Deep Learning for Skeleton-Based Human Action Recognition

Sohaib Laraba

sohaib.laraba@umons.ac.be

Tuesday 20th October, 2020

A dissertation submitted to the Faculty of Engineering

of the University of Mons, for the degree of Doctor of Philosophy in Engineering Science

Supervisor: Prof. T. Dutoit

This thesis was supported by the ERDF DigiStorm Program

Jury members

Prof. **Xavier Siebert** - Université de Mons, President
Prof. **Thierry Dutoit** - Université de Mons, Supervisor
Dr. **Jöelle Tilmanne** - Hovertone, Co-Supervisor
Prof. **Bernard Gosselin** - Université de Mons
Prof. **Hicham Sahli** - Vrije Universiteit Brussel (VUB)
Prof. **Abdelmalik Taleb-Ahmed** - Université Polytechnique Hauts-de-France
Prof. **Hazem Wannous** - Institut Mines-Telecom Lille-Douai

”A candle loses none of its light by lighting another.”
Rumi

Abstract

HUMAN action recognition from videos has a wide range of applications, including video surveillance and security, human-computer interaction, robotics, health care, etc. Nowadays, 3D skeleton-based action recognition has drawn increasing attention thanks to the availability of low-cost motion capture devices, and accessibility of large-scale 3D skeleton datasets, in addition to real-time skeleton estimation algorithms. In the first part of this thesis, we present a novel representation of motion capture sequences for Three Dimensions (3D) skeleton-based action recognition. The proposed approach consists of representing the 3D skeleton sequences into RGB image-like data and leveraging recent convolutional neural networks (CNNs) to model the long-term temporal and spatial structural information for action recognition. Extensive experiments have shown the superiority of the proposed approach over the state-of-the-art methods for 3D skeleton-based action recognition.

In order to extract skeleton sequences, different devices extract first the depth information using multiple technologies (stereo, time-of-flight, etc.), then 3D skeleton poses are extracted using different algorithms. In the very late years, new researches proposed to extract skeleton sequences directly from Red, Green, and Blue (RGB) videos. The most precise methods extract Two Dimensions (2D) skeletons in real-time and with high accuracy.

In the second part of this thesis, we leverage these tools to extend the use of our proposed approach to RGB videos. We first extract 2D skeleton sequences from RGB videos, and then, following approximately the same process as in the first part, we use CNNs for human action recognition. Different experiments showed that the proposed method outperforms different state-of-the-art methods on a large benchmark dataset.

Another contribution of this thesis relates to the interpretability of deep learning models. Deep learning models are still considered alchemy due to the lack of understanding of their internal operations. Interpretability is a crucial task to understand and trust the decisions made by the machine learning model. Thus, we propose in the third part of this thesis to use CNN interpretation methods to understand the behavior of our classifier and extract the most informative joints during the execution of a particular action. This method allows us to see from the Convolutional Neural Network (CNN) point of view the most important joints, and understand why certain actions are confused by the proposed classifier.

Acknowledgements

CHÈRE mère, cher père, je n'oublierai jamais les sacrifices par lequels vous êtes passés pour moi. Vous avez goûté à l'amertume de cette vie afin que nous, vos enfants, puissions nous délecter de son nectar. Qu'importe le nombre de "merci" que je peux vous adresser, il n'égalera jamais toute la gratitude et la reconnaissance qu'éprouve à votre égard. Mon bonheur et ma joie ne se traduisent que par les vôtres.
Abderrahman, Oussama, Maroua, vous avez été pour moi un soutien sans pareil. Je me sens chanceux de vous avoir pour frères et sœur.

ISLEM, ma bien-aimée, je ne pense pas pouvoir te remercier assez pour tout ce que tu as fait pour moi. Tu es arrivée au moment où j'en avais le plus besoin. Tu as comblé un vide qui s'est inscrit dans ma vie. Ton soutien, ta motivation et ta patience sont les clés de ma réussite. Un "merci" ne sera jamais assez pour exprimer ma reconnaissance !

JE ne peux évidemment pas faire sans remercier mes proches, familles et amis, qui n'ont jamais cessé de me soutenir jusqu'au bout, et continueront sans doute à le faire. Je cite en particulier mon oncle Salim et toute sa famille, je n'oublierai jamais votre soutien et vos orientations depuis toujours. Je cite également ma belle-famille, et particulièrement ma belle-mère, qui m'a soutenu et m'a épaulé. Merci infiniment à tous ceux qui m'ont encouragé, qui ont eu une pensée ou une prière pour moi...

I Am also grateful for the time spent in the ISIA Lab (ex-TCTS Lab). I believe I belong to the best lab with the best working environment. I would like to thank all my colleagues who have contributed to this work in one way or another. I feel very lucky to be a part of this team with whom I share too many memories and had great human and technical exchanges.

Last but not least, I would like to express my deep and sincere gratitude to my supervisor Thierry Dutoit, and my co-supervisor Jöelle Tilmanne. Thank you for the opportunity you gave me to be a part this lab. You have always showed your attention and have been patient with me, and your trust made me who I am today. Your dynamism, vision, organization and management have always inspired me.

Contents

List of Figures

List of Tables

List of acronyms

2D	Two Dimensions
HMM	Hidden Markov Model
RGB	Red, Green, and Blue
Hz	Hertz
CNN	Convolutional Neural Network
Seq2Im	Sequence to Image
PCA	Principal Component Analysis
FTP	Fourier Temporal Pyramid
HOJ3D	Histogram Of 3D Joints Locations
DTW	Dynamic Time Warping

Introduction

In the 1960s, artificial intelligence has known its first steps and became an academic discipline. Some researchers were extremely optimistic about the future of this field, and they believed that it would take no more than 25 years to develop a machine that is as intelligent as a human being. MIT professor Seymour Papert believed that a group of MIT students could solve the machine vision problem in a few months, and he launched the Summer Vision Project [18]. Almost 60 years later, we are still very far from solving computer vision. Computer Vision tasks are difficult due to huge variations of appearance, motion patterns, viewpoint angles, lighting conditions, etc. Computers were primarily designed to perform fast and well defined computational tasks, and not complicated tasks requiring "reasoning". Thus, the main goal of computer vision is to improve computer abilities in the interpretation of image and video information. Such abilities will play a key role in future intelligent machines such as robots and vehicles. Analyzing human activities, and particularly from videos, was one of the challenges of the Summer Vision Project, and it is crucial in a wide range of applications such as video surveillance, human-computer interaction, entertainment, etc. In recent years, action analysis has become more and more popular thanks to the availability of video and motion capture data. Action recognition is one of the most important components of intelligent systems. The ability to interpret such information will ultimately bring computers one step closer to human skills. This thesis focuses particularly on the recognition of human actions from skeleton data. In this chapter, we first introduce some definitions of action recognition, in addition to an introduction of deep learning (DL). Furthermore, we follow by defining the problem to be solved in this thesis and the challenges that we face. We close this chapter by listing the contributions.

Definitions of Action Recognition

Human activities can be categorized into four different levels: gestures, actions, interactions, and group activities, depending on their complexity [19].

Gesture: A gesture is an elementary movement of a body part such as 'raising a leg'.

Action: An action is composed of several gestures of one person that are organized in time, such as 'walking' or 'waving'.

Interactions: An interaction involves two or more persons and/or objects. For example, 'a person pushing another person' is an interaction between two people, and 'a person lifting a box' is a human-object interaction. Finally, a group activity is an activity performed by multiple persons, such as 'a group of people is running'. Some action datasets contain both one-person actions and two-persons interactions. In this thesis, we focus on the analysis of single person actions.

Action recognition: One of the most important tasks in human activity analysis is action recognition. Action recognition can be defined as the process of automatically decoding and recognizing the action happening in a given sequence of images. In other words, if we assume that we have a set of sequences $\mathbb{S}$ and a set of corresponding action labels $\mathbb{L}$, and we assume that each sequence $S \in \mathbb{S}$ contains only one action l_S. Thus the goal of action recognition is to predict the label l_S based on the sequence representation S.

Action detection: Action detection problem can be considered as an extension of action recognition. In this problem, we assume that a sequence S can contain more than one action. Thus, each sequence $S \in \mathbb{S}$ has a corresponding set of labels L_S. The goal then is to predict all labels for a given sequence S, as well as the time when a given action starts and ends.

In this thesis, we focus on the action recognition challenge, as we claim that to solve the action detection problem, we need to get an action recognition system that is able to achieve high accuracy.

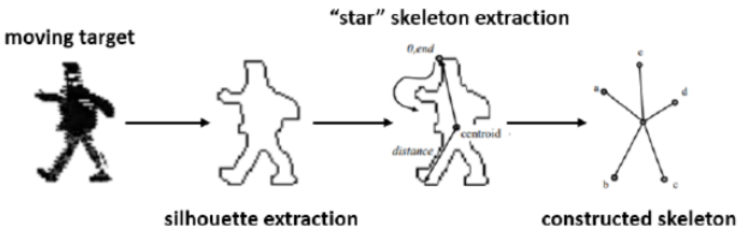


Figure 0.1. "Star" skeleton extraction. [1]

Background on 3D Skeleton Data

The human body is an articulated system that is formed by rigid segments connected by joints. A human action is considered as a continuous evolution of the spatial configuration of these segments. Back in 1975, G. Johansson [20] conducted some experiments and showed that humans could recognize activities with only seeing the light spors attached at the person's main joints. Multiple researchers tried then to extract body joints or to detect body parts and track them in the temporal domain. Fujiyoshi and Lipton [1] attempted to extract a human body skeleton from video streams by first extracting the silhouettes. They produced then a "star" skeleton by detecting extremal points on the boundary of the target, as shown in Figure 0.1. Ben-Arie et al. [21] labeled main body parts, such as arms, legs, torso, and head, based on a pattern matching technique, for activity recognition.

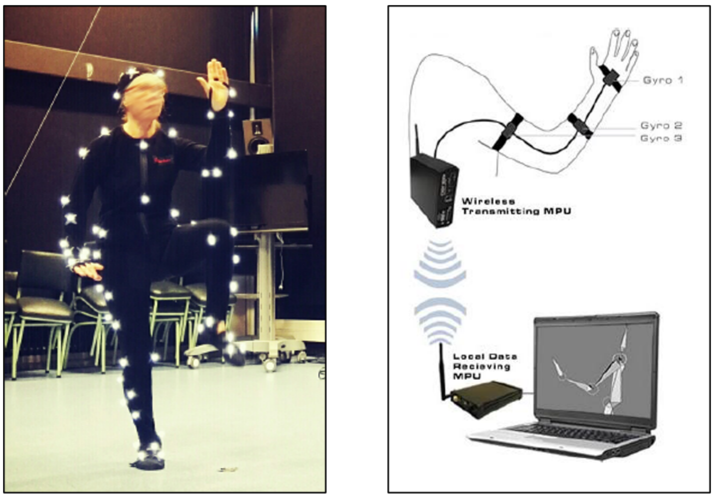
Due to the additional 3D geometric information that depth imagery can provide, many methods are developed to build a 3D human skeleton model. 3D skeleton-based representations demonstrate promising performances in real-world applications, including Kinect-based gaming, as well as in computer vision research [22–27]. 3D skeleton-based representations can model the relationship of human joints and encode whole body configuration. They are robust to scale and illumination changes, and can be invariant to camera view as well as human body rotation and motion speed.

Acquisition of 3D Skeleton Data

While several commercial devices, including motion capture systems, time-of-flight sensors, and structured-light cameras, allow for direct retrieval of 3D skeleton data, multiple approaches have been designed to automatically construct a skeleton model from perception data through pose recognition and joint estimation. Some of these methods are based on RGB imagery, while others leverage the extra-information available in the depth images. The majority of these methods are based on body part recognition, and then fit a flexible model to the 'known' body part locations. An alternative methodology is starting from the 'known' prior, and fitting the silhouette or point cloud to the prior after the person is localized [28, 29]. This section provides a brief overview of technologies allowing direct acquisition of 3D skeletal data and autonomous skeleton construction methods based on visual data.

High-precision motion capture systems
Most of high-precision motion capture systems identify and track markers that are attached to a human subject's joints or body parts to obtain 3D skeleton information. For simplicity matters, we note these marker-based motion capture systems as "MoÇap systems". There are two main categories of MoCap systems, based on either visual cameras or inertial sensors. Optical-based systems, like Qualisys[1], Vicon[2] or OptiTrack[3], employ multiple cameras that track, in 3D space, reflective markers attached to the human body (Figure 0.2, *a.*). These are generally high-speed cameras that can track with high accuracy the locations of the markers. Qualisys Oqus systems[4] for example feature high-speed digital cameras with a rate of 180-500 frames per second (FPS) at full resolution and 360-1750 FPS at reduced resolution, and a precision of less than 1mm. In MoCap systems based on inertial sensors, each 3-axis inertial sensor estimates the rotation of a body part with respect to a fixed point. This information is collected to obtain the skeleton data without any optical device around the subject (Figure 0.2, *b.*).

[1]Qualisys Motion Capture System: https://www.qualisys.com/
[2]Vicon Motion Capture System: https://www.vicon.com/
[3]OptiTrack Motion Capture System: http://optitrack.com/
[4]Qualisys Oqus systems: https://www.qualisys.com/cameras/oqus/



a. *b.*

Figure 0.2. *a.* Reflective markers attached on a person's body to be tracked by the optical MoCap system. b. Illustration of the inertial motion capture systems.

Although MoCap systems, especially based on multiple cameras, can provide very accurate 3D skeleton information at a very high speed, they are typically expensive and can only be used in well-controlled indoor environments.

Structured-light cameras

A typical structured-light stereo color-depth (RGB-D) system such as the Microsoft Kinect V1 [30] and ASUS Xtion [31], consists of a projector and two camera sensors (color and infrared cameras). The projector consists of an infra-red (IR) light source that emits a known pattern that is captured by the IR camera sensor. Depth information of the sensing environment can then be inferred based on how the light patterns are distorted in the IR images [32]. The RGB camera captures the color information and can be aligned with the IR camera by estimating the extrinsic parameters between them, thereby providing color-depth information at each pixel of a frame or 3D color point clouds. Several drivers are available to provide access to the RGB-D data acquired by the sensor, including the Microsoft Kinect SDK [33], the OpenKinect Library [34], etc.

The Kinect SDK also provides 3D human skeletal data using the method described by Shotton et al. [2]. These sensors are not expensive and can provide 3D skeleton information in real-time. On the other hand, since structured light cameras are based on infrared light, they can only work in an indoor environment. The frame rate (30Hz) and resolution of depth images (320×240) are also relatively low. The skeletal data provided is generally noisy and can be obtained correctly only when the subject is facing the camera.

Time-of-Flight (ToF) sensors
Time-of-Flight sensors are able to acquire accurate depth data at a high framerate, by emitting light and measuring the time it takes for light to return. The light is given a modulation envelope by rapidly turning the light source on and off. Distance measurement is achieved by measuring the phase of the modulation envelope of the transmitted light as received at the pixel array [35]. Compared to other ToF sensors, the Microsoft Kinect V2 camera offers an affordable alternative to acquire depth data using this technology. Furthermore, a color camera is integrated into the Kinect V2 sensor to provide registered color data. The color and depth data can be accessed using the Kinect SDK 2.0 [33]. The Kinect V2 camera provides depth images at a resolution of (512×424) at 30Hz. Moreover, the camera can provide 3D skeleton data by estimating positions of 25 human joints, with better tracking accuracy than the Kinect V1 camera. However, similarly to the Kinect V1 sensor, the skeletal data provided is noisy and cannot be used if the subject is not facing the camera.

3D skeleton estimation from depth imagery
In 2011, and with the development of depth sensors, Shotton et al. [2] proposed an extremely effective method to extract 3D body joint locations from a single depth image (independent of previous frames). The human body is labeled as body parts based on the per-pixel classification results using a randomized decision forest classifier. The parts include LU/RU/LW/RW head, neck, L/R shoulder, LU/RU/LW/RW arm, L/R elbow, L/R wrist, L/R hand, LU/RU/LW/RW torso, LU/RU/LW/RW leg, L/R knee, L/R ankle, and L/R foot (L: Left, R: Right, U: Upper, W: Lower) (Figure 0.3). Each branch in the forest is determined by a simple relation between the target pixel and various others. The pixels that are classified into the same category form the body part and the joint is inferred by the mean-shift method from a certain body part, using the depth data to 'push' them into the silhouette.

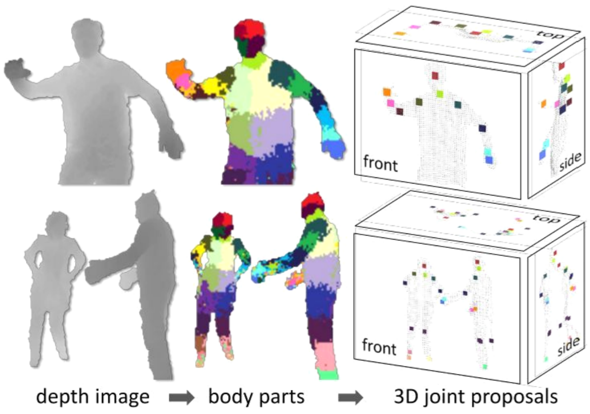


Figure 0.3. 3D joints extraction from a single depth image. [2]

While training the decision forests takes a large number of images (around 1 million) as well as a considerable amount of computing power, the fact that the branches in the forest are very simple allows this algorithm to generate 3D human skeleton models within about 5ms. Girshick et al. [36] published later an extended work with both accuracy and speed improved.

Plagemann et al. [37] built a 3D mesh to find geodesic extrema interest points, which are classified into three parts: head, hand, and foot. Their method provides both a location and orientation estimate of these parts. Holt et al. [38] proposed Connected Poselets for 3D human pose estimation from depth data. This approach uses the idea of poselets [39], are parts that are tightly clustered in both appearance and configuration space. This approach is widely applied for pose estimation from RGB data. For each depth image, a multi-scale sliding window is applied, and a decision forest is used to detect poselets and estimate human joint locations. Using a skeleton prior inspired by pictorial structures [40], the method begins with a torso point and connects outwards to body parts. By applying kinematic inference to eliminate impossible poses, they were able to reject incorrect body part classifications and improve their accuracy.

Another widely used methodology for the construction of 3D human skeletons from depth images is based on nearest neighbor matching [41–44].

Using the point clouds of a person with known poses as a model, several approaches [41, 45] apply Iterative Closest Point (ICP) method to fit the unknown poses by estimating the translation and rotation to fit the unknown body parts to the known model. While these approaches are relatively accurate, they suffer from several drawbacks. ICP is computationally expensive for a model with as many degrees of freedom as a human body. Additionally, it can be difficult to recover from tracking loss. Typically, the previous pose is used as the known pose to fit to; if tracking loss occurs and this pose becomes inaccurate, then further fitting can be difficult or impossible. Finally, skeleton construction methods based on the ICP algorithm generally require an initial T-pose to start the iterative process.

RGB Videos vs. Skeleton Sequences

In the past decades, researchers have focused mainly on recognizing human actions from RGB videos. The same action class can be found in different RGB videos containing different backgrounds, different viewpoints, and different illumination conditions. These conditions make RGB-based action recognition very challenging.

Nowadays, 3D skeleton-based action recognition has attracted increasing attention from the computer vision community, due to the development of accurate motion capture systems and real-time skeleton estimation algorithms. Skeleton sequences provide more comprehensive information on the actions. Back in 1975, Johansson G. [20] conducted some experiments and showed that humans could recognize activities with only seeing the light spots attached at the person's main joints. With additional 3D geometric information that depth imagery can provide, 3D skeleton-based representations are able to model the relationship of human joints and encode whole body configuration. They are robust to scale, illumination changes and clustered backgrounds, and can be invariant to camera view as well as human body rotation and motion speed. Besides, skeleton sequences are very attractive for real-time applications since the dimensionality of skeleton sequences are much lower than those of RGB videos. On this basis, this thesis mainly focuses and skeleton-based action recognition.

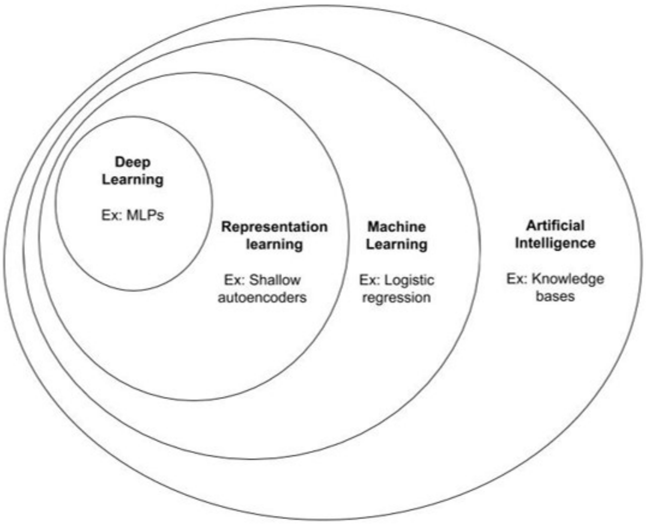


Figure 0.4. Relationship between different disciplines of artificial intelligence (AI).

Deep Learning

Deep Learning is an approach to Artificial Intelligence (AI). Specifically, it is a type of machine learning (M̧L), a technique that allows computer systems to learn and improve with experience and data [46]. Deep Learning can achieve great power and flexibility by learning to represent the world as a nested hierarchy of concepts, where each concept is defined in relation to simpler and more abstract ones. Figure 0.4 and Figure 0.5 show the relationship between different AI disciplines and give a high-level diagram of how they work.

Deep learning methods have been receiving a lot of attention in the last few years due to the high performances they achieve in a variety of problems such as image classification, object detection, speech recognition, natural language processing, and action recognition. The particularity of deep learning architectures is the fact that they can learn representations straight from data, which results in better performances than other methods that use hand-crafted features.

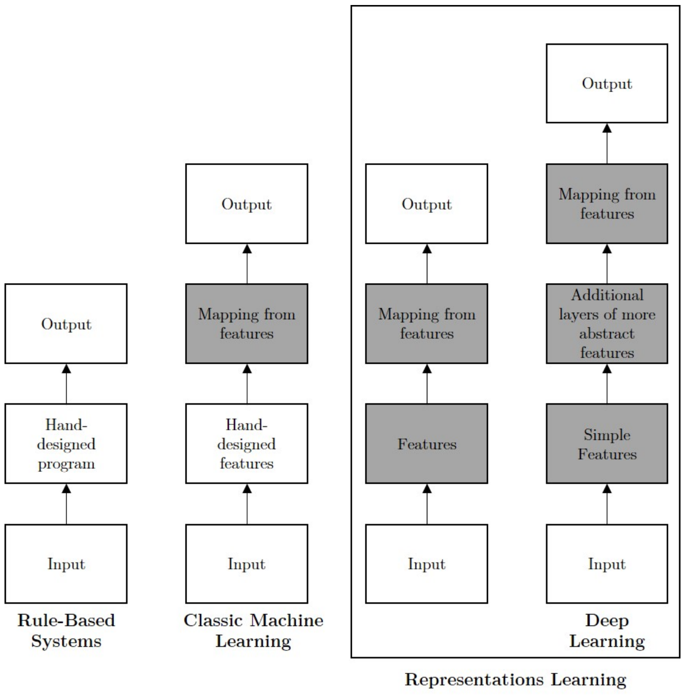


Figure 0.5. A diagram showing how different disciplines of AI work. [3]

The increase of dataset sizes and the rise of the age of "Big Data" has made machine learning and particularly deep learning, much easier, and their performances outperformed previous approaches. Moreover, the improvement of computational power played a crucial role in the development of this domain, allowing the machines to process extremely large datasets in a short time. A dramatic moment in the meteoric rise of deep learning came when a model called Convolutional Neural Network (CNN), designed for computer vision problems, won the largest yearly contest in object recognition (ImageNet Large Scale Visual Recognition Challenge - ILSVRC) for the first time and by a wide margin, bringing down the state-of-the-art top-5 error from 26.1% to 15.3% [6]. This means that the proposed network produces a ranked list of possible categories for each image, and the correct category appeared in the first five entries of this list for all the evaluation dataset except for 15.3% of it.

Since then, these competitions are consistently won by deep convolutional networks, and as of this writing, advances in deep learning have brought the latest top-5 error rate down to 1.3% [47]. Pursuing these advances, this thesis naturally follows this trend and imports its impact on the development of deep learning for action recognition.

Problem Definition

The main problem we wish to solve in this thesis is the recognition of human actions from sequences of skeleton data. In other words, given an ordered sequence of human joint coordinates, representing a person performing an action, the goal is to classify this sequence into one of the possible action categories. A model is developed and trained using multiple examples of each action category, and the sequence to be decoded is unknown to this model, i.e. it has not been used to train the model. Moreover, the motion capture sequences are captured using low-cost sensors (Microsoft Kinect). This means that the data provided can be very noisy.

The second addressed problem is the lack of understanding of deep learning models, and particularly Convolutional Neural Networks (CNNs), that are used in this thesis. CNN models are highly successful in solving complex vision problems. However, they are perceived as “black box” methods, considering the lack of understanding of their internal functioning. Consequently, when today’s intelligent systems fail, they fail spectacularly without warning or explanation, leaving the user with an incoherent output. Therefore, we want to generate a visual explanation from the proposed model for action recognition in order to increase the transparency of deep learning models and get more insights into the performance of different actions.

Research Challenges

The main challenges of the problems we are trying to solve are the following. First, the sequences that we want to classify are not of fixed length. However, their length can only be in a range of a few seconds. Usually, the sequences have a length of less than ten seconds.

Our first goal is then to propose a skeleton data representation that takes into account the length of the sequences. The representation should be able to deal with the temporal and spatial variations of motion capture sequences. Moreover, the models we wish to use in this thesis, namely Convolutional Neural Networks (CNNs), take generally as input 2D images. To this end, we propose a novel spatio-temporal representation that transforms 3D motion capture sequences into 2D images. This representation allows us to use different state-of-the-art image classifiers for our task of action recognition. Furthermore, this representation should be able to deal with the present noise in the data, as we use mainly low-cost sensors (like the Microsoft Kinect).

Many sequences of people performing actions from the same class can look very different when compared to each other. This can also be the case for one person performing the same action multiple times. That is because there are a lot of intra-class variations present in the data we deal with. The methods we use to learn representations from the motion capture sequences should deal with intra-class variations and focus on learning discriminative representations to distinguish between different action classes. We try to achieve this by proposing an architecture that can be trained in an end-to-end supervised way and can learn high-level discriminative features. To achieve this last challenge, we need first to face another challenge that is related to data. The dataset that we need to use should be large enough to train deep learning models. Moreover, this dataset should include different variations of action representations from different subjects and different view angles. The current state-of-the-art dataset that contains all of these aspects is the NTU RGB+D dataset, which contains more than 56,000 action sequences. It includes 60 action classes performed by 40 subjects and captured using three Kinect V2 sensors at the same time covering three views (-45°, 0°, 45°). An extension of this data was proposed recently, which doubled the size of the data, and provided more action classes.

Another challenge that we want to tackle is the recognition of human actions from RGB videos. While 3D skeleton datasets require special devices to be captured, RGB videos can be captured using standard cameras which are widely available and are very cheap compared to motion capture devices. The main problem that we face using video datasets is the detection of people in the video and processing the obtained data, then before training the classifier.

We try to handle this by using the state-of-the-art real-time people detection from 2D videos and process the obtained 2D skeletons before applying the proposed representation for action recognition. The proposed approach should be able to deal effectively with 2D data and noise.

Finally, the networks we are aiming to use are considered as black boxes due to the lack of understanding of their internal operations. For this reason, we use model interpretation algorithms in order to debug our models. Model interpretation algorithms can help the designers to analyze the models behaviors to improve their performances by extracting insights from the classifiers. In action recognition, interpretation algorithms may help, for example, in the localization of the most important joints that a person uses during the performance of a particular action. This information can give the designers insights on how to orient their classification algorithms by selecting the appropriate features, for instance.

Original Contributions of This thesis

The original contributions of this thesis are listed below:

1. A novel representation of skeleton sequences into RGB images, namely Sequence To Image (*Seq2Im*) is proposed. The new representation allows to capture the spatial and temporal information of the skeleton sequences. It is also translation, scale, and rotation invariant.

2. We propose to fine-tune state-of-the-art image classification models on our generated dataset. The fine-tuning process allows the transfer of knowledge from one task to another and is very effective when the dataset of the new task is not big enough to train new CNN models. The proposed method achieves state-of-the-art performance on different benchmark datasets captured using different motion capture devices, including the large scale NTU RGB+D dataset.

3. We propose a new method of human action recognition from RGB videos. We extract 2D poses from different videos in the dataset and propose a skeleton configuration that allows to apply the *Seq2Im* method. We fine-tune state-of-the-art image classification models on our new generated dataset.

