Human Motion Prediction

Yanran Li

Thesis of Doctoral Philosophy

March, 2020

Copyright Statement

Abstract

Motion data has been extensively used in the computer animation industry, feature films, pedestrian tracking, and surveillance. The capacity to understand and predict humans' future movements is much sought-after, for it would have a range of practical applications in fields such as autonomous vehicles and interactive robotics. The complicated constraints of the human body and its high-dimensional dynamics, however, mean that human motion prediction is extremely challenging.

A growing body of approaches are being used to develop various models for motion prediction. Traditional methods, such as Boltzmann machine and Markov chains, proposed possibility models for predicting the sequences of human movement. Furthermore, deep-learning models are introduced and most of them treat motion prediction in similar ways to machine translation problems. However, recent deep learning approaches adopted RNN, CNN, or Fully Connected Networks to learn motion features that do not fully exploit the hierarchical structure of human anatomy. Existing models suffer from the mean pose problem, and oversmoothing problem. Mean pose problem is the models prone to produce the fix pose for a time period. The oversmoothing problem is spefic exist in the graph neural network models for motion prediction. It means that when the neural network goes deeper, the feature learned from network are not distinguished anymore.

In this PhD research, these problems are pursued and two new models for motion prediction are proposed. These models are state-of-the-art: not only are they highly efficient in computation and memory, but they produce realistic motion visualizations too. The first model used the hierarchical structure of human body to reduce the parameter size significantly. The second model introduced a densely connected GCN model to reduce the oversmoothing problem.

To be more specific, a convolutional hierarchical autoencoder model for motion prediction is proposed. Its novel encoder incorporates 1D convolutional layers and hierarchical topology. The new network is smaller and faster than existing deep-learning models. The qualitative and quan-

titative results show that these models outperform state-of-the-art methods in both short-term and long-term prediction.

Following the recent success of graph neural network, an advanced GCN based framework for motion prediction is proposed which connects all the GCN blocks directly. Therefore, the output feature of each GCN block skips the middle layers and jumps to the final layers. Compared to the existing GCN model for motion prediction, this model requires almost the same level of parameters. But it significantly enlarged the feature maps utilization and increased the impact of earlier layers' feature map. Moreover, this densely connected structure makes the model for motion prediction easier to train and able to go deeper.

All these models are evaluated by conducting extensive comparison experiments on the standard benchmarks for motion prediction, which are Human3.6M and the CMU. The performance evaluated from both the angle and 3D position aspects, demonstrated these models' superiority over the state-of-the-art works.

Acknowledgements

First of all, I would like to express my gratitude to my supervisors Dr. Xiaosong Yang and Prof. Jiang J. Zhang for their continued support and advices. Their guidance and help, from research to life, are worth my lifelong gratitude. I am really grateful to meet such supervisors as them, who are always patient to listen to my thoughts and ideas. This work is the result from our countless meetings, especially with Dr. Yang, who encouraged me throughout these four years. Also, I would like to thank Li Wang and Yingyu Nie, for the collaboration and daily discussions.

Part of the inspirations leading to this work also arise from discussions with Dr. Zhao Wang, and Dr. Yunfei Fu. Their critical comments, kind minds and resource sharing were always there when I needed them most.

Memorable friendships have been established during my 4 years study in NCCA. Thanks to Prof. Xiaoguang Han, Prof. Lihua You, Dr. Zhidong Xiao, Dr. Tao Jiang, Dr. Jing Wang, Dr. Kun Qian, Dr. Shaojun Bian and Dr. Ming Jiang for their help and support. Thanks to Ming Zhang, Liangyu Liu, Nan Xiang, Lingteng Qiu, Jinglu Zhang, Yao Lv, Jianyuan Sun, Mengqing Huang, Shaodi Dong, Ruibing Wang, Yu Xia, Shuangbu Wang, Zhangchan Ding and other friends for the amazing time. Thanks to Jan Lewis, Sunny Choi, Sonia Ashby, Jane Whitaker and all graduate school staffs, who handled all the administrative tasks and supported all my visiting.

Gratitude also goes to the BU PhD Fundings, SWTCN fellowships for PhD studentship and financial support of my activities.

Finally, I am greatly indebted to my family, including my parents and boyfriend, Sebastian Iulian Poiana.

Declaration

This thesis has been created by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated.

Contents

List of Figures

List of Tables

List of Publications

1. **Yanran Li**, Zhao Wang, Xiaosong Yang, Meili Wang, Sebastian Iulian Poiana, Ehtzaz Chaudhry and Jianjun Zhang. Efficient convolutional hierarchical autoencoder for human motion prediction, The Visual Computer. 2019 Jun 1;35(6-8):1143-56. Accepted by Computer Graphics International 2019.

2. **Yanran Li**, Lingteng Qiu, Li Wang, Fangde Liu, Zhao Wang, Sebastian Iulian Poiana, Xiaosong Yang and Jianjun Zhang. Densely connected GCN model for motion prediction. Computer Animation and Virtual Worlds CASA 2020 special issue. & Computer Animation and Virtual Worlds.

Chapter 1

Introduction

1.1 Background

Motion prediction is a crucial task for animation production, which aims to forecast humans' future movements based on input sequences. The question of how to predict motion is one of the most important contemporary research problems which also has applications in many different areas such as Interactive Robotics, Autonomous Vehicle, Surveillance and Animation Generation (described in the followings).

There are four main areas in which motion prediction can be applied:

1. **Interactive Robotics.** Robots that interact with humans and virtual characters in computer games are supposed not only to respond to opponents' movements, but also be able to predict and pre-empt future movements. To give an example from adversarial sports, such as badminton and fencing, or collaborative scenarios, such as paired figure skating and Waltz dancing. In these fields, an intelligent agent is expected to anticipate human athletes' actions accurately and rapidly using historical data [Gui et al. [2018b], Pérez-D'Arpino & Shah [2016]].

2. **Autonomous Vehicle.** Research into autonomous vehicles has attracted tremendous attention in both academia and industry, due to its large potential commercial benefits. One of the essential

challenges in this area is that autonomous vehicles must be able to predict pedestrians' impending movements on the road. Through this capacity, they will be able to take advance actions to avoid traffic accidents. An effective motion prediction algorithm is therefore necessary for autonomous vehicle systems [Liang et al. [2019]].

3. **Surveillance.** Visual surveillance systems are usually installed in public areas or individual houses to track targets' behaviour. They can be used, for example, to detect abnormal behaviour, manage crowds, and ensure public security. Motion prediction techniques can contribute to fulfilling these goals by providing an enhanced capacity for understanding human behaviour with stationary cameras [Ristic et al. [2008] Golda et al. [2019]].

4. **Animation Generation.** In animated productions, the animators create gesticulations for virtual characters, such that they move like human beings. Traditionally, motion sequences are generated by interpolating keyframes or recording the actions of human beings. Lots of contemporary films that feature visual effects apply motion data generated indoors, such as *The Avengers*, *Avatar*, and *Doctor Strange*, etc. However, these methods require expensive equipment and intense manual effort. Motion prediction technology, in contrast, can be used in tools that animate motions automatically. For example, the user can simply draw a route that a given visual character is to take and the tool will generate running or walking movements automatically [Holden et al. [2016];Holden et al. [2017]]. This new technique saves a great deal of time and labour for the animation industry.

