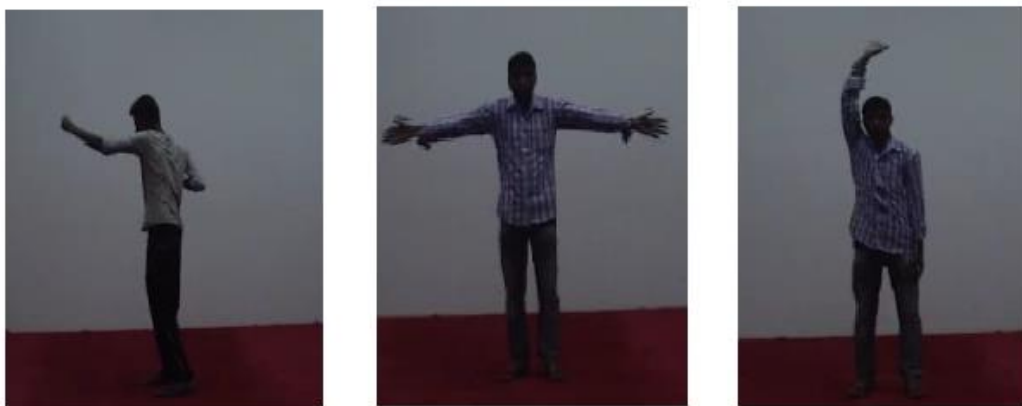


Figure 3.23: Snapshots of frames from AAMAZ Human Action Recognition Dataset

3.8 Conclusion

The purpose of this chapter was to acquire preliminary knowledge in the field of surveillance systems, human action recognition, neural networks, and pose estimation, as well as how these technologies

can be used to design an intelligent system for prevention of crime. As previously stated, crime in South Africa affects all communities and has the potential of long lasting physical and psychological damage. Crime prevention is one of the solutions of addressing this issue.

Surveillance systems, e.g. CCTV systems, are a great solution to gather evidence when a crime is committed. The advantage of a CCTV system is that it comes as a full package and they are easy to install and set up. The system includes a video recorder and the ability of local and remote monitoring. This enables security officers to scroll through footage and gather evidence of an incident. CCTV systems are an excellent security feature, but it has limitations. The main limitation is that such a system can only record incidents; it does not provide any solution for preventing crime. Intelligent surveillance systems aim to address some of the shortcomings of traditional surveillance systems. An intelligent surveillance system has abilities such as human action recognition, intrusion detection, robbery prevention, people counting, loitering detection, etc. Such features can be used to alert security officers to respond early, thus preventing a crime from happening.

The purpose of human action recognition systems is to analyse a scene and to recognise human actions. In order to achieve this, extraction is needed to compute information captured from images or video frames. Human action recognition provides innovative technologies than can be used in crime prevention, but it also comes with many challenges. Videos, for example, are not always captured in perfect conditions; therefore, one must consider many factors when deciding on which feature extraction technique to use. The technique needs to be versatile, robust, and able to deal with unstable footage and noisy footage.

Human pose estimation is a technique used to detect human joints in images and videos. This technique is commonly used to extract human features in videos, because it can be run in real time

and it can detect multiple humans in a frame. This technique can be used to extract information about a human in a frame and for later classification.

Neural networks have the ability to learn underlying relationships in a set of data. This data usually comes from a dataset in which features have been extracted. Once the neural network has been trained, it is able to classify data it has not seen. This can be implemented in a Human Action Recognition System to classify human actions.

Chapter 4: Video Feature Extraction

4.1 Introduction

Phase 3 of this study focused on the development of video feature extraction that was responsible for pre-processing the video files. Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. Video datasets are very large and therefore require a lot of computing resources to process. Feature extraction effectively reduces the amount of data that must be processed, while still accurately and comprehensively describing the original data set. Figure 4.1 shows the overall study design, highlighting Phase 3, which involved the development of the video feature extraction.

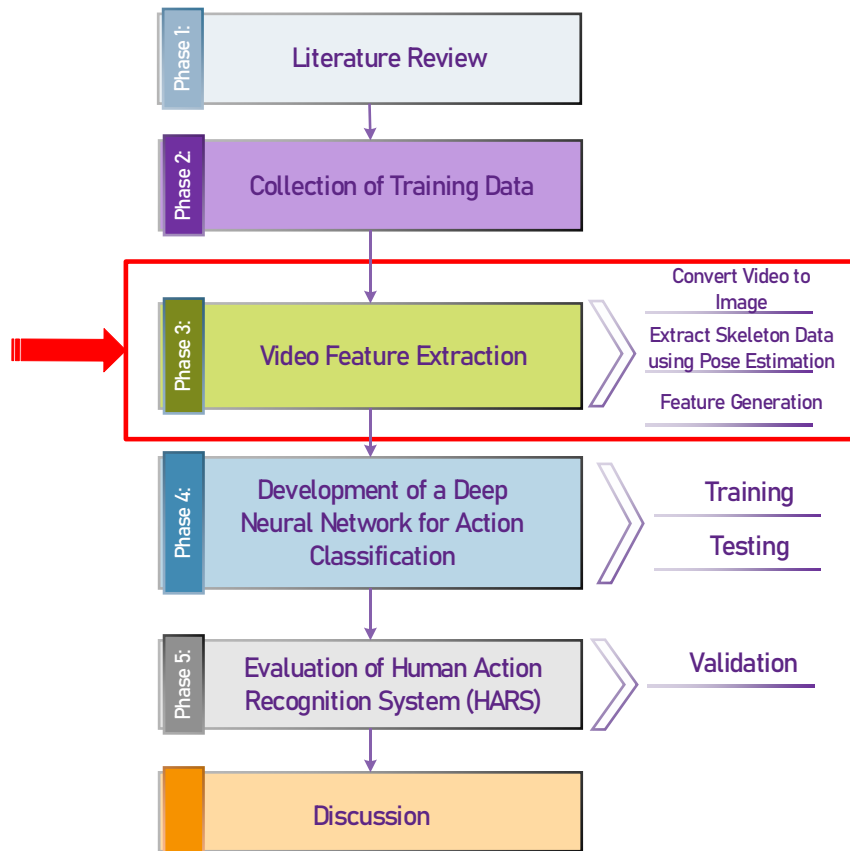


Figure 4.1: Overall study design

Before the training could start, several preparatory steps needed to be taken. The first step was to convert the video files to a series of JPEG images. The raw skeleton data was then extracted, using OpenPose which map out the joints data of the person in the image. In the second step the raw skeleton data was preprocessed. The third step involved feature extraction. The extracted features were then used to train the deep neural network. The layout of the logical design for the video feature extraction is shown in Figure 4.2.

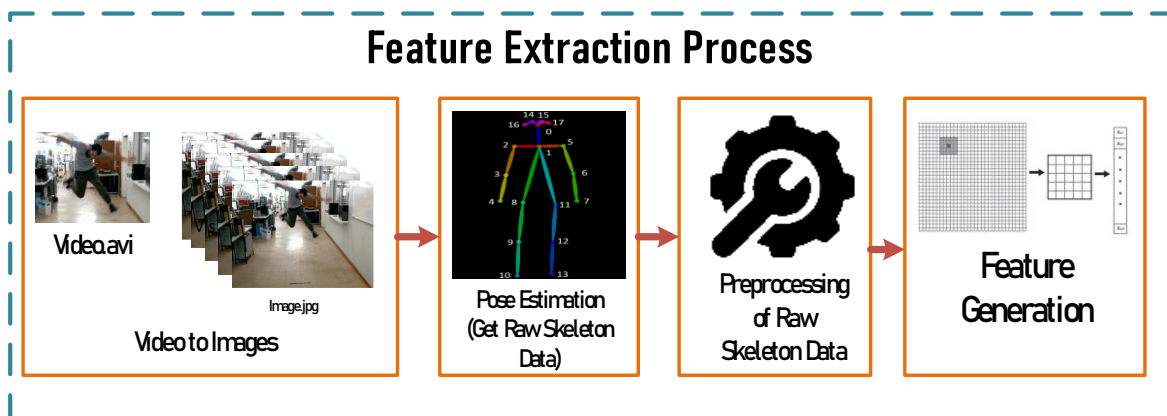


Figure 4.2: Logical layout of feature extraction

4.2 Software development environment

The four programs used for video feature extraction were developed by using Anaconda. Anaconda is a free open-source distribution of the Python programming language for data science, machine learning applications, large-scale data processing, predictive analytics, etc. It aims to simplify package management and deployment [88]. The distribution includes data science packages suitable for Windows, Linux, and macOS. The software tools used to create the programs were:

- Python programming language.** Python is a high-level general-purpose programming language. Python is designed with an emphasis of code readability and notable use of significant whitespace. Its language constructs an object-orientated approach, and is designed to help programmers write clear, logical code for small and large-scale projects.

- **Spyder Integrated Development Environment (IDE).** This is an open source cross-platform for scientific programming in the Python language. It provides comprehensive facilities for software development, such as a source code editor, build automation tools and a debugger.
- **Open-source Computer Vision (OpenCV).** OpenCV is a computer vision and machine learning software library. It provides a common infrastructure for computer vision applications to accelerate the use of machine perception. The library provides more than 2 500 optimised algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.
- **TensorFlow.** TensorFlow is an open-source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for classification, perception, understanding, discovering, prediction and creation. OpenPose is an open-source, real-time, multi-person key point detection library for human body, facial, hand and feet estimation.
- **OpenPose.** OpenPose was developed by researchers at Carnegie Mellon University. It is considered a state-of-the-art approach for real-time human pose estimation. The code base is open source. OpenPose is originally written in C++ and Caffe but has been adapted to work with Python using TensorFlow.

4.3 Convert videos to images

Videos are a series of digital images. They can, however, not be used for data extraction in their raw format. This step converts the video to digital images. The format that the images are converted to and stored as JPEG. JPEG was used because it provides smaller file sizes compared to other standards like BMP and PNG. It also allows users the ability to adjust the degree of compression of a digital image. This enables the user to find the correct balance and trade-off between the image quality and file size.

The logical design of the program was developed in Python 3 programming language, using the OpenCV library, and consisted of three logical steps. These steps involved initialisation of variables, reading video file from file path, and extracting each image frame from the video and writing an image file. The flow diagram in Figure 4.3 shoes how the program operates.

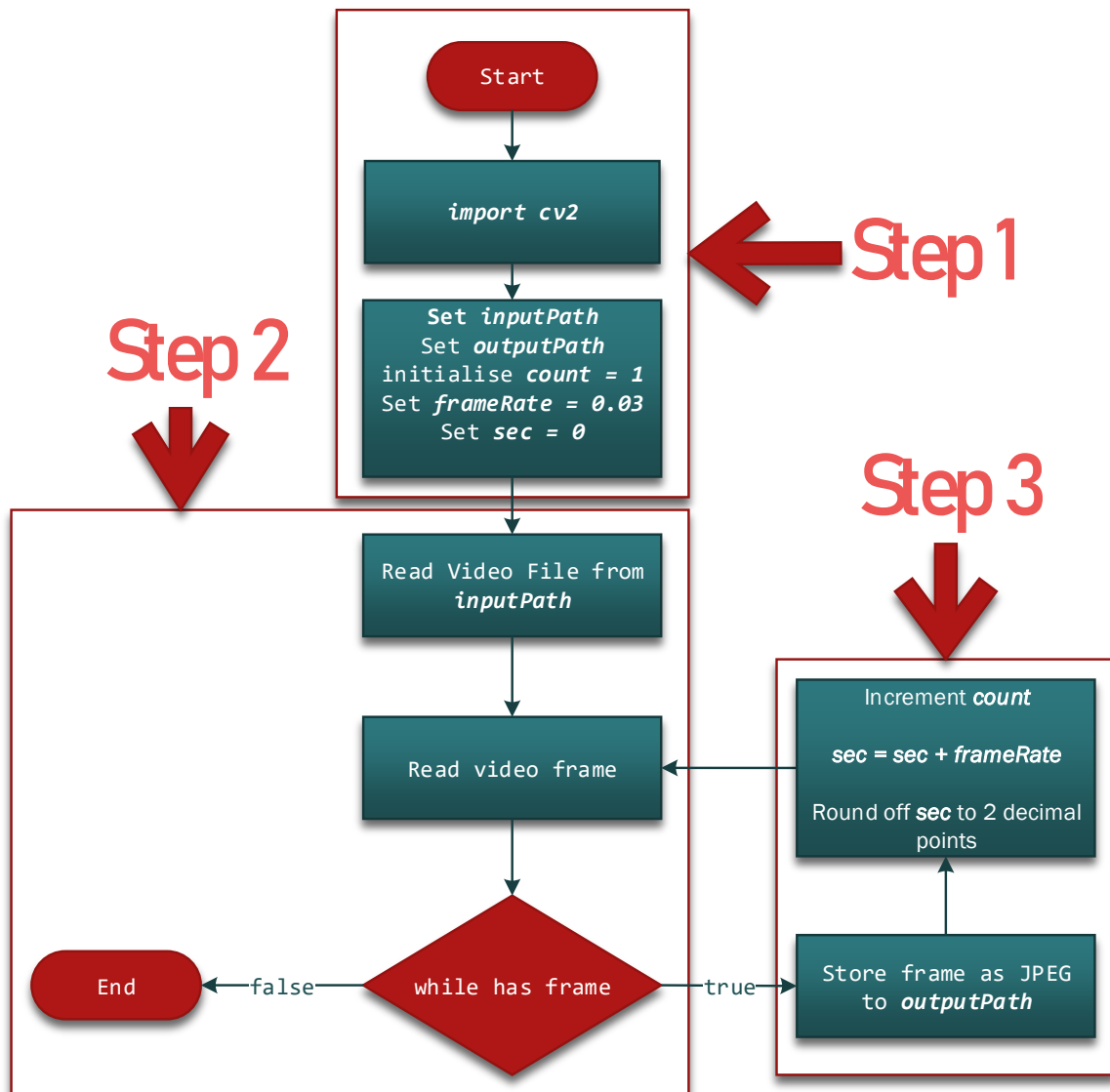


Figure 4.3: Flow chart of converting video to images

The details of the three logical steps are as follows:

- **Step 1: Initialisation of variables.** The first step in the program was to initialise variables. Firstly, we imported the OpenCV library, using the `import cv2` command. Secondly, the

inputPath and *outputPath* were set. The *inputPath* is the file path where the video is stored and the *outputPath* is the path for the location folder where the images would be stored. The *count* variable was initialised to 1. This variable was used to count the number of frames extracted from the video. The *frame rate* was set to 0,03. Frame rate, i.e the frequency at which consecutive image frames appear, was decoded. For the videos in this program the frame rate was 30 frames per second. The *sec* was set to 0. This variable represents seconds. It will determine the second at which each frame is extracted.