The proposed method achieves state-of-the-art performance on the large scale NTU RGB+D dataset.

4. We propose a novel method for debugging our CNN classifiers, which allows to make our models more transparent, and to explain their behavior for skeleton-based human action recognition. The proposed approach allows to explain why some actions are confused by the classifier. Moreover, it allows to highlight the most important joints during the performance of a specific action.

Organization of This Dissertation

This dissertation is organized in three main chapters as follows:

Chapter 1: Background on Convolutional Neural Networks
In this chapter, we cover the relevant concepts that are necessary for the next chapters. We start by introducing neural networks, their basic terminology, and show some examples. Then, we introduce the principals of convolutional neural networks, their different components, and related concepts. We finish by defining six state-of-the-art CNN architectures that are used in this thesis.

Chapter 2: Novel Representation of Skeleton Sequences for 3D Action Recognition Using Convolutional Neural Networks
In this chapter, we tackle the problem of 3D skeleton-based human action recognition. We start by briefly introducing the literature of 3D skeleton-based action recognition methods using hand-crafted features and deep learning networks. Then, we propose a new representation of skeleton sequences, namely Sequence To Image (*Seq2Im*), for human action recognition. Each motion capture sequence is first pre-processed to be invariant to the environment, body orientation, and pose size, then transformed into an image by normalizing different values in the range of [0, 255]. The obtained images are then used to train CNN models. We leverage different state-of-the-art image classifiers and use transfer learning methods for better performances. The proposed method is extensively tested on four challenging benchmark datasets. Experimental results consistently demonstrate the superiority of the proposed Seq2Im representation and classification using pre-trained CNNs for 3D action recognition compared to existing techniques.

Chapter 3 Human Action Recognition From RGB Videos
In this chapter, we tackle the problem of human action recognition from RGB videos. We start first by briefly introducing related works. We propose then to use a state-of-the-art people detection framework to extract 2D human poses accurately. These poses are pre-processed to be invariant to the environment, body orientation, and pose size. We also process the data to add a third dimension, which allows us to generate RGB images using the proposed *Seq2Im* method.

Then following the same process as in the previous chapter, we train different state-of-the-art CNN models for action classification. This method is extensively tested on a large benchmark dataset. Experimental results demonstrate the superiority of the proposed approach for human action recognition from RGB videos compared to existing techniques.

Chapter 4 Towards Human Interpretable Deep Learning Models for Human Action Recognition
In the third chapter, we tackle the problem of transparency of deep learning models. We start by defining this challenging task, then introduce some related methods used for the interpretation of CNN models. We use a visualization method to extract insights and understand the behavior of our deep classifier. More specifically, the Grad-CAM visualization method is used to generate a heatmap on the image that corresponds to the zones that are the most salient to the CNN classifier. Moreover, we can benefit from the transparency provided by such algorithms to understand the classification process. With an inverse mapping of the generated heatmap into the skeleton sequence, we can highlight the joints that are used for the performance of a specific action.

Conclusion This chapter discusses the contributions of this thesis and gives directions for future work.

Chapter 1

Background on Convolutional Neural Networks

Contents

1.1 Introduction

The main problem we wish to solve in this thesis is the recognition of human actions from sequences of skeleton data. Our proposed approach consists of representing these sequences into RGB images that can take into account the spatial structure of the skeleton, and the long-term temporal dependency. The obtained images are then fed into pre-trained convolutional neural networks (CNNs) that are retrained using transfer learning and fine-tuning process.

In this chapter, we cover the relevant concepts that are necessary for the rest of this manuscript. In the first section, we introduce the basics of neural networks and different related components. Neural networks (NNs) are the elementary blocks of CNNs, and they share the same principal building blocks. In the second section, we go deeper into the principals of convolutional neural networks, and we introduce different components and related concepts. In the last section, we define six state-of-the-art CNN architectures that are extensively used for different experiments of this thesis.

1.2 Neural Networks Basics

Neural networks can be considered as a set of basic processing units, which are tightly interconnected. These units operate on the given inputs to generate the desired outputs. They can be grouped into two generic categories: feed-forward networks and feed-back networks [48]. In the feed-forward networks, the information flow only in one direction (such as multilayer perceptron - MLP and CNNs), while in the feed-back networks, the information flow in cycles (or loops) (such as Recurrent Neural Networks - RNNs and Long-Short Term Memory - LSTM). Feed-back network architecture allows us to have memorization ability and can store information and sequence relationships in their internal memory. Since our main focus in this thesis is on CNNs for action recognition, RNNs, and feed-back networks, in general, are out of scope, and we refer interested readers to [49].

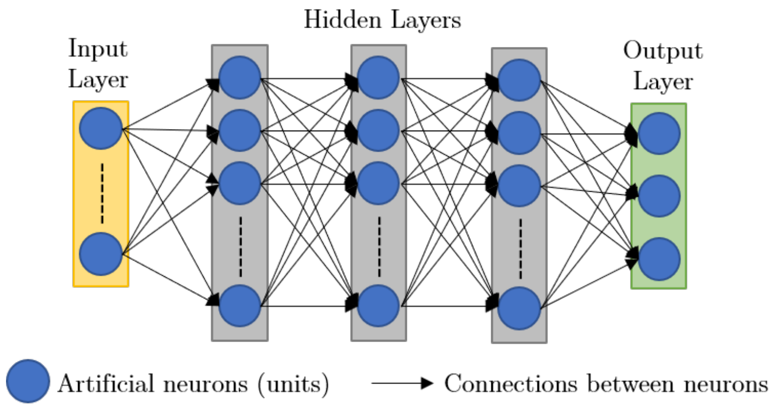


Figure 1.1. MLP network example with an input layer, an output layer and 3 hidden layers.

1.2.1 Multilayer Perceptrons

The simplest architecture of feed-forward networks is the multilayer perceptron (MLP). Figure 1.1 shows an example of a MLP network that consists of an input layer, an output layer, and three hidden layers in between. This network can be considered as a black box that operates on the input data and generates some outputs [50]. We highlight the most interesting aspects of this architecture below.

Layered architecture: Neural networks comprise a hierarchy of processing levels. Each level, called a "network layer", consists of a number of processing units called "nodes" or "neurons". Typically, the *input layer* is fed with the input data, and the final layer, called *"output layer"*, makes predictions. The intermediate layers process the information in a hierarchical way and are referred to as *"hidden layers"*.

Nodes: are the individual processing units in each layer. For each given node, its output value is computed by applying a function (ψ) to the outputs of the nodes belonging to the previous layers.

Therefore, the structure of a neural network (NN) is determined by the number of layers and the functions that determine the outputs of layers.

Dense connections: The nodes in a neural network are interconnected. The connections between different nodes are referred to as weights. The weights express the strength of these connections. As in feed-forward networks, the information flow in one direction from the input to the output layers, each node in one layer (k) is directly connected to all the nodes in the immediate previous layer $(k-1)$.

1.2.2 Parameter Learning

As mentioned in the previous section, the weights of the neural network define the connections between nodes. In order to obtain the desired output, these weights need to be set appropriately. In practice, the number of weights is huge (millions), which requires an automatic procedure to tune their values appropriately for a given task. This procedure is called *"learning"* and is accomplished during the *training* process. This process involves showing the network examples of the data with the desired output so that it learns to identify the relationships between the input and the output. This is the case for the *supervised learning*, for example. In the *unsupervised learning*, the network learns to find previously unknown patterns in the data without pre-existing labels.

The training of the network is generally achieved using the *back-propagation* algorithm, also known as the generalized delta rule. The basic delta rule, proposed by Widrow et al. [51], updates the network parameters based on the difference between the target output and the predicted output. This difference is computed in terms of the Least Mean Square (LMS) error. The output units are a linear function of the inputs, i.e.,

$$\hat{y}_i = \sum_j w_{ij} x_j. \tag{1.1}$$

If $\hat{y}_n$ and y_n denote the predicted and the target outputs respectively, then the error can be calculated as follows:

$$E = \frac{1}{2} \sum_n (y_n - \hat{y}_n)^2, \tag{1.2}$$

The delta rule calculates the gradient of this error function with respect to the parameters of the network $\frac{\delta E}{\delta w_{ij}}$. Thus, the weights are updated iteratively using the following learning rule:

$$w_{ij}^{t+1} = w_{ij}^{t} + \eta \frac{\delta E}{\delta w_{ij}} = w_{ij}^{t} + \eta (y_i - \hat{y}_i) x_j. \tag{1.3}$$

where t denotes the previous iteration of the learning process, and η denotes the step size of the parameter update. The parameters are updated so that the predicted outputs get closer to the target outputs. In other words, after a number of iterations, the training process is said to *converge* when the parameters do not change any longer after updating.

The delta rule only computes linear combinations between the input and the output pairs. This limits its use to only linear networks with single layers, while neural networks generally use nonlinear activation functions at each processing unit. To overcome this limitation, the generalized delta rule (backpropagation algorithm) is proposed, which is an extension of the delta rule. The backpropagation algorithm makes use of the nonlinear activation functions to model nonlinear relationships between the input and the output pairs. The parameters of the multi-layered neural network are updated similarly as the delta rule, i.e.,

$$w_{ij}^{t+1} = w_{ij}^{t} + \eta \frac{\delta E}{\delta w_{ij}} \tag{1.4}$$

However, the errors using the generalized delta rule are recursively sent backward through the layers. This is the reason why it is also called *'backpropagation algorithm'*.

Given the error function 1.2, its gradient with respect to the parameters in the output layer $l^{(o)}$ for each node i can be computed as follows:

$$\frac{\delta E}{\delta w_{ij}^{l}} = \theta_i^{l(o)} x_j, \tag{1.5}$$

$$\theta_i^{l(o)} = (y_i - \hat{y}_i) f'(\sigma_i), \tag{1.6}$$

where, $\sigma_i = \sum_j w_{ij} x_j + b_i$ is the activation which is the input to the neuron (prior to the activation function), x_j's are the output from the previous layer, $\hat{y}_i = f_i(\sigma_i)$ is the output from the neuron, $f(.)$ denotes the nonlinear activation function and $f'(.)$ is its derivative.
Similarly, we can compute the error for the intermediate hidden layers by back-propagation of the errors as follows:

$$\theta_i^{l(k)} = f'(\sigma_i^{l(k)}) w_{ij}^{l(k+1)} \theta_j^{l(k+1)} \tag{1.7}$$

where $k \in \{1...L-1\}$, and L is the total number of layers in the network. The overall update equation for the MLP parameters can written as:

$$w_{ij}^{t+1} = w_{ij}^{t} + \eta \theta_i^{l(k)} x_j^{l(k-1)}, \tag{1.8}$$

where $x_j^{l(k-1)}$ is the output from the previous layer, and t denotes the number of previous training iteration.

1.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks or CNNs, also known as **Convolutional Networks** [52], are a specialized kind of neural networks for processing high-dimensional data that has a known, grid-like topology such as image data that has a 3D grid of pixels. CNNs operate in a way that is similar to the standard neural networks. A key difference, however, is that each unit in a CNN layer is a filter of at least two dimensions. This filter is convolved with the input of that layer using the mathematical operation *"convolution"*.

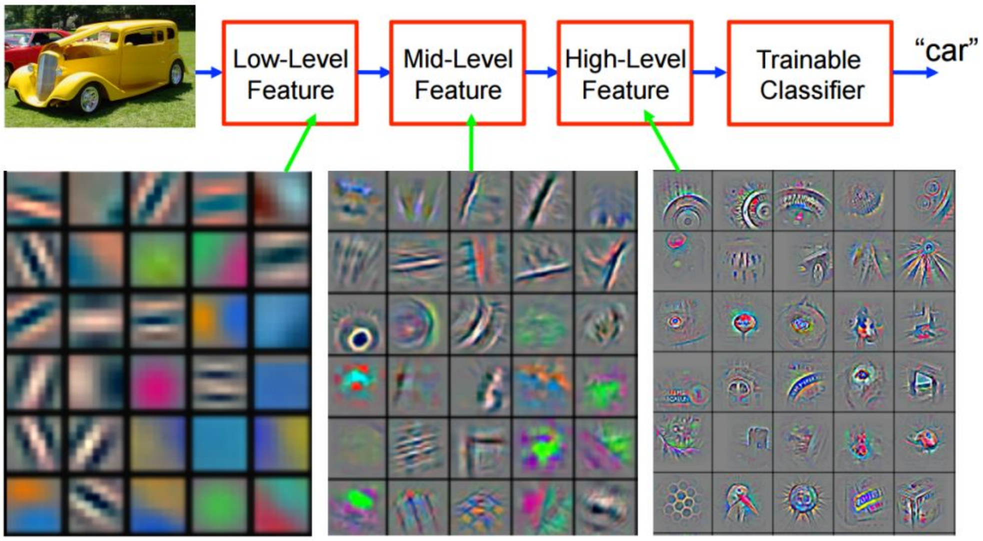


Figure 1.2. Feature visualization of convolutional net trained on ImageNet. [4]

CNN filters incorporate spatial context and use parameter sharing to significantly reduce the number of learned variables.

CNNs are useful for both supervised and unsupervised learning paradigms. An example of supervised learning is "image classification", as shown in Figure 1.2. The CNN learns to map a given image to its corresponding class by learning a number of abstract feature representation from simple to more complex ones. These features are then used to predict the correct category of an input image. Compared to classical machine learning algorithms, CNNs automatically learn a hierarchy of useful feature representations. Moreover, feature extraction and classification stages are trainable in one single pipeline in an end-to-end way. This reduces the need for manual design and expert human intervention.

1.3.1 CNN Layers

A typical layer of a CNN consists of three main stages, that can be considered themselves as building layers, as shown in Figure 1.3.

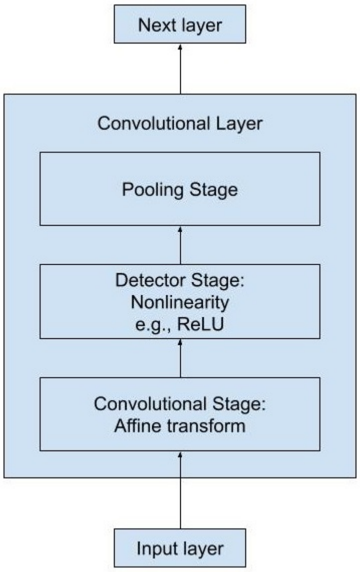


Figure 1.3. The components of a typical convolutional neural network layer.

In the first stage, the layer performs one or multiple convolutions in parallel between the filters and the input of the layer to produce a set of linear activations. In the second stage, each linear activation is run through a non-linear activation function, such as the Sigmoid or a rectified linear activation function. It is also called sometimes the detector stage. In the third stage, a *pooling function* is used to modify the output of the layer further.

Convolutional layers

The convolutional layer is the most important component of a CNN. It comprises a set of *filters*, also called *kernels*. Each filter in a convolutional layer is a grid of discrete numbers. If we consider, for example, a 2×2 filter, as shown in Figure 1.4, the weights of each filter (the numbers in the grid) are learned during the training process of the CNN. The filters are convolved with the given input to generate an output that is referred to as a feature map.

- ***What is a convolution?:*** We mentioned earlier that in a convolutional layer, one or multiple convolution operations are performed between the filters and the input of the layer.

0	1
-1	2

Figure 1.4. Example of a 2×2 CNN filter.

The convolution is performed by applying the filter sliding it over the image, and for each position computing the element-wise multiplication of the two matrices and then summing up the result. To illustrate this with an example, given an input feature map of 4×4 and a convolution filter of 2×2, as shown in Figure 1.5, the convolution layer multiplies the filter with the highlighted patch (of 2×2) of the input feature map and sums up all the values to generate one value in the output feature map. The filter slides along the width and the height of the feature map until the end.

It is important to note that, in signal processing literature, this operation is called "correlation" and not "convolution". During the convolution, the difference is that the filter rotated by $180°$, which means that it is flipped along its width and the height before multiplication. In machine learning, we rarely make a distinction between the two, and we consider both operations as equivalent. In this thesis, we follow the machine learning convention and do not make a distinction between the two operations, i.e., a convolution layer performs the correlation operation.

In the above example, the filter is slid with a step of 1 pixel along the horizontal or vertical position. This step is called the *stride* and can be different than 1 if required. The stride can have a value between 1 and the size of the filter, so all the feature map is covered. For example, if the filter has a size if 3x3, the stride value can be 1, 2, or 3 pixels. A bigger stride results in a smaller output feature map. Such a reduction in dimensions provides a moderate invariance to scale and pose of the objects and can be referred to as *sub-sampling*. However, in some applications, we want to keep the same spatial size after the convolution. This is important for applications that require more dense predictions at the pixel level. Moreover, it allows to design deeper networks by avoiding a quick collapse of the feature maps dimensions. This can be achieved by applying a *zero-padding* around the input feature map.

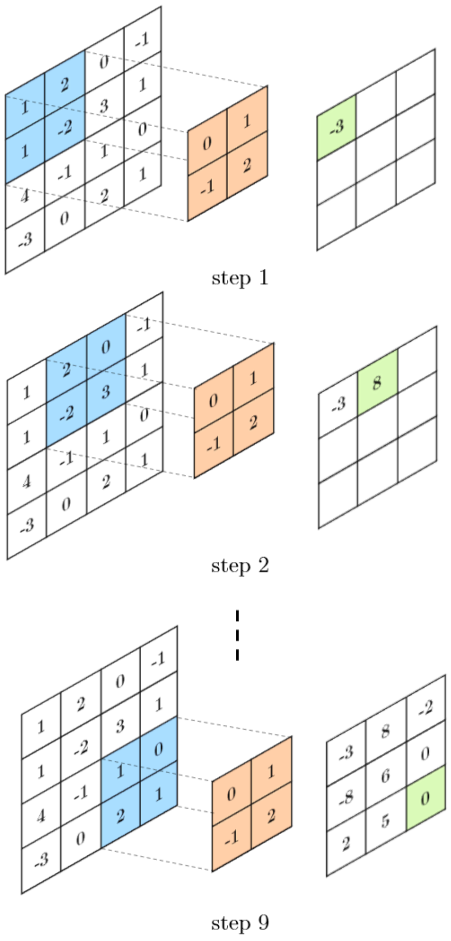
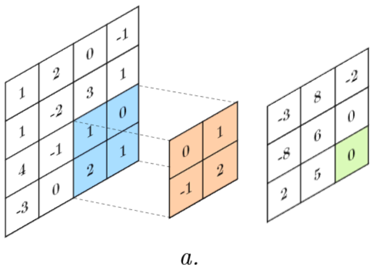


Figure 1.5. Example showing the convolution operation of a input feature map of 4×4 and a filter of 2×2.

This process allows to increase the dimensions of the input feature map by adding zeros along the horizontal and vertical dimensions and, thus, increase the dimensions of the output feature maps, as shown in Figure 1.6.

- ***Hyper-parameters:*** Hyper-parameters are the parameters of the convolution layer which need to be set by the user prior to the filter learning (such as the stride and padding).



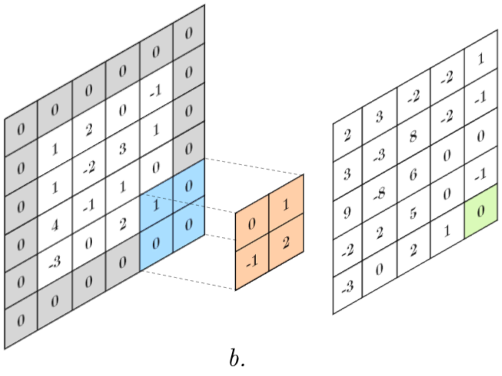


Figure 1.6. Illustration of the convolution operation with a stride of 1: *a*) without zero padding. *b*) with zero-padding.

They can be interpreted as the design choices of our network architecture based on a given application.

Pooling layers

A pooling layer operation allows to down-sample the input feature map to obtain a compact feature representation that is invariant to moderate changes in object scale, pose, and translation in an image [46]. The pooling layer combines the feature activations of blocks in the input feature map using a pooling function such as the average or the max function. Similar to the convolution operation, we need to specify the size of the pooled region and the stride. Figure 1.7 shows an example of the max pooling operation where the size of the pooling region is 2×2, and the stride is 1. The max-pooling operation selects the maximum activation within the selected region. The window is then slid across the input feature map with a step size of 1.

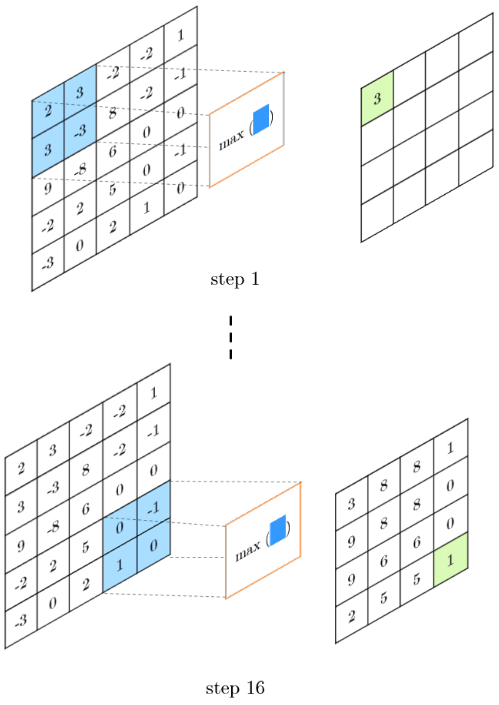


Figure 1.7. Illustration of the max-pooling layer with a size of 2 × 2 and a stride of 1.

The mean-pooling operation has the same process as the max-pooling, except that it calculates the mean of the activations of the selected region instead of the maximum.

Nonlinearity

The weight layers in a CNN are followed by a nonlinear activation function. This function takes a real-valued input and squashes it within a small range (generally [0, 1] or [-1, 1]). This nonlinear function allows the network to learn nonlinear mappings between the input and the output. Certain functions arise often such as the *logistic sigmoid*, the *Hyperbolic Tangent (Tanh)* or the *Rectified Linear Unit (ReLU)*.

- ***Sigmoid*** (Figure 1.8 *a.*): The sigmoid takes in a real number and outputs a number in the range of [0, 1]. It is defined as follows:

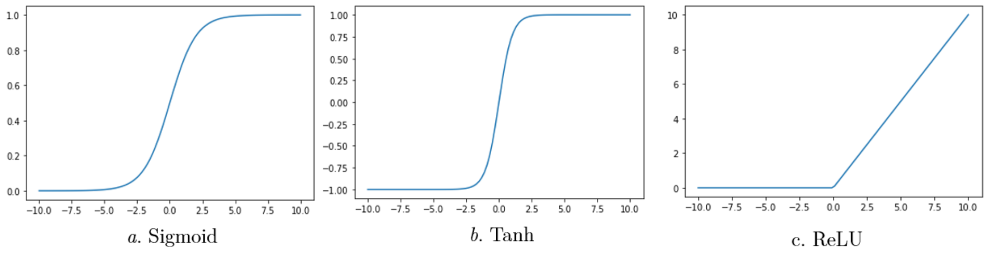


Figure 1.8. Three common activation functions that are used in deep learning.

$$f_{sigm}(x) = \frac{1}{1+e^{-x}} \tag{1.9}$$

The sigmoid function saturates when its argument is very positive or very negative, meaning that the function becomes very flat and insensitive to small changes in its input. It saturates to a high value when x is very positive, saturates to a low value when x is very negative and is only strongly sensitive to its input when x is near to 0. The widespread saturation of the sigmoid function can make gradient-based learning very difficult. For this reason, its use as a hidden unit for nonlinearity in feed-forward networks is now discouraged.

- ***Tanh*** (Figure 1.8 *b.*): The hyperbolic function (eq. tanh) is also very used to produce the nonlinearity. It takes in a real number and outputs a number in the range of [-1, 1]. It is defined as follows:

$$f_{tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.10}$$

The sigmoid and the hyperbolic tangent functions are closely related because $f_{tanh}(x) = 2f_{sigm}(2x) - 1$. However, when a sigmoidal activation function must be used, the hyperbolic tangent activation function typically performs better than the logistic sigmoid. It resembles the identity function more closely, in the sense that $f_{tanh}(0) = 0$ while $f_{sigm}(0) = \frac{1}{2}$. This makes training of the *tanh* network easier.

- ***Rectified linear unit (ReLU)*** (Figure 1.8 *c.*): A ReLU function maps the input to 0 if it is negative and keeps its value unchanged if it is positive. This can be represented as follows:

$$f_{relu}(x) = max(0, x), \tag{1.11}$$

ReLUs are easy to optimize because they are very similar to linear units. The only difference between the linear unit and the rectified linear unit is that a rectified linear unit outputs zero across half of its domain. This makes the derivatives through a rectified linear unit remain large whenever the unit is active.

Fully connected (FC) layers:
As the name suggests, two consecutive layers are fully connected when all the nodes in the previous layer are connected to all the nodes to the next layer (Figure 1.9). Fully connected layers can be considered as convolutional layers with filters of size 1×1. Typically, fully connected layers are placed at the end of the architecture. Note that a fully connected layer is identical to layers that we studied previously in the case of Multi-Layer Perceptrons (Section 1.2.1). Its operation can be represented as simple matrix multiplication and addition, and applying an element-wise nonlinear function f:

$$y = f(W^T x + b), \tag{1.12}$$

where x and y are the input and output vectors, respectively, W represents the weights matrix, and b is the bias vector.

1.3.2 Training the CNN

Now that we have discussed different architecture blocks of the CNN, we will briefly describe the mechanisms that are used to set the weights in deep neural networks. Most of the CNN layers involve parameters which require to be tuned appropriately for a given task.

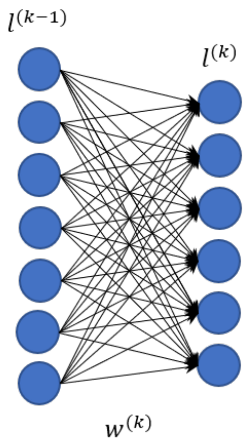


Figure 1.9. Graph representation of two fully connected layers, $l^{(k-1)}$ and $l^{(k)}$, connected by the weight matrix $w^{(k)}$.

The CNN learning process aims to tune the parameters of the network so that the input and the output spaces are correctly tuned [53]. As discussed earlier, at each training step, the estimate of the output variable is matched with the desired output (also called the ground-truth). The training of the CNN involves to minimize this matching function called the *"loss function"* or the *"error function"*. Different variants of CNN optimizers can be found in the literature:

Gradient-based learning: CNNs are nonlinear models that are hard to optimize due to a large number of tunable parameters. Intuitively, instead of searching for a global optimal solution, the optimization is performed so that the loss function is progressively reduced to a minimum value by searching for a local optimal solution at each step. The gradient-based methods are a natural choice [52]. The size of the update step is called the *learning rate*, and the iteration in which the update of the parameters uses the complete training set is called a *training epoch*. Each training iteration at the time t can be written as follows:

$$\theta_t = \theta_{t-1} - \eta\delta_t, \tag{1.13}$$

$$\delta_t = \nabla_\theta \mathcal{F}(\theta_t) \tag{1.14}$$

where $\mathcal{F}(.)$ denotes the function represented by the neural network with parameters θ, ∇ represents the gradient, and η represents the learning rate.

Batch gradient descent: The gradient descent algorithms, as discussed in the previous paragraph, compute the objective function with respect to the neural network parameters, which are updated in the direction of the steepest descent. The basic version of the gradient descent algorithms is the *batch gradient descent*, which computes the gradient on the entire training set [54]. However, the training sets can be very large in computer vision problems, which can make the learning very slow.

Stochastic gradient descent (SGD): The SGD updates the parameters for each set of input and output that are present in the training dataset. This allows a much faster convergence compared to the batch gradient descent [54]. The problem with this algorithm is that its convergence behavior is usually unstable, especially when the learning rate is relatively high.

Mini-batch gradient descent: The mini-batch gradient descent method is an improved version of the SGD, which provides a trade-off between convergence efficiency and convergence stability [54]. The training dataset is divided into a number of mini-batches, each consisting of a relatively small number of examples. The parameter update is then performed after computing the gradients on each mini-batch.

Momentum-based optimization: SGD can be improved with better convergence properties using a momentum optimization [55]. The SGD can oscillate close to a local minima resulting in an unnecessarily delayed convergence as illustrated in Figure 1.10 *a*. The momentum adds the gradient computed at the previous step a_{t-1}, weighted by a parameter γ to the weight update equation as follows:

$$\theta_t = \theta_{t-1} - a_t, \tag{1.15}$$

$$a_t = \eta \nabla_\theta \mathcal{F}(\theta_t) + \gamma a_{t-1}, \tag{1.16}$$

where $\mathcal{F}(.)$ denotes the function represents by the network with parameters θ, ∇ represents the gradient, and η represents the learning rate.