In a conclusion, motion prediction techniques have broad impacts and applications. What is more, these techniques are pivotal to understanding and reusing MoCap data. MoCap data is widely used in visual effects and athletic training (Böhm et al. [2001] Huang et al. [2012] Feng et al. [2014] and Xiao et al. [2015]). Given the great commercial benefits of MoCap data, and the growth of feature films and animations, demand for motion data has increased. Hence, understanding motion data is crucial

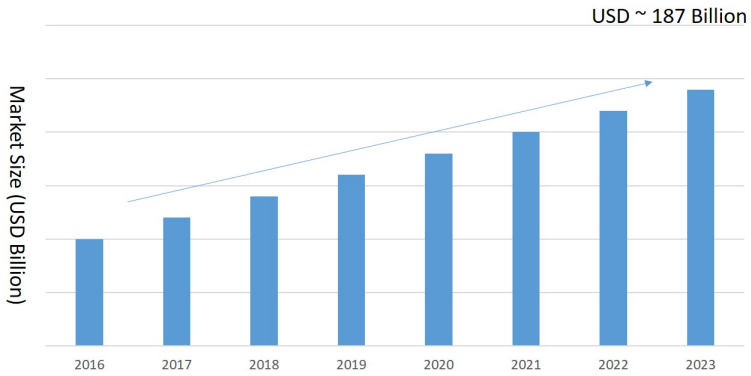

Figure 1.1: *Global 3D Motion Capture System Market Research Report - Global Forecast 2023. Please refer to Web [2020]*

for both academia and industry. Existing data suggests that the motion market will continue expanding as it allows artists and other users to capture and model diverse motions conveniently (Figure 1.1).

Human beings can forecast human motion as part of their natural intelligence. It remains challenging, however, because human motion data is high dimensional and is subject to bio-mechanical constraints. The task of motion prediction is extremely challenging, for it involves techniques for representing, analysing, and generating motion. Existing models suffer from the mean pose and oversmoothing problems. Demand for an efficient and effective motion prediction algorithm is increasing day by day, along with the development of artificial intelligence systems. It is imperative, therefore, that researchers develop an effective and efficient motion prediction method, which produces realistic and high-fidelity motion sequences. This is the key motivation behind this PhD research.

1.2 Main Challenges

Although existing approaches have taken a painstaking effort to address the motion prediction problem, thus far they have had limited success. This is because motion prediction is a very challenging problem. The main challenges are as follows (CL refer to Challenge):

CL1: Efficiency Since the amount of motion data is increasing rapidly, existing motion modelling methods require a huge amount

of memory. The existing motion prediction methods are not fast enough to meet industry requirements. For the most part, existing methods are based on RNNs or Fully Connected Networks (FCNs), which are hard to train because the models converge very slowly. Moreover, large models and parameters that run into the millions are costly, requiring significant computational and memory resources.

CL2: Temporal Spatial information modelling Motion data contains both temporal and spatial information. Therefore, the motion models that either treat motion data as images or language do not exploit the temporal as well as spatial dependencies, nor the ways in which temporal and spatial data interact. Unlike images and language, motion data has a distinct hierarchical spatial structure, as well as unique temporal relationships.

CL3: Complex human body constraints modelling Motion data is captured from real human beings, allowing for realistic motion data that adheres closely to body and sports mechanics. Although humans can intuitively recognise these constraints, computers struggle to define and understand them precisely, due to the complex scenarios in question.

CL4: Oversmoothing problem Recently, researchers have proposed Graph Neural Networks (GNNs) for motion prediction. As the Graph Convolutional Network (GCN) layers go deeper, the distinctiveness of the graph's node features diminishes. This is usually called the oversmoothing problem.

CL5: Generalization Predicting a single type of action is much easier than predicting multiple types because the model can more readily learn the hidden rules. However, in practice, human actions are diverse in types. Multiple forms of motion data significantly increase the complexity involved in prediction. It is therefore both necessary and challenging that researchers build a single model that can be generalised for all kinds of actions.

1.3 Research Aims

The aim of this research is to explore a motion modelling method that addresses the aforementioned challenges to understand and predict motions. This research will model motion sequences in a novel way, offering a more effective and efficient solution for motion prediction. This thesis explores new ways to model the spatial-temporal characteristics of motion data. To validate the performance of the proposed methods, this thesis presents qualitative and quantitative experiments. Accordingly, this research will benefit the animation industry and research on motion.

1.4 Research Objectives

To achieve the aim set out above, this thesis pursues the objectives outlined below (OBJ refers to Objective):

OBJ1: Literature Review Review current approaches to motion classification, retrieval, prediction, and refinement; identify the limitations and challenges of existing motion modelling methods, and review state-of-art deep learning methods, especially research on spatial-temporal modelling.

OBJ2: Motion Modelling Propose an effective and efficient motion modelling method, which can represent motion sequence information and improve the accuracy of motion prediction.

OBJ3: Improve the efficiency of the Motion Prediction Model Develop a motion prediction model that is more efficient than the existing model from both perspectives of time and memory consumption and can generate accurate results at the same time.

OBJ4: Alleviate the oversmoothing problem Deeply investigate the existing oversmoothing problem and tackle this challenge by exploring a new motion prediction model and preserve the prediction performance at the same time.

OBJ5: Analyse Each Components and Strategies of Model

To validate the proposed idea and understand the problems this PhD research face in more depth, ablation studies have been conducted. In this way, we sought to determine the impact of the model's components and parameters and the contribution made by each of its parts. Accordingly, this thesis establishes a clear understanding of how the model works.

OBJ6: Model evaluation and results visualization Conduct experiments to investigate the proposed model and evaluate their effectiveness and limitations.

1.5 Contributions

The key contributions from this PhD research are the following:

1. To address motion prediction problems and outperform the previous state-of-the-art methods, a new Convolutional Hierarchical Autoencoder (CHA) model is proposed. This model is significantly more efficient than the CNN model. It generates high-fidelity motion sequences for both short-term and long-term prediction using the CMU and H3.6M datasets. Being mindful of motion data's characteristics, 1D convolutional layers are incorporated with hierarchical structures to exploit the the human body constrains.

2. A new Densely GCN-based model is proposed so as to address the problem of motion prediction. This model reuses the multi-scale feature maps from every block to enlarge the receptive field and reduce the overfitting problem.

3. Extensive comparison experiments have been conducted on the standard benchmarks for motion prediction: Human3.6M and the CMU motion capture dataset. Having been evaluated with regard to both angle and 3D position, the model surpasses the state-of-the-art performance on all datasets.

1.6 Structure of Thesis

This thesis is organised into five chapters. The relationship between each chapter and the research questions and objectives are as follows:

- Chapter 2 presents a literature review. This chapter has four parts relating to motion data, motion features, motion prediction methods, and related deep learning methods. In the first section, a detailed description of motion data is put forward and recent public motion datasets are presented. The second section discusses how existing approaches conduct motion feature engineering by reviewing research on motion recognition, retrieval, and synthesis. The third section reviews previous approaches to the motion prediction problem and outlines their limitations. Finally, relevant deep learning methods are summarised. This chapter developed a range of insights that feed into this research. Therefore, objective 1 has been addressed in this chapter.

- In Chapter 3, a new framework is described, which is able to predict motion efficiently. This is unlike existing frameworks, which borrow ideas from RNNs and CNN structures. This new framework has a network architecture that has been specifically designed for motion data. In this chapter, objectives 2, 3, 5 and 6 are achieved and contributions 1 and 3 are presented. This chapter also addresses challenge 1, 2, and 3 by elaborating a hierarchical structure and incorporating 1D convolutional operations. What is more, it presents details relating to the CHA.

- In Chapter 4, a densely connected GCN model is introduced. The chapter fulfils objectives 2, 4 and 6 by designing a GNN to address the motion prediction problem. It is able to do this because the human body itself is an intuitive natural graph. This framework is related to contribution 2. The chapter addresses challenge 4 by introducing a dense link for the motion prediction framework. The related ablation and comparative studies of the heavily benchmarked datasets demonstrate that this model addresses challenge 5 as well. The chapter discusses and evaluates the model.

- Chapter 5 presents ideas about the future ideas so as to improve the research. The assessment of future work that is to be undertaken consists of the following: 1. The long term error accumulation problem is considered. A Cycle Strategy Motion Prediction Model (CSMPM) will be designed to address this challenging problem so that the performance of motion prediction can be lifted. 2. The multi-possible future of human motion prediction is considered and exploited deeper. The latent vector in this framework will be delved to find an interpretable representation. 3. The oversmoothing problem of the existing framework will be further pursued. A new network structure will be proposed in the future to enhance the feature map utilization. 4. Explore a better loss function rather than the existing 3D coordinate loss function and Euler Angle loss function. New loss functions will be designed to minimize the visual difference and numerical errors at the same time.