- **Step 2: Reading video file and extracting image frame.** In the second step the video was read from the *inputPath* variable given in the Step 1. A frame from the video was extracted, based on the value of the variable *sec*. An if-statement then checks if the frame exists and if it does not exist the program ends.
- **Step 3: Writing image file.** In the third step an if-statement checks If the frame does exist and if it does not exist, the image frame is then stored in the *outputPath* folder as a JPEG. After the program has stored the program, it performs the following operations in preparation of reading the next frame. It increments the *sec* by the value of the frame rate. Then it then performs a rounding function which rounds off the value for *sec* to two decimal places. The program then returns to reading the next image frame.

After the completion of all the logical steps, the output of the steps will be stored in the *outputPath*.

This program must be applied on all videos that will be used for training data.

4.4 Extracting skeleton data using OpenPose

The program used to extract skeleton data from the images, was OpenPose. OpenPose is human pose estimation algorithm that has been implemented, using TensorFlow. Pose estimation is a key component in enabling machines to understand people in images and videos. OpenPose is a real-time approach that detects two-dimensional pose of multiple people in an image. This method uses a

non-parametric representation, which we refer to as Part Affinity Fields (PAFs), to learn to associate body parts with individuals in the image.

OpenPose takes a colour image of size $w \times h$ as the input (Figure 4.4(a)) for a convolutional neural network (CNN) to jointly predict and produce two-dimension locations of anatomical key points for each person in the image (Figure 4.4(e)). The system further predicts a set of two-dimensional confidence maps (Figure 4.4(b)) of body part locations and a set of two-dimensional vector fields of part affinity fields (PAF) as seen in Figure 4.4(c), which encodes the degree of association between parts. The set also has confidence maps, one per part, whereas vector fields have one per limb. They system further refers to part pairs as limbs for clarity, but some pairs are not human limbs, e.g. the face. Lastly, the parsing step performs a set of bipartite matchings (Figure 4.4(d)) to associate body part candidates to output the 2D key points for all people in the image.

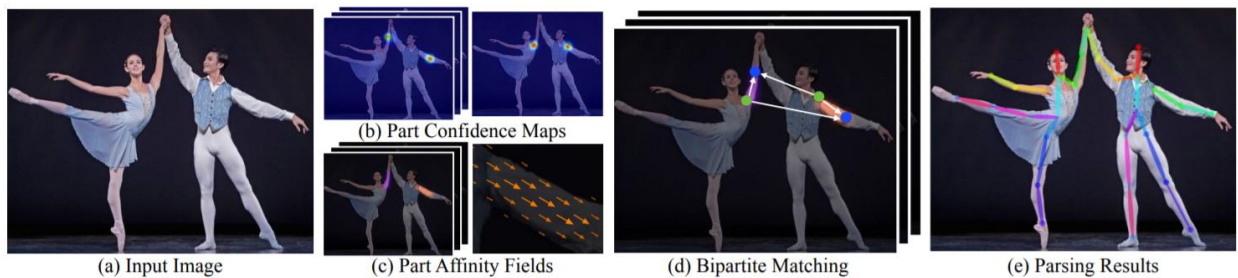


Figure 4.4: Logical design for OpenPose algorithm[89]

Each skeleton has 18 joints which include the head, neck, arms and legs. The joints are labelled as seen in *Table 4.1*.

Table 4.1: Skeleton Joints Label output by OpenPose

Joints		Skeleton
Name	Index	
Head:	0	
Eyes:	14, 15	
Ears:	16, 17	
Neck:	1	
Left Arm:	5, 6, 7	
Right Arm	2, 3, 4	
Left Leg:	11,12, 13	
Right Leg:	8, 9, 10	

4.5 Preprocessing of raw skeleton data

The data generated by OpenPose is unusable in its raw format. OpenPose uses different units for its x and y co-ordinates. It also generates head joints which are not necessary for this study. There are also frames which the algorithm does not detect, or it partially detects a human. Software components were designed to address these issues. The program was designed to perform four logical steps. These steps involved scaling joint co-ordinates, removal of head joints, discarding invalid frames and filling in missing joints from incomplete skeletons. The flow diagram of how the program operates, is shown in Figure 4.5

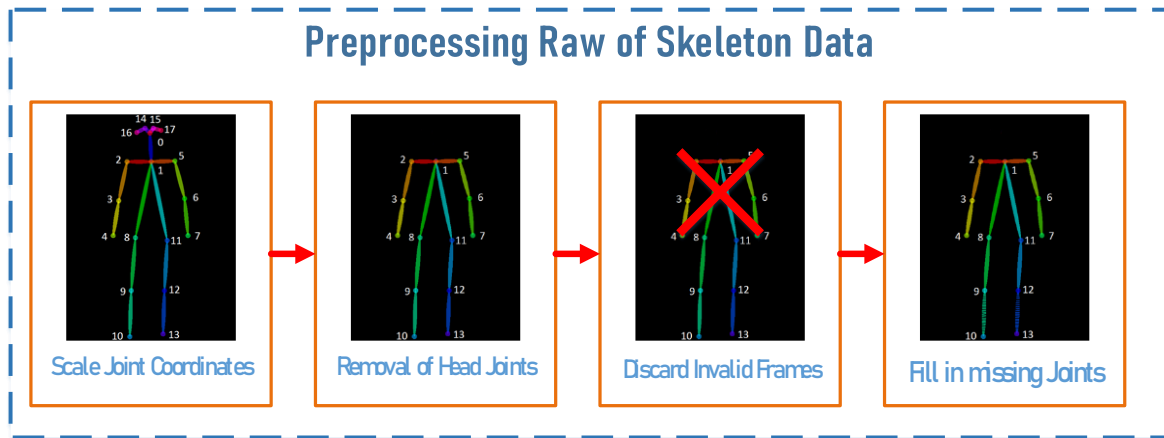


Figure 4.5: Preprocessing raw of skeleton data

There were four software components designed to preprocess the raw skeleton data. Firstly, software to scale joint co-ordinates; secondly, software to remove the head joints of the skeleton; thirdly, discarding frames where humans were not detected and frames where insufficient joints were detected; and, lastly, filling in missing joints on frames that were not invalid but had missing joints.

4.5.1 Scale joint co-ordinates

In the first step of the program, it takes raw skeleton data as an input. The original joint positions that OpenPose outputs, uses a different unit for the x co-ordinate and y co-ordinate. The co-ordinates need to be normalised to the same unit to address images with different height/width ratios.

4.5.2 Removal of head joints

In the second step of the program, the 18 joints that OpenPose outputs, are taken as an input; the head joints are removed and a skeleton without head joints remains. As seen in *Table 4.1*, OpenPose also detects five head joints. The joints include one head joint, two eye joints and two ear joints. Head features were removed, as they are of little significance for classification in actions recognition.

4.5.3 Discard of invalid frames

The third step of the program determines the validity or invalidity of frames, to be used later in training. The criteria used to determine the validity of the frame is twofold: firstly, if no human skeleton is detected by OpenPose in a frame as seen in in Figure 4.6 (b), and, secondly, if the detected skeleton has a neck or leg joints as seen in Figure 4.6 (a). If a frame did not meet the criteria, it is considered invalid, and discarded. The reasoning behind the criteria is blank frames and frames with missing limbs do not add any value to the training process. This will also cause errors as all feature vectors need to be of the same length.



Figure 4.6: Example of Frames that have been discarded

4.5.4 Fill in missing joints

In some of the frames OpenPose fails to detect a complete skeleton. This results in some blanks in the joint positions. In the fourth step of the program, it fills in missing joint information. This is necessary to maintain a fixed-size feature vector for the following feature classification. One may decide to discard an incomplete skeleton frame, but this would mean the final system would never be able to detect the action when the person is standing sideways and not facing the camera. In order to fill in the missing joint, we calculate a joint's position based on its relative position in the previous frame with regard to the neck. Suppose in the previous frame the hand is 10 pixels to the right of the neck; then, in the current frame if the hand joint is missing, it will set at 10 pixels to the right side of the neck of the current frame.

Equation 1: Calculate missing joint x-coordinate

$$x_{i(\text{curr})} = x_{\text{neck}(\text{curr})} + (x_{i(\text{pre})} - x_{\text{neck}(\text{pre})})$$

Where: x_i = x-coordinate of the missing joint

i = joint index (0 - 17)

x_{neck} = x-coordinate of the relative neck

Equation 2: Calculate missing joint y-coordinate

$$y_{i(\text{curr})} = y_{\text{neck}(\text{curr})} + (y_{i(\text{pre})} - y_{\text{neck}(\text{pre})})$$

Where: y_i = y-coordinate of the missing joint

i = joint index (0 - 17)

y_{neck} = y-coordinate of the relative neck

After step four all the feature vectors will be of the same size without any missing coordinates which is important for the training of the neural network.

4.6 Feature generation

A feature generation program needed to be developed after all the skeleton joints had been preprocessed. Feature generation is the process of creating new features from one or multiple existing features, potentially for using in statistical analysis or machine learning algorithms.

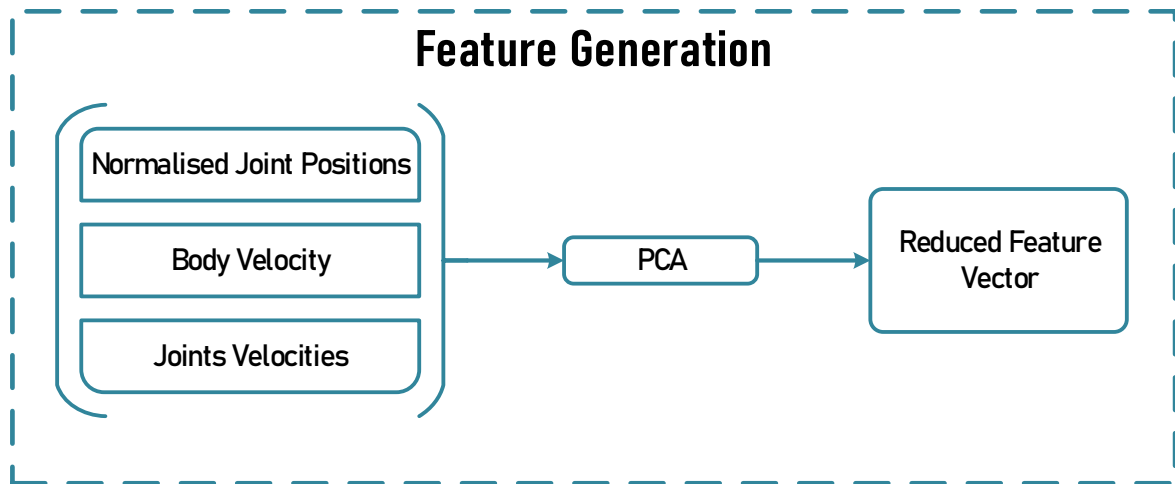


Figure 4.7: Logical design for feature generation

The logical design of the program used to generate features, was developed in four logical steps. Joint positions data from five frames (n) from the preprocessed data was used to generate each feature vector. The first step was to normalise the joint positions of the skeletons. Secondly, the velocity of the skeleton body was calculated. Thirdly, the velocity of the joints was calculated. Lastly, all the data was concatenated to create one feature vector and the principal component analysis (PCA) algorithm was applied to reduce the dimensions of the feature vector. The flow diagram of how the program works, is seen in Figure 4.7.

4.6.1 Normalise joint positions

In Step 1 of the program, it takes five skeleton frames (n) as an input and normalises all the joint positions, using Equation 3. This was necessary, as skeletons may vary in size and height. There therefore was a need to change the values of the skeleton joints to a common scale, without distorting differences in the ranges of values.

Equation 3 Normalise skeleton joint positions

$$X = \frac{X_i - \bar{X}}{H}$$

where: X = normalised joint positions

X_i = concatenated joints of n frames

\bar{X} = the sample mean

H = average skeleton height of X_i

4.6.2 Calculate body velocity

In the second step of the program a calculation of the body velocity was performed, using Equation 4.

This feature gives additional data about the changes in movement of the skeleton as a whole.

Equation 4 Calculate body velocity

$$V_{body} = \frac{V_{neck}}{H}$$

Where: V_{body} = body velocity

V_{neck} = velocity of the neck

H = average skeleton height

4.6.3 Calculate joint velocities

In the third step of the program a calculation of the joint's velocity was performed, using Equation 4.

This feature provides additional data about the changes in movement of the skeleton joints.

Equation 5 Calculate velocity of joints

$$V_{joints} = X[t_k] - X[t_{k-1}]$$

where: V_{joints} = joint velocity

X = normalised joint coordinated

t_k = current frame

4.6.4 Principal component analysis (PCA)

In the third step of the program the PCA algorithm was applied to the dataset. The purpose of this was to reduce the dimensionality of a dataset while retaining the variation present in the dataset up to the maximum extent. PCA performs a linear mapping of the data to a lower-dimensional space with the goal of maximizing the data's variance in the low-dimensional representation. The data's covariance matrix is built, and the matrix's eigenvectors are computed. Principal components, or eigenvectors that correspond to the greatest eigenvalues, can now be used to recreate a considerable portion of the variance of the original data. Additionally, because the first few eigenvectors frequently contribute the great bulk of the system's energy, especially in low-dimensional systems, they can often be interpreted in terms of the system's large-scale physical behaviour. The original space (with a dimension of the number of points) has been reduced to a space spanned by a few eigenvectors (with data loss but presumably maintaining the most significant variance).

The program takes the normalised skeleton joints of $n = 5$ frames, the body velocity, and joints velocities as an input. This results in an input vector of 314 dimensions. The PCA algorithm then reduces it to 50 dimensions.

4.7 Results of feature extraction

In this section the results obtained from the video feature extraction are presented, showing how each module processes the images.

4.7.1 Convert videos to images

The aim of this sub-phase was to convert videos from dataset into images. In Figure 4.8 we see the input path that was used to locate the videos by class. All videos in the data set were stored in avi

format. Figure 4.9 shows the converted videos stored in the output path and sorted by class. These images are now ready to be further processed.