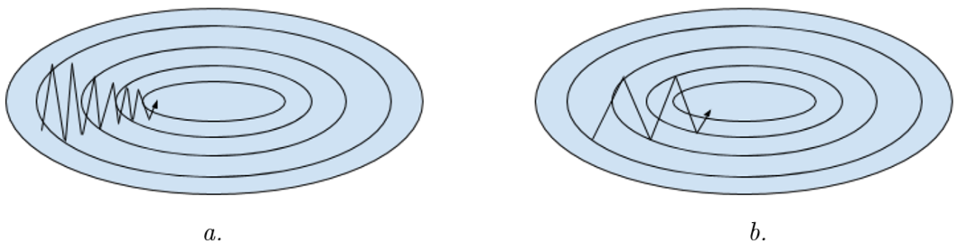


Figure 1.10. Illustration of the convergence behavior of the SGD without (*a.*) and with (*b.*) momentum. [5]

Figure 1.10, (*b*). illustrates how the addition of the momentum can help to speed up the convergence because unnecessary oscillations are avoided. Physically speaking, the momentum quickly magnifies the dimensions whose gradients point in the same direction, while it suppresses the dimensions whose gradients keep on changing directions. Typically, the momentum is set to 0.9 during SGD-based learning.

1.3.3 Transfer Learning and Domain Adaptation

Machine Learning models have been traditionally developed under the assumption that a model works well when the training and the test data are from the same feature space and the same distribution. If the feature space or the distribution of data changes, we would need to build a new model. Developing a new model every time from scratch and collecting a new set of training data is expensive. A very successful practice in such cases is to first train the neural network on a related but different problem, where a large amount of training data is already available. Afterward, the learned model can be "adapted" to the new task by initializing with weights pre-trained on the larger dataset. This process is called *"fine-tuning"* and is a simple, yet effective, way to transfer learning from one task to another (sometimes interchangeably referred to as domain transfer or domain adaptation) [56]. In summary, transfer learning and domain adaptation refer to the situation where the knowledge that has been leaned for one task (i.e., distribution *P1*) is exploited to improve generalization on another task (say distribution *P2*).

Transfer learning can be performed using off-the-shelf pre-trained models (e.g., AlexNet, DenseNet, SqueezeNet). These models are publically available on the internet and are trained on huge datasets such as the famous ImageNet dataset (with 1.2 million images) [57]. The pre-trained model can be adapted for a given task by changing the dimensions of the output neurons to fit the new dataset.
If the dataset for the end-task is similar to the original one on which the model has been pre-trained, the learning implies to freeze all the parameters of different layers and to train the final few layers from scratch (2 or 3 layers are enough for most cases). This can be referred to in some cases as *shallow retraining*. On the other hand, if the end-task dataset is different, more layers are retrained. The complete model can be fine-tuned on the new dataset if this one is very different from the original dataset, or if it is sufficiently large. For this purpose, a small learning rate is used so that the learning previously acquired is not lost. This can be referred to as *deep retraining*.
If one opts for a customized CNN architecture, transfer learning can still be helpful. The architecture can be first trained using a large annotated dataset and the resulting model be retrained in the same manner as described above.

1.4 Six Comparative Classification Architectures

In this work, six state-of-the-art image classification architectures are used with their other varieties. In total, 12 pre-trained models are modified to fit our classification problem (i.e., changing the last fully connected layer and the classification layer to fit the number of action classes). We compare both strategies (shallow and deep retaining) in addition to training the models from scratch (i.e., with random values).

1.4.1 AlexNet

The AlexNet architecture [6] achieved significantly improved performance over the other non-deep learning methods for ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012. This success has revived the interest in CNNs in computer vision.

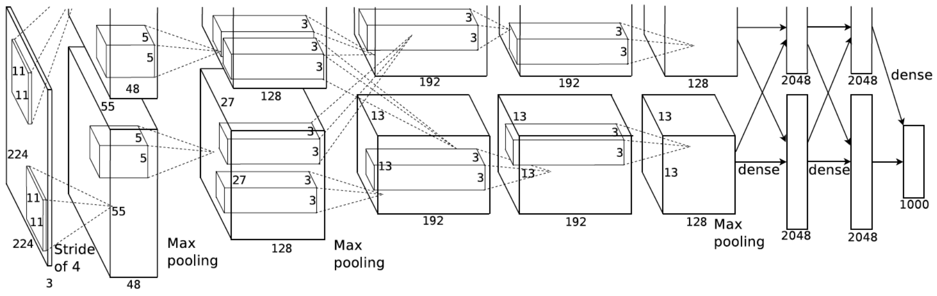


Figure 1.11. Illustration of AlexNet architecture. [6]

AlexNet has a very similar architecture to LeNet by LeCun et al. [52] with more filters per layer, and with stacked convolutional layers. As shown in Figure 1.11, it has five convolutional layers, three pooling layers, and two fully-connected layers. Each one of these layers is followed by a nonlinear function called ReLU (Rectified Linear Unit) to enable faster training. This gives in total approximately 62 million parameters. The input layer of AlexNet accepts images of 227×227 pixels.

1.4.2 Inception V3

Inception V3 [7] is a variant of GoogleNet architecture [58]. GoogleNet was the winner of the ImageNet ILSVRC challenge in 2014. It includes a new module called "Inception", which is a sub-network consisting of parallel convolutional filters whose outputs are concatenated. This makes the networks deeper with fewer parameters to be inferred. Original GoogleNet is composed of 22 layers (27 layers if we count the pooling layers), but has 12 times fewer parameters than AlexNet). Inception V3 has 42 (Figure 1.12) layers, but the computation cost is only 2.5 higher than the original GoogleNet. This input layer of Inception V3 accepts images of 299×299 pixels.

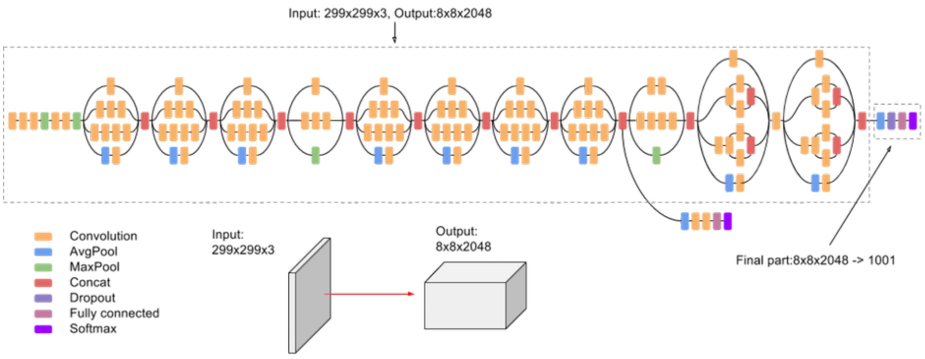


Figure 1.12. Illustration of Inception V3 architecture. [7]

1.4.3 VGGNet

VGGNet [8] was placed second in ILSVRC 2014. It is a very deep network with respect to GoogleNet. It includes at least 16 convolutional and FC layers. Each convolutional layer uses small 3×3 convolution filters, and a pooling layer is placed between each group of 2 or 3 convolutional layers (Figure 1.13). In our experiments, we consider three VGG models (i.e., VGG11, VGG16, and VGG19), with 11, 16, and 19 weight layers, that are available as pre-trained models. The input layers of these models accept images of 224×224 pixels.

1.4.4 ResNet

ResNet [9] was the winner of ILSVRC2015. It is about 20 times deeper than AlexNet and eight times deeper than VGGNet. ResNet introduces a new kind of layers, called “Residual layers”, considered as a kind of “network-in-network” architectures [59]. The residual layers include shortcut connections as shown in (Figure 1.14, *up*) which is the main different compared to a classical plain network (Figure 1.14, *down*). Another novelty is the use of global average pooling layers instead of FC layers at the end of the network. The input layer of ResNet accepts images of 224×224 pixels.

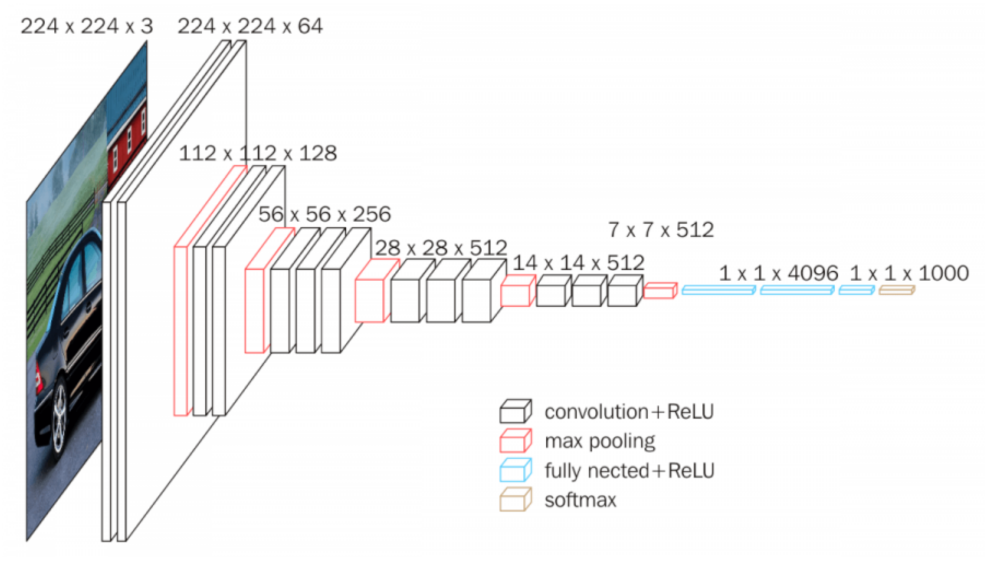


Figure 1.13. Illustration of VGGNet (with 16 weight layers). [8]

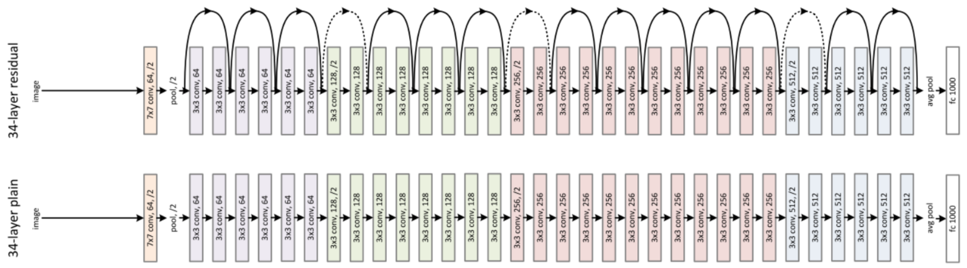


Figure 1.14. Illustration of the ResNet architecture with 34 parameter layers (*up*). For a comparison, a plain network with 34 parameter layers without shortcut connections (*down*). [9]

1.4.5 DenseNet

DenseNet [10] is an evolution of ResNet which connects each layer to every other layer in a feed-forward way. This increases the number of connections from L layers to $L\frac{L+1}{2}$ layers.

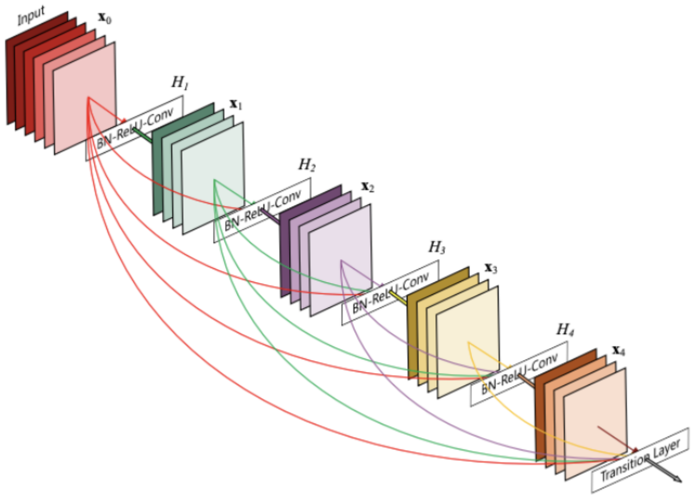


Figure 1.15. Illustration of a 5-layer dense block from DenseNet architecture. Each layer takes all preceding feature-maps as input. [10]

DenseNet obtained significant improvements over other state-of-the-art models at the cost of an increased computation requirement. The input layer of DenseNet accepts images of 224×224 pixels (Figure 1.15).

1.4.6 SqueezeNet

SqueezeNet [11] is a very compact model made of only 18 layers and is able to gain performance similar to AlexNet with 50 times fewer parameters, and a total size of around 0.5MB. New modules are introduced called “Fire” modules, which consist of two layers: A *Squeeze layer*, containing 1×1 filters, followed by an *Expand layer* containing a mix of 1×1 and 3×3 filters. The input layer of SqueezeNet accepts images of 227×227 pixels (Figure 1.16).

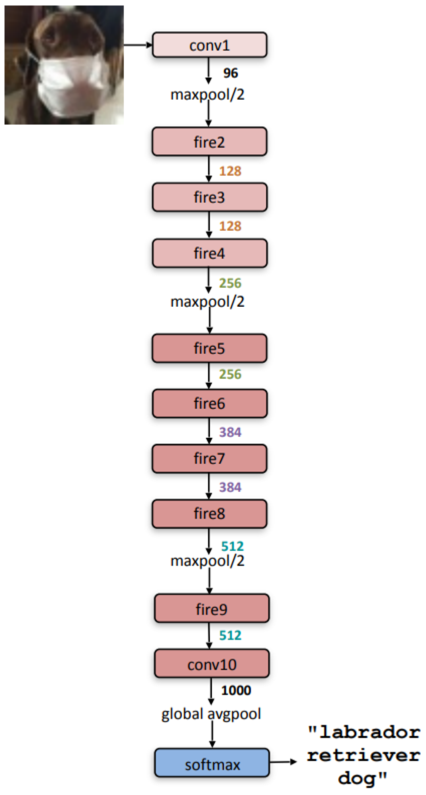


Figure 1.16. Illustration of SqueezeNet architecture. Each *fire* block consists of two layers: Squeeze Layer (with 1×1 filters) and Expand layer (with 1×1 and 3×3 filters). [11]

Chapter 2

Novel Representation of Skeleton Sequences for 3D Action Recognition Using Convolutional Neural Networks

Contents

This chapter is based on the following publications:

- 3D skeleton-based action recognition by representing motion capture sequences as 2D-RGB images. Computer Animation and Virtual Worlds. 2017. [60]
- Leveraging Pre-trained CNN Models for Skeleton-Based Action Recognition. In International Conference on Computer Vision Systems. Springer, Cham. 2019. [61]

2.1 Introduction

3D skeleton data capture the trajectories of human skeleton joints and are view and illumination invariant [62]. As the motion capture devices got democratized, in addition to the high accuracy provided, action recognition based on 3D skeleton sequences has attracted increasing attention [23, 62–64].

Recognizing human actions from videos requires managing the temporal information of the sequences to understand the dynamics of the human postures [65–67]. In skeleton action recognition, the spatial structure of joints is an important clue to exploit. A skeleton sequence provides only trajectories of different joints.

One of the recent approaches that was designed to deal with temporal sequences is called Recurrent Neural Network (RŅN). RNNs have been successfully used for many tasks involving sequential data such as machine translation, sentiment analysis, image captioning, time-series prediction etc. In our context, Recurrent Neural Networks (RNNs) can be used to deal with the trajectories of skeleton joints as time series with improved RNN models called Long-Short Term Memory (LSTM) neurons [68] to explore the temporal structure of the skeleton sequences for action recognition [69–72]. LSTMs enable training on long sequences overcoming the recurring problems of RNNs like vanishing gradients. However, it is difficult for LSTM networks to memorize the information of the entire sequence with many timestamps even if they are designed to explore long-term dependencies [73, 74]. Moreover, it is also difficult for LSTM networks to extract high-level features [75, 76].

On the other hand, Convolutional Neural Networks (CNNs) have shown their great power in learning patterns from images [6,9,58,77,78]. However, for video action recognition, it still lacks the capacity to model the long-term temporal dependency of the entire video [79]. In this chapter, we use CNNs for skeleton-based action recognition, but instead of dealing with each frame alone, we represent the whole skeleton sequence as one image. The generated image takes into account the spatial and temporal information. With the generated image, the spatial and the long-term temporal structure of the skeleton sequence can be effectively learned using deep CNNs. More specifically, for each skeleton sequence, we generate three gray images corresponding to the three channels of the 3D coordinates of the sequence using a normalization process.

By merging these three images, we obtain one RGB image-like representation. Each pixel in the generated image represents one joint in one frame. The generated images can be further resized to fit the developed CNN architecture (see Figures 2.1 and 2.4). Since the temporal information of the skeleton sequence is incorporated in the generated image, the CNN will extract features to learn the temporal and the spatial structure. The generated feature vector is fed in a fully connected neural network for action recognition.

The main contributions of this chapter are summarized as follows: (1) We propose to transform each skeleton sequence to a new representation to allow a global long-term temporal and spatial modeling of the skeleton sequences using deep CNNs to learn features from images. (2) We propose to use transfer learning and fine-tune existing CNN architectures that achieve state-of-the-art results in image classification to our task of human action recognition. The fine-tuning process allows the transfer of knowledge from one task to another and is very effective when the dataset of the new task is not big enough to train new CNN models. (3) The proposed method is validated on multiple challenging benchmark datasets (such as the large scale NTU RGB+D dataset [17, 72]), and achieves state-of-the-art performances.

The rest of this chapter is organized as follows: In Section 2.2, we review the works related to skeleton-based human action recognition. In Section 2.3, we present our proposed skeleton sequence representation, the methodology used to train action classification models, in addition to different experiments and analysis. We follow by a conclusion of this chapter in Section 2.4.

2.2 Related Works

In this section, we cover the relevant literature of 3D skeleton-based action recognition methods using hand-crafted features and deep learning networks.

2.2.1 Classical Approaches of 3D Skeleton-Based Action Recognition

Classical approaches in skeleton-based action recognition involve hand crafting features from the time-series data and training machine learning models.

The 3D representations of human actions provide more complete information than 2D RGB videos [71, 80–83]. Many works have investigated 3D skeleton-based action recognition due to the availability of 3D data acquisition devices and real-time skeleton estimation algorithms, as shown in the previous section. A skeleton sequence is formed by a series of frames containing 3D coordinates of the skeleton joints.

To recognize an action class from the skeleton sequence, the most important factor in classical approaches is to design robust features that describe the spatial structure of the skeleton joints in each frame. Another important factor is to extract temporal information among multiple frames of the sequence.

As mentioned in the introduction, Shotton et al. [2] proposed a pose estimation and detection framework for extracting three-dimensional body joint locations from single depth map images. A total of 20 3D joint positions are extracted. The success of the tracking algorithms resulted in the development of the Microsoft Kinect sensor, which offers the ability to easily access skeletal data.

Lv and Nevatia [84] designed a set of spatially local features based on single joints, or a combination of joints. The developed framework is able to encode features in real-time. The authors first applied a normalization for invariance to the environment, body orientation, pose size, and viewpoint. They suggest that only pose vectors may cause a loss of relevant information, and thus, the encoded features are less discriminative. The authors generated a set of features corresponding to the motion of a single joint or combination of related multiple joints. This results in a high dimensional 141 vector. To efficiently represent each feature and action class, a Hidden Markov Model (HMM) [85] is constructed to model the temporal dynamics, where each action class is learned with one HMM. A set of HMMs is combined into a weak classifier, namely AdaBoost [86], to improve the feature discriminative power. This method, like others using raw joints or a combination of joints, is view-variant and struggles more when the subjects are not facing the camera.

Numerous view-invariant skeleton-based approaches have been proposed. One of the pioneering works has been introduced by Xia et al. [87]. The authors proposed to encode the 3D joints locations in a different way. They presented a new method called Histogram Of 3D Joints Locations (HOJ3D), which encodes the spatial occupancy information relative to a predefined joint, ideally the hip center or the skeletal root as they are the most stable ones.

The 3D space is partitioned into n bins in a modified spherical coordinate system. The authors also used HMMs to model action classes and achieved good classification results. However, since the absolute position of joints is used, these features are sensitive to anthropometric variability. To resolve this issue and preserve view-invariance, some approaches proposed to describe actions using the distance between joints. For instance, J. Wang et al. [19] computed the pairwise relative positions of each joint with other joints to represent each frame of the skeleton sequences. The temporal patterns are modeled using Fourier Temporal Pyramid (FTP). Similarly, X. Yang and Y. L. Tian [88] used the pairwise relative positions of the joints to characterize posture features, motion features, and offset features of the skeleton sequences. They applied Principal Component Analysis (PCA) to normalize the features and compute a novel descriptor called EigenJoints.

Zhang et al. [89] used instead the angles between every pair of selected joints in addition to head–floor distance (the distance between the head joint and the floor plane) as features to train a set of SVM-based classifiers. Sempena et al. [90] built a feature vector from joint orientations along time series and applied Dynamic Time Warping (DTW) to recognize some daily human actions.

To deal with viewpoint variability and increase accuracy, other approaches have modeled human actions using more sophisticated geometric tools. Evangelidis et al. [91] proposed a novel view-invariant representation by introducing a descriptor based on the relative position of joint quadruples. Also, Vemulapalli et al. [63] suggested a new representation called Lie Algebra Representation of body-Parts (LAŖP) by computing the geometric transformation between each pair of skeleton body-parts. These descriptors are implicitly unaffected by the viewpoint variability as they are defined using invariant features such as the distance between joints, angles, transformation matrices, etc. However, an alignment pre-processing can be simply applied before undertaking the descriptor computation. For example, we cite the work of Ghorbel et al. [92], where the motion has been modeled by computing and interpolating kinematic features of joints. In this case, the Kinematic Spline Curves (KŞC) descriptor is not view-invariant by nature; thus, the skeletons are initially transferred to a canonical pose.

Although these representations have shown their effectiveness in terms of computation time and accuracy, it has been demonstrated that hand-crafted features could only perform well on specific datasets [93].

This means that features that have been hand-crafted to work on a specific dataset may not be used on other datasets. This makes it difficult for action recognition to be generalized into wider applications. Moreover, as hand-crafted methods can prevent overfitting well, they may lack the ability to learn from bigger datasets. With the availability of larger benchmark datasets lately, future research trend is more likely to shift to using deep learning features.

2.2.2 Deep Learning-Based Skeleton Action Recognition

Deep learning has enabled the replacement of hand-crafted features by learned features, and the learning of whole tasks in an end-to-end way. Several approaches of deep learning have been proposed for skeleton-based human action recognition. They can be categorized into two main categories: Recurrent Neural Networks (RNNs) methods and Convolutional Neural Networks (CNNs) methods.

RNN-Based Methods

RNNs are adopted to capture temporal information from spatial sequences. Basic RNN architectures are notoriously difficult to train [94, 95], and more elaborate architectures are commonly used instead, such as the LSTM (Long Short-Term Memory) [96] and the GRU (Gated Recurrent Unit) [97]. Applications of these networks have shown promising results in skeleton-based action recognition. Du et al. [23, 69] designed an end-to-end hierarchical RNN architecture for skeleton-based action recognition. They divided the human skeleton into five main parts in terms of physical body structure and fed them into five independent bidirectional RNNs/LSTMs for local feature extraction in the first layer. In the following layers, the outputs of the RNNs were concatenated to represent the upper body and lower body, and then each was further fed into another set of RNNs. The global body representation was obtained and fed to the next RNN layer. These features are fed into a fully connected layer, followed by a softmax layer for classification. This method explicitly encodes the spatio-temporal structural information into high-level representation.

Veeriah et al. [70] presented a differential RNN that extends the LSTM structure by modeling the dynamics of states evolving over time. They proposed to add a new gating mechanism for LSTM to model the derivatives of the memory states and explore the salient action patterns.

In this method, all of the input features were concatenated at each frame and were fed to the differential LSTM at each step. This work is one of the first aimed at demonstrating the potential of learning complex time-series representations via high-order derivatives of states. Zhu et al. [71] designed two types of regularizations to learn effective features and motion dynamics. In the fully connected layers, they introduced regularization to learning effective features of the joints at different layers. Moreover, the authors derived a new dropout and applied it to the LSTM neurons in the last LSTM layer. This helps the network to learn complex motion dynamics.

Instead of keeping a long memory of the entire body's motion, Shahroudy et al. [72] proposed a part-aware extension of LSTM to exploit the physical structure of the human body. They split the LSTM memory cell to sub-cells to push the network towards learning the context representations for each body part separately. It is argued that keeping the context of each body part independent and representing the output of the sub-cells as a combination of independent body part context information is more efficient. These RNN-based 3D-action recognition methods only model the long-term contextual information in the temporal domain. However, they neglect the spatial configurations of articulated skeletons where the joints are strongly dependent. To exploit this dependency, Liu et al. [98] proposed a spatio-temporal LSTM (ST-LSTM) network, which extends the traditional LSTM-based learning to both temporal and spatial domains. They explicitly modeled the dependencies between the joints and applied recurrent analyses over spatial and temporal domains concurrently instead of concatenating the joint-based input features. Furthermore, they introduced a trust gate mechanism to make the LSTM robust to noisy input data. Song et al. [99] also proposed an approach that exploits both the spatial and the temporal domain. The proposed a spatio-temporal attention model with LSTM to automatically mine the discriminative joints and learn the respective and different attentions of each frame along the temporal domain.

Motivated by the need to distinguish fine-grained action classes that are intractable using only one network, Li et al. [100] proposed an adaptive and hierarchical framework for fine-grained and large-scale skeleton-based action recognition. In their framework, multiple RNNs are incorporated in a tree-like hierarchy to mitigate the discriminative challenge and use a divide-and-conquer strategy.

CNN-Based Methods

RNN architectures are generally used for modeling sequential data. However, one of their major drawbacks is the exploding and vanishing gradient problem and the difficulty of parallelizing their training. Bai et al. [101] shows that convolutional networks can perform as well as or even better than recurrent networks in many tasks such as speech recognition [102], some tasks of Natural Language Processing (NĻP) like neural machine translation [103,104], classification of long sentences [105], etc. Convolution Neural Networks (CNNs) were successfully introduced for image and video classification. Inspired by this success, particularly for RGB image-based action recognition [66,106,107], there is a growing trend of using deep neural networks for skeleton-based gesture recognition.

The challenge for CNN-based methods is to effectively capture the spatio-temporal information of a skeleton sequence using image-based representation. The main step is then to convert the skeleton sequences into images where the spatio-temporal information is reflected in the image properties, including color and texture. In fact, it is inevitable to lose temporal information during the conversion of 3D information into 2D information.

Wang et al. [108] proposed to encode the skeleton sequences into multiple texture images, namely, Joint Trajectory Map (JŢM), by mapping the trajectories into HŞV (Hue, Saturation, Value) space. Pre-trained models over ImageNet dataset were fine-tuned over the JTMs to extract features and recognize actions. Similarly, Hou et al. [109] drew the skeleton joints with a pen of a specific size and colors to three orthogonal planes. Each resulting image serves as an input to a CNN for classification. Li et al. [110] proposed to encode the pairwise distances of skeleton joints into texture images, namely, Joint Distance Maps (JDM).

They encoded the pairwise distances between joints over skeleton motion sequences into color variations to capture temporal information. These images were fed into a CNN for action recognition. Compared to the first two methods, JDM is less sensitive to view variations. Liu et al. [111] introduced an enhanced skeleton visualization method to represent a skeleton sequence as a series of visual and motion enhanced color images, which implicitly encode the spatio-temporal information of skeleton joints. Their method allows for dealing with the problem of view variation. A multi-stream CNN fusion model is then designed to extract and fuse deep features from enhanced color images, and conduct recognition. Ke et al. [112] represented each skeleton sequence as a clip with several gray images for each channel of the 3D coordinates, which reflects multiple spatial structure information of the joints. The generated images are fed to a CNN to learn high-level features. The features of the three clips at each timestamp are concatenated in a feature vector, which represents the temporal information of the entire sequence. A Multi-Task Learning Network (MTLN) is then adopted to jointly process the feature vectors of all the time-steps in parallel to conduct action recognition. Wang et al. [108] encoded spatio-temporal information carried in 3D skeleton sequences into multiple 2D images, referred to as Joint Trajectory Maps (JTM). Each joint trajectory is projected to the three orthogonal planes, i.e. three Cartesian planes, to form three JTMs. Authors proposed to use the hue, saturation and brightness of color components to encode motion direction and magnitude. CNNs are then adopted to train a classification model. Liu et al. [113] encoded skeleton sequences into spatial and temporal volumes and used 3DCNNs to learn features.