Chapter 2

Literature Review

This chapter reviews motion-related literature that bears upon four themes. First, existing work on motion data and MoCap systems are introduced, presenting widely used motion datasets. In this PhD research, evaluations based on these datasets have been conducted. Usually, motion prediction includes three steps: motion feature engineering, future prediction, and the synthesis of future motion sequences. Therefore, the current work related to motion feature engineering are reviewed, such as motion recognition and retrieval, in section 2. This section summarises approaches to motion synthesis. Then, in section 3, existing models for motion prediction and their limitations are discussed. Finally, we review recent deep learning methods in section 4, so as to inspire motion prediction research.

2.1 Motion Data

In computer vision, human motion capture was defined as "the process of capturing the large scale body movements of a subject at some resolution." (Moeslund & Granum [2001]). Motion data is widely used in academia and various industries for the purposes of robotics, education, and filmmaking. The human motion data used for this PhD research work is a sequence of 3D skeletons poses, as presented in Figure 2.1. The 3D human skeletons always have a hierarchical structure.

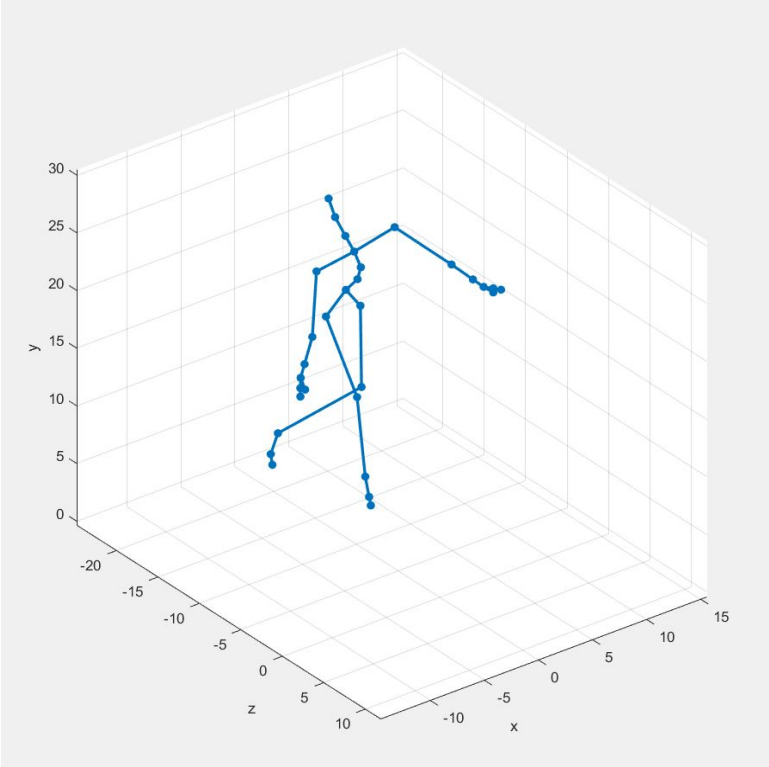

Figure 2.1: *The Motion Data Example: a ballet action. Here, a stick figure performs a ballet dance in the window.*

2.1.1 3D skeletal data

MoCap data is usually provided in three formats: depth images, RGB images, and 3D skeletons. A lot of 3D human skeleton MoCap databases have been released to the public. Given their convenience, robustness, and realtime performance, these have been widely used in researches. This report focuses on studies of skeleton-based MoCap data. 3D skeletal data usually derives from two sources: it is either captured directly using equipment or reconstructed from images/videos by means of computational methods. The following sections summarise these two ways of acquiring 3D human skeletal data.

Capturing by Special Equipment

Skeletal data can be directly obtained using equipment such as Kinect v1/v2, PrimeSensor. Three main types of equipment are used to detect the position of joints and movement, as well as record 3D skeletal data. These are motion capture systems, structured-light cameras, and time-of-flight sensors.

There are two traditional ways of producing Motion Capture data: through the Optical 3D Motion Capture System and through the Non-Optical 3D Motion Capture System. The Optical 3D Motion Capture System has several subcategories: active, passive, markerless, and underwater. The Non-Optical 3D Motion Capture System has three subtypes: mechanical, inertial, and electromagnetic. A number of results pertaining to their performance and properties have been listed in table 2.1:

The markerless, active and passive variants of the first system are used to capture motion indoors. The inertial version of the latter system is used capable to capture motion outdoors. Whereas indoor capture systems are expensive, the outdoor system is cheaper. Because conditions are most suitable indoors, the markerless, active, and passive subcategories are more accurate than the inertial system. Moreover, inertial equipment captures the largest scope of movement. When it comes to capturing the movement of multiple targets, the performance of passive systems remains unsurpassed.

Table 2.1: *The comparison of different methods of Motion Capture (Please ref to MoC [2020])*

Property	Markerless	Active	Passive	Inertial
Accuracy	High	High	High	High
Efficiency	Low	Low	Low	High
Scope of Movement	Small	Medium	Medium	Large
Multiple Target Capture	Low	Medium	High	Medium
Environment Requirements	Sunlight, heat source interference	Strong light interference	Sunlight interference	Sensor Noisy interference
Cost	Low	Medium	Medium	Low

Current motion capture systems attach markers or sensors to human joints, thereby immediately providing a 3D skeleton. If participants perform actions naturally while wearing the devices, the coordinates of their joints transmit into the computer. Whereas optical systems use cameras to track joints' positions by analysing light, inertial systems record the rotations of inertia sensors. Although optical systems are usually more accurate, inertial systems can work outdoors and capture a larger scope of movement. Motion capture systems are effective when it comes to producing high quality 3D skeletal data. That said, they are expensive (optical cameras systems range from $150,000 to $250,000).

Another way to acquire 3D skeletal data is to capture information related to colour and depth using structured-light cameras. Structured-light cameras do not require markers because they estimate the surface area of 3D objects by projecting infrared light. Some software, such as NITE, construct a 3D skeleton model using colour-depth information. These are cheaper than motion capture devices, for they do not entail wearing special equipment. However, they only work indoors.

Time-of-flight cameras obtain depth information in a different way than structured-light cameras. These systems use light in a manner comparable to the way in which a radar uses sound, calculating reflection times. They reconstruct depth images at around 160 frames per second,

more quickly than structured-light cameras. They detect joints in the colour-depth data, from which a 3D skeleton data can be constructed. Kinect v2 (released in 2014) is a typical camera that works in this way. In comparison to Kinect v1, it is significantly faster and more accurate.

Reconstruction from Images

Additionally, 3D skeleton data can be reconstructed from image data. Media data that involves human actions and poses has grown rapidly. Given its great practical value, the topic of 3D skeleton reconstruction has attracted increasing attention on the part of researchers and software developers. The 3D positions of joints can be estimated from depth image data, single image data, and multi-perspective image data.

Most approaches to estimating 3D skeletons using depth RGB images consist of two steps. The first is to classify the position of each pixel on the surface of an image. The second is to minimise errors in matching data to a 3D skeleton. Recent approaches to deep learning have performed remarkably in this area by improving the first of these two steps. These approaches try to detect the joints either by using visual descriptors or by matching an existing 3D pose to the human body. Wohlhart & Lepetit [2015] have used CNN to classify the descriptor of object views and assign the related 3D skeleton to images. Porzi et al. [2017] have designed a depth-CNN to predict the position of pixels on a depth image. They classify pixels very accurately, using a random sample consensus (RANSAC)-based scheme, which is simpler than traditional optimisation.