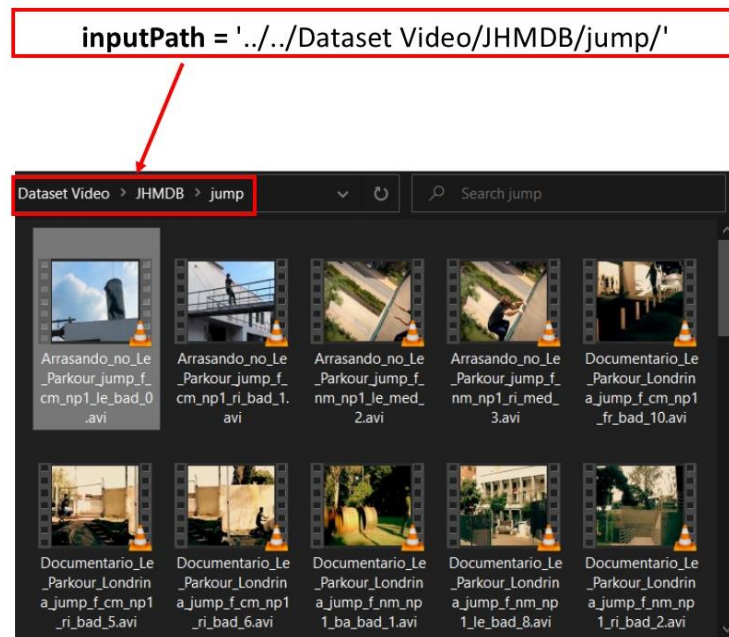


Figure 4.8: Input Videos from Video Dataset

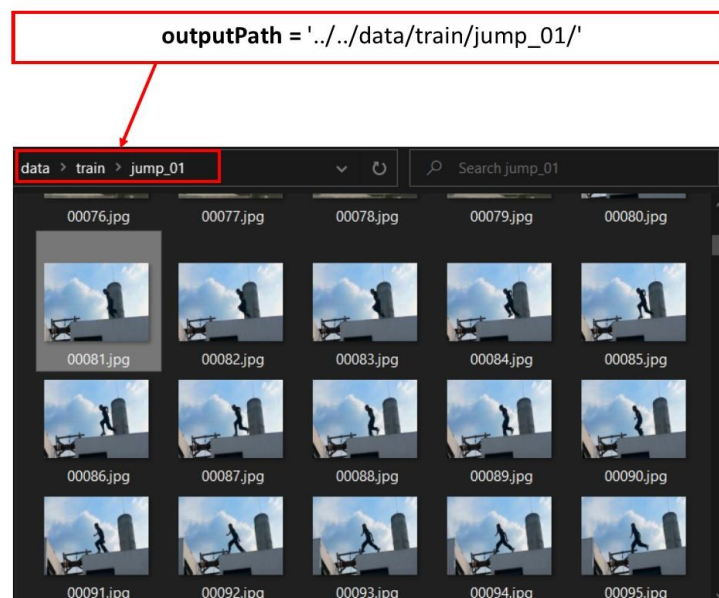


Figure 4.9: Converted JPEG Images

4.7.2 Extracting skeleton data using OpenPose

The purpose of this sub-phase was to extract skeleton data of humans detected in each image frame and store the data of the frame. Figure 4.10 shows the results of the OpenPose algorithm, as well as the input image with the annotated skeleton, heat-map and PAF map.

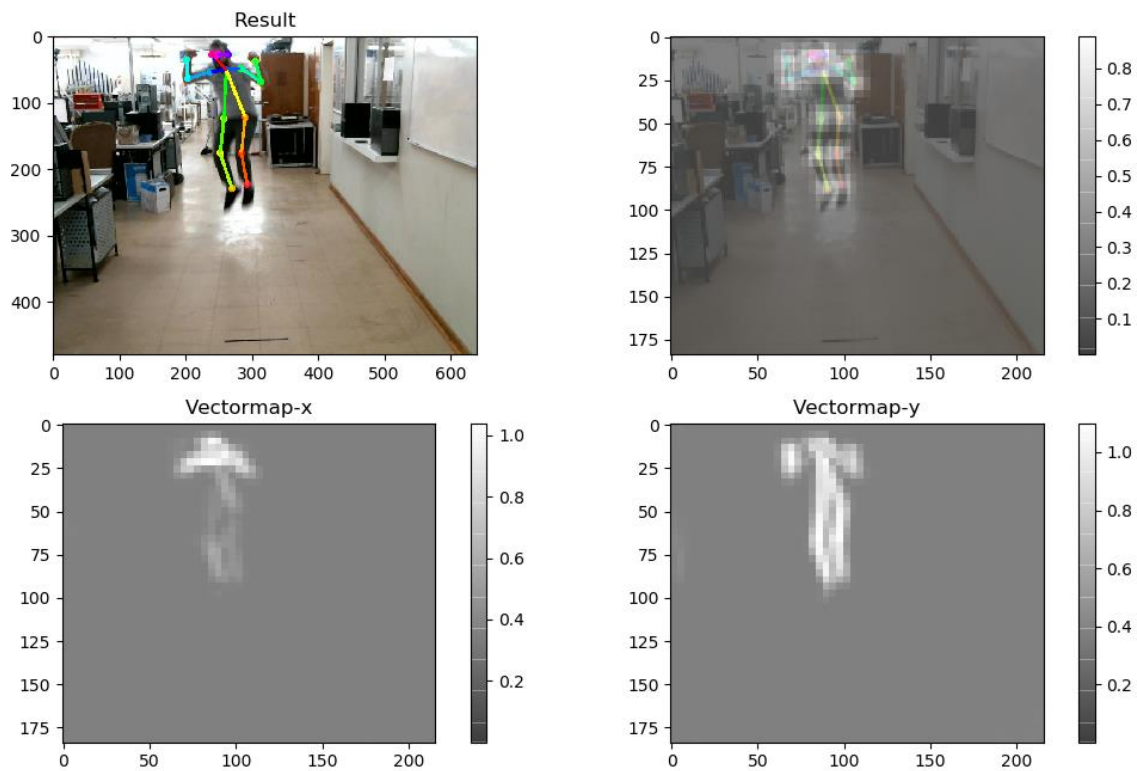


Figure 4.10: Skeleton Extraction from an image using OpenPose

In *Table 4.2* the detected joint co-ordinates detected and the confidence scores in Figure 4.10 are displayed. The algorithm has a 78% average score that it detected the joints correctly. The images above confirm these scores are correct and the skeletons have been detected correctly.

Table 4.2: Detected Skeleton Joints and Score by OpenPose

Body Part No.	Skeleton Joints Coordinates	Score
0	(0.38, 0.07)	0.89
1	(0.41, 0.11)	0.83
2	(0.37, 0.11)	0.71
3	(0.32, 0.14)	0.75
4	(0.31, 0.08)	0.77
5	(0.44, 0.10)	0.74
6	(0.49, 0.15)	0.74
7	(0.48, 0.08)	0.52
8	(0.40, 0.26)	0.64
9	(0.39, 0.37)	0.81
10	(0.42, 0.48)	0.65
11	(0.45, 0.26)	0.71
12	(0.44, 0.37)	0.68
13	(0.46, 0.47)	0.58
14	(0.38, 0.06)	0.82
15	(0.39, 0.06)	0.85
17	(0.41, 0.06)	0.84
Overall Score		0.78

In *Table 4.3* the number of skeletons detected in the dataset is reported. Each skeleton has 18 joints which makes 36 data points (Raw Feature Length).

Table 4.3: Number of skeletons detected in the dataset

Training Data	
Total number of Samples	22489
Raw Feature Length	36
Number of Classes	6

In *Table 4.4* the number of skeletons detected per action class is presented. This table shows the number of samples available for training and testing.

Table 4.4: Number of Skeletons Detected per Class

Number of Sample per Class	
Jump	2331
Kick	2245
Run	2492
Sit	5284
Stand	4692
Walk	5445

4.7.3 Preprocessing Raw Skeleton Data

The aim of the Preprocessing sub-phase was scaling joint coordinates, removal of head joints, discarding invalid frames and filling in missing joints from incomplete skeletons. This resulted in a reduced dataset as seen in *Table 4.5*.

Table 4.5: Number of samples remaining after the reduced dataset

Training Data	
Total number of samples	19730
Raw feature length	36

4.7.4 Feature generation

Table 4.6 shows the results of the applied PCA function. The generated features are reduced from a vector of length 314 to 50 with sum Eigen value of 0.95.

Table 4.6: Number of generated feature after PCA was applied

Generated Sample Features	
Size of training data	(19730, 314)
Sum eigen values	0.95
After PCA`	(19730, 50)

4.8 Discussion

The purpose of the videos feature extraction is to process the dataset videos and prepare them for training and/or classifications. This was achieved through four sub-phases, namely: converting videos to images, extracting skeleton data, pre-processing the skeleton data and, finally, feature generation.

In the first sub-phase the video(s) were converted into images. In the second sub-phase a pose estimation algorithm was used to extract skeleton data. The algorithm that was selected to extract skeleton feature in the images, was OpenPose. The extracted skeletons were not ready to be used for training in their raw format, as they vary in size because the algorithm is unable to detect certain joints in cases where the video does not display the full body or where there was occlusion. As a result, in the third sub-phase the detected skeletons were processed to determine valid and invalid frames and generate missing joints. This reduced the dataset from 22 489 to 19 730 samples. In the last sub-phase a PCA algorithm was applied on the processed skeleton feature to reduce the dimensionality of a dataset, while retaining the variation in the dataset.

After the video feature extraction phase was developed, the data it generated could now be used to train the Deep Neural Network (DNN) in Phase 4, Chapter 5.

Chapter 5: Development of a Deep Neural Network for Action Classification

5.1 Introduction

Phase 4 of this study involves the development of the deep neural network (DNN) for action classification. The purpose of the DNN is to classify human actions based on features generated in Phase 3 of the study. Phase 4 involves the development of a DNN to automatically recognise human actions based on six actions classes. The action classes selected for this study were jumping, kicking, running, standing, sitting, and walking. The DNN will form part of the human action recognition system (HARS). Figure 5.1 shows the overall study design, highlighting Phase 4, which involved the development of the DNN.

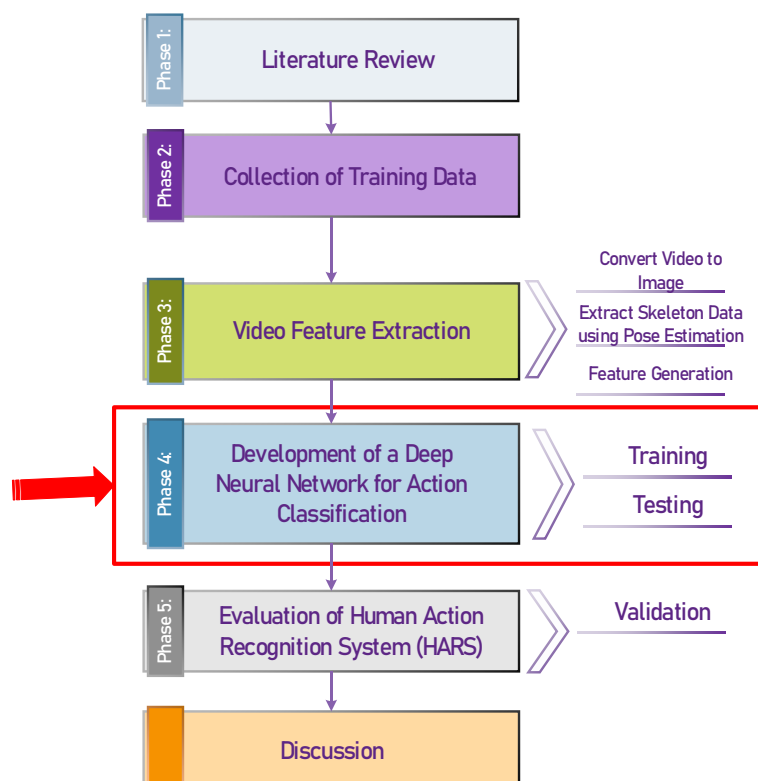


Figure 5.1: Overall Study Design

5.2 Software environment for the development of the Deep Neural Network (DNN)

The software selected for the development and implementation of the DNN was Scikit-learn. Scikit-learn is an open-source machine learning library for the Python programming language. The library features various classification, regression and clustering algorithms. These include support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to operate with the Python numerical and scientific libraries NumPy and SciPy[90].



Figure 5.2: Scikit-Learn Logo

5.3 Training and testing datasets

The dataset used in this research project was created by combining multiple datasets. It was necessary to create a diverse dataset that would include varying lighting conditions, different camera angles, and humans that vary in height and size. The data sets were:

- Joint-annotated Human Motion Data Base (JHMDB)
- North-western-UCLA Multiview Action 3D Dataset
- AAMAZ Human Action Recognition Dataset

Table 5.1 shows that the datasets provided 13 811 samples that can be utilised for training and 5919 testing the DNN.

Table 5.1: Dataset Train-Test Split

Train-Test Split	
Number of training samples	13 811
Number of testing samples	5 919

5.4 Development of the neural network

The development of the DNN was implemented using Scikit-learn. The model used for the development, was the Multi-layer Perceptron (MLP). MLP is a supervised learning algorithm that learns a function $f(\cdot) : R^m \rightarrow R^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. This model used the generated features from the skeleton data which are represented as $X = x_1, x_2, \dots, x_m$ and a target, which is the classes, y to learn a non-linear function approximator for classification. The model has more than one non-linear layer between the input and the output layer, called hidden layers. Figure 5.3 shows a one hidden layer MLP with scalar output[91].

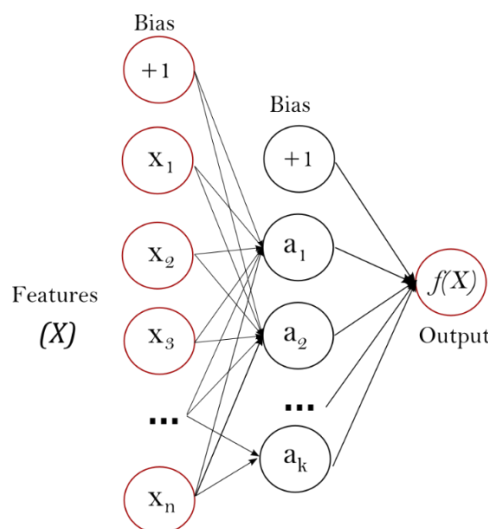


Figure 5.3: One hidden layer MLP[91]

When a multilayer perceptron has a linear activation function in all neurons, i.e. a linear function that maps the weighted inputs to the output of each neuron, then linear algebra shows that any number of layers can be reduced to an input-output two-layer model. In MLPs some of the neurons use a non-linear activation function that was designed to model the frequency of action potentials, or firing, of biological neurons. In deep learning the most used activation function is the Rectifier Linear Unit (ReLU), which is defined as $f(x) = \max(0, x)$ and can be plotted as seen in Figure 5.4.