Another form of Convolutional Neural Networks are called Graph Convolutional Networks (GCNs). GCNs generalize the convolutional operation beyond the grid like representation and deal with graph constructions. There are two main ways of construction a GCN: spatial perspective and spectral perspective. Spatial perspective approach directly perform the convolution filters on the graph vertexes and their neighbors. On the other hand, the spectral perspective methods consider the graph as a form of spectral analysis by using the eigenvalues and eigenvectors of the graph Laplacian matrices. In human action recognition using GCNs, it is common to find the spatial perspective-based methods. The first work that used GCNs was done by Yan et al. [114] in 2018.

Authors constructed a spatial graph based on the natural connections between different body joints and added temporal edges between these joints in consecutive frames. A distance-based sampling function is proposed to construct the graph convolutional layer, which is used as a basic block to construct the final spatio-temporal GCN (ST-GCN). Based on ST-GCN, Shi et al. [115] proposed a two-stream adaptive graph convolutional network (2S-AGCN), which can exploit the second-order information of the skeleton to improve the performance of action recognition. Li et al. [116] proposed the Actional-Structural Graph Convolution Network (AS-GCN) which generates the skeleton graph with actional and structural links.

Conventional GCNs though are all feed-forward networks in which it is not possible for low-level layers to access the semantic information in high-level layers.

2.3 Proposed Approach: Using CNNs for 3D Skeleton-Based Action Recognition

Convolutional Neural Networks are used effectively in image classification and have achieved great success in related fields. However, two problems can be addressed:

- how can we apply these models for the recognition of human actions from the trajectories of skeleton joints?
- how can we tackle the temporal dependency of the motion capture sequence?

To answer these two questions, we want to use one method that addresses them both. Instead of having an architecture that deals with each aspect, spatial and temporal, separately, we propose a spatio-temporal representation of motion capture data that allows us to use CNNs for skeleton-based action recognition. We first represent the skeleton sequences as images of all the frames. With the generated images, CNNs are trained to model different actions in the dataset.

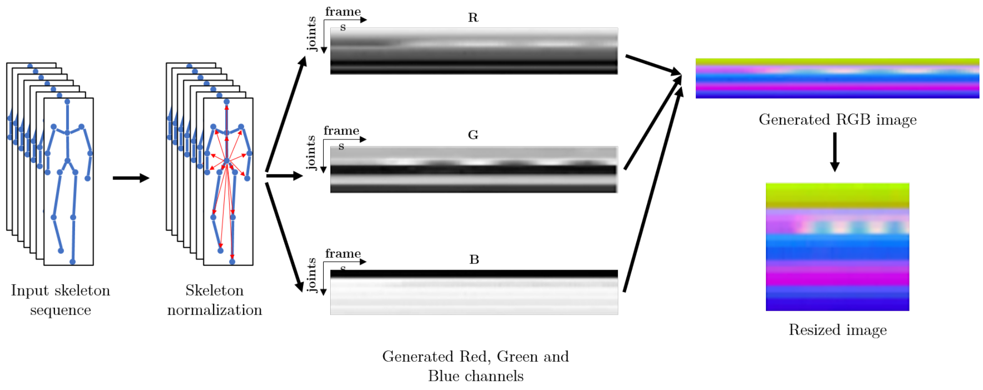


Figure 2.1. Illustration of the proposed sequence-to-image (*Seq2Im*) approach. A skeleton sequence is first normalized to a central joint, then each array of the 3D Cartesian coordinates is transformed into a grayscale image. By merging these images we obtain a RGB color image that is resized to fit the input of the CNN model.

2.3.1 Sequence to Image - Seq2Im: A Novel Representation of 3D Skeleton Sequences

Skeleton sequences only provide the trajectories of the 3D coordinates of joints. The proposed approach allows us to transform the original skeleton sequences of a dataset to a collection of images, thus allows spatial and temporal feature learning using CNNs. Intuitively, one can represent each joint coordinates in each frame as a color pixel in the image. The generated values are obtained by normalizing the X, Y, and Z values between 0 and 255.

As shown in Figure 2.1, the skeleton joint coordinates at each frame are first normalized. Considering that the relative positions of joints provide more useful information than their absolute locations, a reference joint (SpineBase in the case of the Kinect, for example) is used to compute the relative positions of the other joints. This joint is selected as a reference joint since it is stable in most actions. The relative positions of joints are described with the 3D Cartesian coordinates. Three 2D arrays corresponding to X, Y, and Z coordinates, with dimension $J \times F$ are generated, where J is the number of joints, and F is the number of frames in the sequence.

Each array is transformed into a gray image by scaling the coordinate values between 0 to 255 using a linear transformation (eq. 2.1).

$$\begin{cases} r_i(f) = 255 * \frac{x_i(f) - min(X)}{max(X) - min(X)} \\ g_i(f) = 255 * \frac{y_i(f) - min(Y)}{max(Y) - min(Y)} \\ b_i(f) = 255 * \frac{z_i(f) - min(Z)}{max(Z) - min(Z)} \end{cases} \tag{2.1}$$

where: $(x_i(f), y_i(f), z_i(f))$ are the relative coordinates of the joint i to the joint SpineBase, at the frame f. $min(X), min(Y), min(Z)$ are the minimum values for all joints on the 3 axes. $max(X), max(Y), max(Z)$ are the maximum values for all joints on the 3 axes.

$(ri(f), gi(f), bi(f))$ are hence the normalized values of every joint i at the frame f, where each value corresponds to the format of one channel of the RGB space (eq. 2.2, 2.3, and 2.4)

$$R = \begin{pmatrix} r_1(1) & \cdots & r_1(F) \\ \vdots & \ddots & \vdots \\ r_N(1) & \cdots & r_N(F) \end{pmatrix} \tag{2.2}$$

$$G = \begin{pmatrix} g_1(1) & \cdots & g_1(F) \\ \vdots & \ddots & \vdots \\ g_N(1) & \cdots & g_N(F) \end{pmatrix} \tag{2.3}$$

$$B = \begin{pmatrix} b_1(1) & \cdots & b_1(F) \\ \vdots & \ddots & \vdots \\ b_N(1) & \cdots & b_N(F) \end{pmatrix} \tag{2.4}$$

Thus, by merging the three generated images we obtain one RGB image, considering each image as a channel in the new one (X corresponds to the red channel R, Y corresponds to the green channel G, and Z corresponds to the blue channel B).

Depending on the number of frames of the transformed sequence, the generated image-like representation has a very long width compared to its height. If we consider a skeleton sequence of 3 seconds, captured using the Kinect sensor at a framerate of 30 fps, the generated image will have a height of 25 pixels (relative to 25 joints) and a width of 90 pixels. To deal with this problem, we resize this generated image-like representation into a square (128×128, for example) using a bicubic interpolation [117,118]. This makes it possible to use the generated images to fine-tune existing CNN architectures. The generated images are resized to fit the input of the CNN architecture to be trained.

Figure 2.2 shows some results from the NTU RGB+D dataset before the normalization process. The colored triangle on the bottom right of the figure can be considered as a dictionary to interpret these images. The red color represents the X coordinate, the green color represents the Y coordinate, and the blue color represents the Z coordinate. For example, the greener the part of an image is from left to right means that the Y value is getting higher. We can see for example at the images corresponding to the action “Throw” the lines on the top and in the middle (corresponding to the right and left arms joints) are getting, in particular, more of the green color at a certain part of the image then it comes back to the original color. This is the translation of the arms moving up to throw the object before going down again. In the images corresponding to “Falling”, we can see, starting from the middle, that the images are getting more red, to purple. This can be explained by the fact that, when the person falls, all the joints Y values are getting smaller (hence less green).

In order to be invariant to the global position, all 3D joint coordinates are normalized with respect to the “SpineBase” joint (the most stable joint from the Kinect 2). 3D joint coordinates are hence relative coordinates. The order of the joints is also modified in order to have more present visual patterns, and four very noisy joints have been discarded (thumbs and hand tips joints). The final order of joints is the following: 3- Head, 2- Neck, 20- SpineShoulder, 1- SpineMid, 8- ShoulderRight, 9- ElbowRight, 10- Wrist-Right, 11- HandRight, 4- ShoulderLeft, 5- Elbow-Left, 6- WristLeft, 7- HandLeft, 16- HipRight, 17- KneeRight, 18- AnkleRight, 19- FootRight, 12- HipLeft, 13- KneeLeft, 14- AnkleLeft, 15- FootLeft. Figure 2.3 shows some examples of the final representation.

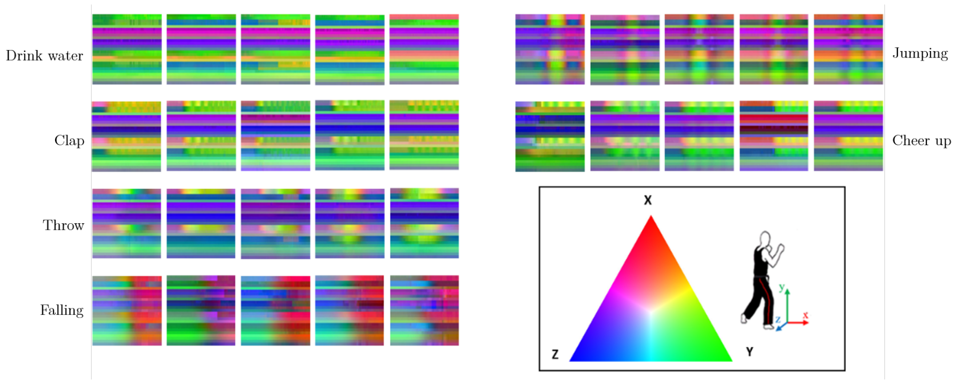


Figure 2.2. Examples showing multiple sequences from the NTU RGB+D dataset after transformation using the proposed *Seq2Im* approach. The colored triangle on the bottom right of the figure can be considered as a dictionary to interpret these images.

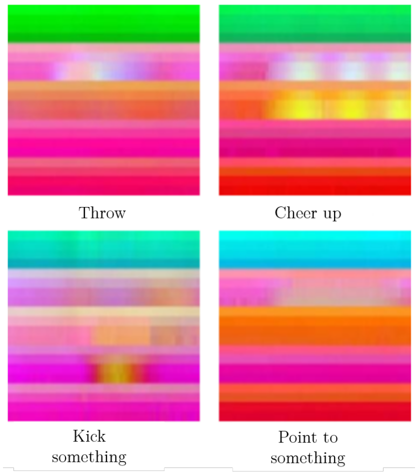


Figure 2.3. Examples showing multiple sequences from the NTU RGB+D dataset transformed into images using the proposed *Seq2Im* approach after normalization to the joint "SpineBase". The normalization allows the skeleton to be invariant to global position.

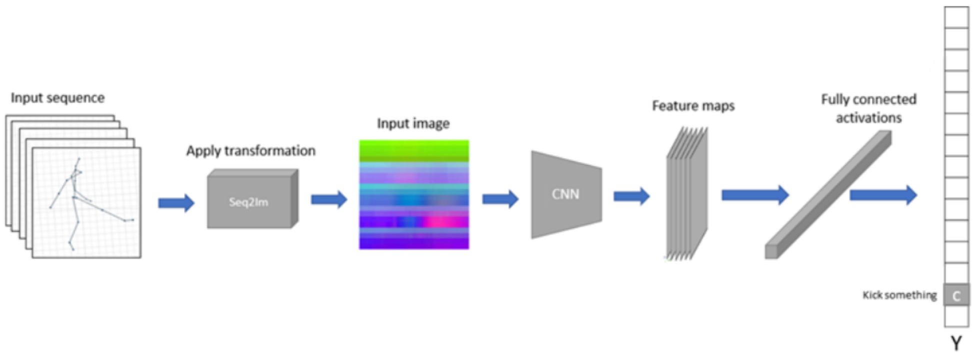


Figure 2.4. Overview of the proposed end-to-end architecture for skeleton-based human action recognition.

The generated images are fed into different models (defined in section 1.4) pre-trained over the ImageNet dataset, and using fine-tuning, we retrain these models to perfom action recognition.

2.3.2 Experimental Results

In this section, the proposed method is first evaluated on the NTU RGB+D [72] dataset and its extension NTU RGB+D 120 [17]. We evaluate different state-of-the-art CNN architectures using different training strategies.

An overview of the end-to-end CNN system used in this work is illustrated in Figure 2.4. After the skeletons are pre-processed, motion capture sequences are transformed into image data representations. These images are fed into different CNN architectures to be trained using different strategies. The CNN part of our models generates automatically features (feature maps) that are fed into a fully connected network for classification.

Data Structure

NTU RGB+D dataset
In order to compare our results with existing works, we evaluate our method mainly on the NTU RGB+D [72].

This dataset was collected using three Kinect V2 sensors at the same time covering three views (-45°, 0°, 45°) and contains more than 56,000 action sequences. A total of 60 different action classes are performed by 40 subjects aged from 10 to 35 years. Among these action classes, 49 are performed by single persons, and 11 are interactions between two people. Only the 49 single person actions were used in our tests (around 47,000 sequences). In addition to depth maps, RGB frames, and infrared (IR) sequences, information of 25 3D joints are available. This dataset is challenging because of the large intraclass and viewpoint variations; however, due to its large scale, it is highly suitable for deep learning.

NTU RGB+D 120 dataset

The NTU RGB+D dataset has been extended later and published under the name NTU RGB+D 120 dataset [17]. This dataset is recorded using the same setups (three Kinect V2 sensors covering three views, -45°, 0°, and 45°). Compared to the first version, this one contains many more action classes. 120 action categories in total are recorded, which are divided into three major groups, including 82 daily actions (eating, writing, sitting down, moving objects, etc.), 12 health-related actions (blowing nose, vomiting, staggering, falling, etc.), and 26 mutual actions (handshaking, pushing, hitting, hugging, etc.). These actions are recorded by 106 subjects aged between 10 and 57 years. Again, only single person actions were used in our experiments (around 110,000 sequences into 94 classes).

The captured Kinect V2 skeletons of both of these datasets have 25 joints in total. The configuration and the given order of joints are shown in Figure 2.5.

Training the Deep Network

In this experiment, 12 state-of-the-art architectures (AlexNet, Inception V3, VGG11, VGG16, VGG19, ResNet34, ResNet50, ResNet152, DenseNet121, DenseNet169, DenseNet201 and SqueezeNet) (see Section 1.4 for more details about these architectures) are trained on the dataset described in the previous section. The numbers after the architectures' names represent the number of layers. This will allow us to analyze the effect of the depth of the models as well. We trained these architectures using three different strategies.

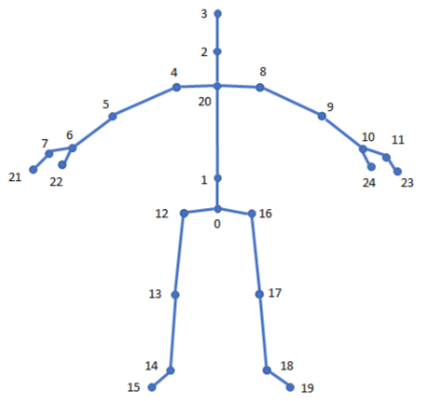


Figure 2.5. Configuration of the 25 joints skeleton captured by the Microsoft Kinect V2.

The first strategy consists in training the CNN from scratch starting from random weights. The other two strategies are based on transfer learning using pre-trained networks. The first transfer learning approach, called shallow retraining approach, consists in fine-tuning only the last added fully connected layer, while the rest of the network is used as a feature extractor. The second approach, called deep retraining approach, fine-tunes all the network layers.

All the 36 training configurations use the same hyperparameters (momentum 0.9, learning rate 0.001, batch size 30, and a number of epochs of 15). The dataset is divided following two evaluation protocols:

- Cross-subject evaluation: the 40 subjects are split into training and testing groups. Each group consists of 20 subjects.
- Cross-view evaluation: the samples of one camera (corresponding to one viewpoint) are used for testing, and samples of the two other cameras are used for training.

All experiments are performed on a machine having the specifications summarized in Table 2.1.

Table 2.1. Specifications of the machine used for our experiments.

	Characteristics
Memory (RAM)	32GB
Processor (CPU)	Intel® Core™ i7-7800X CPU @ 3.50GHertz (Hz) x 12
Graphics (GPU)	2 * GeForce GTX 1080 Ti/PCIe/SSE2 (11GB)
Operating system	Linux Ubuntu 16.04 64 bits

Models Evaluation: Results and Discussion

NTU RGB+D dataset

In this section, we discuss the results obtained for different experiments. We first make an extensive analysis using the first NTU RGB+D dataset. The accuracy and the training time for all the 12 models are shown in Table 2.2 and Table 2.3 for cross-subject and cross-view evaluation protocols respectively.

- ***Cross-subject evaluation protocol***

From Table 2.2, we get the highest scores of accuracy with the models ResNet50 and DenseNet201, around **82%**. This accuracy is obtained using the deep retraining approach. Analyzing this table, we notice that the lowest scores are obtained using the shallow retraining approach. Such low scores are linked to the fact that we retrain only the last added layer, and we keep the rest of the model untouched. The convolutional layers in this situation are used as feature extractors. This can work and give good results in cases where we have images that are similar to the original dataset that was used to pre-train the model (ImageNet). However, our images are more abstract and completely different from the original dataset. The features extracted are hence meaningless in regard to our data.

We can notice relatively higher scores, compared to the shallow retraining, in the case of the retraining from scratch (from random weights). Retraining the whole model allowed it to develop features that are adapted to our data. Nevertheless, the scores are not as high as the deep retraining approached because retraining from scratch may need more data and more time to converge. Transfer learning using deep retraining allowed a quicker convergence.

Table 2.2. Obtained results for the cross-subject evaluation.

Model	From Scratch		Retrain Shallow		Retrain Deep	
	Time (h)	Accuracy (%)	Time (h)	Acc	Time (h)	Accuracy (%)
AlexNet	1.17	65.44	1.09	23.71	1.76	73.19
InceptionV3	1.78	66.07	0.61	25.24	1.68	79.53
VGG11	2.33	68.41	1.95	26.27	3.29	76.91
VGG16	2.82	66.74	2.15	26.96	3.87	77.60
VGG19	3.07	64.90	2.27	22.00	3.70	75.14
ResNet34	1.05	72.02	0.85	37.68	1.05	80.20
ResNet50	1.31	68.09	0.97	38.35	1.30	**82.07**
ResNet152	2.45	67.56	2.09	39.10	2.43	81.18
DensNet121	1.29	74.68	1.53	42.28	1.28	80.96
DensNet169	1.56	76.10	1.49	43.80	1.53	81.72
DeseNet201	1.86	76.22	1.84	44.22	1.82	**82.00**
SqueezeNet	0.61	65.73	0.55	31.45	0.62	72.09

We can also conclude that the use of more layers does not automatically imply a higher accuracy. It can be the opposite in many cases, like VGG and ResNet. VGG architecture with 19 layers, for example, gives lower accuracy than the one with 11 and 16 layers in a deep retraining approach. This can be explained by the fact that the more layers we have, the more risk we have of overfitting. We can finally notice that SqueezeNet gives relatively high scores, particularly in a deep retraining approach. SqueezeNet is a very small network with few parameters. It has a total size of less than 0.5MB. Compared to AlexNet, for example, that has a total size of 240Mb, it makes it easy and practical to fit into embedded systems and smartphones. Evaluation of models' performances comparing training time and accuracy is presented in Figure 2.6.

- ***Cross-view evaluation protocol***
From Table 2.3, we get the highest scores in the "Cross-View" evaluation protocol with the models ResNet34, ResNet152, and DenseNet201 of about **86%**. The same conclusions as in the "Cross-Subject" evaluation can be drawn.

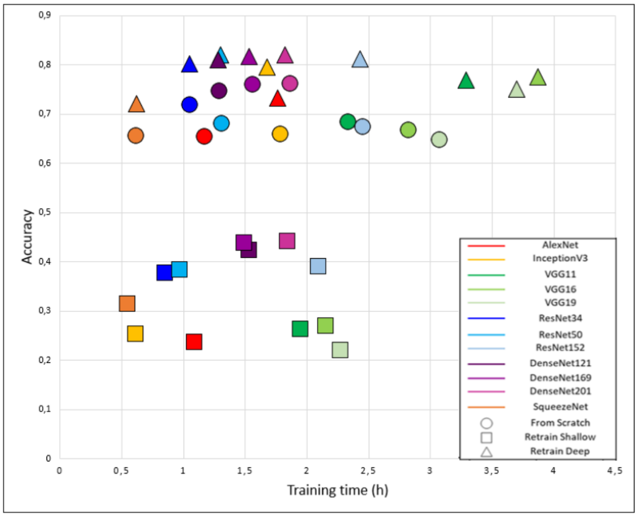


Figure 2.6. Accuracy vs. training time for different models in cross-subject protocol, following the three training strategies: from scratch, shallow retraining, and deep retraining.

Evaluation of the models' performances by comparing training time and accuracy, is also displayed in Figure 2.7, which is similar to the previous evaluation scenario (cross-subject).

The obtained results can be compared to several works in the state of the art that used the same dataset (Table 2.4). The highest score we obtained is 82,07% using ResNet50 in the cross-subject evaluation protocol and 86,54% using DenseNet201 in the cross-view evaluation protocol. Our results outperform most of the state-of-the-art methods in both cross-subject and cross-view protocols. Our results are obtained by transforming motion sequences into images and using pre-trained models without developing new architectures. Specifically, we improve accuracy by 3 to 8% compared to other techniques using CNNs with the image-like transformation of motion sequences. This proves that adapting the weights from pre-trained models for a new task improves the performance of deep learning networks and gives high scores even if the original task is very different.

Table 2.3. Obtained results for the cross-view evaluation.

Model	From Scratch		Retrain Shallow		Retrain Deep	
	Time (h)	Accuracy (%)	Time (h)	Accuracy (%)	Time (h)	Accuracy (%)
AlexNet	0.92	64.00	1.18	22.67	1.15	74.86
InceptionV3	1.73	64.82	0.61	25.73	1.54	80.46
VGG11	1.75	75.73	2.13	26.93	2.27	81.44
VGG16	2.23	76.35	2.18	25.30	2.75	82.69
VGG19	2.50	76.43	2.25	21.60	2.99	80.01
ResNet34	1.01	69.97	0.83	36.72	1.03	**86.00**
ResNet50	1.27	64.57	0.96	37.74	1.29	85.92
ResNet152	2.67	73.75	1.61	39.06	2.38	**86.11**
DensNet121	1.30	78.77	0.94	41.04	1.27	85.50
DensNet169	1.64	77.67	1.10	43.02	1.51	84.89
DeseNet201	1.95	79.24	1.27	44.22	1.80	**86.54**
SqueezeNet	0.4	70.21	0.57	30.65	0.62	77.60

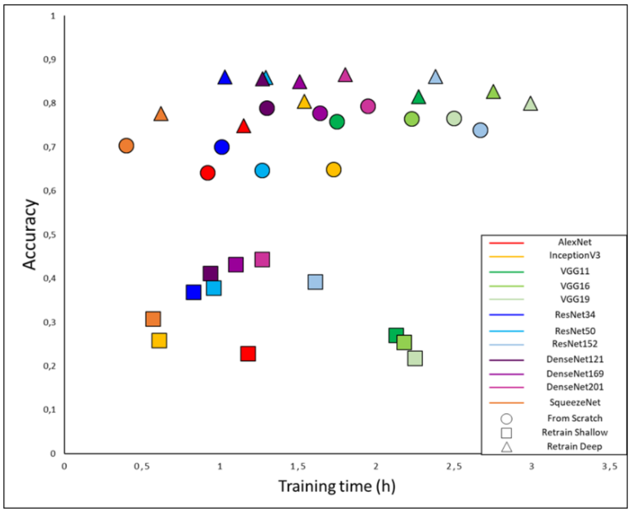


Figure 2.7. Accuracy vs. training time for different models in cross-view protocol, following the three training strategies: from scratch, shallow retraining, and deep retraining.

Table 2.4. Comparison of our proposed approach with the state-of-the-art results.

Method	Cross-Subject Accuracy (%)	Cross-View Accuracy (%)
Using CNNs		
Two streams 3DCNN [113]	66.85	72.58
CNN + MTLN [112]	79.57	84.83
Trajectory maps + CNN [108]	76.32	81.08
Conversion into image + CNN [119]	75.20	82.10
Using other DL methods		
HBRNN [23]	59.07	63.97
Deep RNN [72]	56.29	64.09
Deep LSTM [72]	60.69	67.29
PA-LSTM [72]	62.93	70.27
LieNet [120]	61.37	66.95
ST-LSTM [98]	69.20	77.70
Our Method	**82.07**	**86.54**

NTU RGB+D 120 dataset

We have followed the same evaluation procedures as for the first NTU RGB+D dataset. In order to gain time, we only made a deep retraining of the 12 models, and we followed the two training evaluation protocols: cross-subject and cross-view (or cross-setup). The highest accuracy was obtained with DenseNet161: 73.7% for the cross-subject evaluation, and 76.6% for the cross-setup evaluation. Compared to the state-of-the-art results (Tables 2.5 and 2.6), our method outperforms all the other approaches, even when other modalities are used (RGB, Depth and 3D skeletons). This proves that our approach is highly effective for skeleton-based action recognition, despite its simplicity to realize.

Other Experiments

The above experiments performed on the NTU RGB+D datasets take into account only Kinect V2 skeleton sequences.

Table 2.5. The results of different methods designed for 3D Skeleton-based human activity recognition using the NTU RGB+D 120 dataset.

Method	Cross-Subject Accuracy (%)	Cross-Setup Accuracy (%)
Part-Aware LSTM [72]	25.5	26.3
Soft RNN [121]	36.3	44.9
Dynamic Skeleton [122]	50.8	54.7
Spatio-Temporal LSTM [98]	55.7	57.9
Internal Feature Fusion [123]	58.2	60.9
GCA-LSTM [124]	58.3	59.2
Multi-Task Learning Network [112]	58.4	57.9
FSNet [125]	59.9	62.4
Skeleton Visualization (Single Stream) [111]	60.3	63.2
Two-Stream Attention LSTM [123]	61.2	63.3
Multi-Task CNN with RotClips [126]	62.2	61.8
Body Pose Evolution Map [127]	64.6	66.9
Our method	**73.3**	**76.6**

Table 2.6. The results of other methods combining different modalities (RGB, depth and 3D Skeleton data) for human action recognition using the NTU RGB+D 120 dataset. [17]

Data Modality	Cross-Subject Accuracy (%)	Cross-Setup Accuracy (%)
RGB Video	58.5	54.8
Depth Video	48.7	40.1
RGB Video + Depth Video	61.9	59.2
RGB Video + 3D Skeleton Sequence	61.2	63.1
Depth Video + 3D Skeleton Sequence	59.2	61.2
RGB Video + Depth Video + 3D Skeleton Sequence	64.0	66.1

We wanted to make similar experiments on other sources of data with other qualities and skeleton configurations in order to validate the proposed approach.

MSR action 3D dataset

MSR Action 3D [128] is one of the earliest datasets recorded using a depth sensor. The samples of this dataset were limited to depth sequences. Later, body joints information was added. There are 20 action classes performed by ten subjects.

Table 2.7. The results of different methods designed for 3D Skeleton-based human activity recognition using the MSR Action 3D dataset.

Method	Accuracy (%)
Sequence of Most Informative Joints [129]	29.41
Recurrent neural network [130]	42.50
Dynamic time warping [131]	54.00
Hidden Markov models [84]	63.00
Multiple instance learning [132]	65.70
EigenJoints + NBNN [88]	72.00
Structured Streaming Skeletons [133]	81.70
DBN + HMM [134]	82.00
Conceptors of Skeleton Joint Trajectories [135]	83.40
Our method	**92.18**

Each action was performed twice to three times by each subject. In our experiment, we focused only on the skeletal data. There are in total 557 skeletal sequences, where each sequence has 20 joint positions. In this dataset, we use the cross-subject experiment, where the sequences of five subjects are used in training, and the rest are used for testing. Table 2.7 compares our results with some of the state-of-the-art skeleton-based action recognition approaches. Our proposed approach shows its superiority over other techniques with an accuracy of 92.18%. This proves that our method can deal with noisy data provided by the Kinect V1 sensor.

HDM05 dataset

HDM05 [136] contains 2,343 sequences of 130 classes executed by various actors. Each action was performed 10 to 50 times by each actor. The dataset was recorded using a Vicon mocap system, where 31 reflective markers were placed on the actors' bodies. The 3D positions of these markers are provided. Following Huang et al. [137], we conducted ten evaluations, each of which selects randomly half of the sequences for training and the other half for testing.

Table 2.8. The results of different methods designed for 3D Skeleton-based human activity recognition using the HDM05 dataset.