It is expensive to generate depth images using special cameras. It is unfeasible, then, to generalise this process such that it can be used by private individuals. Another way to provide depth information is to use multi-perspective images. Gall et al. [2009] have proposed an optimisation method for detecting underlying 3D skeletons from multi-view silhouette sequences. Liu et al. [2011] have presented a framework that uses image segmentation and the probability estimation method to reconstruct the 3D skeletons of two people from a multi-view interaction

video.

Like depth images, multi-perspective images involve strict restrictions on the recording environment. Accordingly, it is much less accessible to individuals than the single image method. The challenge post by the latter is that it does not capture depth information. Therefore, many researchers only consider extracting 2D skeleton information. Some approaches use physical and somatological constrains (such as limb length or limited rotation angles) to estimate depth data. This thesis focuses on 3D motion data; accordingly, only 3D skeleton reconstruction work is discussed here.

Early approaches represented images by means of local descriptors, such HoG or SIFT. In this way, they matched descriptors to 3D skeletons. Li & Chan [2014], for instance, have used a CNN to detect human body parts in monocular images. In this case, however, the descriptors are not robust and matching accuracy is low. Another solution for estimating 3D skeleton from single images is to annotate a set of training images and apply a regression model or deep learning framework.

However, it is much harder to collect a large amount of annotated images with 3D skeletons compared to collecting a dataset with 2D skeletons. To address the limitations posed by having insufficient training data for detecting 3D skeletons in images, researchers have tried to retrieve 3D skeletons from estimated 2D skeletons. For example, Yasin et al. [2016] has trained a pictorial structure model (PSM) to estimate the position of a 2D skeleton from a single image initially and then use an iterative procedure to searching the 3D skeleton with the projected 2D similar skeleton. Recently, Tome et al. [2017] have proposed a CNN architecture that extrapolates 3D skeletons from 2D skeletons by calculating probabilities under mechanical and somatological constraints. This framework achieved state-of-the-art results using the Human3.6m dataset, being able to generate 3D skeletons from single images in real-time.

Summary. This section reviews the basic forms of motion data and the typical technologies for obtaining motion data. Motion capture sys-

tems are common in the industry. Nowadays, researchers are trying new approaches to obtain motion data from videos or images. This thesis focuses on motion-related problems. As part of this broader focus, this review section establishes the basic knowledge necessary for understanding motion data.

2.1.2 Motion Dataset Benchmark

MoCap datasets containing 3D skeletal data are widely used in motion research. Some key datasets in this area are as follows:

Msr-Action 3D This dataset, which contains 20 actions, was collected by Li et al. [2010]. They found 10 volunteers, who performed each of the actions several times to create a database of 567 sequences. This dataset not only offers skeleton information but depth images as well. The human skeleton structure used in this dataset contains 20 joint points.

Berkeley MHAD This dataset was presented by Ofli et al. [2013]. It contains 11 actions performed by 12 individuals (7 men and 5 women). Each participant was asked to perform each action 5 times, allowing the researchers to collect 660 motion sequences in total. They used 8 motion capture cameras to trace 43 LED markers, 12 Dragonfly2 cameras to capture multi-view video data, and two Kinect cameras to acquire depth images. Therefore, this dataset provides skeletal, depth, and RGB information. Moreover, they asked each participant to strike a T pose, which they recorded to provide a marker for the skeleton structure.

CMU This dataset is freely available to the public online. (Please refer to the website: The CMU dataset) It presents a large amount of skeleton based MoCap data, including around 2,605 sequences that have been divided into 6 categories and 23 subcategories (actions). The skeletons that it uses consist of 31 joints and related limbs. Unlike the MsrAction 3D and Berkeley MHAD, this dataset contains not only single actions but also interactive actions (including interactions both between peoples and between people and environments. The actions feature various sports and physical activities, such as dancing, football, and golf.

Each category includes a range of different performances: the "walking" category, for example, has more than 100 kinds of different motion sequences, which offer many fine details.

HDM05 Müller et al. [2007] have presented MoCap Dataset HDM05, which has more than three hours' worth of footage spanning five categories. It has 22 subcategories (action types) and each category contains between 10 and 15 motion sequences (action style and direction, in both C3D and AMC data formats). The skeleton that this database uses consists of 31 joints. The actions that it contains were performed by individuals against a clear background (except for a chair and table). It has many actions in common with other CMU datasets, such as walking, running, jumping, etc.

Human3.6m This dataset is provided by Ionescu et al. [2014]. It contains 3.6 million 3D human poses and corresponding images. 11 professional actors were asked to perform in certain scenarios, such as a discussion, and enact certain actions, such as smoking, etc. Their pose and actions were then captured by a depth camera, 3D laser scans, and Kinect Systems. Therefore, this dataset provides motion data with depth, RGB, and skeleton information, as well as a huge range of pose image data. It uses a skeleton of 32 joints to represent all of the participants.

NTU RGB+D Shahroudy et al. [2016] have provided a large dataset including more than 56,000 video samples of actions that were performed by 40 participants. These actions span 60 categories and were recorded in 4 million frames from 80 viewpoints. This data includes all of the modes considered so far: RGB, depth, infrared frames, and skeletal data. The skeleton used in this dataset has 25 major body joints. Unlike all the other dataset mentioned before, which used Kinect V1 only, the researchers enrolled Kinect V2. The recorded actions include various forms of locomotion and a range of sports. Moreover, the dataset includes interactive actions, which also appear in CMU.

3DPW von Marcard et al. [2018] have introduced a new dataset named *3D Poses in the Wild*, which contains challenging scenarios. Indeed, the dataset provides more than 51,000 frames annotated ground

truth 3D poses from 60 natural videos for further analysis. All of these sequences have been sampled at 30Hz. Unlike existing datasets, 3DPW contains videos from more complex situations, such as waiting for a bus, buying coffee, etc.

Summary. This section reviews widely used datasets of motion data. They differ in terms of the number of motion trials conducted and action types. Building a motion dataset is expensive and labour intensive. Therefore, this thesis conducts experiments based on existing benchmarks. In this thesis, **Human3.6m** and **CMU** are selected for experiment, for these two datasets are used mostly widely in recent approaches. Accordingly, this makes it possible to compare our proposed methods with recent approaches.

2.2 Motion Related Topics

Crucial research questions relating to motion data include how motions can be recognised, how motion data can be edited and reused, and how new motion data can be synthesised. With this in mind, the following review addresses the following three topics: motion synthesis, motion recognition (and motion retrieval).

Motion data are usually high dimensional and complex. Therefore, motion-related research typically strives to establish a low dimension manifold to represent motion data and then manipulate its features on the manifold. The same goes for motion prediction, in which finding a representative motion feature is an important step. For this reason, this section reviews motion recognition methods, for they provide abundant insights for motion feature engineering. Moreover, understanding motion properties is essential for the study of motion undertaken in this PhD research. To help establish an understanding of the properties of motion, this section discusses literature on motion retrieval. Finally, motion prediction is a kind of motion synthesis. Accordingly, this section reviews work on motion synthesis so as to offer ideas for motion generation as part of this PhD research.

2.2.1 Motion Recognition

How to recognise the type of a given motion is a basic problem for computer vision. This issue has attracted a lot of attention from researchers over the last few decades. In large part, this is because motion recognition has great potential benefits for developing computer interaction, producing animations, searching for sports videos, developing robotics, and analysing crowds.

The overarching idea of this research is to transform motion recognition into a form of classification. There are usually two steps in a pipeline. The first is the extraction of features from a given motion. The second is the application of classification methods from the field of machine learning, which serves to categorise features according to predefined labels. The challenge is to find a feature that is capable of representing a motion's semantic information, for there is a big gap between high-level semantic meanings and the representation of numerical features. There are two types of feature: handcraft and deep features. Through observations and experiments, researchers manually design handcraft features by setting a suitable metric for representing motion. Deep features, in contrast, can be learned automatically by training a neutral network using labelled data. Comparing these features reveals that whereas deep features can extract abundant information, handcraft features cannot. What is more, although deep features are more generalised than handcraft features, they always require a training dataset that is extensive enough to train a converged network.