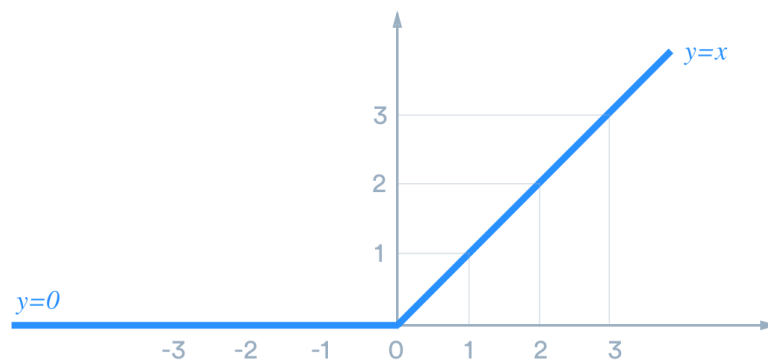


Figure 5.4: Plot of ReLu activation function[92]

The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. ReLU has just enough non-linearity to be almost as simple as a linear activation, yet this non-linearity allows for tremendously complicated representations. Because, unlike in the linear situation, the more non-linear ReLUs you stack, the more non-linear it becomes.

5.4.1 Learning

For classification, the MLPClassifier class was used from the scikit-learn library. Training takes place in the perceptron by adjusting the link weights after processing each piece of data, depending on the amount of error, in the output relative to the expected result. This is a supervised learning approach, and it is achieved through backpropagation, a generalisation of linear perceptron of the least mean

squares algorithm. The degree of error in an output node j in the n^{th} data point can be represented by Equation 6

Equation 6: Error Function

$$e_j(n) = d_j(n) - y_j(n)$$

where d = the target value

y = the value produced by the perceptron

The weights of the node can be modified based on corrections that decrease the error in the output, given by

Equation 7

$$\varepsilon(n) = \frac{1}{2} \sum_j e_j^2(n)$$

The adjustment in each weight is by gradient descent.

Equation 8: Gradient Descent

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial v_j(n)} y_i(n)$$

Where y_i = output of the previous neuron

η = the learning rate

v_j = the induced local field

5.4.2 Parameters

These are the parameters that were used to configure the MLPClassifier in the learning process. See *Table 5.2* for the values and description for each parameter.

Table 5.2: Parameters used to Initialise MLPClassifier

Parameter	Value	Description
hidden_layer_sizes	(100,)	The <i>i</i> th element represents the number of neurons in the <i>i</i> th hidden layer.
activation	relu	Activation function for the hidden layer
solver	adam	The solver for weight optimization. 'adam' refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba[93]
alpha	0.0001	L2 penalty (regularization term) parameter
batch_size	auto	Learning rate schedule for weight updates
learning_rate	constant	'constant' is a constant learning rate given by 'learning_rate_init'.
learning_rate_init	0.001	The initial learning rate used. It controls the step-size in updating the weights.
max_iter	200	Maximum number of iterations. The solver iterates until convergence (determined by 'tol') or this number of iterations.
shuffle	True	Whether to shuffle samples in each iteration.
random_state	None	Determines random number generation for weights and bias initialisation, train-test split if early stopping is used, and batch sampling when solver is adam optimiser.
tol	1e-4	Tolerance for the optimisation

beta_1	0.9	Exponential decay rate for estimates of first moment vector in adam
beta_2	0.999	Exponential decay rate for estimates of second moment vector in adam
epsilon	1e-8	Value for numerical stability in adam
n_iter_no_change	10	Maximum number of epochs to not meet to improvement

5.5 Results

To prove that the set parameters were tuned correctly, after **200 epochs** of training an accuracy of **93%** was achieved on the training set and **83%** accuracy on the testing set. There further is an accuracy report (illustrated in Table 5.3) and confusion matrix (illustrated in Figure 5.5) detailing the performance and accuracy of the DNN training.

5.5.1 Confusion matrix

The confusion matrix is a table used to illustrate the performance of a DNN, using the test data for which the true values are known. Figure 5.5 depicts a confusion matrix of the developed DNN. The confusion takes a tally of the number of samples that the DNN classified correctly and incorrectly. They vertical axis is the **true label**, and the horizontal axis is the **predicted label**. In the kick action the DNN predicted 424 samples correctly and 180 samples incorrectly (9 as stand, 73 as walk, 43 as run, 41 as jump and 14 as sit, all of them being incorrect predictions).

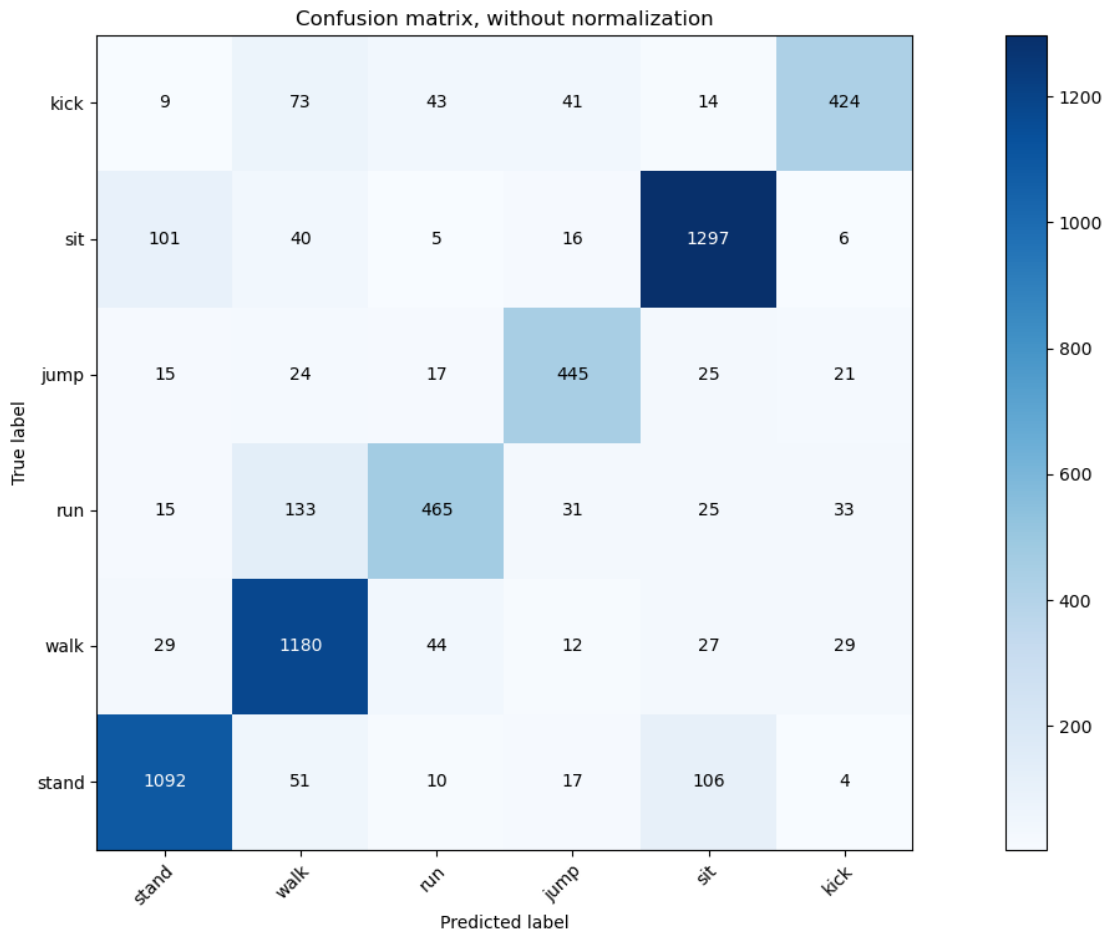


Figure 5.5: Confusion Matrix

Based on the data presented above, the sit, walk and stand actions have more samples in the test dataset compared to the kick, jump, and run actions. The DNN therefore was able to predict a higher number of samples correctly regarding the sit, walk and stand actions. The matrix also shows that the DNN classified 133 samples from the run action as walking. This could be a result of there being fewer samples, and the walk action being similar to the run action. Overall, the performance of the DNN is consistent and it is able to predict most of the samples from the test set correctly.

5.5.2 Accuracy report

The accuracy report (Table 5.3) displays the results for precision, recall, F1-score, and support. In addition, it also reports the macro-average and weighted average for each action. The precision column is the ratio $tp/(tp + fp)$ where tp represents the number of true positives and fp represents the

number of false positives. Precision is instinctively the ability of the classifier not to label a negative sample as positive for that specific action. In this case of action classification, precision illustrates the number of correct results divided by the number of all returned results. Precision takes all retrieved video features into account. The recall column is the ratio $tp / (tp + fn)$ where tp represents the number of true positives and fn represents the number of false negatives. The recall is the instinctive ability of the classifier to find all the positive samples. In this case of action classification, recall is the number of correct results divided by the number of results that should have been returned. This can also be referred to as the sensitivity of the DNN. The precision and recall score are mostly above 80%. This proves that the DNN returns are more correct than incorrect.

Table 5.3: Accuracy Report

Action	Precision	Recall	F1-score	Support
Stand	0.85	0.84	0.84	1280
Walk	0.80	0.88	0.84	1321
Run	0.78	0.68	0.73	702
Jump	0.85	0.80	0.82	547
Sit	0.88	0.89	0.88	1465
Kick	0.76	0.77	0.76	604
Accuracy			0.83	5919
Macro Average	0.82	0.81	0.81	5919
Weighted Average	0.83	0.83	0.83	5919

The F1-score column is a measure of a test's accuracy for each column. It is calculated from the precision and the recall scores. The score can be understood as a weighted harmonic mean of the precision and recall. The highest possible value for F1-score is 1, meaning the DNN has perfect precision and recall, and the worst score is 0. The support column is the number of occurrences of each class. Macro-average is the average of the unweighted mean per label, and the weighted average is the average of the support-weighted mean per label.

Based on the accuracy, report the DNN gives a consistent performance of above 80% across all action classes, except for 'run' and 'kick'. The DNN could find it difficult to differentiate between walking and running, as they tend to be similar in motion. The score for the run action class could be lower, because it had the smallest number of data samples for training.

5.6 Discussion

The purpose of this chapter was to develop a deep neural network (DNN) to classify action features. The chosen classifier was an MLP classifier which was developed using scikit learn libraries and the Python programming language. The DNN was trained to use data samples generated in Phase 3 of the study. There were three main datasets were selected for training, namely Joint-annotated Human Motion Data Base (JHMDB), North-western-UCLA Multiview Action 3D Dataset and AAMAZ Human Action Recognition Dataset.

During the development of the DNN there are many parameters that need to be set. These parameters influence how the network learns and eventually will perform. Important parameters to consider are the activation function, optimisation function, learning rate, etc. (See *Table 5.2*) The neural network was trained, and after 200 epochs it achieved an accuracy rate of 93% on the training set and 83% on the testing set. Further details on the performance are presented on the confusion matrix and accuracy report.

The overall result of the developed neural network is positive and demonstrates that the network can classify the generated features with great accuracy. However, the network struggles to classify the run action. When the results are analysed, it is clear that the network misclassifies the run action as walk. This could be a result of the actions having a similar pose. More training data can be added to the dataset to try and solve this issue.

Chapter 6: Evaluation of the Human Action Recognition System

6.1 Introduction

Phase 5 of this study focused on evaluating the Human Actions Recognition System (HARS). Additional videos were recorded to evaluate the HARS. This step was performed to validate the results produced during the training and testing. The aim of this phase was also to determine how effective HARS would be in real-life applications. Figure 6.1 provides a flow diagram of the overall study design, highlighting Phase 3 which involved the evaluation of the Human Action Recognition System.

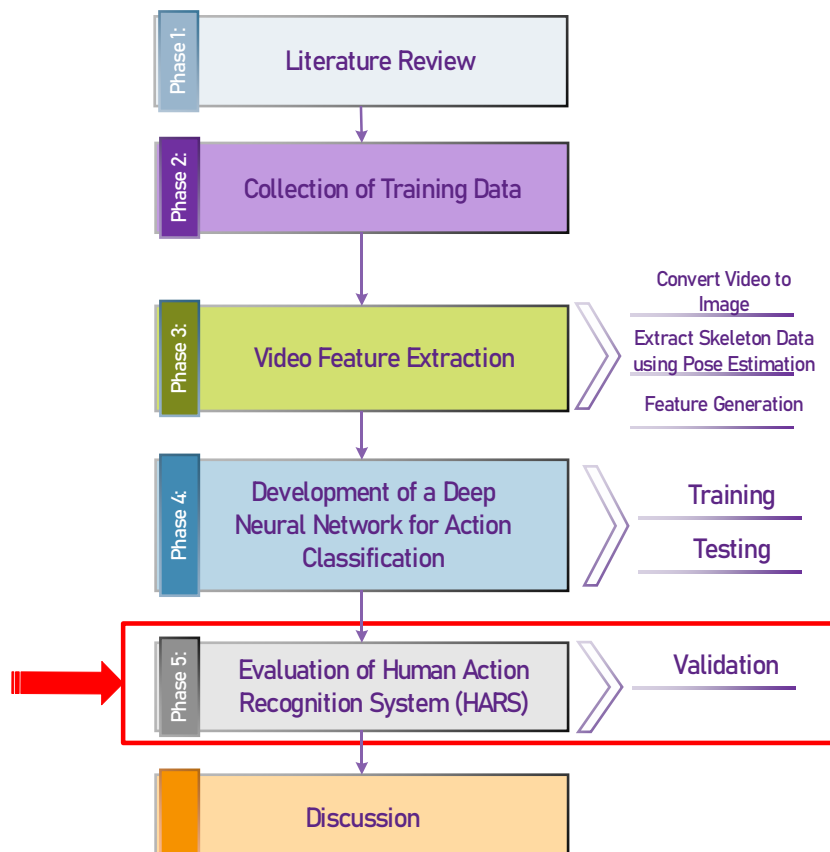


Figure 6.1: Overall study design

The HARS is made up of mainly two parts, namely the video feature extraction and the deep neural network (DNN). The system takes a video as an input. The video is then processed, and features are generated. The generated features are classified using the DNN and a prediction score is displayed. The video feature extraction component has been tested, and the DNN has been trained and tested. The results are in Chapters 4 and 5. Figure 6.2 displays the logical diagram of the HARS and the multiple components that make up the system.