Method	Accuracy (%)
SPDNet [137]	61.45
SE [63]	70.26
SO [138]	71.31
LieNet [120]	75.78
Our method	**83.33**

However, due to the long time it takes to train CNNs, we only ran one experiment for this case. Table 2.8 lists the average accuracy of the proposed method and the results obtained in the previous works. Our approach achieved the highest accuracy scores with 83.33%, compared to other methods.

From these last experiments, we can conclude that our approach can be easily adapted to different types of skeleton data, even when they are provided with low-cost sensors that generally provide noisy and low-quality skeleton sequences.

2.4 Conclusion

In this chapter, we addressed the problem of 3D skeleton-based human action recognition. An effective yet simple method is proposed to represent skeleton sequences into 2D RGB images. Such a representation allows us to use powerful image classifiers to recognize human actions, particularly CNNs, using fine-tuning techniques. The experimental results on different public datasets provided by different sensors showed the efficiency of the representation even without extracting complex features. The NTU RGB+D dataset was recorded using three Microsoft Kinect sensors at the same time. These sensors were placed on different angles in order to test the view-invariance of the proposed method.

In our approach, we pre-processed our skeleton sequences and then normalized each coordinate between 0 and 255 in order to generate RGB image-like representations. The action classification was conducted using multiple state-of-the-art image classification architectures and the process of fine-tuning. Our method outperformed most of the state-of-the-art results on this dataset and was able to prove its view-invariance character.

Moreover, our approach was tested on other action classification datasets provided by different sensors from low-cost (Kinect V1) to high-precision mocap system (Vicon), and with different skeleton configurations, and achieved the highest scores. This proves that our method is independent of the quality of data, and the number of joints forming the skeleton.

Chapter 3

Human Action Recognition From RGB Videos

Contents

This chapter is based on the following publication:

- Action recognition based on 2D skeletons extracted from RGB videos. In MATEC Web of Conferences. 2018. [139]

3.1 Introduction

In the previous chapter, we developed a novel method to represent the 3D motion capture sequences into color images. We proved that pre-trained CNN classifiers could be retrained for our task of action classification, and the results were very promising.

In this chapter, we want to go a step further where another problem was raised. How can we get cheaper devices than a Kinect? And how can we get more motion capture data?

To develop the first question, the cheapest devices that we can get in the market are RGB cameras (webcams, for example). So if we find a way to get skeleton data from RGB videos and succeed in using our approach on these data, we can answer this problem. YouTube CEO Susan Wojcicki reported in 2015 at the VidCon conference [140] that more than 400 hours of videos are being uploaded on YouTube alone every single minute. This means that a huge amount of video data is already available online, and it increases continuously. This can answer our second question: how to get more data? This is a great motivation to try to develop methods that can learn from video data. As a matter of fact, developing such a method can open access to a new area of research where any kind of video capturing devices can be used for motion capture. It also allows to benefit from the huge amount of data already available on the internet. Therefore, human action recognition systems can be improved, and the access to data for this domain can be democratized.

The solution to these questions lies in the development of precise and effective tools to capture human poses from RGB videos. With the advances of deep learning, many attempts have been made to develop such tools. In 2017, a real-time approach to detect the 2D pose of multiple people in an image has been published by Zhe et al. from Carnegie Mellon University [12] and a tool has been developed called OpenPose. OpenPose is publicly available and free for non-commercial use. It is considered as the state-of-the-art approach for real-time human pose estimation, which attracted the research community thanks to its great results and performances. This was the opportunity for us to address our problem of human action recognition from videos.

In this chapter, we propose to follow the same process as in the previous chapter for action recognition.

But first, we use the OpenPose framework to extract a skeleton pose from each frame of the RGB videos to generate skeleton sequences. Then, using the *Seq2Im* method proposed in the previous chapter, we transform these sequences into images where each pixel in the generated image represents one joint in one frame. Pre-trained CNNs are then used to train new models for human action recognition.

The main contributions of this chapter are summarized as follows: 1) We propose to perform human action recognition from videos by extracting skeleton sequences using the OpenPose framework. 2) We propose to transform these skeleton sequences into RGB images and then fine-tune existing CNN architectures to perform human action recognition. The generated skeletons extracted by the OpenPose framework having only two coordinates, we analyze different ways to replace the third coordinate in order to generate a RGB image. 3) The proposed method is validated on the large scale NTU RGB+D benchmark dataset and achieves state-of-the-art performances.

The rest of this chapter is organized as follows: In Section 3.2, we review the works related to human action recognition from videos. In Section 3.3, we present the OpenPose framework for human pose estimation from RGB videos. We follow by presenting our proposed approach of human action recognition from videos in Section 3.4, and conclude our work in Section 3.5.

3.2 Related Works

In video-based human activity recognition, a system takes as input a video clip (a sequence of video frames) and outputs the corresponding class. Human activity recognition from videos is a challenging task due to multiple factors. Human actions are complex and have many different degrees of freedom. As it is intuitive for a human to classify actions, it is not evident for a machine to extract this knowledge. Besides, different people may perform the same action in different ways, and this can be true even for one person performing the same actions multiple times, it can be different at each repetition. Moreover, and similarly to image datasets, different videos from one action class may have varying light-conditions, different sizes, and shapes of objects, occlusions, cluttered background, different camera viewpoints, etc. These elements can be found in the same video from one frame to another.

For this reason, developing a successful model requires to have a good feature representation that is robust to different variations in the data, discriminative for different action classes, and can be generalized to different datasets. Furthermore, the extracted features must include both spatial and temporal information for robust classification. The spatial information indicates where the action is happening, while the temporal information concerns the motion clues of different objects, in addition to the changes in the background.

Multiple human action recognition systems have been proposed in literature which attempted to solve these challenges. In early works, researchers developed hand-crafted features for human activity recognition from videos. Davis and Bobick [141] computed the differences between binary silhouettes and accumulated them over the spatial and temporal domain constructing Motion Energy Image (MEI) and Motion History Image (MHI) to generate action templates. The statistical model of moments (mean and covariance) is used to match unknown sequences to the closest templates. Gorelick et al. [142] stacked the 2D silhouettes in spatio-temporal volume to form 3D shapes. They exploited the properties of the solution to the Poisson equation to extract space-time features like local space-time saliency, action dynamics, shape structure, and orientation. For classification, they used the template matching method by comparing the target sequence to all the sequences of the dataset using the nearest neighbor procedure (with Euclidean distance). Yu and Aggarwal [143] detected first the extremities of the human body, such as the head, the hands, and the feet, from the body contour. They built then feature vectors out of these detected extremities to train a Hidden Markov Model (HMM) for human action recognition. These approaches, in addition to others [143], are holistic methods that rely on people detection (as regions of interest ROI).

Other hand-crafted methods require neither human segmentation nor people detection in order to perform action recognition. A large number of approaches extract local spatio-temporal features in videos. Laptov [144] introduced spatio-temporal interest points, which are an extension of the image Harris detector to video. These interest points are local maxima of a cornerness criterion based on the spatio-temporal second-moment matrix at each video point. Dollár et al. [145] combined a 2D Gaussian filter in space and a 1D Gabor filter in one cornerness function.

In a similar work, Bregonzio et al. [146] proposed to use instead 2D Gabor filters of different orientations. Another successful hand-crafted representation uses Dense Trajectories, as proposed by Wang et al. [147]. Authors densely sampled feature points in each video frame, then they tracked them in the video using optical flow. They computed several descriptors along the trajectories of feature points to capture the shape, the appearance and the motion information. Action classification is then performed with a bag-of-features representation and a SVM classifier.

Encouraged by the great success of deep learning, there have been multiple attempts to use, especially CNNs for action recognition from videos. One of the main advantages of CNNs is that they learn both features and classification boundaries in an end-to-end process. Simonyan and Zisserman [66] proposed a two-stream convolutional network to learn spatio-temporal features. Their network uses visual frames and optical flows between frames as two separate inputs and fuses the obtained scores to obtain the final prediction. This work has been further extended in many other works [148–150].

With the growing computation power of GPUs and the availability of large-scale datasets, 3D CNNs have attracted increasing attention. Tran et al. [151] designed a 3D CNN with 11 layers to learn spatio-temporal features on the Sports-1M dataset [107]. Qiu et al. [152] proposed a Pseudo-3D Residual Net (P3D) to build a deeper 3D CNN model. They decomposed a 3D convolution into a 2D convolution, followed by a 1D one. In another work, Tran et al. [153] proposed a similar architecture that is referred to as 2+1 D. Carreira and Zisserman [143] proposed a method called Two-Stream Inflated 3D ConvNet (I3D) which aims to enlarge the perception field along the temporal direction. This is based on the approach proposed by Tran et al. [143], where the parameters are initialized by inflating the weights of pre-trained 2D CNNs.

Recurrent Neural Networks (RNNs) have also been used in order to model the temporal relationships among video frames. Wu et al. [154] trained two CNNs to extract the spatial and short-term motion features separately. These features are combined in a regularized feature fusion network for classification. On top of these features, RNNs, and particularly Long Short Term Memory (LSTM) networks, are applied to further model longer-term temporal features. Similarly, Yue-Hei Ng et al. [155] used CNNs to learn spatial features for each frame, then LSTMs to model the temporal dynamics.

Luo et al. [156] proposed to use a RNN-based Encoder-Decoder framework to effectively learn a representation that predicts the sequence of basic motions. Similar to previous approaches, they first extract high-level features from each input frame using CNNs.

These methods analyze the whole video sequences and ignore the semantic meaning of human actions, which are structured body movements. Multiple recent studies [157–161] extract first the whole human body or body parts and use them for further analysis.

In non-deep learning methods, we can find the work of Wang et al. [162], where they improved an existing pose estimation method and then designed pose features to represent both spatial and temporal configurations of body parts. Cheron et al. [163] proposed a Pose-based Convolutional Neural Network descriptor (P-CNN) for action recognition. They first use a pose estimation method to extract different body joints, and then they combine motion and appearance-based CNN features computed for each track of body parts.

Zolfaghari et al. [164] estimated poses and video frames and encoded them directly by a multi-stream 3D CNN. Their proposed network architecture integrates multiple cues (raw images, optical flows, and body poses) sequentially via a Markov chain model.

The above methods estimate first the human body pose in one network and do not use the relative positions of different body joints over time. Our proposed method takes advantage of the state-of-the-art human body pose estimation (OpenPose) in order to extract body joints, then a spatio-temporal representation is used to train CNN models for human action classification. Our proposed method achieves state-of-the-art performances on the NTU RGB+D dataset, which is currently the biggest challenging benchmark dataset for action classification.

3.3 OpenPose: Real-time Multi-Person Pose Estimation From RGB Videos

In this section, we will briefly introduce the OpenPose framework. We refer interested readers to [12] for more details.

OpenPose, proposed by researchers at Carnegie Mellon University (CMU) in 2017, is a library for real-time multi-person pose estimation. It is based on a bottom-up approach instead of detection-based approaches in other works. In this approach, the body parts are detected by the model, and a final parsing is used to extract the pose estimation results. OpenPose can jointly detect, and in real-time, the human body, hands, and facial keypoints. In total, 130 keypoints per person and on a single image. Moreover, the computational performance on body keypoints estimation is invariant to the number of detected people in the image. OpenPose framework works following three main steps, which are summarized in Figure 3.1.

1. First, an input RGB image is fed into a two-brunch multi-stage CNN that produces two different outputs. In fact, the image is first analyzed by a pre-trained convolutional neural network such as the first ten layers of VGG-19 (see Section 1.4, *VGGNet*), to produce a set of feature maps F, which are then passed to the mentioned two-brunch multi-stage CNN. The architecture of this CNN is illustrated in Figure 3.2. It consists of two branches: the top branch (beige) predicts the confidence maps (Figure 3.1, *b.*) of different body parts locations such as the right eye, left eye, right elbow, etc. The bottom branch (blue) predicts the affinity fields (Figure 3.1, *c.*), which represent the degree of association between different body parts. The CNN is also composed of multiple stages. At the first stage (first beige and blue blocks on the left part of Figure 3.2), the network produces an initial set of detection confidence maps S and a set of Part Affinity Fields (PAFs) L. Then, in each subsequent stage (second block on the right part of Figure 3.2), the predictions from both branches in the previous stage are concatenated, along with the image Features F and used to produce more refined predictions. In the OpenPose implementation, the number of stages $T = 6$.
2. The second step of the OpenPose pipeline is processing the confidence maps and part affinity fields by a greedy inference (by performing a set of bipartite matchings) to associate body part candidates (Figure 3.1, *d*).
3. Finally, they are assembled into full-body poses for all the people in the input image (Figure 3.1, *d*). OpenPose provides a total of 18 body

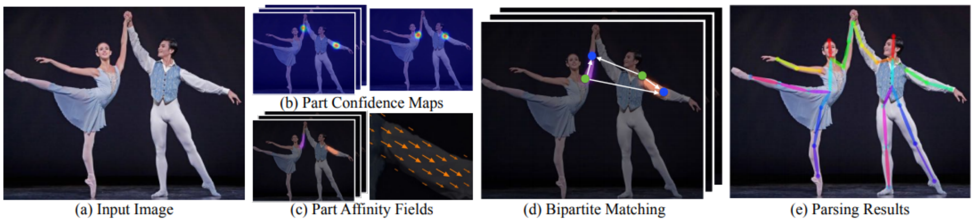


Figure 3.1. Overall pipeline of the OpenPose framework. *a.* The entire image is taken as the input for a CNN to jointly predict confidence maps for body part detection (*a.*) and Part Affinity Fields (PAFs) for part association (*c.*). *d.* The parsing step performs a set of bipartite matchings to associate body part candidates. *e.* Finally, they are assembled into full body poses for all people in the image. [12]

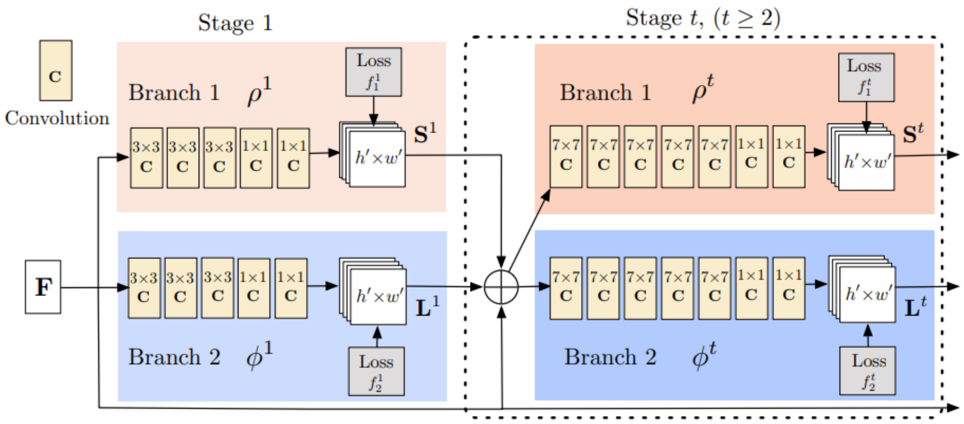


Figure 3.2. Architecture of the two-branch multi-stage CNN. [12]

joints locations for each person in each frame, and the corresponding confidence scores.

OpenPose has many advantages; it can estimate the human body poses in real-time, and it requires only RGB cameras to extract skeletons from a video. Moreover, it allows partial skeleton detection, i.e., unlike the Kinect sensor, which always fits all the joints to the body even if it is partially viewed, OpenPose ignores the joints that are not seen.

However, some disadvantages can also be highlighted, such as the lack of depth information since only 2D coordinates are given. Furthermore, OpenPose coordinates locate each detected landmark into the processed frame according to a global coordinate system, where the origin of the axis is on the upper left frame corner. Therefore, the data are location and size-dependent.

In the next section, we will present our proposed approach in details. First, we will detail the extraction of skeleton sequences, their processing, and how to use them to train our action recognition models.

3.4 Proposed Approach for Human Action Recognition from RGB Videos Using CNNs

In the previous chapter, we proposed a novel spatio-temporal representation of 3D skeleton sequences into RGB images. The X, Y, and Z coordinates were normalized and mapped into the color domain where the red color (R) represents the X coordinate, the green color (G) represents the Y coordinate and the blue color (B) represents the Z coordinate. However, in this new situation, we only have two coordinates (X and Y), and as mentioned earlier, the depth information is not provided. In fact, even if the depth information is not provided directly, the information is included within the given X and Y coordinates. The zoom effect, when a person is moving forward and backward, for example, includes itself the depth information. This because the given joints' coordinates are given in pixels and not in real-world coordinates like the Kinect sensor data. This is similar to the situation where a person can still get the depth information and can estimate the location of an object or another person even if (s)he looks with only one eye.

In order to adapt our proposed approach presented in Chapter 2 to this new kind of skeleton data, and to generate a RGB image, we need to find additional information that replaces the third channel. In the following experiments, we try and analyze different configurations to find the best representation for our data. We use the NTU RGB+D dataset as a reference where the RGB videos were provided as well. This dataset has been described in Section 2.3.2.

Figure 3.3 summarizes the different steps of our proposed approach for action recognition from videos.

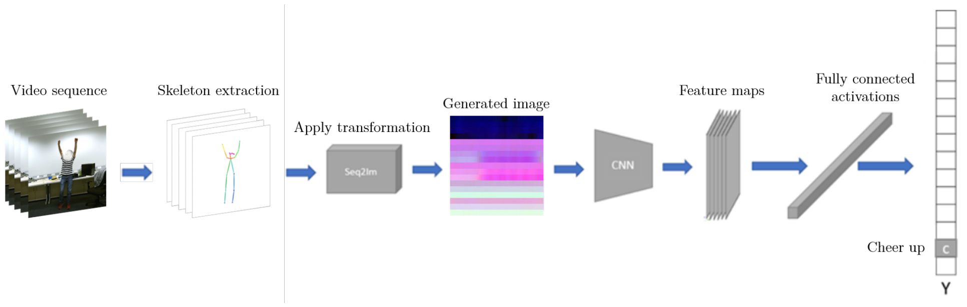


Figure 3.3. Overview of the proposed end-to-end architecture for the classification of human actions from RGB videos.

We can notice that it is very similar to the architecture proposed in Chapter 2, except that we proceed first with the extraction of skeleton sequences directly from videos. Furthermore, we apply the same skeleton transformation by normalizing different joint coordinates to the virtual landmark situated between the two hip joints (the mean of the two joints), as they are among the most stable joints. This allows the joint coordinates to be invariant to the global position. After choosing the final configuration of our data, we transform our sequences into RGB images that are fed into a pre-trained CNN for feature extraction, and then the classification is performed using a fully connected layer, followed by a Softmax layer.

3.4.1 Skeleton Data Extraction and Processing

OpenPose provides the positions of 18 body joints (Figure 3.4) in real-time (Section 3.3). The data is obtained in JSON format containing the 2D body part locations (X, Y) in addition to the detection confidence C of each joint. We extract the skeleton sequences from the videos of the NTU RGB+D dataset [72]. The NTU RGB+D dataset was collected using three Kinect V2 sensors at the same time covering three views (-45°, 0°, 45°) and contains more than 56,000 action sequences as described in Section 2.3.2. In addition to 3D skeleton sequences, this dataset also contains the corresponding RGB videos from the Kinect sensors. We recall that this benchmark dataset is evaluated using two protocols: 1) cross-subject, where the data of a group of subjects is used for training, and the rest is used for validation.

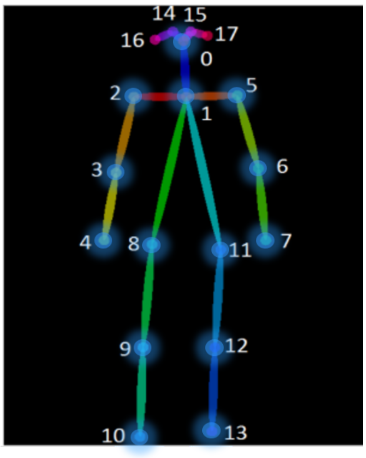


Figure 3.4. Illustration of the body skeleton extracted by the OpenPose framework. To validate the proposed approach, we test the use of all joints (18) vs. only the 14 joints highlighted in blue (all the joints except 14, 15, 16 and 17).

2) cross-view, where the data provided by two cameras is used for training, and the rest is used for validation.

As mentioned earlier, one dimension is missing in order to generate a color image with the three channels red, green, and blue. We empirically test two different configurations in order to replace the third coordinate Z. 1) In the first setup, we use the mean of X and Y coordinates $(\frac{X+Y}{2})$. 2) In the second setup, we take advantage of the confidence score C of each joint provided and use them as a third dimension. For both configurations, we first compute relative positions of different joints to the mean of right and left joints. Then we apply the *Seq2Im* method described in Section 2.3.1 and follow the steps described in Figure 3.3. Moreover, as OpenPose ignores the hidden joints and sets their corresponding values to *unexpected*, i.e., -1, we analyze different joints frame by frame to find those who are the most unstable. We found that the joints corresponding to the ears and eyes are missing in almost 40% of the time. We decide to add a second set of tests where we discard these four joints (see Figure 3.4, where we take into account only the joints that are highlighted in blue). Figure 3.5 shows two examples of sequences transformed into images using *Seq2Im* method.

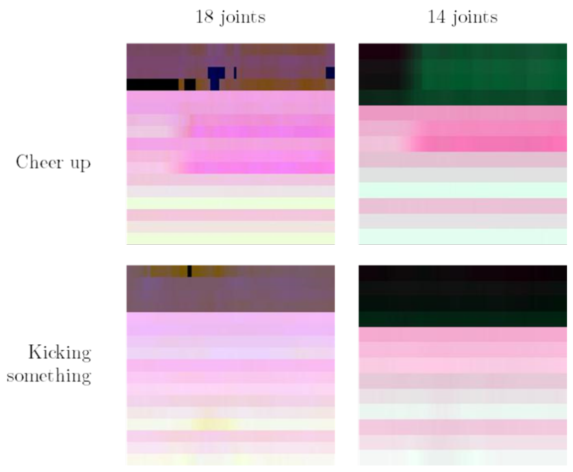


Figure 3.5. Examples showing two sequences transformed into images using *Seq2Im* method. In the first column (left), all the joints are used for the transformation. In the second column (right), only 14 joints are used (highlighted in blue in Figure 3.4).

The first column corresponds to the actions with 18 joints, while the second column, the four joints corresponding to the eyes and ears are ignored. We can notice black lines in the left images in the upper zone. These lines correspond to the joints that were not detected by OpenPose. These joints have been eliminated in the second set of tests and are not seen on the images on the second column any more. We can also notice that we do not have the same range of colors between both kinds of images. This is because by eliminating joints, the range of values after transformation to RGB channels changes.

In order to evaluate these representations, we train the DenseNet201 model (see Section 1.4) for different setups using transfer learning (deep retraining), where the model was pre-trained on the ImageNet dataset [57], as it gave the highest scores in the previous chapter. We only follow the cross-subject protocol for these experiments, and we compare the obtained accuracy scores.

Table 3.1 shows the obtained results for different configurations. First, we notice that in different configurations, the obtained scores are over 80%, which is considered as high compared to the state-of-the-art results from the previous chapter.

Table 3.1. Comparison between different configurations to select the best representation of OpenPose skeleton sequences.

Setup	18 joints	14 joints
$(X, Y, \frac{X+Y}{2})$	80.34%	80.98%
(X, Y, C)	**82.72%**	**82.71%**

We also notice that there is a very small difference between the 18 joints and 14 joints configurations, which makes us believe that the network may ignore the four additional joints. Finally, we notice that the use of the confidence score of joints adds around 2% from using the mean of X and Y coordinates. This may be explained by the fact that the mean of the coordinates is considered redundant information, while the confidence score provides additional information that can help the decision made by the model. For the next experiments, we only focus on the second configuration, where the confidence score is used as a third channel.

3.4.2 Experimental Results and Discussion

In this section, we validate our proposed approach using multiple experiments. These experiments are inspired by Chapter 2, and we follow the same steps. First, we compare different state-of-the-art image classifiers. Then, we study the effect of transfer learning (see Section 2.3.2) for our proposed approach. Finally, we compare the obtained scores with the state-of-the-art results on the same benchmark dataset. All the experiments are performed on the same machine as in the previous chapter, with the specifications summarized in Table 2.1.

CNN Models Comparison

In this section, we compare different models of image classification in terms of accuracy scores. All these models are tested with both data representations (18 and 14 joints), encoding (X, Y) and C as RGB channels.

Table 3.2. Results obtained by deep retraining of different models using 18 joints and 14 joints in cross-subject protocol.

Model (retraining deep)	Accuracy using 18 joints (%)	Accuracy using 14 joints (%)
SqueezeNet	75.79	75.80
AlexNet	74.54	74.05
Inception V3	81.98	81.53
DenseNet169	81.94	82.65
DenseNet201	**82.72**	82.71
ResNet34	82.59	81.36
ResNet50	**83.35**	81.72
VGG13	79.10	78.87
VGG19	78.50	78.98

We evaluate the following models: AlexNet [6], Inception V3 [7], VGGNet [8] (with 13 and 19 layers), ResNet [9] (with 34 and 50 layers), DenseNet [10] (with 169 and 201 layers), and finally SqueezeNet [11], which are described in Section 1.4. These models are trained using transfer learning (the models are pre-trained on the ImageNet dataset first).

The obtained results are summarized in Table 3.2. First, we can confirm our conclusion of the previous section, where the elimination of the four joints corresponding to the ears and eyes do not affect a lot the accuracy of the models. This because these landmarks may not be very important for the decision process. The results obtained with 18 joints are though slightly higher than 14 joints. We can also notice that the highest results were obtained using ResNet50 and DenseNet201 with 83.35% and 82.72% of accuracy, respectively. These two models gave the highest scores for action classification from 3D skeleton sequences as well.

Transfer Learning: Deep Retraining vs. Shallow Retraining vs. Training from Scratch

In this section, we evaluate our models using the three different training strategies: 1) Training from scratch. 2) Transfer learning from pre-trained models and retraining all the weights (deep retraining). 3) Transfer learning from pre-trained models and retraining only the last added fully connected layer (shallow retraining). The same hyperparameters are used for different configurations and are the same as in the previous chapter (momentum 0.9, learning rate 0.001, batch size 30, and a number of epochs of 15). Again, we only focus here on the cross-subject protocol to evaluate the different training strategies.

Table 3.3 summarizes the obtained results. As expected, the deep retraining is by far the best strategy for all the evaluated models. It is also worth noticing that the models were pre-trained on the ImageNet dataset, which is completely different from our dataset, and the obtained results are still very high. Compared to the training from scratch, where the weights are initialized to random values, the pre-trained weights from ImageNet helped to learn knowledge from the image, and thus helped to have a faster convergence and to better avoid local minima. Shallow retraining is the worst strategy as the convolutional parts of the networks are left untouched (weights are frozen during the training). Because the initial dataset (ImageNet) is very different from ours, the learned features by the CNN from the pre-trained model are not relevant for our task. Only retraining the last fully-connected layer cannot help the classifier and do not provide good results. The highest scores are obtained using DenseNet and ResNet models (more than 82%). This shows that the connections between different layers improve the performances even despite the low amount of parameters to learn, compared to the other architectures. SqueezeNet, on the other hand, has very few layers compared to the other architectures (i.e., very few parameters), which explains the low accuracy obtained. It is yet higher than the accuracy obtained by AlexNet that has 50 times more parameters. This is mainly due to the fire modules that are introduced in SqueezeNet [11].

Table 3.3. Training from scratch vs. deep retraining vs. shallow retraining.

Model (retraining deep)	From scratch	Deep retraining	Shallow retraining
SqueezeNet	65.40	75.80	36.39
AlexNet	68.92	74.54	30.15
Inception V3	75.19	81.98	30.04
DenseNet201	77.64	82.72	47.02
ResNet34	77.78	82.59	41.10
ResNet50	73.44	83.35	41.29
VGG13	72.85	79.10	33.37
VGG19	72.34	78.98	26.71

Comparison With the state-of-the-srt Results

In this section, we compared the obtained results with state-of-the-art works done on the same dataset (NTU RGB+D). This dataset, as mentioned earlier, is captured using three different cameras from different angles, and the actions were performed by 40 different subjects. The proposed benchmark protocol suggests two different setups. 1) **Cross-subject**: the data of 20 subjects are used for training, and the data of the other 20 subjects are used for validation. 2) **Cross-view**: the data provided by two cameras (viewpoints) is used for training, and the data provided by the third camera is used for validation. These two setups allow us to test our classifier performances to deal with large variations of action representations from different subjects and different view angles.

The results are shown in Table 3.4, where we also compare with the results obtained for skeleton-based human action recognition in the previous chapter. It can be seen that our proposed method performs better than the proposed methods in literature in both cross-subject and cross-view evaluation protocols. The obtained accuracies are 83.32% and 88.78% for cross-subject and cross-view protocols, respectively.