Handcrafted features

Methods for presenting handcrafted features often have three steps. First, researchers explore the visual information of motion that determines the type of motion or seek to identify those features that are most discriminative in different categories of motions. These features might include moving directions, angles between legs and arms, or the speed at which limbs rotate. Second, researchers convert the visual information into mathematical form, which can be computed. This means that the iden-

tified information will be encoded in a numerical vector, which enables the computer to automatically recognise motions. Third, researchers employ a classification method to categorise numerical features using existing motion labels. Recognition errors occur in this pipeline in three ways: the identified information does not allow for effective discrimination between different motions, the numerical features cannot represent the visual information accurately, and the classification method only finds an approximate decision boundary, not the real one.

Handcrafted features typically adhere to the following ideas:

Keypose

Using keyframes to represent a whole sequence is a standard method of information compression, capable of reducing 2,000 to 3,000 frames data to 10 to 30 frames. This transforms comparisons between different motions into comparisons between images' series. This method is reasonably appropriate given that we can imagine few kick pose frames that enhance a system's capacity to recognise a kicking motion. These key poses can be regarded as entries in a codebook or dictionary, or basic chemical elements. Various types of motions can be recognised by using the way of the statistical histograms contained in these codes. Numerous researchers have proposed frameworks premised on this idea. One such framework, Sequence of the Most Informative Joints (SMIJ) encoded motions using a few skeletal joints, thereby surpassing previous motion recognition models (Ofli et al. [2012]). Through semi-supervised learning and by counting Gaussian mixture models (GMM), every pose can be described as a vector (Qi et al. [2013]).

Markerable subset of joints

In coordinating the joints that determine poses and actions, researchers also trace how joints move to help discriminate different actions. Eweiwi et al. [2014] have proposed a framework that employs a partial least square method for learning a compact representation. All of the joint features will be put in a soft bin and then combined together in a matrix, which is used for classification analysis. A genetic algorithm is presented

to determine the relevance of each joint point to how motions are classified (Climent-Pérez et al. [2012]). Using data mining techniques, Wang et al. [2012] have introduced an action representation that captures spatial and temporal cues by grouping all joints into five body parts. After extracting features, researchers use a kernel Supply Vector Machine (SVM) to classify them. In this "bag-of-pose" approach, joints are defined by kinematic chains, while separate body parts are classified using the nearest-neighbour method.

Geometric Transformation

Another approach considers the geometrical relations between joints. Müller et al. [2005] have designed a series of Boolean features to describe the spatial relations between limbs, with the aim of distinguishing motions at a semantic level. They regard two motions are similar if they have similar features. When it comes to computing, this method is less complex than the dynamic time warping (DTW method, which operates at the level of frames. Similarly, Evangelidis et al. [2014] have introduced a Fisher feature that records the geometrical relations between joint quadruples. By way of a GMM, they generated "skeletal quads" before classifying motions using a SVM model. Furthermore, Vemulapalli et al. [2014] propose a new skeletal representation based on a Lie group. Given that each limb's movement can be mapped as an element in Euclidean Group $SE(3)$, a body movement is a curve in a Lie Group. This method achieved a new peak of accuracy in motion classification on account of its precise numerical presentation of motions. Wang et al. [2012] have presented an "Action Let Ensemble" model to capture intra-class variance in human motion and human interactions with objects. The evaluation of three benchmark datasets – CMU MoCap, MSR-Action3D, and DailyActivity3D – outperform existing methods. In this model, they designed invariant features to record 3D joint positions and a limb occlusion pressure (LOP) feature for local "depth appearance". After that, they described the dynamics variation using a Fourier temporal pyramid, before applying SVM to recognise motions. Another idea is that of using the trajectory of a given motion to aid recognition. Shao & Li [2015] have used integral invariants to represent

motion trajectories and defined a kernel function to measure similarities between trajectories.

Frames spatial information

One set of approaches focuses on capturing spatial information, whether by measuring the distance variation between joints or dissimilarities between frames. Ellis et al. [2013] have proposed algorithms for balancing latency and accuracy in motion classification. They computed a feature set from frame data by choosing the current frame, preceding frame, and first frame in a sequence. If joints in frame t are linked to the same joints in frame $t+1$, a skeleton graph will be obtained naturally. Both Kerola et al. [2014] and Hammond et al. [2009] have proposed wavelet transform functions to extract features from skeleton graphs. This method has significantly improved the performance of motion recognition.

Deep Features

Given that deep learning neural networks (DNN) have achieved a significant breakthrough in image classification (Wan et al. [2014]), researchers have been inspired to apply it to motion classification. A DNN can automatically extract more informative features by training on a dataset. The deep features generated in this way far surpass their handcrafted equivalents (Yu et al. [2015]). Despite this, the DNN framework is more robust and can therefore be generalised so as to solve diverse questions.

Spatial-temporal Convolutions. These methods trained a Recurrent Neural Network (RNN)(Mikolov et al. [2010]) or 3D CNN (Ji et al. [2013]) to identify the position of 3D skeleton joints in order to capture temporal information and use a Spatio-temporal feature for classification. RNN is designed for objects that combine temporal and spatial data, such as videos and languages. Accordingly, its structure is able to accumulate information over time. It is especially effective in capturing periodic data. It is quite straightforward, therefore, to employ RNN such as LSTM networks in extracting motion features (Zhu et al. [2016]). Aperiodic motion data such as "dancing" or "kicking", however, are not appropriate objects for RNN-based analysis. In view of this, some re-

Method	Ofli et al 2014	Vantigodi et al 2013	Vantigodi et al 2014	Kapsouras et al 2014	Chaudhry et al 2013	Chaudhry et al 2013	HBRNN-L
Berkeley MHAD	95.37%	96.06%	97.58%	98.18%	99.27%	100	100

Method	Cho and Chen 2013	DURNN-T	DBRNN-T	DURNN-L	DBRNN-L	HURNN-L	HBRNN-L
HDM05	95.59%	94.63%	94.79%	96.62%	96.70%	96.70%	96.92%

Figure 2.2: *The comparison of Classification accuracy different architectures (Du et al. [2015]), the HBRNN-L beats down all the other methods in two benchmarks.*

cent research has trained separate models for different motions. This framework is redundant, however, and works only for a specific problem. What is more, as the complexity of a neural network increases, it needs more time and training data, which is not always available. Baccouche et al. [2011] have employed a 3D CNN to encode motion sequences into features, which they have used to train an RNN to classifies motions. When applied to the KTH dataset, their work is 94.4% more accurate than previous approaches. A similar work is the BLSTM-RNN that has been proposed for 3D Gesture Classification by Lefebvre et al. [2013]. Furthermore, Du et al. [2015] presented the first RNN framework for combining feature learning with classification. This framework achieved the highest accuracy in skeleton-based motion recognition on the MSR Action3D dataset, reaching 94.49%. Their architecture (HBRNN-L) divided the body into five parts and fed them in five subnets. They fused the features extracted from five hierarchical parts of the body and obtained a classification result using the softmax layer. In addition, they compared the performance of different architectures. This comparison is presented in Figure 2.2.

Fully Connected Neural Network (FCN) In approaching the hierarchical structure of human skeleton data, the fully connected neural network does not break down the hierarchy by means of convolution. Rather, it extracts informative features by means training as well. A fully connected network structure has serval layers and all of the nodes in each two adjacent layers are interconnected. Recently, researchers have

also presented an FCN framework for encoding temporal information. Bütepage et al. [2017] have designed three FCNs – S-TE, C-TE, and H-TE – which have achieved 92% accuracy in prediction and 78% accuracy in classification on the 1035 CMU database. S-TE is a symmetrical structure with five layers. Of these five layers, C-TE and H-TE change only the first. C-TE replaces it with a time scale convolutional bottleneck layer; H-TE with a hierarchy bottleneck layer. This outperforms the deep sparse autoencoder method, which is able to extract features with around 94% accuracy.