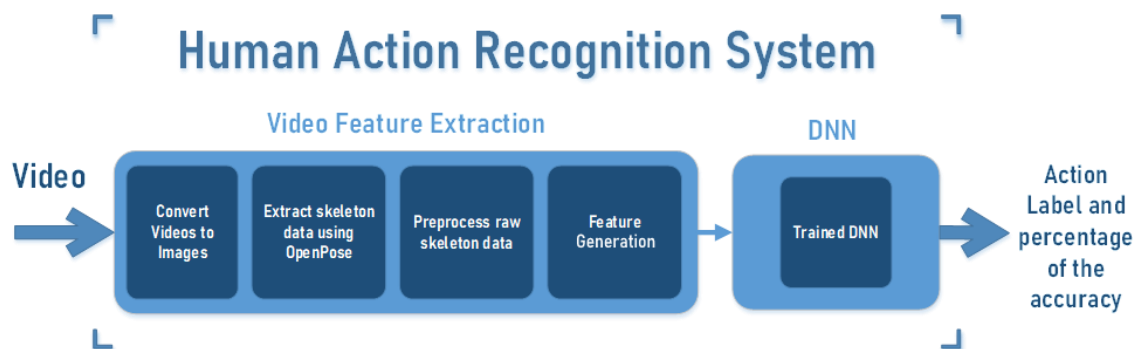


Figure 6.2: Logical diagram of the HARS system

6.2 Validation

The two parts that make up the HARS, i.e. the video feature extraction and the DNN have been tested individually and the results are displayed and discussed in Chapters 4 and 5. In this chapter the two parts are combined into one system, and it is tested on new data to see if the system performs consistently.

For the validation test, three test subjects were selected, and video recorded while performing all six actions the system is able to classify. The videos were recorded in a room under LED lighting, using a 720p webcam. Test 1 contains the results of the first subject, Test 2 the results of the second and test 3 the results of the third subject.

6.2.1 Test 1

The purpose of this validation test is to evaluate how the HARS performs on the first test subject. The subject performs all six actions the system is able to detect. The subject performs the same action multiple times, changing the distance from the camera, position within the frame and the direction in which they face the camera.

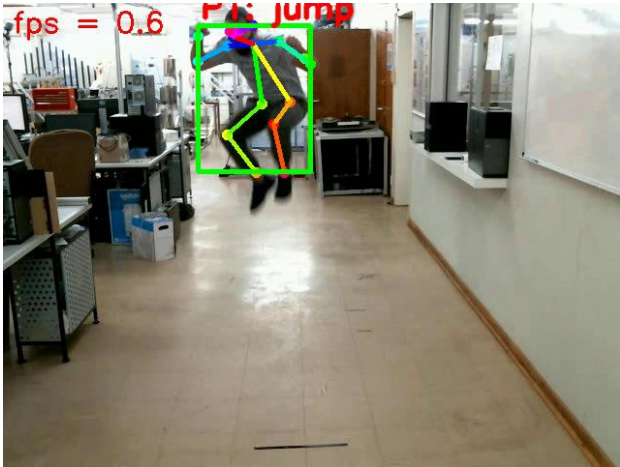
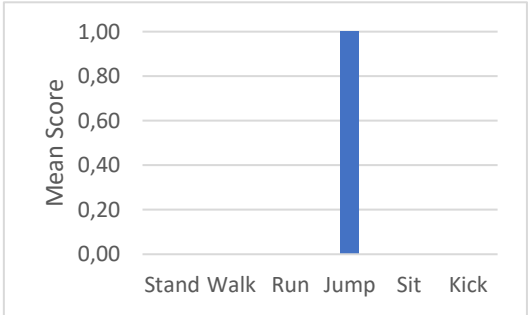
Video Snapshot	Result
 <p data-bbox="389 1223 687 1256">Figure 6.3 Test 1: Jump</p>	 <p data-bbox="1003 1193 1276 1261">Predicted label: Jump Mean score: 100%</p>

Figure 6.3 shows a video snapshot of the jump action and on the right-hand side are the results. The system was able to predict the action correctly with 100% accuracy. The system also predicted the action consistently as the test was repeated five times with the same positive result.

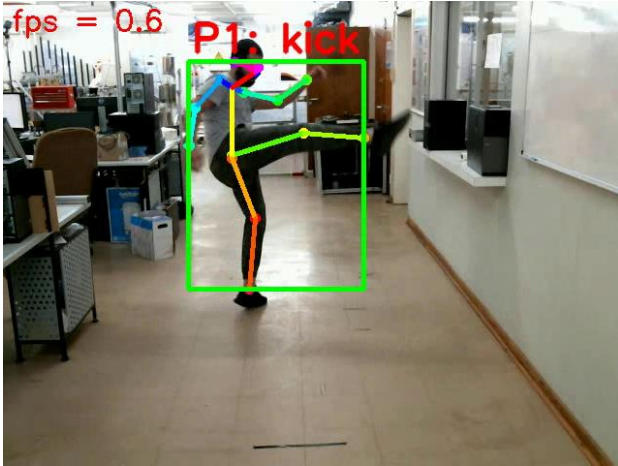
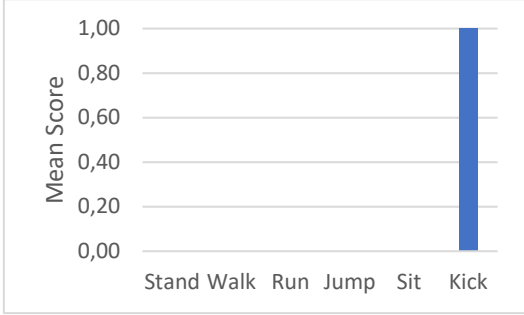
Video Snapshot	Result
 <p data-bbox="395 797 679 831">Figure 6.4 Test 1: Kick</p>	 <p data-bbox="1011 712 1270 779">Predicted label: Kick Mean score: 100%</p>

Figure 6.4 shows a video snapshot of the kick action with, on the right-hand side the results. The system was able to predict the action correctly with 100% accuracy. The system also predicted the action consistently as the test was repeated five times with the same positive result.

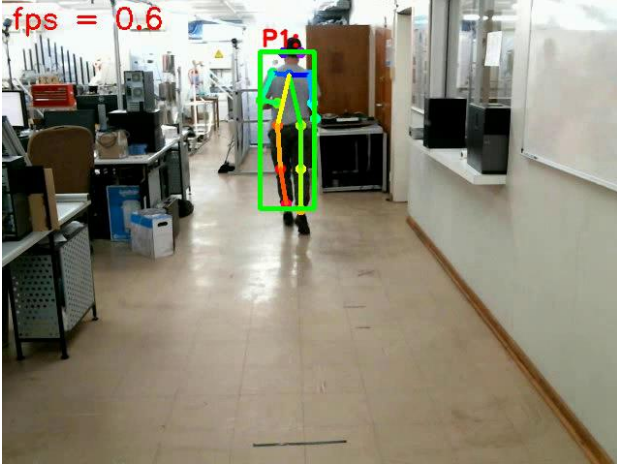
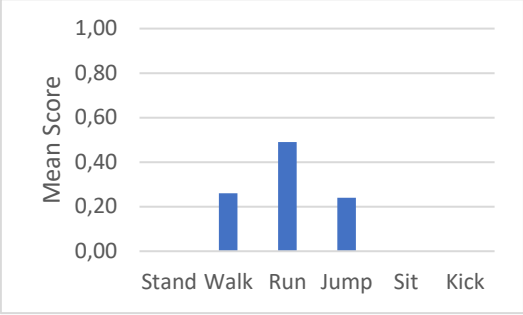
Video Snapshot	Result
 <p data-bbox="395 795 678 833">Figure 6.5 Test 1: Run</p>	 <p data-bbox="1013 712 1268 779">Predicted label: Run Mean score: 49%</p>

Figure 6.5 shows a video snapshot of the run action with the results on the right-hand side. The system was able to predict the action correctly with 49% accuracy. The system also predicted 26% walk and 24% jump. The system predicted the run action inconsistently as the test was repeated five times with the results always fluctuating. This result is in line with the results seen in the previous chapter, because of the lower number of training samples of the run action and the fact that it is similar to the walk action in motion, The system tends to struggle to classify it with great accuracy.

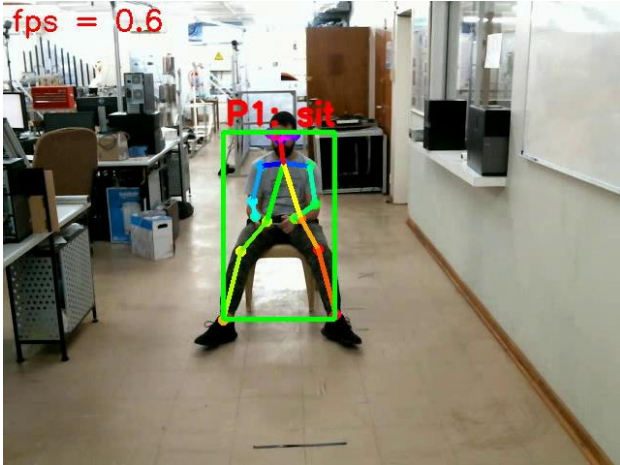
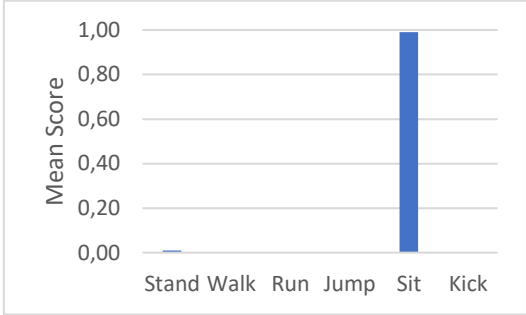
Video Snapshot	Result
 <p data-bbox="405 792 671 831">Figure 6.6 Test 1: Sit</p>	 <p data-bbox="1018 741 1262 808">Predicted label: Sit Mean score: 99%</p>

Figure 6.6 shows a video snapshot of the sit action with, on the right-hand side, the results. The system was able to predict the action correctly with 99% accuracy. The system also predicted the action consistently as the test was repeated five times with the same positive result.

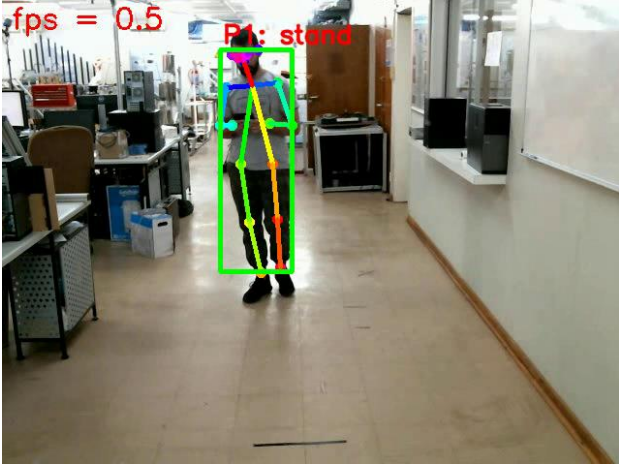
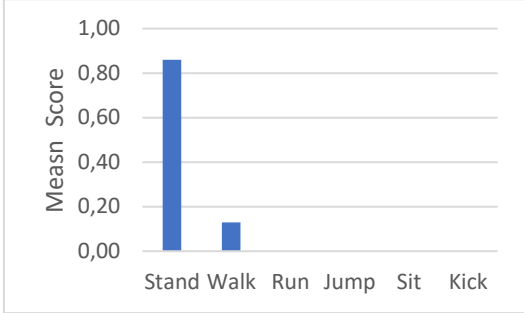
Video Snapshot	Result
 <p data-bbox="384 795 687 833">Figure 6.7 Test 1: Stand</p>	 <p data-bbox="1002 707 1278 779">Predicted label: Stand Mean score: 86%</p>

Figure 6.7 shows a video snapshot of the stand action with, on the right-hand side, the results. The system was able to predict the action correctly with 86% accuracy. The system also predicted 13% walk action which is not an anomaly, because the pose estimation can be similar. This can be ignored, because the stand action was consistently predicted as the test was repeated five times with the similar positive result.


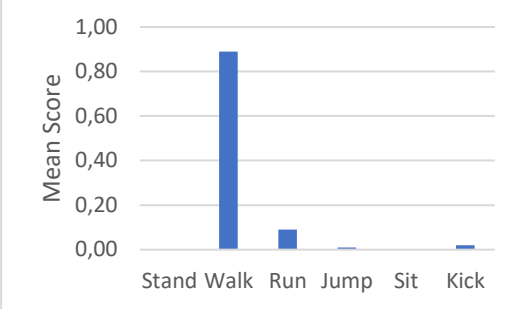
Video Snapshot	Result
 <p data-bbox="391 792 683 831">Figure 6.8 Test 1: Walk</p>	 <p data-bbox="1007 674 1273 741">Predicted label: Walk Mean score: 89%</p>

Figure 6.8 shows a video snapshot of the walk action with, on the right-hand side, the results. The system was able to predict the action correctly with 89% accuracy. The system also predicted 9% run, 1% jump and 2% kick. These values are insignificant, as the system was able to predict the correct action consistently as the test was repeated five times with the same positive result.

6.2.2 Test 2

The purpose of this validation test is to evaluate how the HARS performs on the second test subject. The subject, as in the first test, performs all six actions the system is able to detect. The subject performs the same action multiple times, changing the distance from the camera, the position within the frame and the direction in which they face the camera.

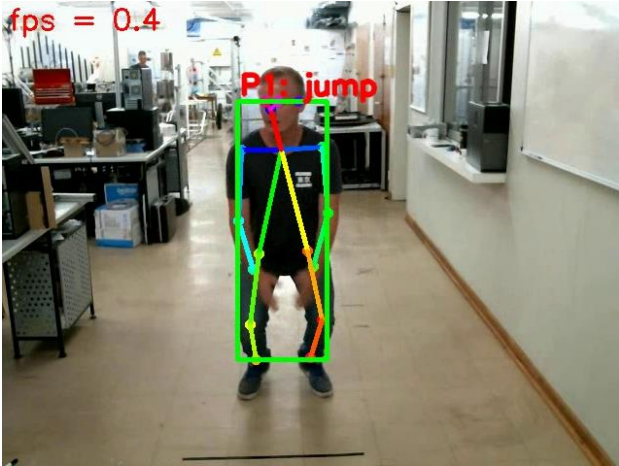
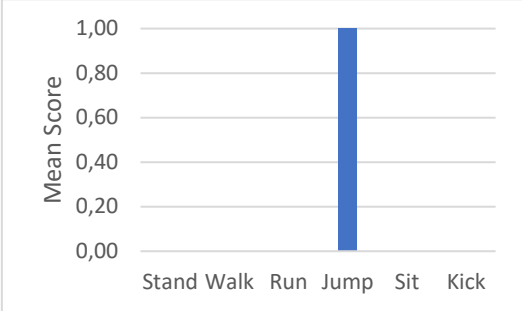
Video Snapshot	Result
 <p data-bbox="386 795 687 833">Figure 6.9 Test 2: Jump</p>	 <p data-bbox="1002 707 1276 779">Predicted label: Jump Mean score: 100%</p>

Figure 6.9 shows a video snapshot of the jump action with, on the right-hand side, the results. The system was able to predict the action correctly with 100% accuracy. They system also predicted the action consistently, as the test was repeated five times with the same positive result.