The superiority of performances of the proposed method is due to our novel spatio-temporal representation of skeleton data.

Table 3.4. Comparison with the state-of-the-art results.

	Cross-subject accuracy (%)	Cross-view accuracy (%)
Skeleton-based methods		
Two streams 3DCNN [113]	66.85	72.58
CNN + MTLN [112]	79.57	84.83
Trajectory maps + CNN [108]	76.32	81.08
HBRNN [23]	59.07	63.97
Deep RNN [72]	56.29	64.09
Deep LSTM [72]	60.69	67.29
PA-LSTM [72]	62.93	70.27
LieNet [120]	61.37	66.95
ST-LSTM [98]	69.20	77.70
Our Skeleton-based approach (Chapter 2)	82.07	86.54
Video-based methods		
RNN Encode-Decoder + CNN [156]	/	56.00
C3D [151]	63.5	70.3
ResNet50+LSTM [165]	71.3	80.2
Pose-Driven Attention (RGB) [165]	75.6	80.5
Pose-Driven Attention (Pose+RGB) [165]	**84.8**	**90.6**
Multi-Stream 3D CNN [164]	80.80	/
Our approach	**83.32**	**88.78**

Compared to different methods that model the temporal information separately using RNNs (or LSTMs), the use of CNNs to learn both spatial and temporal features of 2D skeleton sequences from the generated RGB images is more robust to noise and temporal variations. This is mainly due to the convolution and pooling operations, which results in better performances.

There is a very clear gap between our approach and other approaches using RGB data such as C3D [151] (+19.82% in cross-subject and +18.48% in cross-view), or Pose-Driven Attention (using only RGB) [165] (+7.72% in cross-subject and +8.28% in cross-view). By conditioning the spatial attention model to the 3D pose features, the authors of this last work [165] obtained 84.8% in cross-subject protocol and 90.6% in cross-view protocol. This shows that the use of poses improves the performances of the classifier.

In a future work, we will attempt to add a layer to our approach, taking into account the RGB frames in parallel to see how it can improve our results.

Compared to the results obtained for the 3D skeleton-based method proposed in Chapter 2, the performance is improved by 1.25% for cross-subject protocol and 2.24% for cross-view protocol. The two representations are different, but the improvement is may be due to the use of the joints' confidence scores for the transformation of our motion sequences.

3.5 Conclusion

In this chapter, we proposed a method that allows us to classify human actions from RGB videos. We propose to use a tool, namely OpenPose, to extract 2D skeletons from video sequences, then using our *Seq2Im* (Sequence To Image) method, we transform the obtained sequences into RGB image-like representations. As the obtained skeletons are in two dimensions, we use the confidence score of each joint as a third information to be added to our representation. We propose then to use pre-trained CNN models to learn features from these images and perform action classification. In order to validate our approach, we have tested it on the NTU RGB+D dataset which includes two benchmark protocols, cross-subject, and cross-view . Experimental results have shown the effectiveness of the proposed approach for human action recognition and its superiority over most of the state-of-the-art approaches. This method allows us to perform human action recognition from videos that can be captured using any camera, which makes it effective to have low-cost systems. Moreover, our method is flexible and can use any skeleton extraction tool from videos that can be developed in the future.

Chapter 4

Towards Human Interpretable Deep Learning Models for Human Action Recognition

Contents

4.1 Introduction

Over the years, machine learning (ML) and particularly deep learning (DL) has come a long way, from its existence as experimental research in an academic setting to wide industry adoption as a way of automating solutions to real-world problems. Businesses and organizations across diverse domains in the industry are building large-scale applications powered by artificial intelligence (ĄI). Nevertheless, these algorithms are still considered as alchemy due to the lack of understanding of the inner operations of these models. A few questions are raised to think about here: “Do we trust decisions made by such models?” and “How does machine learning or deep learning models make their decisions?” When a model predicts our insights, it takes certain decisions and choices. Model interpretation tries to understand and explain these decisions by the response function, i.e., the what, why, and how.

- What drives model predictions? We should have the ability to find out latent feature interactions to get an insight into which features might be important in the decision-making process. This ensures the **fairness** of the model.

- Why did the model takes a certain decision? We should be able to validate and justify why certain features were responsible for driving certain decisions during the predictions. This ensures the **accountability** and reliability of the model.

- How can we trust the model predictions? We should also be able to evaluate and validate any data point, and a model takes decisions on it. This ensure the **transparency** of the model.

A successful model interpretation method requires transparency that allows humans to question the model’s decisions and to easily understand them. This can also be referred to as human-interpretable interpretations (HII) of machine learning/deep learning models. Its simplistic definition is the extent to which a human (including non-experts in machine learning) can understand the choices taken by models in their decision-making process (the what, the how, and the why).

Existing works on deep learning-based action recognition have shown superior performances. Nevertheless, as these models are considered as black boxes, they lack of interpretation and clarity on how they work. Model interpretation algorithms can help the designers to analyze the models' behaviors in order to improve their performances by extracting insights from the classifiers. In action recognition, interpretation algorithms may help, for example, in the localization of the most important joints that a person uses during the execution of a particular action. This information can give the designers insights on how to orient their classification algorithms by selecting the appropriate features, for instance. In a domain where physical expertise is required, such as martial arts, this information can help both experts and beginners to analyze their actions and define the joints on which they should focus in order to improve their performances.

In literature, many algorithms are proposed for the interpretation of deep learning classifiers [4, 15, 166–168]. In our work, we focus on a particular algorithm called "Gradient-weighted Class Activation Mapping" (Grad-̧CAM) [16]. This method can be applied to our RGB image-like representations of skeleton sequences to get, from the models' point of view, the regions that are the most important for classification. By an inverse transformation to the 3D coordinates domain, we can obtain the most important joints at a specific temporal interval.

The main contributions of this chapter are summarized as follows: 1) We propose a novel method for debugging our CNN classifiers, which allows us to make our models more transparent, and to explain their behavior for skeleton-based human action recognition. 2) The proposed approach allows us to explain why some actions are confused by the classifier. 3) The proposed approach allows us to highlight the most important joints during the performance of a specific action.

The rest of this chapter is organized as follows: In Section 4.2, we discuss the importance of interpretability in machine learning. In Section 4.3, we present some visualization methods that are used in ML, and particularly DL, for interpretability. We follow by presenting our proposed approach and different experiments for the interpretation of CNN models in Section 4.4, and we finish by concluding our chapter in Section 4.5.

4.2 The Importance of Interpretability

If a machine learning model performs well enough, why don't we just trust it and ignore why it made certain decisions? The problem is that only one metric, like classification accuracy, is not enough to evaluate a model in most real-world tasks [169]. When it comes to predictive modeling, we have to make a trade-off : do we just want to know **what** is predicted? Or do we want to know **why** the prediction was made? In some cases, it is not important to know why the decision was made; it is enough to know that the performance on the test dataset is good. This because they are generally used in a low-risk environment, which means that a mistake will not have important consequences. However, in other cases, knowing the 'why' can help us learn more about the problem, the data, and the reason why the model might fail.

This means that for certain problems, it is not enough to get the prediction (the **what**). The model must also explain how it came to the prediction (the **why**) because a correct prediction only partially solves the original problem. Machine learning models take on real-world problems that require safety measures and testing. Imagine a self-driving car that automatically detects cyclists using deep learning models. We want to be sure that the system learns the abstraction with the smallest possible error, ideally 0%. If the detection is quite bad, using model interpretation methods can reveal why there are errors. An explanation might be that the most important learned feature is to recognize the two wheels of a bicycle, and this leads to thinking about exceptions like bicycles with side bags covering partially the wheels. Moreover, machine learning models pick up biases from the training data. This can turn these models into racists, as it has been reported many times. A relevant example that we can cite is the website "**Imagenet Roulette**" . The website's algorithm was trained to identify faces and then label them using the 2833 subcategories of people within the ImageNet's taxonomy ("adult male", "pilot", "widower", etc.). However, in many situations, the website was going racist, misogynist, and cruel. A man can be classified as "rape suspect", a person with a black skin as "negro", an Asian person as "chink", etc. The main reason for such behavior is due to the fact that the ImageNet dataset was created from images scraped from the internet, then categorized by the Amazon Mechanical Turk [170]. The dataset was very badly labeled by fallible and underpaid humans.

The prejudices and biases of these crowdsourced laborers are inevitably reflected in the AI system that they helped create. In this situation, interpretability is a useful debugging tool for detecting biases in machine learning models. In fact, the website was actually developed for this reason, to show the biases on the ImageNet dataset, and is now offline.

In summary, model interpretability gives explanations to ensure that the effects of gaps in the formalization of the problems are visible to us. [169]

4.3 Visualization Methods for Model Interpretability

As mentioned earlier, interpretability is an important task to understand and trust the decisions made by the machine learning model. Visualizing the learned features by making them explicit is one of the approaches that is widely used for this aim.

One of the biggest advantages of deep learning models is their ability to automatically learn high-level features without the need for feature engineering. The first layers learn low-level features that can be understood by a human and can be directly projected into the pixel space, such as colors, edges, etc. However, the learned features are increasingly complex and abstract the deeper we go into the network. The correlation between these features and the input image pixels is also complex. In this chapter, we will study the use of one particular method to visualize and understand the behavior of our human action classifier, namely *Grad-CAM*, which stands for "Gradient-weighted Class Activation Mapping". However, we will first introduce some visualization methods that can benefit the reader to have a global idea of the state of the art. These approaches are applied mainly to higher layers of convolutional neural networks.

4.3.1 Activation Maximization

Activation Maximization (AM) method [13] is proposed to synthesize an input image that maximizes the activation of a given hidden unit (or a neuron).

This generated image represents the input pattern that certain neurons prefer most. To synthesize such an image, each pixel of the CNN input is iteratively changed in order to maximize the activation of the neuron.

The fundamental algorithm was proposed by Erhan et al. in 2009 [13], where they visualized the preferred input patterns for the hidden layers of the Deep Belief Network [171] and the Stacked Denoising Auto-Encoder [172] learned from the MNIST digit dataset [173]. In 2013, Simonyan et al. applied this method on convolutional neural networks [166].
In order to synthesize an input image x^* that maximizes the activation a_i^l of the i^{th} neuron of the l^{th} layer, Activation Maximization approach proceeds three sequential steps:

$$x^* = argmax_x(a_i^l(x)) \tag{4.1}$$

1. Initialize the input with an image $x = x_0$ with random pixel values.

2. Compute the gradients $\frac{\delta a_i^l}{\delta x}$ using the backpropagation algorithm, while the parameters of the CNN are fixed.
3. Change each pixel of the input image iteratively following the direction of the gradient $\frac{\delta a_i^l}{\delta x}$ to maximize each activation with a step η.

$$x \leftarrow x + \eta \frac{\delta a_i^l}{\delta x} \tag{4.2}$$

Steps 2 and 3 are repeated until the generated images x no longer changes (the change is less than a certain threshold). This image x maximizes then the activation of the i_{th} neuron and represents the preferred input pattern for this neuron. Typically, a_i^l is the unnormalized activation rather than the probability returned by the SoftMax function, because the SoftMax normalizes the final layer output to values between zero and one.

In Figure 4.1, we can see some examples from the work done by Erhan et al. [13]. 36 neurons from the different layers of the Deep Belief Network, trained on the MNIST dataset, are shown. We can clearly notice the increase of complexity from the first layer (left) to the last layer (right).

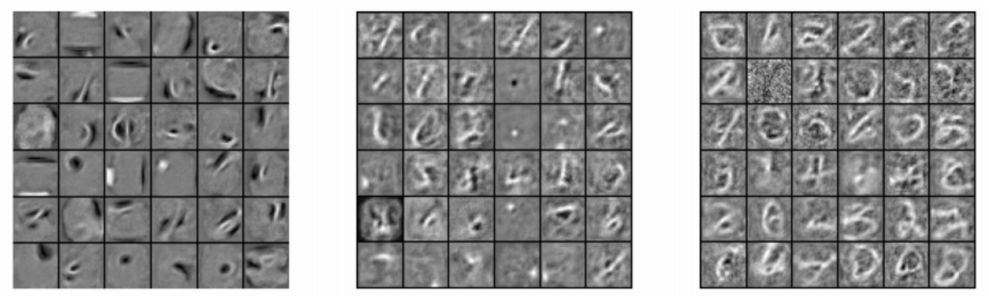


Figure 4.1. Activation Maximization Results on the MNIST Dataset. [13]

In the first layer, we can notice only rough patterns. In the second layer, we can already see some shapes that look like digits, while in the last layer, we can clearly notice the shapes of different digits. This means that the neurons in this layer have been correctly trained to recognize hand-written digits.

Modern CNNs though are more complex and can deal with input images of higher dimensions. The MNIST dataset contains only gray-scale images of 28×28 pixels while recent CNNs can have as inputs RGB images of higher sizes (For instance: Inception V3 [7] - 299×299 pixels, DenseNet [10] - 224×224, etc.). These recent CNNs are also very deep, which makes the visualized patterns in higher layers unrealistic and uninterpretable. To overcome this issue, a regularization method can be applied to improve the interpretability of the patterns. As proposed by Yosinski et al. [174], the regularization function can be applied in step 3 of the AM process:

$$x \leftarrow r_\theta\left(x + \eta\frac{\delta a_i^l}{\delta x}\right) \tag{4.3}$$

where r_θ denotes the regularization function. Different regularization methods are adopted, such as L_2 decay, which tends to penalize large values and prevent a small number of extreme pixel values from dominating the visualized patterns [166]. Another regularization technique is Gaussian Blur [174], where the high-frequency information in the visualized patterns are penalized. The contribution of a pixel is measured by setting to zero and checking how much the activation increases or decreases.

These regularization methods, in addition to others that we did not cite here, can be applied to the Activation Maximization individually or cooperatively.

4.3.2 Deconvolution

The Deconvolution method was introduced by Zeiler and Fergus in 2014 and allowed, unlike the Activation Maximization method that interprets the CNNs from the neurons perspective, to explain the CNNs from the input image perspective [4, 175]. The deconvolution method finds the selective patterns from the input image that activate a specific neuron in the convolutional layers [176]. The patterns are reconstructed by projecting the low dimension feature maps back to the image dimension. This projection process is implemented by a *DeconvNet* structure, which contains deconvolutional layers and unpooling layers, performing the inverse computation of the convolutional and pooling layers. The *DeconvNet* based visualization demonstrates a straightforward feature analysis in an image level instead of analyzing directly the interests of the neurons.

The first proposed DeconvNet structure by Zeiler et al. [175] aims to capture certain features to reconstructing the natural image by projecting a set of low-dimensional feature maps to high dimension. Later in [177], the authors used the DeconvNet structure to decompose an image in a hierarchical way. This allows to capture multi-scale information from edges (low-level) to object parts (high-level). The DeconvNet method became an effective approach to visualize the convolutional neural networks when applied to interpret the hidden layers.

Figure 4.2 shows the architecture of a DeconvNet with a forward pass on the right (a standard ConvNet) and a backward pass on the left. The results of one layer are passed to the convolution operation of the next layer to produce features maps. These feature maps are passed through a ReLU function, followed by a max-pooling layer. The max-pooling layer decreases the size of the feature maps removing in each step all the values except one (that corresponds to the maximum), which makes this operation non-reversible. To deal with this issue, Zeiler and Fergus [4] proposed to use a set of *switches* that save the positions of the maximum values during the max-pooling operation.

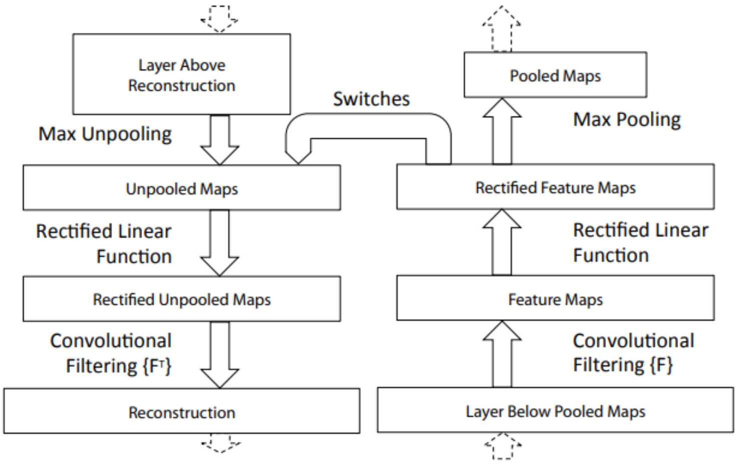


Figure 4.2. A DeconvNet layer (*right*) attached to a ConvNet layer (*left*). [4]

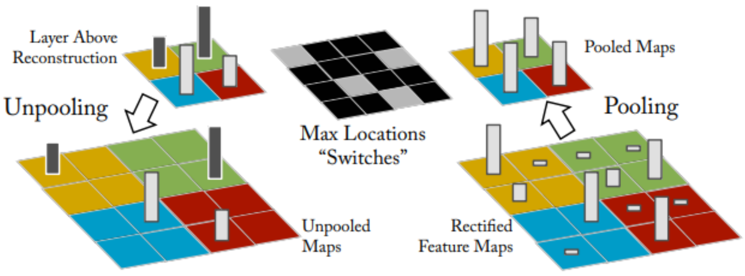


Figure 4.3. Illustration of the unpooling operation in the DeconvNet. The unpooling uses *switches* that record the local max in each pooling region during pooling in the ConvNet. [4]

These switches are used in the backward pass of the DeconvNet during the unpooling layer by restoring the maximum values in their saved positions and filling the rest of the values with zeros, as illustrated in Figure 4.3.

Finally, the "unpooled" maps are passed through a ReLU to obtain the final reconstruction.

Figure 4.4 shows Randomly chosen features from several convolutional layers obtained by the Deconvolution method during training. In the first layer, we can see basic features like lines, gradients, etc.

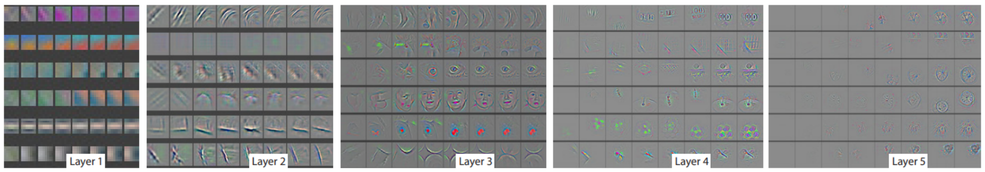


Figure 4.4. Evolution of a randomly chosen subset of model features through training. [4]

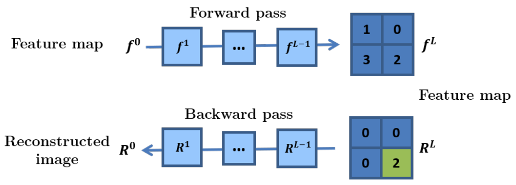


Figure 4.5. Illustration of the forward and backward passes. [14]

The features become more and more complex in higher layers and can have objects like eyes, faces, car wheels, etc.

4.3.3 Guided Backpropagation

Guided Backpropagation was introduced in 2014 by Springenberg et al. [14]. This is a modified Deconvolution approach that can be applied to a wider range of neural network structures. Moreover, it provides more accurate reconstructions of features, especially from higher layers. The authors proposed to replace the max-pooling layer with a convolutional layer with a bigger stride without losing accuracy.

To reconstruct a feature from a higher layer using DeconvNet or Guided Backpropagation, the input images are passed to the network up to the given layer. Only one value of the desired neuron from the obtained feature map is left non-zero and then passed backwards through the network to obtain a reconstruction showing the part of the image that is strongly activating this particular neuron.

In fact, the "deconvolution" in a DeconvNet is similar to the backpropagation approach, and they mainly differ in the way they handle the ReLU nonlinearity [166].

In the backpropagation algorithm, the gradients are computed based on the bottom input data, i.e., the values that are set to zero correspond to negative values positions in the forward pass (Figure 4.6, second row).

If we denote the gradient of feature i in the layer l as R_i^l then:

$$R_i^l = \sigma(z_i^l).R_i^{l+1} \tag{4.4}$$

where $\sigma(z_i^l)$ is the forward ReLU function.

In DeconvNet (Section 4.3.2), the ReLU nonlinearity is applied in the backward pass using the top gradients for the backpropagation. This zeros the negative values in the top gradients (4.6, third row), where the resulting gradient is computed as follows:

$$R_i^l = \sigma(R_i^{l+1}).R_i^{l+1} \tag{4.5}$$

The Guided Backpropagation combines the standard Backpropagation algorithm with the Deconvolution approach by using both bottom input and top gradients to compute the propagation through the ReLU nonlinearity in the backward pass (Figure 4.6, second row).

The gradient is therefore computed as follows:

$$R_i^l = \sigma(z_i^l).\sigma(R_i^{l+1}).R_i^{l+1} \tag{4.6}$$

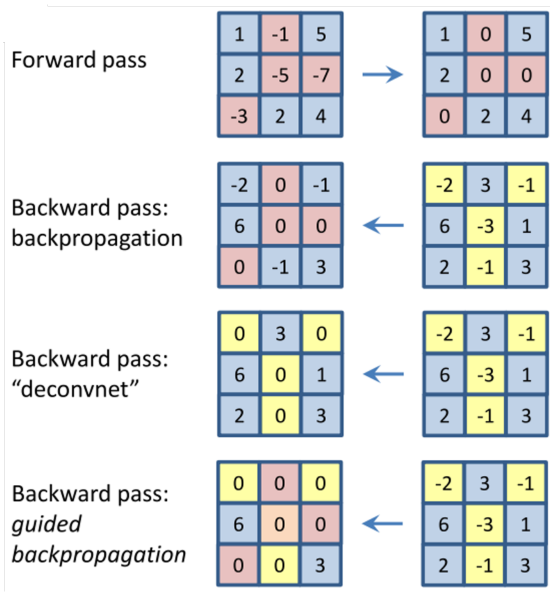


Figure 4.6. Comparison of different methods of backward passes through a ReLU nonlinearity. [14]

4.3.4 Class Activation Maps (CAM)

Class Activation Maps (CAM) method was introduced by Zhou et al. [15]. It allows producing heatmaps that indicate the discriminative regions in the input images that motivated the decision of the network for a given class. CAM can be obtained from fully-convolutional neural networks, in addition to a Global Average Pooling (GAP) layer at the end of the network. The GAP outputs the spatial average of the feature map of each unit at the last convolutional layer, as illustrated in Figure 4.7.

A weighted sum of these values is used to generate the final output. Similarly, a weighted sum of the feature maps of the last convolutional layer is computed to obtain the class activation maps.

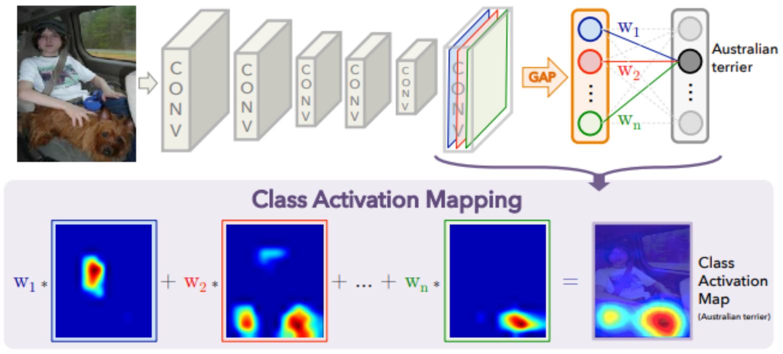


Figure 4.7. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions. [15]

4.3.5 Gradient-Weighted Class Activation Mapping (Grad-CAM)

Gradient-weighted Class Activation Mapping (Grad-CAM) is an extension of the CAM method that allows to avoid the limitation of using only fully-convolutional networks. Grad-CAM was proposed by Selvaraju et al. [16] who modified the CAM method to be applied to a wider range of networks, including common convolutional neural networks with fully-connected layers at the end. It uses the gradients of any target to produce a coarse localization map highlighting the discriminative regions in the input image for the prediction of this target.

Grad-CAM can be applied to any task-specific network (image classification, image captioning, visual questioning-answering, etc.). For the image classification task, in order to obtain the class-discriminative localization map Grad-CAM L^c for any class c as shown in Figure 4.8, we need first to compute the gradient of the score for this class, y^c (before the Softmax) with respect to the feature maps A^k of a convolutional layer, i.e., $\frac{\delta y^c}{\delta A^k}$.

From these gradients we compute the global average pooling (GAP) layer to obtain the neuron importance weights α_k^c:

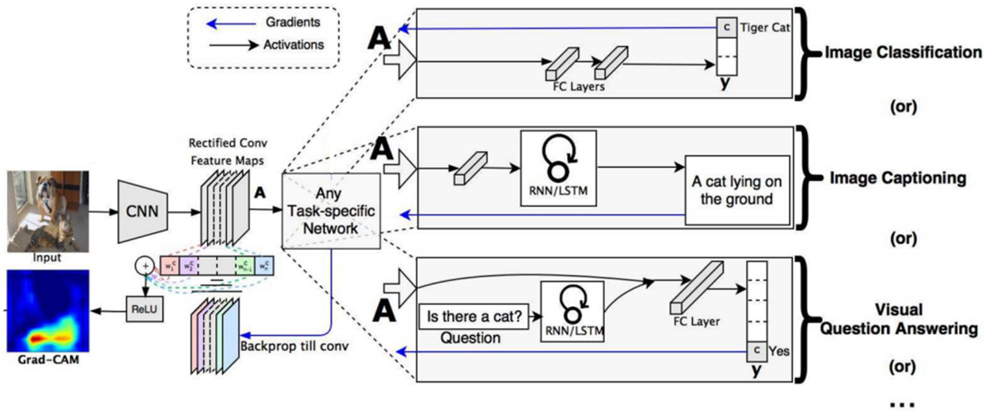


Figure 4.8. Illustration of the Grad-CAM visualization method. [16]

$$\alpha_k^c = \frac{1}{Z}\sum_i \sum_j \frac{\delta y^c}{\delta A_{ij}^k} \tag{4.7}$$

A weighted combination of forward activation maps is then performed, followed by a ReLU to obtain only the features that have a positive influence on the class of interest as follows:

$$L^c_{Grad-CAM} = ReLU(\sum_k \alpha_k^c A^k) \tag{4.8}$$

Negative pixels are likely to belong to other categories in the image. This results in a coarse heatmap of the same size as the convolutional feature maps that can be resized the fit the size of the input image.

4.4 Proposed Approach of Interpretation of CNN Models Built for Skeleton-Based Human Action Recognition

In the previous chapters, skeleton sequences were transformed into RGB images, and then CNNs we used for deep feature extraction. In this section, we use a visualization method in order to extract insights and understand the behavior of our deep classifier. More specifically, the Grad-CAM visualization method (Section 4.3.5) is used in order to generate a heatmap on the image that corresponds to the zones that are the most salient to the CNN classifier. Moreover, we can benefit from the transparency provided by such algorithms to understand the classification process. With an inverse mapping of the generated heatmap into the skeleton sequence, we can highlight the joints that are used for the execution of a specific action.

In this chapter, we will only focus on the 3D skeleton-based action recognition, which is related to Chapter 2. To the best of our knowledge, visual interpretability methods have not been used before to deal with skeleton-based human action recognition.

4.4.1 Proposed Approach

Saliency (or attention) maps generation using Grad-CAM is an analytic method that allows the estimation of the importance of each pixel of an image, using only a forward, then a backward pass through the network [16]. The intuition behind this method is that, if one pixel is important in respect to the node corresponding to the ground truth y, then changing the value of this pixel leads to a big change in that node. In other words, the higher the absolute value of the gradient in a pixel, the more important this pixel is. Grad-CAM uses the gradients of any target, flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting this target. Unlike other approaches, it is applicable in a wide variety of CNN models without architectural changes or retraining.

Figure 4.9 shows an overview of the Grad-CAM approach for the extraction of a visual explanation from the human action recognition process.