Summary. This section reviews relevant motion recognition methods. Motion Recognition is a basic classification problem. The success of approaches to it are highly dependent on feature engineering methods. The first of these approaches reviewed here are handcrafted features. However, today deep features are the most usual approach. This section surveys abundant sources of inspiration and insight for our research on how to encode motion into features. Following ongoing trends, this thesis extracts deep features for encoding input motion clips.

2.2.2 Motion Retrieval

To address the problem of motion retrieval, researchers have been developing methods that fall into the following three categories: text-based motion retrieval, motion clips-based motion retrieval, and sketch-based motion retrieval. Text-based motion retrieval requires a large number of labels, which are laborious to collect. What is more, text input is not comprehensive when it comes to describing high-dimensional motion data. Accordingly, this review focuses mainly on motion clips-based motion retrieval and sketch-based motion retrieval (SBMR). Motion clips-based retrieval always requires input in the form of motions performed by a human or puppet. SBMR requires that the user draws several free-hand sketches for motion searches. In retrieving targeted motion capture data, motion clips are the most precise information to use as input. Retrieving a target motion from a large repository is very challenging due to the complexity of data structures and the fact that semantically simi-

lar motions are not always numerically similar too. In designing motion clip-based motion retrieval frameworks, researchers usually incorporate the following steps:

1. Extracting motion features or finding low-dimensional representations of motions.
2. Proposing a metric function to measure the differences between two motions.
3. Searching for a target motion in a large database according to a similarity ranking.

Given that motion features have already been summarised in the preceding section, this section focuses on reviewing methods that aim to match similar motions and search for a target motion in large databases. Approaches to motion retrieval compare similarities in two ways. (1) By designing a distance function so as to measure the difference between two motions and select the closest motion to the query motion from a repository. (2) By applying ranking or learning methods, such as DNN and SVM. These models are trained to learn the parameters that are best suited to finding the metric. Finally, the result for the query motion can be found.

Distance-based methods A distance function can be defined manually by measuring differences between key poses, joint angles, positions, Euclidean distance, or trajectories. The most straightforward way to measure the difference between two motions is to establish the Euclidean distance between two representations. This method is not effective enough, however, because two motions can be semantically similar but have different length sequences. To cope with a tolerated level of temporal variance, the DTW and its variants have been widely applied in motion retrieval processes. (DTW is a time series technology that uses optimisation methods to match two series that differ in length.) For example, Qi et al. [2014] have represented motions by using strings, which allowed them to match the similar strings with different lengths by means of DTW. Barnachon et al. [2014] have employed dictionary methods to

represent motions by means of histograms of poses and DTW to compare features. Müller & Röder [2006] and Müller et al. [2009] have proposed a motion template representation. This involves constructing a matrix of Boolean features and using DTW in the training process. The DTW, however, has a quadratic complexity and its effectiveness depends on the consistency of semantic periods. Motion clips can be semantically similar but not temporally aligned. Recently, other time warping technologies have been proposed to address this misalignment problem, such as Isotonic Canonical Correlation Analysis (ISOCCA) (Shariat & Pavlovic [2011]), Canonical Time Warping(CTW) (Zhou & Torre [2009]), and Correlation-Optimized Time Warping(CoTW) (Etemad & Arya [2015]). But the computational complexity of these methods is high and it is time consuming to set their parameters. Another idea is to measure similarities by means of histogram matching (Barnachon et al. [2012], Eweiwi et al. [2014], Fotiadou & Nikolaidis [2014]). Barnachon et al. [2012] among others, have constructed pose histograms for a number of motions and defined Bhattacharyya distances so as to measure the differences. Fotiadou & Nikolaidis [2014] compared two pose histograms by comparing pose matrixes. Histograms can address the misalignment problem by jettisoning the rich information contained in particular dynamics.

Large scale searching methods Given that computing complexity dramatically increases in tandem the volume of data, it is necessary to have an effective method that searches for target motions in a large dataset. Previous work has proposed several searching methods. The most commonly used method is called K-nearest neighbour searching. However, it cannot be applied to real-time searches of large databases due to the increasing computing time and memory. Because motion sequences are segmented into motion clips to allow for comparison, the database size will increase to two million level. Therefore, many approaches use index techniques for speeding up searching processes. Krüger et al. [2010] Tang et al. [2014] , for example, have employed KD-Trees to structure the data. Sedmidubsky et al. [2018] have used the M-index method to search sequences that contain certain keyframes.

What is more, they have addressed the real-time searching problem by means of a fast and effective disoriented method, enabling their framework to search for the target motion in 20-million data in minutes.

Summary. This section reviews prevailing approaches to motion retrieval. Two steps are usually needed to address the motion retrieval problem. First, a representation of motion trials is required. Second, a metric is needed so as to find the best matching query. This thesis benefits from this review in that it has allowed for a better understanding of motion properties. The metric proposed for motion retrieval has inspired us to measure similarities between the predicted motion and ground truth motion in this thesis.

2.2.3 Motion Synthesis

In reusing MoCap data, a typical problem is that of new motion data, which satisfies users' needs, can be generated. For example, animators may want to create new motion by controlling trajectories or producing kicking and punching motions by offering the target location. Therefore, motion synthesis has attracted increasing attention in recent years. Motion synthesis is very challenging because the techniques need to both generate realistic motions that resemble the captured data and adhere to users' control information (such as trajectories and style). There are two groups of methods for synthesising new motions: data-driven methods and physical simulation methods. Whereas data-driven methods generate new motions by reusing existing data, physical simulation computes skeletons' positions by means of mechanical and somatological knowledge. This review focuses on data-driven methods, because they usually offer more user interactions and entail less computing complexity. This section surveys two aspects of motion synthesis approaches that relate to our work: stylised motion synthesis and deep learning-based motion synthesis.

Stylized Motion Synthesis

This method aims to generate new motions with certain styles, such as "old", "proud", and "drunk". Similar to image, researchers applied style translation to motions so as to generate new sequences by combining styles from one sample and the content from another. Stylising the motion synthesis problem is challenging for two reasons. First, the time-invariant style information is hard to extract and define; second, it is extremely difficult to automatically produce a synthesised motion that looks natural. Usually, there are two kinds of solution to these issues: style translation and parameterising style factors.

Hsu et al. [2005] have proposed a system for motion style transfers and an iterative motion warping (IMW) technique for eliminating temporal misalignments between two motions. Their system, which combines a linear time-invariant (LTI) model and post-processing techniques, is able to expose time-invariant stylistic differences and translate the style of one motion to another online. Ikemoto et al. [2009] have employed the similar heuristic technique to produce plausible synthesised motions in their motion editing system. Using Gaussian process models, their system is also able to generate a stylised motion by generalising an input edit of a short clip to an entire sequence. That said, this work has only addressed labelled, homogeneous motion data. In the case of such homogeneous data, correctly aligning two motions with different styles is vital for translation performance. Recently, another project on motion style transfer [Xia et al. [2015]] has employed a deep learning framework (instead of the linear time-invariant model) to generate realistic results using unlabelled, heterogeneous motion data.

Xia et al. [2015]'s research on motion synthesis elaborates an innovative approach to generating stylistic human motion in real time. They have also applied an online learning algorithm used to build a series of local mixtures of auto regressive models (MAR), capturing the complex relationships among motion styles. They produced local MAR models by searching for examples from the database that most closely resembled each input pose. One of the model parameters have been estimated from

the training data; the model is capable of adapting to a current pose using linear transformations. In addition, they introduced an efficient local regression model to predict the timings of synthesised poses in the output style. A demonstration has been provided for the purpose of testing this approach, in which stylistic human motions have been transferred for a wide variety of actions, such as walking, running, punching, kicking, etc.