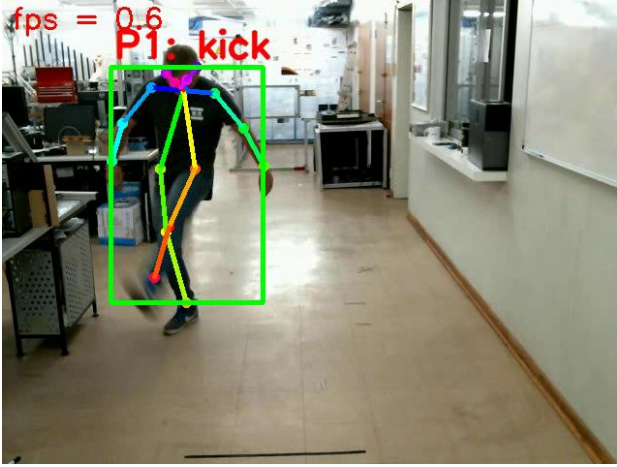
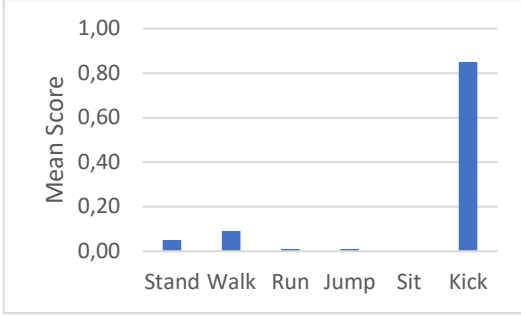
Video Snapshot	Result
 <p data-bbox="386 795 689 833">Figure 6.10 Test 2: Kick</p>	 <p data-bbox="1008 712 1268 779">Predicted label: Kick Mean score: 85%</p>

Figure 6.10 shows a video snapshot of the kick action with, on the right-hand side, the results. The system was able to predict the action correctly with 85% accuracy. The system also predicted 6% stand and 9 % walk. Overall, the system predicted the correct action consistently, as the test was repeated five times with the same positive result.

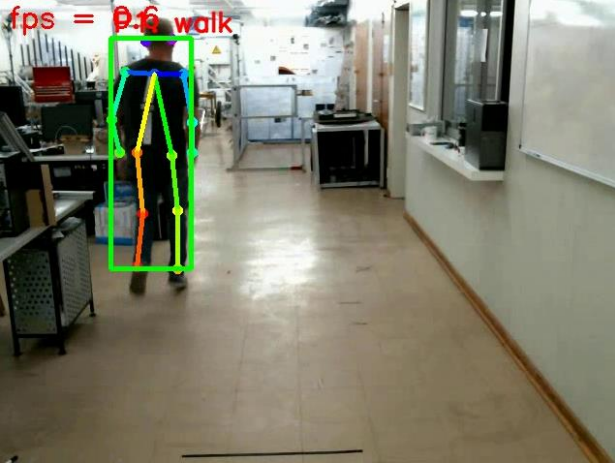
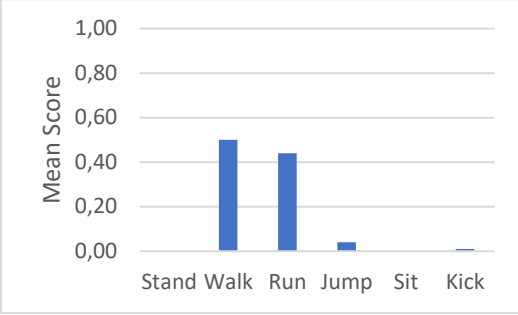
Video Snapshot	Result
 <p data-bbox="391 795 686 833">Figure 6.11 Test 2: Run</p>	 <p data-bbox="1005 705 1276 779">Predicted label: Walk Mean Score: 50%</p>

Figure 6.11 shows a video snapshot of the run action with, on the right-hand side, the results. The system was unable to predict the action correctly. It predicted 50% walk, 44% run and 5% jump. The system predicted the run action inconsistently, as the test was repeated five times with the results varying. This result is in line with the results seen in the previous chapter, because of the lower number of training samples of the run action. In addition, the run action is similar to the walk action in motion, a fact causing the system to struggle to classify it with great accuracy.

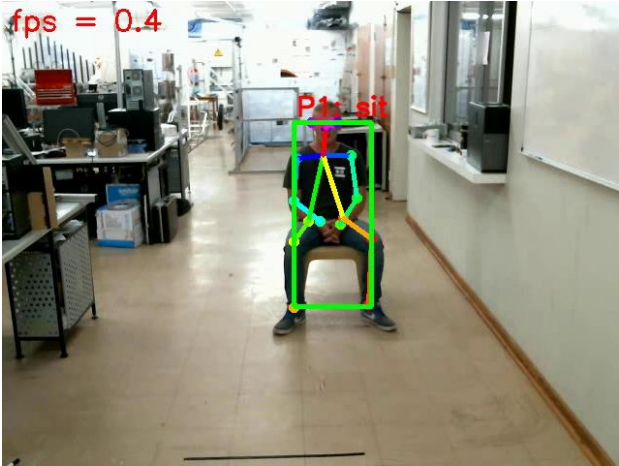
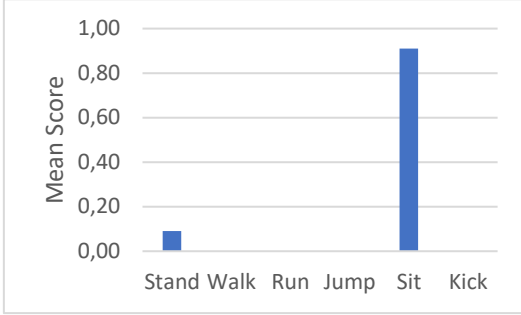
Video Snapshot	Result
 <p data-bbox="397 795 676 833">Figure 6.12 Test 2: Sit</p>	 <p data-bbox="1018 707 1259 779">Predicted label: Sit Mean score: 91%</p>

Figure 6.12 shows a video snapshot of the sit action with, on the right-hand side, the results. The system was able to predict the action correctly with 91% accuracy. The system also predicted 9% stand action. This can be ignored, because the stand action was consistently predicted as the test was repeated five times with the similar positive result.

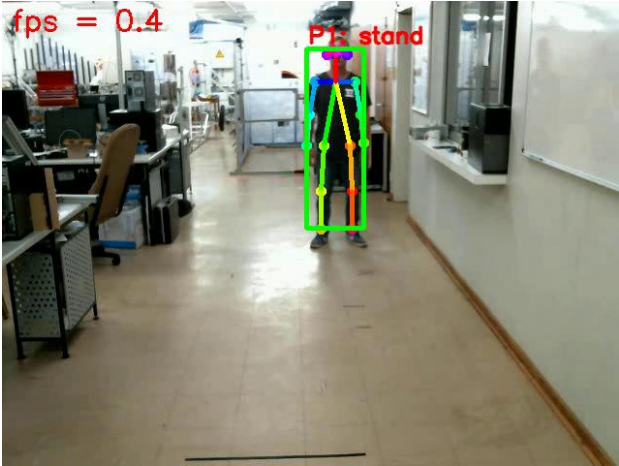
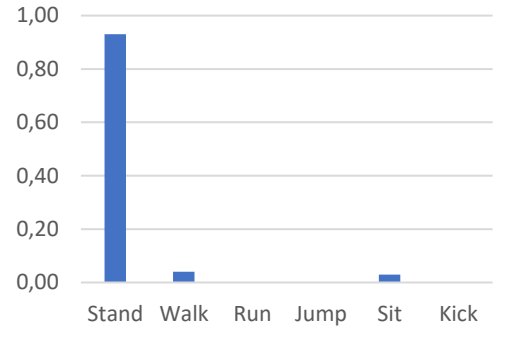
Video Snapshot	Result
 <p data-bbox="379 797 695 831">Figure 6.13 Test 2: Stand</p>	 <p data-bbox="1002 730 1278 801">Predicted label: Stand Mean score: 93%</p>

Figure 6.13 shows a video snapshot of the stand action with, on the right-hand side, the results. The system was able to predict the action correctly with 93% accuracy. The system also predicted 4% walk action, which is not an anomaly because the pose estimation can be similar. This can be ignored, because the stand action was consistently predicted as the test was repeated five times with the similar positive result.

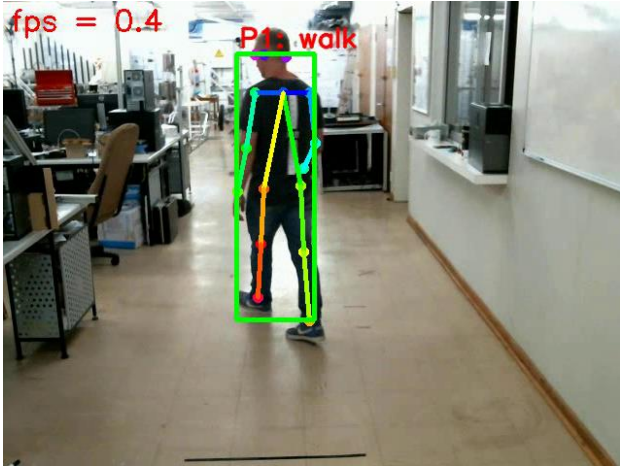
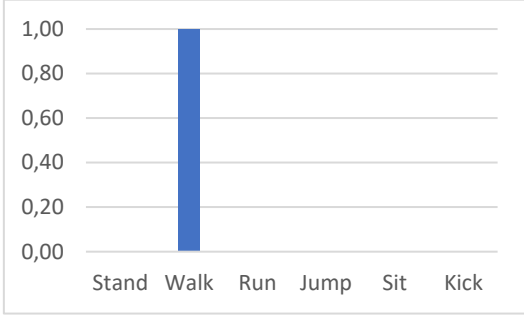
Video Snapshot	Result
 <p data-bbox="384 797 691 835">Figure 6.14 Test 2: Walk</p>	 <p data-bbox="1007 712 1273 779">Predicted label: Walk Mean score: 100%</p>

Figure 6.14 shows a video snapshot of the walk action with, on the right-hand side, the results. The system was able to predict the action correctly with 100% accuracy. They system also predicted the action consistently, as the test was repeated five times with the same positive result.

6.2.3 Test 3

The purpose of this validation test is to evaluate how the HARS performs on the third test subject. The subject, as in the first and second tests, performs all six actions the system is able to detect. The subject performs the same action multiple times, changing the distance from the camera, the position within the frame and the direction in which they face the camera.


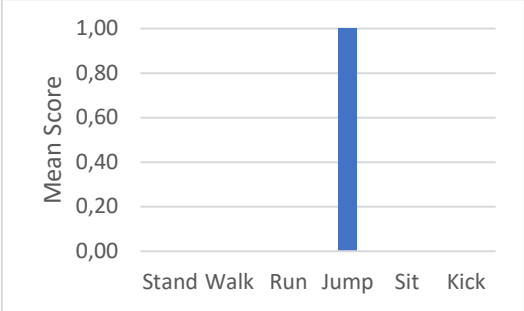
Video Snapshot	Result
 <p data-bbox="379 797 692 831">Figure 6.15 Test 3: Jump</p>	 <p data-bbox="1003 710 1273 779">Predicted label: Jump Mean score: 100%</p>

Figure 6.15 shows a video snapshot of the jump action with, on the right-hand side, the results. The system was able to predict the action correctly with 100% accuracy. The system also predicted the action consistently, as the test was repeated five times with the same positive result.

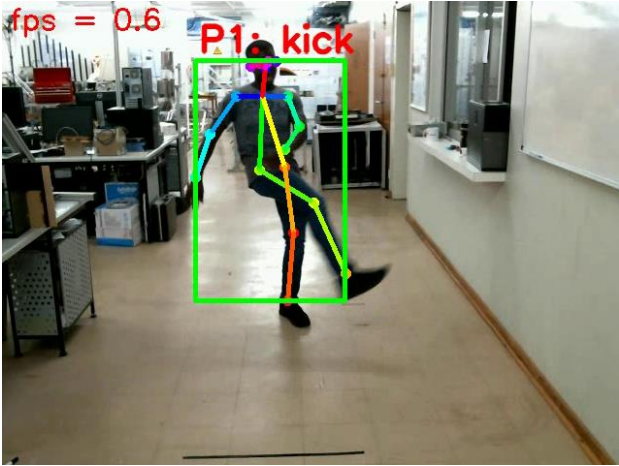
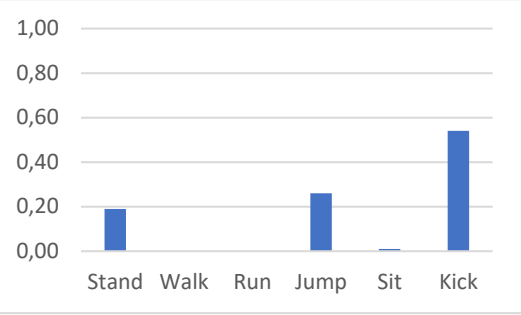
Video Snapshot	Result
 <p data-bbox="387 795 687 831">Figure 6.16 Test 3: Kick</p>	 <p data-bbox="1011 712 1270 779">Predicted label: Kick Mean score: 54%</p>

Figure 6.16 shows a video snapshot of the kick action with, on the right-hand side, the results. The system was able to predict the action correctly with 54% accuracy. The system also predicted 19% stand, 26% jump and 1% sit. Overall, the system predicted the correct action consistently, as the test was repeated five times with a similar positive result.

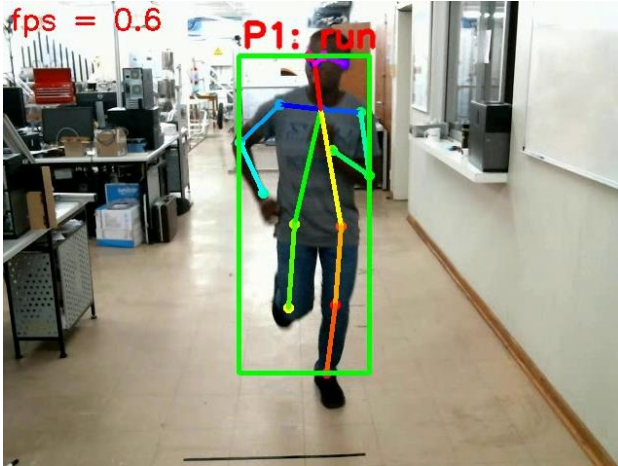
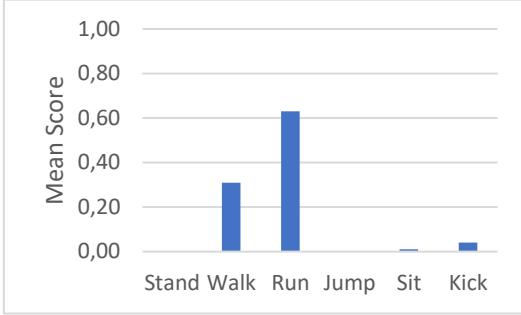
Video Snapshot	Result
 <p data-bbox="389 797 683 831">Figure 6.17 Test 3: Run</p>	 <p data-bbox="1011 712 1267 779">Predicted label: Run Mean score: 63%</p>

Figure 6.17 shows a video snapshot of the run action with, on the right-hand side, the results. The system was able to predict the action correctly with an accuracy of 63%. It also predicted 31% walk, 4% kick and 1% sit. The system continues to predict the run action inconsistently, as the test was repeated five times with the results always varying.