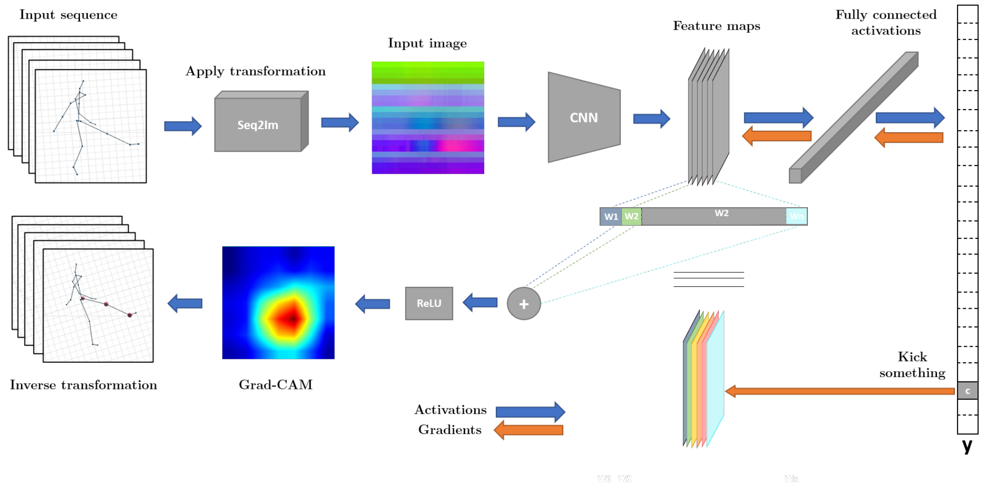


Figure 4.9. Overview of Grad-CAM visualization algorithm for human action recognition.

Given an input skeleton sequence and a target class (for example, ’kicking something’) as input, we first transform the sequence into a RGB image using the *Seq2Im* approach. Then, the generated image is forward propagated through the CNN layers of our model, followed by the fully-connected layer to obtain a raw score of the category. All the gradients are set to zero for all classes except the target class (’kicking something’), which is set to 1. This information is then backpropagated to the rectified convolutional feature maps, which are combined to compute a coarse localization (heatmap). The blue values in the generated heatmap correspond to pixels with low impact (values near to zero) where the red regions correspond to pixels with high impact (values near to 1). By applying a *Seq2Im* (Section 2.3.1) inverse transformation of these heatmaps, we can obtain values for each skeleton joint at each frame. The higher the value, the more important the joint is at a specific frame during the performance of the action.

Figure 4.10 shows Grad-CAM saliency maps visualizations for ’throw’, ’clap’, ’cheer up’ and ’kick something’ actions from the NTU RGB+D dataset. The models DenseNet and SqueezeNet (Section 1.4) have been tested for both protocols, cross-subject, and cross-view. The first row in Figure 4.10 corresponds to examples of original images for four actions.

The next two rows correspond to Grad-CAM heatmaps in cross-subject and cross-view configurations using DenseNet201. The fourth row corresponds to the mapping of the obtained heatmaps on original mocap sequences, and the next three rows correspond to the same experiences using the SqueezeNet model. We can notice that for each action, specific zones are highlighted by the models corresponding to specific patterns in the images. These zones correspond to specific joints, as shown in the fourth and the last rows. For an intuitive understanding of these zones, each joint in the skeleton is represented with a red sphere where the radius is relative to the generated values from the mapping of the heatmaps to the original mocap sequences.

We can notice that there is not a big change in the heatmaps from cross-subject and cross-view configurations. However, different models give different results. SqueezeNet model heatmaps have narrower and localized results, while DenseNet ones are wider.

By analyzing the mapping on the original sequences, we can notice that the SqueezeNet model focuses only on a few joints to make the classification. Furthermore, it shows that the decision is taken by only focusing at the beginning of the execution of the action. Conversely, DenseNet models tend to make a decision on a bigger window (longer sequence) and use more joints for classification. The classification results in the previous chapters show that DenseNet gives higher accuracy compared to SqueezeNet, and the information used by both models, analyzed by the Grad-CAM approach, may explain the reason behind that. We conclude that for each action sequence, there is only a set of informative joints, which are the most important as they contribute to action analysis and classification, while the other joints are irrelevant to this action.

Furthermore, as we take into account that the columns in the generated images using *Seq2Im* correspond to time (or frames), we notice that the highlighted zones start at a specific frame and end at a specific frame. Analyzing the corresponding skeleton sequence, we can clearly see that the interval of activation corresponds to the time where the person concretely executes the action, especially for the DenseNet model. This allows us to generalize our proposed approach of human action classification for online action segmentation, which needs to be confirmed in future works. One way to do this is by using a sliding window of a few frames and apply the classification process defined in the previous sections.

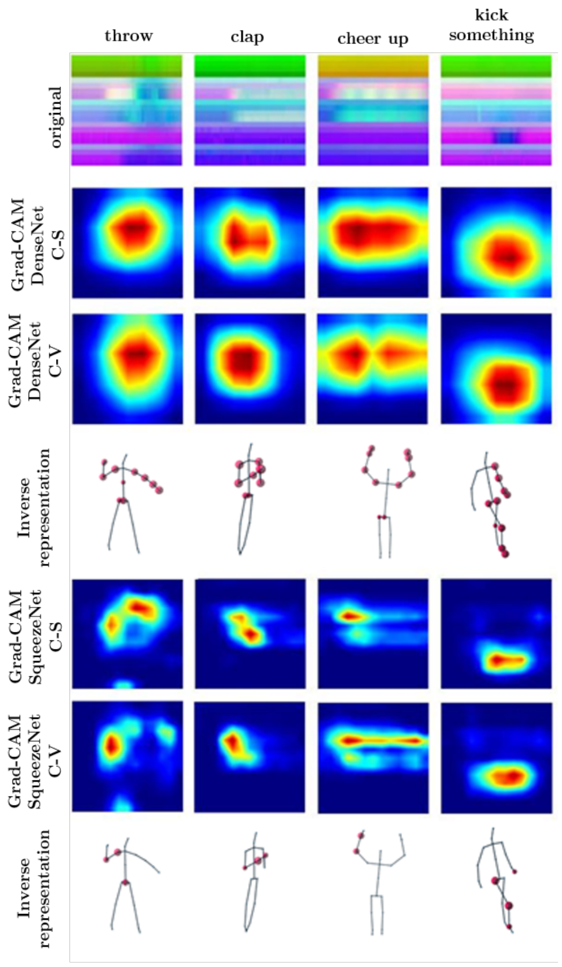


Figure 4.10. Examples of Grad-CAM visualizations of four different actions (throw, clap, cheer up and kick something). 1st row: original images generated using *Seq2Im*. 2nd and 3rd rows: generated heatmaps using DenseNet201 following the cross-subject and the cross-view protocols consecutively. 5th and 6th rows: generated heatmaps using SqueezeNet following the cross-subject and the cross-view protocols consecutively. 4th and 7th rows: mapping of the obtained heatmaps on the original skeleton sequences.

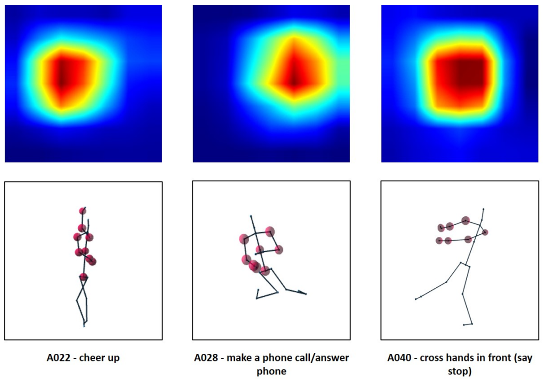

A022 - cheer up
A028 - make a phone call/answer phone
A040 - cross hands in front (say stop)

Figure 4.11. Examples of situations where the model correctly performs classification even when the data is noisy. 1st row: generated heatmaps using Grad-CAM. 2nd row: mapping of the obtained heatmaps on the original skeleton sequences.

Another interesting thing to notice, and the assumption was made in Chapter 2 is that the used CNN models can give high accuracy scores independently from the quality of the data. Figure 4.11 shows examples of three different actions where they present noises, particularly on the legs. Grad-CAM only highlights the zones that are related to the upper body joints (which are the most important joints, from the human perspective, that are responsible for the execution of the related actions). This proves that the CNN models learn patterns from the data even if an important noise is present.

4.4.2 Analysis of the miss-classifications

The proposed approach in Chapter 2 has achieved an accuracy on the NTU RGB+D dataset of 82.00% for cross-subject evaluation and 86.54% for cross-view evaluation using the model DenseNet201. The confusion matrix obtained for the cross-subject evaluation is shown in Figure 4.12. It shows that the model provides overall a good classification performance for different action classes.

Table 4.1 shows the five best recognized actions and the five most confused pairs of actions using DenseNet201 model.

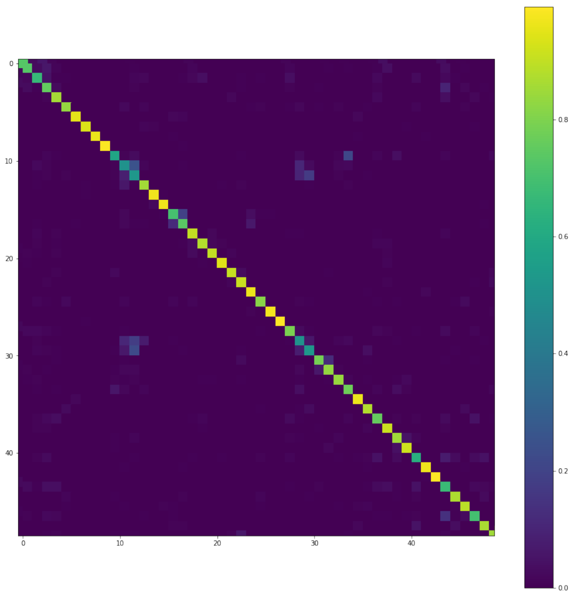


Figure 4.12. Confusion matrix of test results on the NTU RGB+D dataset (cross-subject evaluation).

We can notice that the model is not perfect and confuses multiple actions. The first four actions are generally confused with each others and are very similar, like *reading* (mostly confused with '*writing*'), '*writing*' (with '*typing on keyboard*'), '*playing with phone/tablet*' (with '*writing*'). The fifth action ('*clapping*') is also confused with a very similar action that is '*rubbing two hands together*'.

Figure 4.13 and Figure 4.14 show the heatmaps generated by Grad-CAM method for the ten actions mentioned in Table 4.1. Figure 4.13 presents three columns corresponding, from left to right, to the original images generated from the mocap sequences, the generated heatmaps, and the highlighted joints by mapping the heatmaps on the original skeletons sequences. As mentioned in the previous section, the model takes into account only a set of joints to make the decision. For the first action 'jumping up', all the joints are considered as important except the head.

Table 4.1. Top 5 accurate actions and 5 confused pairs for the proposed model, including the recognition accuracy per class.

Top 5 recognized actions		Top 5 confused actions	
1) A027: Jumping Up	99.26%	1) A011: Reading → A012: Writing (23.75%)	50.19%
2) A043: Falling	99.25% %	2) A029: Playing with phone/tablet → A012: Writing (16.36%)	50.93%
3) A009: Standing Up	98.53	3) A012: Writing → A030: Typing on keyboard (18.22%)	52.63%
4) A026: Hopping	97.43%	4) A030: Typing on keyboard → A012: Writing (20.99%)	53.05%
5) A014: Wearing jacket	96.99%	5) A010: Clapping → A034: Rubbing two hands together (20.82%)	58.36%
Average accuracy = 98.29%		Average accuracy = 52.40%	

Taking into account the time, we can notice from the generated pattern that the arms are first highlighted, then the legs. This is a natural behavior when jumping as a person naturally moves first the arms up before jumping for stability. We can also notice that the model considers the left arm and the right leg more important than the other limbs for this specific sequence.

For the action ’falling’, the model considers the lower body as more important than the upper body. For this specific sequence, the left arm is also important, as in the original sequence, the person tries to use his left arm to find support on the ground. Contrariwise, the model focuses on the upper body joints to classify the action ’standing up’. For the action ’hopping’, the model takes into account almost all of the joints except the leg with the bent knee. Also, the pattern of this specific sequence shows that the model takes into account the right leg joints only at the beginning. Finally, the action ’wearing jacket’, and as expected, the most important joints are those related to the arms. We can notice that left arms joints are more important than the right arms joints, as the person starts by wearing the jacket from the left arm then the right.

These examples demonstrate that the CNN models do not take into account all the joints to make the decision, and only a few joints contribute to action classification, from the CNN viewpoint. They also show how CNNs behave for different actions.

Figure 4.14, on the other hand, visualizes the top five confused pairs of actions. The first and the second columns show the heatmaps combined with original images and corresponding visualization on the skeleton sequence consecutively for correctly classified actions.

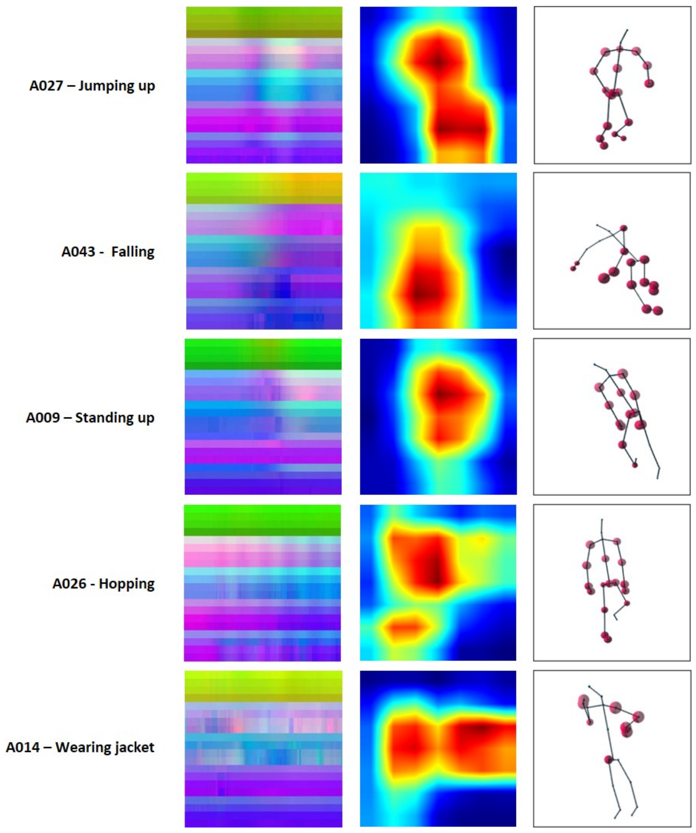


Figure 4.13. Attention visualization of the top 5 recognized actions using Grad-CAM. 1st column: original images. 2nd column: generated heatmaps. 3rd column: one frame with the highlighted joints by mapping the heatmaps on the original skeleton sequences.

The third and the fourth columns, on the other hand, show the same visualizations but with confused actions. The first row, for example, shows visualizations of the action 'reading'. The first heatmap corresponds to a correctly classified sequence, whereas the second one corresponds to a sequence that is confused with the action 'writing'. 23.75% of 'reading' sequences are confused with the action 'writing' (Table 4.1). These two actions are very similar, especially when we do not take into account the objects the person interacts with.

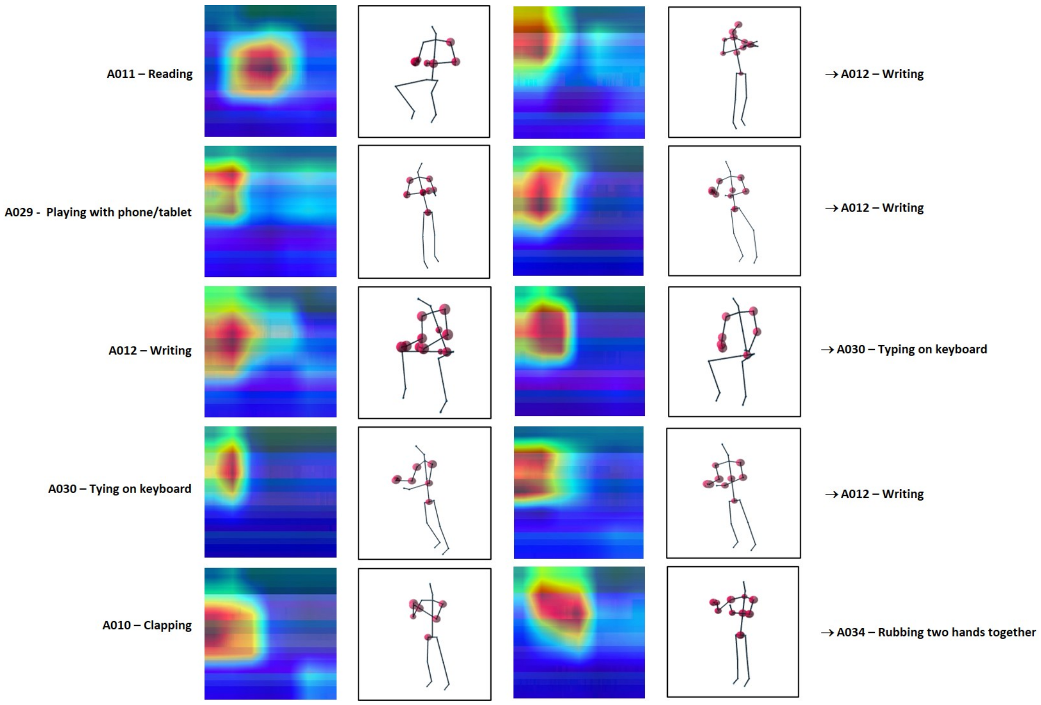


Figure 4.14. Attention visualization of the top 5 confused pairs of gestures using Grad-CAM. 1st column: heatmaps applied on images corresponding to correctly classified skeleton sequences. 2nd column: one frame with the highlighted joints using the inverse transformation of the heatmaps on the original skeleton sequences. 3rd column: heatmaps applied on images corresponding to miss-classified skeleton sequences. 4th column: one frame with the highlighted joints using the inverse transformation of the heatmaps on the original skeleton sequences.

Moreover, the same joints (upper body) are used for both actions, as confirmed by the Grad-CAM visualization. We can notice that for the confused action, the joints corresponding to the head and neck are also highlighted, and the detection is made earlier. By analyzing the video sequences, we can notice that for the correctly classified ones, the persons do not move a lot, while for the miss-classified ones, the person makes small additional movements, like taking the notebook or turning the page and bowing the head. These small movements confuse the model, which "thinks" that the person is writing.

In fact, in a lot of sequences of the class ’writing’, the person moves the hand toward the notebook in order to write. A similar pattern can be found in the third row of Figure 4.14. The same conclusions can be made for the second and the fourth actions ’playing with phone/tablet’ and ’typing on keyboard’ as they are confused with ’writing’ for the same reasons. The action ’writing’ is generally confused with ’typing on the keyboard’, and the main reason is that the person moves his hand with the pen toward the notebook, which is confused with the person moving his hand(s) toward the keyboard to type. The last example corresponds to the action ’clapping’ that is confused in many cases with the action ’rubbing two hands together’.

Logically, these two actions have very good chances to be confused as they have the same process of moving the arm in front and then bring them close to each other. The DenseNet model can classify though both actions with 58.36% and 77.27% of accuracy consecutively. By analyzing the highlighted zones in the confused sequences, we notice two main movements that are done just before and can be the reason of the miss-classification: 1) the person moves a little bit forward before/while clapping the hands, 2) the person changes the frequency of clapping. These two movements can represent biases that make the classification failing.

4.4.3 Evaluation

In this section, we attempt to give a numerical evaluation of our visualizations, to see if the zones highlighted by the CNN model correspond to the human viewpoint. As mentioned earlier, and to the best of our knowledge, visualization methods have never been used before for the interpretation of CNN models for action classification. This makes it complicated to objectively evaluate our method. Instead, we make a subjective evaluation by asking people to annotate manually what are, in their opinion, the most important joints that are responsible for the execution of an action. Annotators have also been asked to specify the beginning and the end time for every annotated joint.

Figure 4.15 shows the interface of the annotation tool that has been developed on top of the MotionMachine framework [178]. The annotators can move the cursor using the mouse and click to select the beginning of the selected sequence and then move and click again to select the end time.

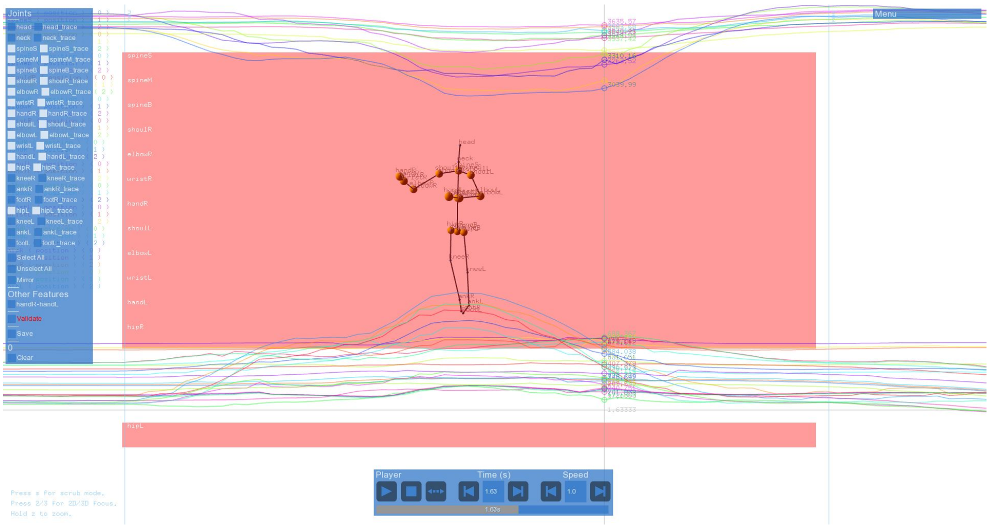


Figure 4.15. Interface of the manual annotation tool for the selection of the most important joints.

In the left panel on the interface, the annotators can select the most important joints, and also the corresponding X, Y and Z trajectories if necessary. In summary, 980 random sequences have been manually annotated (20 annotations by class) by three people. Due to the limited time, we only use these annotated sequences, and we will make more annotations by multiple other people in future works.

From these annotations, we generate black and white images, resized to have the same size as the original images, where black corresponds to 0 (not important), and white corresponds to 1 (important). To proceed to our evaluation, we convert our heatmaps into grayscale images, and then we filter out the less important regions (threshold = 100, empirically chosen).

Figure 4.16 shows an example of heatmaps and their processing for a sequence of the class 'cheer up'. *a)* and *c)* are the heatmaps generated by Grad-CAM from the models DenseNet201 and SqueezeNet. *b)* and *d)* are the corresponding filtered images. *e)* is the manual annotation (ground truth), and f is the same as e excluding the joints corresponding to the head and the spine.

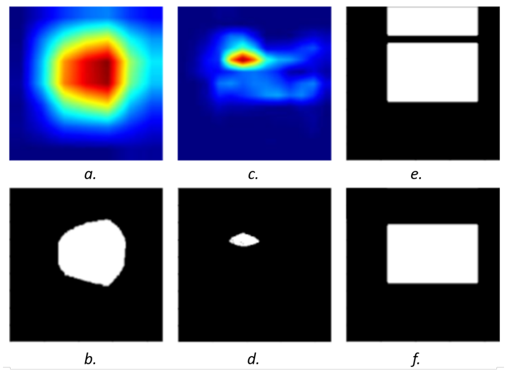


Figure 4.16. Examples showing the generated images for the evaluation of heatmaps (action: 'cheer up'). *a*) generated heatmap by the model DenseNet201. *b*) corresponding filtered image. *c*) generated heatmap by the model SqueezeNet. *d*) corresponding filtered image. *e*) ground truth annotation. *f*) ground truth annotation excluding the head and spine joints.

We choose four metrics to evaluate our results: precision, recall, $F1$ score and the intersection-over-union (IOU), that are defined by the following equations:

$$precision = \frac{TP}{TP + FP} \tag{4.9}$$

$$recall = \frac{TP}{TP + FN} \tag{4.10}$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.11}$$

$$IOU = \frac{TP}{TP + FP + FN} \tag{4.12}$$

where, TP, FP, and FN denote the true positive, false positive, and false negative counts (pixel-wise), respectively.

We compute different metrics for both cross-subject and cross-view protocols of the NTU RGB+D dataset.

Furthermore, while analyzing the obtained results and manual annotations, we noticed that people (annotators) generally consider the head and the spine joints as important for many actions (such as 'making a phone call', 'taking a selfie', some sequences of 'cheering up', etc.).

The obtained heatmaps, on the other hand, show that the CNN models generally focus on arms and legs joints and do not consider head joints as important except in a few situations. For this reason, we decided to add another set of tests where we exclude the head and spine joints from the ground truth annotations (Figure 4.16, *f.*).

Figure 4.17 shows the results obtained for different tests. In the first set of tests, we take into account all joints from the manual annotations (blue bars). We notice that we generally have high precision and low recall, and we have almost no difference between cross-subject and cross-view configurations. Using DenseNet201, we obtain a $F1$ score of 0.57 and an IOU of 0.37, which are considered relatively high for a pixel-to-pixel correspondence between automatic and manual annotation from images. SqueezeNet obviously has very high precision, but very low recall, and by consequence, low $F1$ score and IOU because, as explained earlier, it makes generalization on small observations.

By removing the head and the spine joints from manual annotations and computing different metrics, we obtain the results shown in orange bars in Figure 4.17.

We notice a small drop in precision but a higher recall in all configurations, and by consequence, a higher $F1$ score and IOU. SqueezeNet results are still very low while we get an increase of $F1$ score and IOU of 7% using DenseNet201, for both cross-subject and cross-view configurations. These scores motivate the investigation of the extension of the proposed approach of skeleton-based action classification for online use, as the CNN models can focus on the most informative joints.

4.5 Conclusion

In this chapter, we addressed the problem of the "black-boxiness" of the CNN models and how to use interpretability methods to explain the behavior of these models for skeleton-based human action recognition.

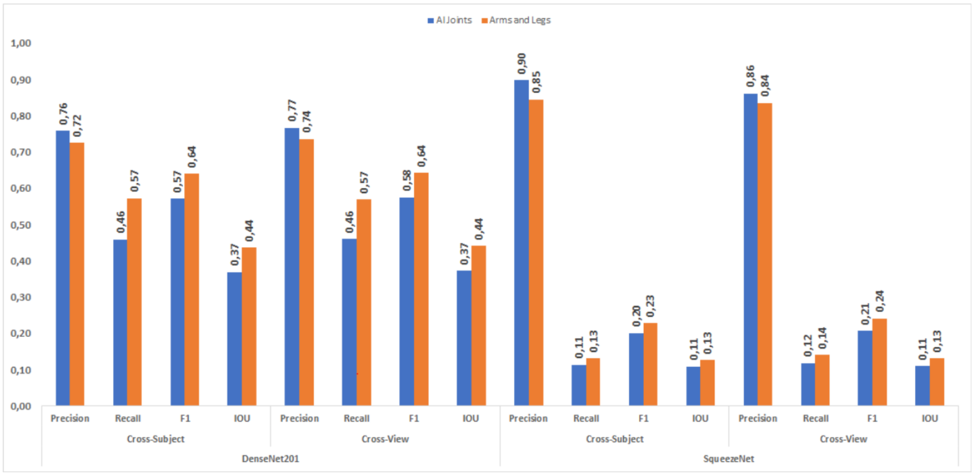


Figure 4.17. Results obtained for the evaluation of heatmaps generated for both DenseNet201 and SqueezeNet models, with and without the head and spine joints.

Investigations of the state of the art show that there is no similar work that addressed the same subject. After recalling the importance of interpretability and visualization methods to improve the transparency of deep learning models, we proposed an approach that helps to understand the decisions made by our models. This approach consists of using a saliency map visualization technique, namely *Gradient-weighted Class Activation Mapping (Grad-CAM)*, for the estimation of the most important joints, from the CNN point of view, that are responsible for the decisions made. Grad-CAM is a technique used for visual explanations of CNN models by making them more transparent. It allows us to visualize, using heatmaps, the regions of the input images that are "important" for predictions. We map the heatmaps generated by the Grad-CAM on the original skeleton sequences to visualize the most informative joints. This mapping allows us to give a score for each joint of the skeleton and on every frame. The higher the score, the more important the joint is at a specific frame. By analyzing these results, we found some biases that were responsible for the miss-classification of some actions.

Moreover, we proposed an evaluation method to compare the generated heatmaps from the subjective perspective of the human.

We asked people to manually annotate a list of sequences where they had to specify, in their opinion, the most important joints that are responsible for the execution of the action, in addition to the time interval. Comparing these annotations to the generated heatmaps, especially those corresponding to the DenseNet201 model, we obtained interesting results. We obtained almost 80% of precision and 50% of recall, in addition to around 40% of IOU (Intersection Over Union) score, which are considered as relatively good results as we make the comparison in a pixel-wise manner. Furthermore, as the CNN model focuses on specific joints at a certain time interval and ignores the other joints, it motivates the investigation of extending our approach of skeleton-based human action recognition to real-time. This is one of the axis that will be investigated in future works.