Another method is that of parameterising style factors by using statistical models to establish the difference between motion causes according to factors such as style, gait, identity, etc. Brand & Hertzmann [2000] , for instance, have used a Hidden Markov Model (HMM) to control several stylistic parameters and thus generate various stylistic motions. Similarly, Wang et al. [2007] have found a set of low-dimensional factors with which motion styles can be parameterised. By employing nonlinear basis functions in a multifactor Gaussian process model, they identified style-specific distributions. Min et al. [2010] have presented a generative motion model with two parameters for controlling variations in "identity" and "style". As compared with the two previously mentioned approaches, this generative model can reduce visual artefacts to a remarkable degree. What is more, users can edit "style" and "identity" more explicitly, given that the style parameters are hidden in the other two approaches. Ma et al. [2010] have employed a geo-statistical model, Universal Kriging, to generate motions according to style and variation at the same time. They introduced a latent parameter for each group of joints and used Bayesian networks to analyse the relationship between user-defined styles and latent parameters. This model is able to generate unlimited variants of a certain style in real time.

Deep Learning based Motion Synthesis

In adopting the deep learning method, researchers have done much to address analogical synthesis problems, such as image and speech synthesis. In the light of this, a group of approaches have been developed to employ or design deep learning frameworks for motion synthesis too. As mentioned above, RNN, CNN, and hybrid methods have achieved a state-of-art skeleton-based motion classification accuracy when used on

benchmark databases. This section focuses on summarising the deep learning approaches applied to motion synthesis problems.

Although generating realistic motion sequences allows one to avoid a great deal of labour and expenditure, it poses a very challenging problem. Motion data can be manually segmented and combined together to generate new sequences by alignment and labelling. Afterwards humans need to manually smooth the synthesised sequences. But these processes are infeasible in an automatic process. Automatic techniques are necessary, therefore, in addressing two challenges: that of designing a method that generates motions by controlling information and of finding a method able to make sequences appear plausible.

Taylor & Hinton [2009] and Taylor et al. [2011] have used a Conditional Restricted Boltzman Machine (CRBM) model to design a stylistic motion sequence from a set of parameters. The CRBM method has a similar structure to a three-layer neutral network. Mittelman et al. [2014] have adjusted a probabilistic time-series model (SRTRBM) to learn the dependency structures in the datasets and demonstrated its performance in motion data simulation. The experiment validating this model can reduce motion prediction errors. Fragkiadaki et al. [2015] propose a recurrent network model for motion prediction, surpassing the best optical flow method. Their deep learning framework, Encoder-Recurrent-Decoder (ERD), is a variant of a typical LSTM network and it is trained jointly. Furthermore, Jain et al. [2016] have proposed an S-RNN, which considered spatial-temporal graphic information and RNN features at the same time. Their framework employs an RNN for each spatial-temporal factor and uses "factor sharing" to maintain learning capacity. Hence, this extended RNN outperforms the ERD method in motion prediction. However, this framework combined a set of RNNs to allow the model to manage significant computational complexity. Bütepage et al. [2017] have designed three fully connected deep learning frameworks composed of five layers for motion prediction. Their DNN structure is relatively simple and does not need to train jointly. As such, it outperforms the S-RNN. Besides, CNN and FCN can generalise easily and are unaffected by previous frames. For example, users prefer to edit current frames without

propagating the influence on remaining sequences. Accordingly, Holden et al. [2016] adopted the CNN structure proposed by Holden et al. [2015] rather than RNN structures. They presented a framework that uses high-level parameters to synthesise character movements, while respecting the human motion manifold trained on a large motion dataset. The network can produce realistic motion sequences from parameters, such as a target location for punching and kicking. The user can easily switch between feedforward networks, in accordance with their desired interface, without retraining the motion manifold. This is made possible by the fact that both the network and manifold are trained independently. The generated motion can be edited by performing optimisation in the space of the motion manifold. This allows the user to enforce kinematic constraints or transform the motions' style, while guaranteeing that the rewritten motion remains natural.

Summary. This section reviews previous approaches to motion synthesis methods. They usually find a motion manifold and reconstruct new motion sequences from this manifold's data. Motion prediction is among the typical motion synthesis problems. Therefore, this thesis can learn from technologies and methods of motion synthesis.

2.3 Motion Prediction Methods

This section presents a comprehensive survey of existing research works related to motion prediction. Attempts to understand and predict motion sequences have a long history. Various approaches have been painstakingly applied in previous efforts at addressing this problem. We summarise existing methods based on their different ways of modelling sequences. First are traditional approaches, which use statistic methods to model sequences. In the deep-learning era, researchers have proposed diverse models based on RNN, CNN, or GCN frameworks.

2.3.1 Traditional method

Motion data is a typical time series. Accordingly, traditional research follows the HMM statistic model [Brand & Hertzmann [2000]] , which is widely used in machine translation and speech recognition.

However, the HMM method is based on a single, discrete K state multinomial. As input information increases, this method gives rise to the exponential explosion problem. Therefore, a series of CRBM methods have been introduced in this field. Taylor et al. [2007] , for example, use binary latent variables to address both the exponential explosion problem and the task of modeling non-linear dynamics at the same time. Furthermore, Urtasun et al. [2008] and Wang et al. [2008]have proposed Gaussian processes models, which map human motions into a low-dimension latent space.

Following this approach, Taylor & Hinton [2009] have developed the CRBM by introducing a three-way interactions scheme, which simultaneously reduces the size of parameters and extends the model's ability. Another impressive work by Taylor et al. [2010] puts forward their design for an implicit mixture CRBM model, which has an enhanced ability to express multiple activities and difficult sequences.

Moreover, Sutskever et al. [2009] have studied the existing Temporal Restricted Boltzmann Machine (TRBM) model, though its exact inference ability is limited. What is more, it encounters numerous complexities when computing a Gibbs update. Based on this observation, Sutskever et al. have modified the original TRBM model by introducing a recurrent design that enhances expressiveness.

In addition to the Boltzmann Machine method, Lehrmann et al. [2014] have used a nonlinear and non-parametric method based on an expressive Markov model. As compared to existing HMM-based models, it provides more realistic results when it comes to inference.

These traditional models have achieved impressive feats in modelling human motions. However, all of them are hard to generalise to more diverse and complicated actions. Modifications to these traditional models

seem to have very limited capacity to address real, high-dimensional, and complex human movements.

Recently, a growing body of research has been dedicated to developing a deep learning framework for motion prediction. These works have significantly improved performance, both quantitatively and qualitatively. Recent deep learning approaches to motion prediction can be summarised in the following way:

2.3.2 RNN based models

Most existing network architectures adopt the RNN module so as to solve time series problems such as those of machine translation (Kalchbrenner & Blunsom [2013] Kalchbrenner et al. [2016]) and stock forecasting (Kamijo & Tanigawa [1990] Rather et al. [2015]).

The Encoder-Recurrent-Decoder (ERD) Fragkiadaki et al. [2015] is a typical encoder-decoder model. To incorporate representation learning and temporal dynamics learning, it installs an encoder and a decoder network before and after recurrent layers. The cyclical way of predicting frames iterates errors and generates unrealistic poses. They also considered a noise schedule as a means of tackling these problems. However, the noise schedule is very inconvenient in practice.

Jain et al. [2016] have proposed an SRNN model that combines multiple RNN structures with the spatial-temporal graphs to improve performance. However, it is very time consuming to train these multiple RNN models. ERD and SRNN are action-specific models which are trained separately for each action type, meaning that these models do not enrol the real strength of deep learning techniques: namely, that one can benefit from a large dataset. Furthermore, other researchers have developed multi-action models to predict motions for diverse types of actions.

Ghosh et al. [2017] have introduced a Dropout Autoencoder Network (DAE) to learn inherent human body structures along with a 3-layer LSTM to learn temporal dynamics. Martinez et al. [2017] have achieved high accuracy in both short-and long-term motion prediction by apply-

ing three changes to the typical RNN model. They have introduced a sequence-to-sequence and residual architecture, as well as a sampling-based loss plan to ensure robustness and avoid hyper-parameter tuning.