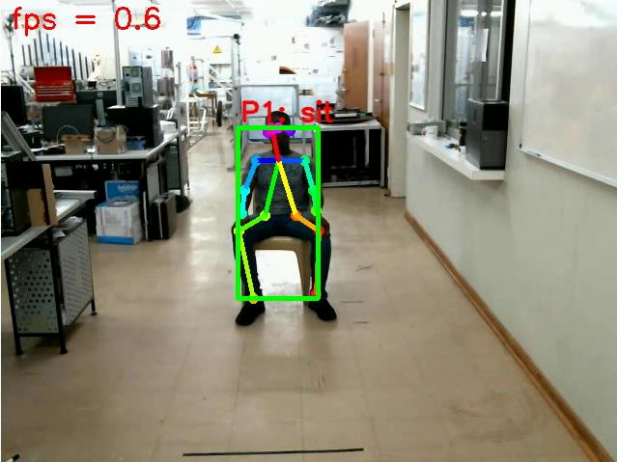
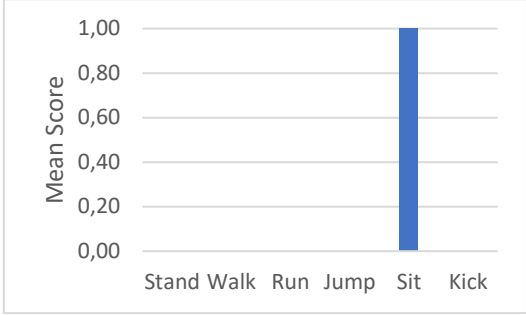
Video Snapshot	Result
 <p data-bbox="395 795 678 833">Figure 6.18 Test 3: Sit</p>	 <p data-bbox="1018 712 1262 779">Predicted label: Sit Mean score: 100%</p>

Figure 6.18 shows a video snapshot of the sit action with, on the right-hand side, the results. The system was able to predict the action correctly with 100% accuracy. They system also predicted the action consistently, as the test was repeated five times with the same positive result.

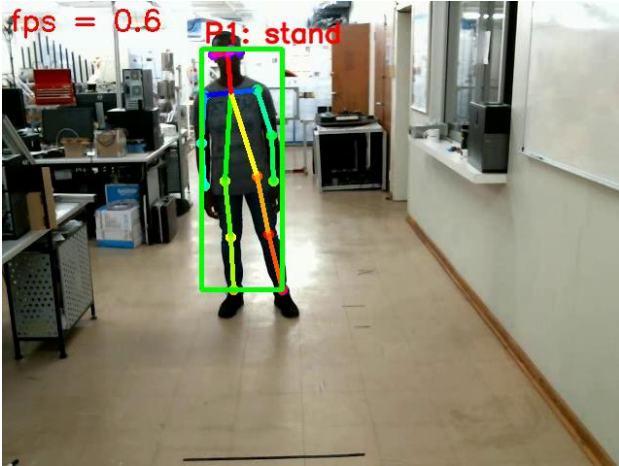
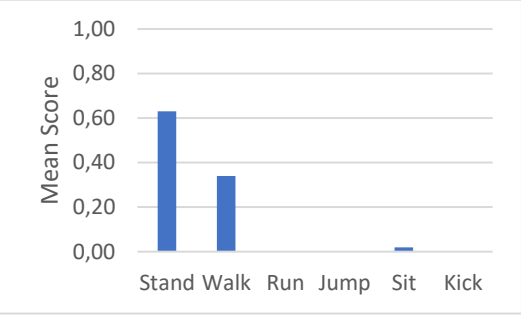
Video Snapshot	Result
 <p data-bbox="379 797 695 831">Figure 6.19 Test 3: Stand</p>	 <p data-bbox="1002 712 1278 779">Predicted label: Stand Mean score: 63%</p>

Figure 6.19 shows a video snapshot of the stand action with, on the right-hand side, the results. The system was able to predict the action correctly with 63% accuracy. The system also predicted 34% walk action which is not an anomaly, because the pose estimation can be similar. This can be ignored, because the stand action was consistently predicted as the test was repeated five times with the similar positive result.

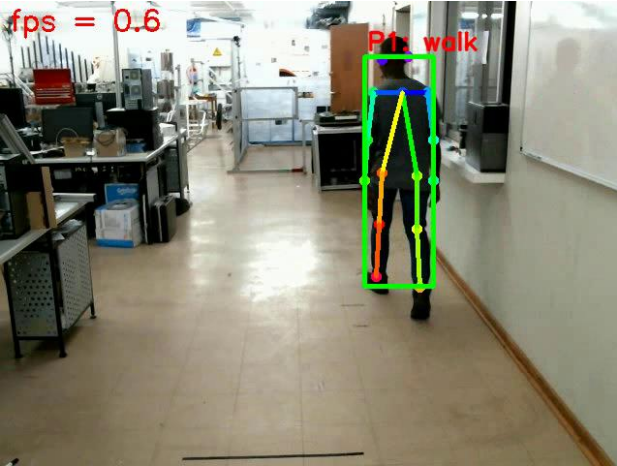
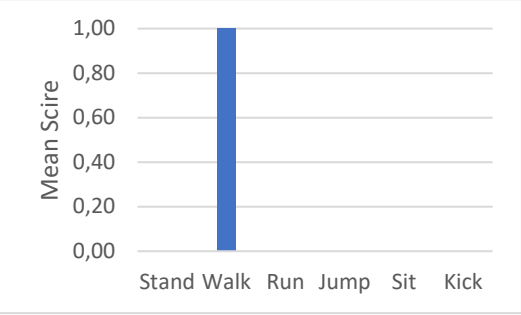
Video Snapshot	Result
 <p data-bbox="384 797 691 831">Figure 6.20 Test 3: Walk</p>	 <p data-bbox="1007 712 1273 779">Predicted label: Walk Mean score: 100%</p>

Figure 6.20 shows a video snapshot of the walk action with, on the right-hand side, the results. The system was able to predict the action correctly with 100% accuracy. They system also predicted the action consistently, as the test was repeated five times with the same positive result.

6.3 Discussion

The purpose of this chapter was to evaluate the Human Actions Recognition System (HARS). This was achieved by taking three human subjects performing three tests. The tests were designed to evaluate each human action the system is able to classify and see how the HARS would perform overall. Each human subject was requested to perform all six actions (jump, kick, run, sit, stand, walk) to see if the system would be successful at recognising the different actions. This was also to demonstrate how the system would function when applied in real-life situations.

The results prove that the videos extraction methods applied, and the deep neural networks are functioning as expected, and can extract features and classify them with a high accuracy. However,

the system still has challenges in classifying the run action. This was apparent in all three tests. There are moments where the system can classify the run actions with high accuracy (above 80%), but in many cases it confuses the action with a walk action. This could be as result of not having sufficient data samples in the training dataset.

Chapter 7 : Discussion and Conclusion

7.1 Introduction

The purpose of this chapter is to recapitulate the research project, revisit the research goals and objectives, express the contributions made, identify future work to be done and draw conclusions to the research.

Chapter 1 contains a brief introduction to the research project, putting it in perspective by stating the problem, determining the aim and, finally, listing the objectives of the study. In Chapter 2, the study design, a description of the different aspects of this research project is presented. The project was divided into five phases. The phases include literature review, collection of training data, video feature extraction, development of the deep neural network and evaluation of the Human Action Recognition System (HARS). Chapter 3 contains a review of the field of study, which includes various aspects of video surveillance systems used in our modern and age. The study further reviews literature on pose estimation, video feature extraction techniques and deep neural networks, and action video datasets. Chapter 4 documents the methodologies followed to develop video feature extraction. It specifies the software development tools utilised to design and develop the video feature extraction process, and further stipulates how these functions integrate with the HARS. Lastly, it illustrates the results generated by the videos feature extractor which will be used in further phases of the project. Chapter 5 documents the development of the deep neural network which was used to predict the human action. It further specifies the neural network tools and environment utilised, the performance of the neural network and how it integrates with the HARS. Chapter 6 presents tests and procedures that were identified to evaluate the HARS, as well as their results.

7.2 Human action recognition system

The objective of the study was to create an intelligent vision system that can identify a range of human actions within surveillance videos. This would offer security officers additional data of activities occurring in the videos and enable security officers to access specific incidents faster and provide early detections of crimes. Training data from existing video datasets was gathered. In addition, images were extracted from the video datasets. From the images, features were extracted and used to train the neural network to classify the action features. Finally, an evaluation of the neural network from known data (labelled data) was performed to test for real world applications.

These objectives were accomplished by firstly selecting software tools and developing the software modules that are collectively required to build the entire system. The system was built using methods discussed in Chapters 4 and 5. Afterwards, tests were established in Chapter 6 and were set up in such a way to verify each separate component of the system. After the results had been obtained, the results were analysed and discussed in Chapters 4, 5 and 6. These chapters delivered satisfactory results regarding each module and the operation of the system. They also highlighted the system's shortfalls and areas that can be researched in the future. Most importantly, the results verify the operation of the system and that it can be utilised in real-world applications.

A primary challenge of this study was insufficient data. Datasets should include sufficient scale ranges, occlusion, intra and inter-class differences, etc., to be able to develop efficient techniques. However, that is not that case with most human action recognition datasets. Most datasets consist of a limited number of labelled training and test samples. This is because the task of annotating a large dataset is time consuming. This results in neural networks performing poorly because of a lack of data. This was highlighted in Chapters 5 and 6, as the run action classification performed inconsistently because of insufficient data.

A secondary challenge of this study was occlusion. Pose estimation was the technique utilised to extract data about the human in the frame. This algorithm has limitations. It has difficulty detecting the human if there is an object blocking or partially blocking the camera's view. As a result, after an occlusion has occurred, the system will wrongly identify the initially tracked object as a new object.

7.3 Concluding remarks and future work

The purpose of this research project was to design a human action recognition system able to detect jumping, kicking, running, sitting, standing, and walking in videos. In designing and developing a human action recognition system (HARS), many of the challenges posed by typical video surveillance systems, were circumvented. Using a HARS within an intelligent surveillance system will allow the system to collect metadata on events that occur within a video scene. The HARS will also contribute to crime prevention, as the data it collects, can be used towards detecting suspicious behaviour. Further studies are necessary to tackle the challenges the system still faces. There is a need to develop a system to automate the process of collecting data, as there is a need for more training data. Pose estimation techniques are approaching the state of the art in computer vision. These methods have concrete applications in human action recognition. However, due to occlusion of joints and anomalous angles, further work needs to be done to provide solutions. The HARS is only capable of classifying six action classes. Further studies are necessary to determine how the system would perform with additional action classes with more complexity.

References

- [1] T. (CSIR) Kruger, L. Lancaster, K. (CSIR) Landman, S. (CSIR) Liebermann, A. (The institute for S. S. I. Louw, and R. (The institute for S. S. I. Robertshaw, *Making South Africa Safe A Manual for Community-based Crime Prevention*. 2016.
- [2] B. Bowman, S. Kramer, S. Salau, E. Kotze, and R. Matzopoulos, "Linking criminal contexts to injury outcomes: findings and lessons from a national study of robbery in South Africa," *Int. J. Public Health*, vol. 63, no. 8, pp. 977–985, 2018, doi: 10.1007/s00038-018-1129-z.
- [3] "P0340 Governance and Access to Justice GOVERNANCE , PUBLIC SAFETY AND JUSTICE SURVEY Governance , Public Safety and Justice Survey : 2018 / 19," 2019.
- [4] G. D. Breetzke, K. Landman, and E. G. Cohn, "Is it safer behind the gates? Crime and gated communities in South Africa," *J. Hous. Built Environ.*, vol. 29, no. 1, pp. 123–139, 2014, doi: 10.1007/s10901-013-9362-5.
- [5] UNODC, *Handbook on the Crime Prevention Guidelines: Making them work*. 2010.
- [6] H. Kruegle, *CCTV Surveillance: Video Practices and Technology*. Elsevier, 2011.
- [7] M. Torabi *et al.*, "Advance Intelligent Video Surveillance System (AIVSS): A Future Aspect," *Intech*, vol. i, p. 13, 2016, doi: <http://dx.doi.org/10.5772/57353>.
- [8] G. Cheng, Y. Wan, A. N. Saudagar, K. Namuduri, and B. P. Buckles, "Advances in Human Action Recognition: A Survey," no. January, 2015, [Online]. Available: <http://arxiv.org/abs/1501.05964>.

- [9] Y. Kong and Y. Fu, "Human Action Recognition and Prediction: A Survey," vol. 13, no. 9, 2018, [Online]. Available: <http://arxiv.org/abs/1806.11230>.
- [10] E. L. Piza, B. C. Welsh, D. P. Farrington, and A. L. Thomas, "CCTV surveillance for crime prevention: A 40-year systematic review with meta-analysis," *Criminol. Public Policy*, vol. 18, no. 1, pp. 135–159, 2019.
- [11] B. Quadros, R. Kadam, K. Saxena, W. Shen, and A. Kobsa, "Dashbell: A Low-cost Smart Doorbell System for Home Use," 2017, [Online]. Available: <http://arxiv.org/abs/1706.09269>.
- [12] "Video Doorbell 3 – Ring ZA." <https://za-en.ring.com/collections/video-doorbells/products/video-doorbell-3> (accessed Jun. 30, 2021).
- [13] S. Rho, W. Rahayu, and U. T. Nguyen, "Intelligent video surveillance in crowded scenes," *Inf. Fusion*, vol. 24, no. C, pp. 1–2, 2015.
- [14] A. Ben Mabrouk and E. Zagrouba, "Abnormal behavior recognition for intelligent video surveillance systems: A review," *Expert Syst. Appl.*, vol. 91, pp. 480–491, 2018, doi: 10.1016/j.eswa.2017.09.029.
- [15] A. C. Nazare and W. Robson Schwartz, "A scalable and flexible framework for smart video surveillance," *Comput. Vis. Image Underst.*, vol. 144, pp. 258–275, 2015, doi: 10.1016/j.cviu.2015.10.014.
- [16] I. Jegham, A. Ben Khalifa, I. Alouani, and M. A. Mahjoub, "Vision-based human action recognition: An overview and real world challenges," *Forensic Sci. Int. Digit. Investig.*, vol. 32, p. 200901, Mar. 2020, doi: 10.1016/j.fsidi.2019.200901.