To conclude, this proposed approach allowed us to partially answer the three interpretability questions: 1) ”What” drives model predictions? We found that the model focuses on specific regions and ignores others in order to make the decision. 2) ”Why” did the model takes a certain decision? We could understand why the model correctly classifies some sequences and why it fails in some cases. 3) How can we trust the model predictions? Using the Grad-CAM method, we could debug our model and get some insights into its behavior.

Conclusion

This dissertation has presented a number of novel research in human action recognition. In each of the topics that were addresses, significant contributions materialized by international publications have been brought to the state-of-the-art, or being under submission. In this chapter, we summarize the main contribution and future research directions.

4.6 Contributions

This thesis focuses particularly on human action recognition from skeleton sequences. Skeleton sequences provide 3D coordinates of human skeletal joints. Compared to RGB videos, skeleton sequences are robust to illumination changes and clustered backgrounds. Early approaches of skeleton-based human action recognition are mainly based on hand-crafted features, which requires a lot of processing and feature engineering. Other approaches focus on recurrent neural networks. Following these categories, it is difficult to capture the complex spatial structure and long-term temporal information of skeleton sequences. In this thesis, we present a series of approaches to improve action recognition.

- We present a new representation of 3D skeleton sequences called "Sequence to Image" (*Seq2Im*) that allows to generate RGB images that model both of the spatial and temporal information. This representation allows to leverage the state-of-the-art image classification models, and particularly Convolutional Neural Networks (CNNs), to extract deep features and model human actions using the process of transfer learning. The proposed approach showed that fine-tuning pre-trained models for our task of action recognition provides high accuracy scores even if the original task is very different. Extensive experiments on multiple benchmark datasets showed the superiority of our approach over other literature methods despite its simplicity. (Chapter 2)

- As human pose detection from single images got a lot of advances due mainly to the development of deep learning and compute power, we propose to extended our approach into the RGB videos domain. We propose to extract 2D skeletons from video sequences, and then pre-process this data to be adapted to our *Seq2Im* method. Following the same steps as in 3D skeleton action recognition, we train CNN models using transfer learning process on the generated image dataset. The obtained results on a challenging benchmark dataset showed the superiority of our approach over other video-based action recognition methods, and even 3D skeleton-based approaches on the same dataset. (Chapter 3)
- We present a new method to interpret the behaviour of CNN models used for skeleton-based human action recognition. Deep learning models, and especially CNNs are considered generally as black boxes where they lack in understanding of their inner operations. The proposed approach allows to highlight the most important regions in the generated image (using *Seq2Im* method) that influence the decision of the classifier. By applying an inverse transformation and mapping to the original skeleton sequences, we can obtain the most important joints that are used during the performance of a specific action. This method allowed us to conclude that the CNN models focus only on specific patterns on the input images and ignores the rest of the pixels. These spatio-temporal pattern contains information of a few informative joints at a specific time interval. (Chapter 4)

4.7 Future Works

This thesis focused on action recognition using 3D skeleton sequences. The proposed approach consists of transforming the skeleton sequences into RGB images and use state-of-the-art pre-trained models in order to perform the classification using transfer learning technique. The generated images are resized to fit the input of the desired architecture. One of the first works to do in future is to study the effect of the resizing on the model performance. We will also try different resizing methods to obtain the best results. Moreover, we will try different configurations to transform our skeleton sequences into images.

Instead of using only three coordinates, including the joint orientations may improve the performances. Also, instead of using the red, green and blue channels, we will test other color representations (ex: HSV, or HSB for hue, saturation, and value, or brightness) to transform our sequences.

For some actions, and particularly those involving interactions with objects, RGB videos provide more useful information, which can be explored to recognize the action more accurately. Future research directions could be to design models for action recognition based on both RGB videos and skeleton sequences. This includes how to learn features from RGB videos and how to combine the information of the RGB and skeleton sequences for better action recognition.

In Chapter 4, we showed that the CNN models focus on only few joints at a specific time interval in order to make a decision. This means that, for a longer sequence, the CNN model can still ignore non-important information and focuses on meaningful visual patterns. In the future, we will exploit this valuable information to extend our proposed approach to real-time. One possible way would be to use a sliding time window and apply the proposed approach on each obtain sub-sequence, then merge the obtained decisions. We may also exploit 3D CNNs on the generated clips to take into account the relationship between successive *Seq2Im* frames.

Appendix A

Publications related to this thesis

A.1 Journals with Peer Review

1. **S. LARABA**, T. DUTOIT. "*Towards Human Interpretable Deep Learning Models for Human Action Recognition*". [to be submitted]

2. A. GRAMMATIKOPOULOU, **S. LARABA**, O. SAHBENDEROGLU, K. DIMITROPOULOS, S. DOUKA, N GRAMMALIDIS. "*An adaptive framework for the creation of exergames for intangible cultural heritage (ICH) education*". Journal of Computers in Education. (2019)

3. **S. LARABA**, M. BRAHIMI, J. TILMANNE, T. DUTOIT. "*3D skeleton-based action recognition by representing motion capture sequences as 2D-RGB images*". Computer Animation and Virtual Worlds 28 (3-4), e1782. (2017)

4. **S. LARABA**, J. TILMANNE. "*Dance performance evaluation using hidden Markov models*". Computer Animation and Virtual Worlds 27 (3-4), 321-329. (2016)

A.2 Journals without Peer Review

1. M. Tits, **S. LARABA**, E. CAULIER, J. TILMANNE, T. DUTOIT. "*UMONS-TAICHI: A multimodal motion capture dataset of expertise in Taijiquan gestures*". Data in brief journal. (2018)

A.3 Papers in International Conference with Peer Review

1. **S. LARABA**, J. TILMANNE and T. DUTOIT. "*Leveraging Pre-trained CNN Models for Skeleton-Based Action Recognition*". International Conference on Computer Vision Systems, 1, 612-626. (2019)
2. S. AUBRY, **S. LARABA**, J. TILMANNE, T. DUTOIT. "*Action recognition based on 2D skeletons extracted from RGB videos*". MATEC Web of Conferences 277, 02034. (2019)
3. M. MANCAS, **S. LARABA**, A. BANDRABUR, P-H. DE DEKEN, K. HAGIHARA, N. LEBLANC, SB. Y. TASDEMIR, B. MACQ, T. DUTOIT. "*People Groups Analysis for AR Applications*". International Conference on 3D Immersion (IC3D). (2018)
4. A. GRAMMATIKOPOULOU, S. LARABA, O. SAHBENDEROGLU, K. DIMITROPOULOS, N. GRAMMALIDIS. "*An adaptive framework for the creation of bodymotion-based games*". International Conference on Virtual Worlds and Games for Serious Applications (VS-Games). (2017)
5. **S. LARABA**, J. TILMANNE and T. DUTOIT. "*Adaptation procedure for HMM-based sensor-dependent gesture recognition*". ACM SIGGRAPH Conference on Motion in Games. (2015)

Bibliography

[1] H. Fujiyoshi, A. J. Lipton, and T. Kanade, "Real-time human motion analysis by image skeletonization", *IEICE TRANSACTIONS on Information and Systems*, vol. 87, no. 1, pp. 113–120, 2004.

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

[3] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A guide to convolutional neural networks for computer vision", *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, pp. 1–207, 2018.

[4] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks", in *European conference on computer vision.* Springer, 2014, pp. 818–833.

[5] "Momentum and learning rate adaptation", https://www.willamette.edu/~gorr/classes/cs449/momrate.html, accessed: 2020-07-13.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, *arXiv preprint arXiv:1409.1556*, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[11] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size”, *arXiv preprint arXiv:1602.07360*, 2016.

[12] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[13] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, “Visualizing higher-layer features of a deep network”, *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[14] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net”, *arXiv preprint arXiv:1412.6806*, 2014.

[15] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.

[16] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra *et al.*, “Grad-cam: Visual explanations from deep networks via gradient-based localization.” in *ICCV*, 2017, pp. 618–626.

[17] J. Liu, A. Shahroudy, M. L. Perez, G. Wang, L.-Y. Duan, and A. K. Chichung, “Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding”, *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[18] S. A. Papert, “The summer vision project”, 1966.

[19] J. Wang, Z. Liu, Y. Wu, and J. Yuan, “Mining actionlet ensemble for action recognition with depth cameras”, in *2012 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2012, pp. 1290–1297.

[20] G. Johansson, “Visual motion perception”, *Scientific American*, vol. 232, no. 6, pp. 76–89, 1975.

[21] J. Ben-Arie, Z. Wang, P. Pandit, and S. Rajaram, “Human activity recognition using multidimensional indexing”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1091–1104, 2002.

[22] A. Yao, J. Gall, G. Fanelli, and L. Van Gool, “Does human action recognition benefit from pose estimation?” in *BMVC 2011-Proceedings of the British Machine Vision Conference 2011*, 2011.

[23] Y. Du, W. Wang, and L. Wang, “Hierarchical recurrent neural network for skeleton based action recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1110–1118.

[24] M. Brahimi, M. Arsenovic, S. Laraba, S. Sladojevic, K. Boukhalfa, and A. Moussaoui, “Deep learning for plant diseases: detection and saliency map visualisation”, in *Human and machine learning.* Springer, 2018, pp. 93–117.

[25] S. Laraba and J. Tilmanne, “Dance performance evaluation using hidden markov models”, *Computer Animation and Virtual Worlds*, vol. 27, no. 3-4, pp. 321–329, 2016.

[26] A. P. Twinanda, E. O. Alkan, A. Gangi, M. de Mathelin, and N. Padoy, “Data-driven spatio-temporal rgbd feature encoding for action recognition in operating rooms”, *International journal of computer assisted radiology and surgery*, vol. 10, no. 6, pp. 737–747, 2015.

[27] G. Stavropoulos, D. Giakoumis, K. Moustakas, and D. Tzovaras, “Automatic action recognition for assistive robots to support mci patients at home”, in *Proceedings of the 10th international conference on pervasive technologies related to assistive environments*, 2017, pp. 366–371.

[28] S. Ikemura and H. Fujiyoshi, “Real-time human detection using relational depth similarity features”, in *Asian Conference on Computer Vision.* Springer, 2010, pp. 25–38.

[29] L. Spinello and K. O. Arras, “People detection in rgb-d data”, in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, 2011, pp. 3838–3843.

[30] Z. Zhang, “Microsoft kinect sensor and its effect”, *IEEE multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

[31] “Xtion rgb and depth sensor”, https://www.asus.com/3D-Sensor/, accessed: 2020-07-13.

[32] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, “Depth mapping using projected patterns”, Apr. 3 2012, uS Patent 8,150,142.

[33] “The microsoft kinect sdk”, https://developer.microsoft.com/windows/kinect/, accessed: 2020-07-13.

[34] “Openkinect project”, https://openkinect.org/, accessed: 2020-07-13.

[35] S. B. Gokturk, H. Yalcin, and C. Bamji, “A time-of-flight depth sensor-system description, issues and solutions”, in *2004 conference on computer vision and pattern recognition workshop.* IEEE, 2004, pp. 35–35.

[36] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, “Ef-

ficient regression of general-activity human poses from depth images", in *2011 International Conference on Computer Vision.* IEEE, 2011, pp. 415–422.

[37] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun, "Real-time identification and localization of body parts from depth images", in *2010 IEEE International Conference on Robotics and Automation.* IEEE, 2010, pp. 3108–3113.

[38] B. Holt, E.-J. Ong, H. Cooper, and R. Bowden, "Putting the pieces together: Connected poselets for human pose estimation", in *2011 IEEE international conference on computer vision workshops (ICCV workshops).* IEEE, 2011, pp. 1196–1201.

[39] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations", in *2009 IEEE 12th International Conference on Computer Vision.* IEEE, 2009, pp. 1365–1372.

[40] M. Andriluka, S. Roth, and B. Schiele, "Pictorial structures revisited: People detection and articulated pose estimation", in *2009 IEEE conference on computer vision and pattern recognition.* IEEE, 2009, pp. 1014–1021.

[41] D. Grest, J. Woetzel, and R. Koch, "Nonlinear body pose estimation from depth images", in *Joint Pattern Recognition Symposium.* Springer, 2005, pp. 285–292.

[42] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, "Accurate 3d pose estimation from a single depth image", in *2011 International Conference on Computer Vision.* IEEE, 2011, pp. 731–738.

[43] Q. Zhang, X. Song, X. Shao, R. Shibasaki, and H. Zhao, "Unsupervised skeleton extraction and motion capture from 3d deformable matching", *Neurocomputing*, vol. 100, pp. 170–182, 2013.

[44] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-

driven approach for real-time full body pose reconstruction from a depth camera", in *Consumer Depth Cameras for Computer Vision.* Springer, 2013, pp. 71–98.

[45] Y. Zhu, B. Dariush, and K. Fujimura, "Controlled human pose estimation from depth image streams", in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops.* IEEE, 2008, pp. 1–8.

[46] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[47] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy: Fixefficientnet", *arXiv preprint arXiv:2003.08237*, 2020.

[48] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey", *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[49] A. Graves, "Supervised sequence labelling", in *Supervised sequence labelling with recurrent neural networks.* Springer, 2012, pp. 5–13.

[50] J. E. Dayhoff and J. M. DeLeo, "Artificial neural networks: opening the black box", *Cancer: Interdisciplinary International Journal of the American Cancer Society*, vol. 91, no. S8, pp. 1615–1635, 2001.

[51] B. Widrow and M. E. Hoff, "Adaptive switching circuits", Stanford Univ Ca Stanford Electronics Labs, Tech. Rep., 1960.

[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[53] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology", *Insights*

into imaging, vol. 9, no. 4, pp. 611–629, 2018.

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

[55] N. Qian, "On the momentum term in gradient descent learning algorithms", *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[56] S. J. Pan and Q. Yang, "A survey on transfer learning", *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[57] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *2009 IEEE conference on computer vision and pattern recognition.* Ieee, 2009, pp. 248–255.

[58] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[59] M. Lin, Q. Chen, and S. Yan, "Network in network", *arXiv preprint arXiv:1312.4400*, 2013.

[60] S. Laraba, M. Brahimi, J. Tilmanne, and T. Dutoit, "3d skeleton-based action recognition by representing motion capture sequences as 2d-rgb images", *Computer Animation and Virtual Worlds*, vol. 28, no. 3-4, p. e1782, 2017.

[61] S. Laraba, J. Tilmanne, and T. Dutoit, "Leveraging pre-trained cnn models for skeleton-based action recognition", in *International Conference on Computer Vision Systems.* Springer, 2019, pp. 612–626.

[62] F. Han, B. Reily, W. Hoff, and H. Zhang, "Space-time representation of people based on 3d skeletal data: A review", *Computer Vision and Image Understanding*, vol. 158, pp. 85–105, 2017.

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

[64] L. L. Presti and M. La Cascia, “3d skeleton-based human action classification: A survey”, *Pattern Recognition*, vol. 53, pp. 130–147, 2016.

[65] A. Kar, N. Rai, K. Sikka, and G. Sharma, “Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3376–3385.

[66] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos”, in *Advances in neural information processing systems*, 2014, pp. 568–576.

[67] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, “Action recognition in video sequences using deep bi-directional lstm with cnn features”, *IEEE Access*, vol. 6, pp. 1155–1166, 2017.

[68] A. Graves, “Neural networks”, in *Supervised sequence labelling with recurrent neural networks.* Springer, 2012, pp. 15–33.

[69] Y. Du, Y. Fu, and L. Wang, “Representation learning of temporal dynamics for skeleton-based action recognition”, *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3010–3022, 2016.

[70] V. Veeriah, N. Zhuang, and G.-J. Qi, “Differential recurrent neural networks for action recognition”, in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4041–4049.

[71] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, “Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks”, in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[72] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, “Ntu rgb+ d: A large scale dataset for 3d human activity analysis”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1010–1019.

[73] J. Gu, G. Wang, and T. Chen, “Recurrent highway networks with language cnn for image captioning”, *arXiv preprint arXiv:1612.07086*, 2016.

[74] J. Weston, S. Chopra, and A. Bordes, “Memory networks”, *arXiv preprint arXiv:1410.3916*, 2014.

[75] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, “How to construct deep recurrent neural networks”, *arXiv preprint arXiv:1312.6026*, 2013.

[76] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long short-term memory, fully connected deep neural networks”, in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2015, pp. 4580–4584.

[77] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[78] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning”, *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[79] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, “Temporal segment networks: Towards good practices for deep action recognition”, in *European conference on computer vision.* Springer, 2016, pp. 20–36.

[80] J. Liu, G. Zhang, Y. Liu, L. Tian, and Y. Q. Chen, “An ultra-fast human detection method for color-depth camera”, *Journal of Visual Communication and Image Representation*, vol. 31, pp. 177–185, 2015.

[81] J. Liu, Y. Liu, G. Zhang, P. Zhu, and Y. Q. Chen, "Detecting and tracking people in real time with rgb-d camera", *Pattern Recognition Letters*, vol. 53, pp. 16–23, 2015.

[82] J. Liu, Y. Liu, Y. Cui, and Y. Q. Chen, "Real-time human detection and tracking in complex environments using single rgbd camera", in *2013 IEEE International Conference on Image Processing.* IEEE, 2013, pp. 3088–3092.

[83] G. Zhang, L. Tian, Y. Liu, J. Liu, X. A. Liu, Y. Liu, and Y. Q. Chen, "Robust real-time human perception with depth camera." in *ECAI*, 2016, pp. 304–310.

[84] F. Lv and R. Nevatia, "Recognition and segmentation of 3-d human action using hmm and multi-class adaboost", in *European conference on computer vision.* Springer, 2006, pp. 359–372.

[85] L. Rabiner and B. Juang, "An introduction to hidden markov models", *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[86] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting", in *European conference on computational learning theory.* Springer, 1995, pp. 23–37.

[87] L. Xia, C.-C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3d joints", in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops.* IEEE, 2012, pp. 20–27.

[88] X. Yang and Y. L. Tian, "Eigenjoints-based action recognition using naive-bayes-nearest-neighbor", in *2012 IEEE computer society conference on computer vision and pattern recognition workshops.* IEEE, 2012, pp. 14–19.

[89] C. Zhang and Y. Tian, "Rgb-d camera-based daily living activity recognition", *Journal of computer vision and image processing*, vol. 2, no. 4,

p. 12, 2012.

[90] S. Sempena, N. U. Maulidevi, and P. R. Aryan, “Human action recognition using dynamic time warping”, in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics.* IEEE, 2011, pp. 1–5.

[91] G. Evangelidis, G. Singh, and R. Horaud, “Skeletal quads: Human action recognition using joint quadruples”, in *2014 22nd International Conference on Pattern Recognition.* IEEE, 2014, pp. 4513–4518.

[92] E. Ghorbel, R. Boutteau, J. Boonaert, X. Savatier, and S. Lecoeuche, “Kinematic spline curves: A temporal invariant descriptor for fast action recognition”, *Image and Vision Computing*, vol. 77, pp. 60–71, 2018.

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

[94] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult”, *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[95] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks”, in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[96] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[97] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches”, *arXiv preprint arXiv:1409.1259*, 2014.

[98] J. Liu, A. Shahroudy, D. Xu, and G. Wang, “Spatio-temporal lstm with trust gates for 3d human action recognition”, in *European conference on*

computer vision. Springer, 2016, pp. 816–833.

[99] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data", in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[100] W. Li, L. Wen, M.-C. Chang, S. Nam Lim, and S. Lyu, "Adaptive rnn tree for large-scale human action recognition", in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1444–1452.

[101] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling", *arXiv preprint arXiv:1803.01271*, 2018.

[102] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. L. Y. Bengio, and A. Courville, "Towards end-to-end speech recognition with deep convolutional neural networks", *arXiv preprint arXiv:1701.02720*, 2017.

[103] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation", *arXiv preprint arXiv:1611.02344*, 2016.

[104] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time", *arXiv preprint arXiv:1610.10099*, 2016.

[105] H. Adel and H. Schütze, "Exploring different dimensions of attention for uncertainty detection", *arXiv preprint arXiv:1612.06549*, 2016.

[106] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition", *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[107] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural

networks", in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[108] P. Wang, Z. Li, Y. Hou, and W. Li, "Action recognition based on joint trajectory maps using convolutional neural networks", in *Proceedings of the 2016 ACM on Multimedia Conference.* ACM, 2016, pp. 102–106.

[109] Y. Hou, Z. Li, P. Wang, and W. Li, "Skeleton optical spectra-based action recognition using convolutional neural networks", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 3, pp. 807–811, 2016.

[110] C. Li, Y. Hou, P. Wang, and W. Li, "Joint distance maps based action recognition with convolutional neural networks", *IEEE Signal Processing Letters*, vol. 24, no. 5, pp. 624–628, 2017.

[111] M. Liu, H. Liu, and C. Chen, "Enhanced skeleton visualization for view invariant human action recognition", *Pattern Recognition*, vol. 68, pp. 346–362, 2017.

[112] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "A new representation of skeleton sequences for 3d action recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3288–3297.

[113] H. Liu, J. Tu, and M. Liu, "Two-stream 3d convolutional neural network for skeleton-based action recognition", *arXiv preprint arXiv:1705.08106*, 2017.

[114] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition", *arXiv preprint arXiv:1801.07455*, 2018.

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

tion, 2019, pp. 12 026–12 035.

[116] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3595–3603.

[117] H. Prashanth, H. Shashidhara, and B. M. KN, "Image scaling comparison using universal image quality index", in *2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies*. IEEE, 2009, pp. 859–863.

[118] J. Titus and S. Geroge, "A comparison study on different interpolation methods based on satellite images", *International Journal of Engineering Research & Technology*, vol. 2, no. 6, pp. 82–85, 2013.

[119] C. Li, S. Sun, X. Min, W. Lin, B. Nie, and X. Zhang, "End-to-end learning of deep convolutional neural network for 3d human action recognition", in *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2017, pp. 609–612.

[120] Z. Huang, C. Wan, T. Probst, and L. Van Gool, "Deep learning on lie groups for skeleton-based action recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6099–6108.

[121] J.-F. Hu, W.-S. Zheng, L. Ma, G. Wang, J. Lai, and J. Zhang, "Early action prediction by soft regression", *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 11, pp. 2568–2583, 2018.

[122] J.-F. Hu, W.-S. Zheng, J. Lai, and J. Zhang, "Jointly learning heterogeneous features for rgb-d activity recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5344–5352.

[123] J. Liu, A. Shahroudy, D. Xu, A. C. Kot, and G. Wang, "Skeleton-based

action recognition using spatio-temporal lstm network with trust gates", *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 3007–3021, 2017.

[124] J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot, "Global context-aware attention lstm networks for 3d action recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1647–1656.

[125] J. Liu, A. Shahroudy, G. Wang, L.-Y. Duan, and A. C. Kot, "Skeleton-based online action prediction using scale selection network", *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 6, pp. 1453–1467, 2019.

[126] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "Learning clip representations for skeleton-based 3d action recognition", *IEEE Transactions on Image Processing*, vol. 27, no. 6, pp. 2842–2855, 2018.

[127] M. Liu and J. Yuan, "Recognizing human actions as the evolution of pose estimation maps", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1159–1168.

[128] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points", in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops.* IEEE, 2010, pp. 9–14.

[129] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, "Sequence of the most informative joints (smij): A new representation for human skeletal action recognition", *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 24–38, 2014.

[130] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization", in *Proceedings of the 28th international conference on machine learning (ICML-11).* Citeseer, 2011, pp. 1033–1040.

[131] M. Müller and T. Röder, "Motion templates for automatic classification

and retrieval of motion capture data", in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006, pp. 137–146.

[132] C. Ellis, S. Z. Masood, M. F. Tappen, J. J. LaViola, and R. Sukthankar, "Exploring the trade-off between accuracy and observational latency in action recognition", *International Journal of Computer Vision*, vol. 101, no. 3, pp. 420–436, 2013.

[133] X. Zhao, X. Li, C. Pang, X. Zhu, and Q. Z. Sheng, "Online human gesture recognition from motion data streams", in *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 23–32.

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

[135] J. Bao, L. Pei, and X. Zhao, "Action recognition based on conceptors of skeleton joint trajectories", in *Revista de la Facultad de Ingeniería U.C.V*, 2014, pp. 11–22.

[136] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber, "Documentation mocap database hdm05", Universität Bonn, Tech. Rep. CG-2007-2, June 2007.

[137] Z. Huang and L. Van Gool, "A riemannian network for spd matrix learning", in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[138] R. Vemulapalli and R. Chellapa, "Rolling rotations for recognizing human actions from 3d skeletal data", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4471–4479.

[139] S. Aubry, S. Laraba, J. Tilmanne, and T. Dutoit, "Action recognition based on 2d skeletons extracted from rgb videos", in *MATEC Web of Conferences*, vol. 277. EDP Sciences, 2019, p. 02034.

[140] "Vidcon conference for and about online video", https://www.vidcon.com/, accessed: 2020-07-15.

[141] J. W. Davis and A. F. Bobick, "The representation and recognition of human movement using temporal templates", in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE, 1997, pp. 928–934.

[142] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes", *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 12, pp. 2247–2253, 2007.

[143] E. Yu and J. K. Aggarwal, "Human action recognition with extremities as semantic posture representation", in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops.* IEEE, 2009, pp. 1–8.

[144] I. Laptev, "On space-time interest points", *International journal of computer vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[145] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features", in *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance.* IEEE, 2005, pp. 65–72.

[146] M. Bregonzio, S. Gong, and T. Xiang, "Recognising action as clouds of space-time interest points", in *2009 IEEE conference on computer vision and pattern recognition.* IEEE, 2009, pp. 1948–1955.

[147] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition", *International journal of computer vision*, vol. 103, no. 1, pp. 60–79, 2013.

[148] R. Christoph and F. A. Pinz, "Spatiotemporal residual networks for video action recognition", *Advances in Neural Information Processing Systems*, pp. 3468–3476, 2016.

[149] C. Feichtenhofer, A. Pinz, and R. P. Wildes, “Spatiotemporal multiplier networks for video action recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4768–4777.

[150] Y. Zhu, Z. Lan, S. Newsam, and A. Hauptmann, “Hidden two-stream convolutional networks for action recognition”, in *Asian Conference on Computer Vision.* Springer, 2018, pp. 363–378.

[151] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks”, in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[152] Z. Qiu, T. Yao, and T. Mei, “Learning spatio-temporal representation with pseudo-3d residual networks”, in *proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5533–5541.

[153] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition”, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459.

[154] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue, “Modeling spatial-temporal clues in a hybrid deep learning framework for video classification”, in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 461–470.

[155] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[156] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei, “Unsupervised learning of long-term motion dynamics for videos”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017,

pp. 2203–2212.

[157] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao, “A key volume mining deep framework for action recognition”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1991–1999.

[158] S. Ma, L. Sigal, and S. Sclaroff, “Space-time tree ensemble for action recognition”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5024–5032.

[159] G. Gkioxari and J. Malik, “Finding action tubes”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 759–768.

[160] S. Singh, C. Arora, and C. Jawahar, “First person action recognition using deep learned descriptors”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2620–2628.

[161] M. Ma, H. Fan, and K. M. Kitani, “Going deeper into first-person activity recognition”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1894–1903.

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

[163] G. Chéron, I. Laptev, and C. Schmid, “P-cnn: Pose-based cnn features for action recognition”, in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3218–3226.

[164] M. Zolfaghari, G. L. Oliveira, N. Sedaghat, and T. Brox, “Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection”, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2904–2913.

[165] F. Baradel, C. Wolf, and J. Mille, “Human activity recognition with

pose-driven attention to rgb", in *Proceedings of the 29th British Machine Vision Conference*, 2018.

[166] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps", *arXiv preprint arXiv:1312.6034*, 2013.

[167] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.

[168] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4829–4837.

[169] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning", *arXiv preprint arXiv:1702.08608*, 2017.

[170] "Amazon mechanical turk", https://www.mturk.com/, accessed: 2020-07-13.

[171] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets", *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[172] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion", *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[173] Y. LeCun and C. Cortes, *The MNIST Database of Handwritten Digits*, 1998.

[174] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization", *arXiv preprint arXiv:1506.06579*, 2015.

[175] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks", in *2010 IEEE Computer Society Conference on computer vision and pattern recognition.* IEEE, 2010, pp. 2528–2535.

[176] Z. Qin, F. Yu, C. Liu, and X. Chen, "How convolutional neural network see the world-a survey of convolutional neural network visualization methods", *arXiv preprint arXiv:1804.11191*, 2018.

[177] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning", in *2011 International Conference on Computer Vision.* IEEE, 2011, pp. 2018–2025.

[178] J. Tilmanne and N. d'Alessandro, "Motion machine: A new framework for motion capture signal feature prototyping", in *2015 23rd European Signal Processing Conference (EUSIPCO).* IEEE, 2015, pp. 2401–2405.