- [17] J. K. Aggarwal and M. S. Ryoo, "Human Activity Analysis: A Review," *ACM Comput. Surv.* (CSUR, vol. 43, no. 3, pp. 1–43, 2011.
- [18] R. Poppe, "A survey on vision-based human action recognition," *Image Vis. Comput.*, vol. 28, no. 6, pp. 976–990, 2010, doi: 10.1016/j.imavis.2009.11.014.
- [19] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld, and I. Laptev Marcin Marszalek Cordelia Schmid Benjamin Rozenfeld, "Learning Realistic Human Actions from Movies," pp. 1–8, 2008, doi: 10.1109/CVPR.2008.4587756i.
- [20] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior Recognition via Sparse Spatio-Temporal Features." Accessed: Jul. 28, 2020. [Online]. Available: <http://vision.ucsd.edu>.
- [21] H. Wang and C. Schmid, "Action Recognition with Improved Trajectories," 2013, doi: 10.1109/ICCV.2013.441.
- [22] Y. Z. Cheong and W. J. Chew, "The Application of Image Processing to Solve Occlusion Issue in Object Tracking," doi: 10.1051/mateconf/201815203001.
- [23] N. Ikidler-Cinbis, R. G. Cinbis, and S. Sclaroff, "Learning Actions From the Web," 2009.
- [24] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce, "Automatic Annotation of Human Actions in Video." Accessed: Jul. 31, 2020. [Online]. Available: www.weeklyscript.com.
- [25] S. Zhang, Z. Wei, J. Nie, L. Huang, S. Wang, and Z. Li, "A Review on Human Activity Recognition Using Vision-Based Method," 2017, doi: 10.1155/2017/3090343.
- [26] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-Time Tracking of the Human Body," 1997. Accessed: Aug. 16, 2020. [Online]. Available:

- <http://ppnder.www.media.mit.edu/projects/ppnder/>.
- [27] D. Koller *et al.*, “Towards robust automatic traffic scene analysis in real-time Towards Robust Automatic Traac Scene Analysis in Real-Time,” doi: 10.1109/ICPR.1994.576243.
- [28] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, 1999, vol. 2, pp. 246–252.
- [29] B. D. Lucas and T. Kanade, “Iterative Image Registration Technique With an Application To Stereo Vision.,” vol. 2, pp. 674–679, 1981.
- [30] Xin Lu, Qiong Liu, and S. Oe, “Recognizing non-rigid human actions using joints tracking in space-time,” in *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, 2004, vol. 1, pp. 620-624 Vol.1.
- [31] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, “Actions as space-time shapes,” in *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, 2005, vol. 2, pp. 1395-1402 Vol. 2.
- [32] A. Klaser, M. Marszalek, C. Schmid, A. Kläser, and M. Marszałek, “A Spatio-Temporal Descriptor Based on 3D-Gradients,” 2008. Accessed: Aug. 17, 2020. [Online]. Available: <https://hal.inria.fr/inria-00514853>.
- [33] X. Peng, L. Wang, X. Wang, and Y. Qiao, “Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice,” *Comput. Vis. Image Underst.*, vol. 150, pp. 109–125, 2016, doi: 10.1016/j.cviu.2016.03.013.

- [34] X. Peng, C. Zou, Y. Qiao, and Q. Peng, "LNCS 8693 - Action Recognition with Stacked Fisher Vectors," 2014.
- [35] C. G. Harris, M. Stephens, and others, "A combined corner and edge detector.," in *Alvey vision conference*, 1988, vol. 15, no. 50, pp. 10–5244.
- [36] I. Laptev and T. Lindeberg, "On Space-Time Interest Points *," 2003.
- [37] G. Willems, T. Tuytelaars, and L. Van Gool, "LNCS 5303 - An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector," 2008.
- [38] N. D. and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005.
- [39] W. L. Lu and J. J. Little, "Simultaneous tracking and action recognition using the PCA-HOG descriptor," *Third Can. Conf. Comput. Robot Vision, CRV 2006*, vol. 2006, 2006, doi: 10.1109/CRV.2006.66.
- [40] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld, and I. Laptev Marcin Marszalek Cordelia Schmid Benjamin Rozenfeld, "Learning Realistic Human Actions from Movies," pp. 1–8, 2008, doi: 10.1109/CVPR.2008.4587756i.
- [41] O. Oreifej and Z. Liu, "HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences," 2013, doi: 10.1109/CVPR.2013.98.
- [42] J. Shotton *et al.*, "Real-Time Human Pose Recognition in Parts from Single Depth Images."
- [43] "Kinect for Windows SDK Beta," *Microsoft*, 2011. <https://www.microsoft.com/en-us/research/project/kinect-for-windows-sdk-beta/>.

- [44] J. Sell and P. O'Connor, "The xbox one system on a chip and kinect sensor," *IEEE Micro*, vol. 34, no. 2, pp. 44–53, 2014, doi: 10.1109/MM.2014.9.
- [45] Y. Zhao, Z. Liu, L. Yang, and H. Cheng, "Combing RGB and Depth Map Features for Human Activity Recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 4043–4048, 2018, Accessed: Aug. 18, 2020. [Online]. Available: <http://www.di.ens.fr/>.
- [46] X. Yang, C. Zhang, and Y. Tian, *Recognizing Actions Using Depth Motion Maps-based Histograms of Oriented Gradients*. 2012.
- [47] A. Jalal, S. Kamal, and D. Kim, "A depth video-based human detection and activity recognition using multi-features and embedded hidden Markov models for health care monitoring systems," *Artic. Int. J. Interact. Multimed. Artif. Intell.*, vol. 4, pp. 4–54, 2017, doi: 10.9781/ijimai.2017.447.
- [48] H. Chen, G. Wang, J.-H. Xue, and L. He, "A novel hierarchical framework for human action recognition," 2016, doi: 10.1016/j.patcog.2016.01.020.
- [49] J. Wang, S. Member, Z. Liu, S. Member, Y. Wu, and J. Yuan, "Learning Actionlet Ensemble for 3D Human Action Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, 2014, doi: 10.1109/TPAMI.2013.198.
- [50] S. Amir, G. Wang, T.-T. Ng, and Q. Yang, "Multimodal Multipart Learning for Action Recognition in Depth Videos."
- [51] "A Beginner's Guide to Neural Networks and Deep Learning | Pathmind." <https://wiki.pathmind.com/neural-network> (accessed Aug. 18, 2020).

- [52] “Classification Using Neural Networks | by Oliver Knocklein | Towards Data Science.”
<https://towardsdatascience.com/classification-using-neural-networks-b8e98f3a904f>
(accessed Aug. 18, 2020).
- [53] M. Bassiouni, M. Ali, and E. A. El-Dahshan, “Ham and Spam E-Mails Classification Using Machine Learning Techniques,” *J. Appl. Secur. Res.*, vol. 13, no. 3, pp. 315–331, 2018, doi: 10.1080/19361610.2018.1463136.
- [54] A. S. Aski and N. K. Sourati, “Proposed efficient algorithm to filter spam using machine learning techniques,” *Pacific Sci. Rev. A Nat. Sci. Eng.*, vol. 18, no. 2, pp. 145–149, 2016, doi: 10.1016/j.psra.2016.09.017.
- [55] E. J. Cheng *et al.*, “Deep Sparse Representation Classifier for facial recognition and detection system,” *Pattern Recognit. Lett.*, vol. 125, pp. 71–77, 2019, doi: 10.1016/j.patrec.2019.03.006.
- [56] B. Ríos-Sánchez, D. Costa-da-Silva, N. Martín-Yuste, and C. Sánchez-Ávila, “Deep learning for facial recognition on single sample per person scenarios with varied capturing conditions,” *Appl. Sci.*, vol. 9, no. 24, 2019, doi: 10.3390/app9245474.
- [57] K. T. Islam and R. G. Raj, “Real-time (Vision-based) road sign recognition using an artificial neural network,” *Sensors (Switzerland)*, vol. 17, no. 4, pp. 14–16, 2017, doi: 10.3390/s17040853.
- [58] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, “How far are we from solving pedestrian detection?,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 1259–1267, 2016, doi: 10.1109/CVPR.2016.141.
- [59] J. Mao, Y. Jiang, T. Xiao, and Z. Cao, “What can help pedestrian detection?,” *Proc. - 30th IEEE*

- Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6034–6043, 2017, doi: 10.1109/CVPR.2017.639.
- [60] S. Shuffy, R. Applications, C. Lee, S. Member, and J. Moon, “Robust Lane Detection and Tracking for Real-Time Applications.”
- [61] H. Yousif, J. Yuan, R. Kays, and Z. He, “Fast human-animal detection from highly cluttered camera-trap images using joint background modeling and deep learning classification,” *Proc. - IEEE Int. Symp. Circuits Syst.*, 2017, doi: 10.1109/ISCAS.2017.8050762.
- [62] F. Moutarde, “Deep Learning for Hand Gesture Recognition on Skeletal Data,” *13th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG 2018)*, pp. 106–113, 2018, doi: 10.1109/FG.2018.00025.
- [63] P. J. Rani, J. Bakthakumar, B. P. Kumar, U. P. Kumar, and S. Kumar, “Voice controlled home automation system using Natural Language Processing (NLP) and Internet of Things (IoT),” in *ICONSTEM 2017 - Proceedings: 3rd IEEE International Conference on Science Technology, Engineering and Management*, 2017, pp. 368–373, doi: 10.1109/ICONSTEM.2017.8261311.
- [64] K.-L. Du, “Clustering: A neural network approach \$,” *Neural Networks*, vol. 23, pp. 89–107, 2010, doi: 10.1016/j.neunet.2009.08.007.
- [65] X. H. Han, L. Quan, X. Y. Xiong, M. Almeter, J. Xiang, and Y. Lan, “A novel data clustering algorithm based on modified gravitational search algorithm,” *Eng. Appl. Artif. Intell.*, vol. 61, no. November 2016, pp. 1–7, 2017, doi: 10.1016/j.engappai.2016.11.003.
- [66] Z. Li, Y. Zhao, R. Liu, and D. Pei, “Robust and Rapid Clustering of KPIs for Large-Scale

- Anomaly Detection,” *2018 IEEE/ACM 26th Int. Symp. Qual. Serv. IWQoS 2018*, 2019, doi: 10.1109/IWQoS.2018.8624168.
- [67] “Neural Networks for Regression (Part 1)—Overkill or Opportunity? - MissingLink.ai.” <https://missinglink.ai/guides/neural-network-concepts/neural-networks-regression-part-1-overkill-opportunity/> (accessed Aug. 18, 2020).
- [68] J. Stangierski, · D Weiss, and · A Kaczmarek, “Multiple regression models and Artificial Neural Network (ANN) as prediction tools of changes in overall quality during the storage of spreadable processed Gouda cheese,” vol. 245, pp. 2539–2547, 2019, doi: 10.1007/s00217-019-03369-y.
- [69] “A Beginner’s Guide To Understanding Convolutional Neural Networks – Adit Deshpande – Engineering at Forward | UCLA CS ’19.” <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/> (accessed Aug. 19, 2020).
- [70] “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science.” <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (accessed Aug. 19, 2020).
- [71] T. N. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran, “DEEP CONVOLUTIONAL NEURAL NETWORKS FOR LVCSR.”
- [72] K. Liu, G. Kang, N. Zhang, and B. Hou, “Breast cancer classification based on fully-connected layer first convolutional neural networks,” *IEEE Access*, vol. 6, pp. 23722–23732, 2018.
- [73] “A 2019 guide to Human Pose Estimation with Deep Learning.” <https://nanonets.com/blog/human-pose-estimation-2d-guide/> (accessed Jul. 20, 2020).

- [74] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Sep. 2014, pp. 1653–1660, doi: 10.1109/CVPR.2014.214.
- [75] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks." Accessed: Jul. 14, 2020. [Online]. Available: <http://code.google.com/p/cuda-convnet/>.
- [76] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using Convolutional Networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Oct. 2015, vol. 07-12-June-2015, pp. 648–656, doi: 10.1109/CVPR.2015.7298664.
- [77] S. E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 4724–4732, 2016, doi: 10.1109/CVPR.2016.511.
- [78] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 4733–4742, 2016, doi: 10.1109/CVPR.2016.512.
- [79] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9912 LNCS, pp. 483–499, 2016, doi: 10.1007/978-3-319-46484-8_29.
- [80] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11210 LNCS, pp. 472–487, 2018, doi: 10.1007/978-3-030-01231-1_29.

- [81] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” Accessed: Jul. 16, 2020. [Online]. Available: <http://image-net.org/challenges/LSVRC/2015/>.
- [82] K. Sun, B. Xiao, D. Liu, and J. Wang, “Deep High-Resolution Representation Learning for Human Pose Estimation,” 2019, [Online]. Available: <http://arxiv.org/abs/1902.09212>.
- [83] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, “Realtime multi-person 2D pose estimation using part affinity fields,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, no. Xxx, pp. 1302–1310, 2017, doi: 10.1109/CVPR.2017.143.
- [84] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “{HMDB}: a large video database for human motion recognition,” 2011.
- [85] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, “Towards understanding action recognition,” in *International Conf. on Computer Vision (ICCV)*, Dec. 2013, pp. 3192–3199.
- [86] “Datasets.” https://wangjiangb.github.io/my_data.html (accessed Jul. 20, 2020).
- [87] “AAMAZ Human Action Recognition Dataset | Kaggle.” <https://www.kaggle.com/ismail19/aamaz-human-action-recognition-dataset/metadata> (accessed Jul. 20, 2020).
- [88] “Anaconda | Individual Edition.” <https://www.anaconda.com/products/individual> (accessed Dec. 02, 2020).
- [89] Z. Cao, “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields *.”
- [90] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in {P}ython,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

- [91] “1.17. Neural network models (supervised) — scikit-learn 0.23.1 documentation.” https://scikit-learn.org/stable/modules/neural_networks_supervised.html (accessed Jun. 17, 2020).
- [92] “A Practical Guide to ReLU - Danqing Liu - Medium.” <https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7> (accessed Jun. 23, 2020).
- [93] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.