Following Martinez's work, Gui et al. [2018a] have added two discriminators to their system to improve the quality of the predicted motions and alleviate the error accumulation problem. Although these RNN architectures surpass traditional Markov methods, they are still very complex and their performance is limited when it comes to aperiodic actions. Moreover, they are prone to generate mean poses in long-term prediction.

2.3.3 CNN & FC Net & GCN based Models

Given RNN architectures' inability to extract spatial information and tendency to produce noisy data, researchers have proposed supplements, such as co-training, handcrafted spatial graphs, and adversarial networks. These strategies alleviate but do not completely solve the limitations of RNNs. Some recent approaches have therefore considered different encoder layers so as to replace recurrent layers entirely.

Li et al. [2018] have proposed a first convolutional sequence-to-sequence model for predicting motions and validating effectiveness. It has been applied to both the HM3.6M and CMU datasets. Going beyond a typical encode-decode model, they have designed a long-term and a short-term encoder for all of the input frames and short-term neighbouring frames respectively. In addition, they have added a simple two-layer fully connected network to serve as a discriminator and thus enhance the quality.

However, Bütepage et al. [2017] have argued that convolutional layers are less effective than hierarchical layers, for 2D convolutional computation does not correspond to the human body's hierarchical structure. They prove that the fully connected hierarchical layer works significantly better in prediction tasks. But the deep fully connected networks are of a high computational complexity and prone to overfitting. Moreover, the decoder's design means that their framework sacrifices qualitative performance so as to improve quantitative performance. Limb length varies during training and prediction. What is more, noise may disturb the

output sequence, for the reason that the framework lost logical coherent information between frames in the decoder.

Researchers (Du et al. [2015]; Shao et al. [2018]; Garbade & Gall [2016]) have also used a hierarchical human structure made up of five body parts. However, this hierarchical structure increased computation complexity. Therefore, in Chapter 2 we propose a new model, which contains a more reasonable convolutional hierarchical encoder structure. Indeed, this structure is much less complex than all of the other modules and achieves a state-of-the-art performance both quantitatively and qualitatively.

2.3.4 Stochastic Motion Prediction

All of the aforementioned models are deterministic in that they take human motion prediction as one solution task. However, they generate few possible results for future motions in the real case. Therefore, researchers have introduced a latent vector to generate multiple possible prediction results. To do so, they are mostly inspired by Generative Adversarial Networks (GAN) or Conditional Variational Autoencoders (CVAE).

The first class of model introduced GAN models. By randomly sampling the distribution factor, their generator synthesises multiple possible sequences. For example, Barsoum et al. [2018] have modified Wasserstein generative adversarial networks (WGAN-GP) such that they address the human motion prediction problem. Their model, HP-GAN, employs a RNN as generator and a multilayer network as discriminators.

Compared to the HP-GAN, which only introduces the latent factor as input for the generator, Kundu et al. [2019] have proposed a novel BiHMP-GAN model that also takes account of the latent factor in the conditional discriminator. Their discriminator simultaneously regresses the latent vector and supervises the output sequence. Extensive experiments show that this BiHMP-GAN has achieved significant advances in comparison to HP-GAN.

All of the existing GAN frameworks are effective at short-term pre-

diction but less effective when it comes to long-term prediction. Having noticed this, Hernandez et al. [2019] have introduced a motion inpainting framework: a conditional GAN model that produces not only body poses but also absolute positions. The GAN hosts three discriminators that cooperate to enhance the model's ability to predict long-term sequences. Their motivation to design these discriminators are learned from real human beings.

Another body of work learns from the Variation Auto Encoder (VAE) to build frameworks.

To address the limitations of results generated by existing models, Walker et al. [2017] have estimated several future poses by VAE models, which they used as initial frames to generate multiple possible video motions.

Bütepage et al. [2018] have proposed a model based on a conditional variational autoencoder, so as to predict human motions from RGB depth images. Their framework is able to extract a few samples of future possible movements to facilitate interaction between humans and robots.

All of these models, however, are prone to generating discontinuous sequences. Furthermore, Yan et al. [2018b] have designed a Motion Transformation Variational Auto-Encoder (VAE) to generate future motion sequences. Compared to the existing VAE based model for motion prediction, their model produces results of a much higher quality. It does so by using the RNN's hidden states as a conditioning variable and concatenating it with the latent vector.

Aliakbarian et al. [2020] have comprehensively studied existing stochastic motion prediction methods. They pointed out that these models consider the random latent vector through the descriminator's manner, meaning that the model can ignore the latent vector. To tackle this issue, they proposed a stochastic conditioning scheme that randomly combines the latent vector with past pose information. In the experiments, this scheme drastically improved the accuracy of human motion prediction.

Summary. This section reviews state-of-the-art motion prediction meth-

ods. Deep learning methods are presented here largely because they overwhelmingly outperform traditional methods. However, existing deep learning models for motion prediction have limitations, such as the mean pose and oversmoothing problems. This thesis therefore develops deep learning models, proposing new models for addressing the task of motion prediction.

2.4 Deep learning methods

2.4.1 Convolutional Neural Networks

The invention of CNNs has revolutionised the field of computer vision. It has achieved tremendous breakthroughs in image recognition and benefits a wide range of applications related to images. The parameter size of existing fully connected networks is growing exponentially, greatly limiting their ability to train and making them easy to get overfitting. Researchers have proposed pooling layers and convolutional operations as a way of addressing these limitations.

Researchers have learned from the organisation of animals' visual cortexes and came up with the receptive field for convolutional operations. Hubel & Wiesel [1962] first created this concept by studying the behaviour of individual cells. They revealed that the functional architecture of the visual cortex is complex and overlapping. Following this observation, researchers have introduced the receptive field in neural networks for image recognition as well. In the convolutional layers, the neurons in the same receptive field share weights, allowing for a significant reduction in the model's parameter size and the network's complexity. Normal GPU devices can therefore simultaneously train CNNs more efficiently and maintain high performance.

In 1998, LeCun et al. [1998] first proposed a graph transformer network (LeNet) to allow for the recognition of handwritten characters. It is the first network to contain basic components of CNNs: a pooling layer, FC layer, and convolutional layer. The following approaches have built on this structure in improving CNN networks.

One famous work in the development of CNNs is the AlexNet proposed by Krizhevsky et al. [2012]. AlexNet significantly outperforms all existing traditional methods on the large benchmark ImageNet LSVRC-2010, improving recognition accuracy by 10.9%. As compared with LeNet, it proposed important changes to the network, which are still widely used. First, it introduced a ReLU function (replacing a Tanh function) to address the gradient vanishing problem. Second, it proposed a regularisation strategy named Dropout to reduce the overfitting problem. Krizhevsky et al. designed the network to incorporate five convolutional layers and three fully connected layers. What is more, they proved that performance is quite sensitive to a given network's depth.

There is a trade-off problem for CNN networks. If a network goes deeper, the accuracy of a training dataset will improve. This is because the model will have a better fitting ability. The overfitting problem will increase, however, because the size and complexity of the model will also increase drastically. To deal with this problem, remarkable approaches have sought to make the network go deeper while maintaining performance when working with a test dataset. In 2014, Simonyan & Zisserman [2014] introduced a VGG network with 19 layers. It outdid what was the state-of-the-art in terms of performance. They used smaller convolutional filters (3×3 as compared with the 7×7 used in AlexNet) to reduce each layer's parameters. Moreover, they designed the single convolutional filter in the network to increase the model's nonlinearity without affecting the receptive field. Furthermore, they reorganised the combination of activated functions and convolutional layers to improve performance.

Another important challenge restricting the depths of CNN models is the limitation posed by computational resources. When the model goes deeper and becomes larger, it requires hardware with very large capacity. This is sometimes not feasible in practical scenarios of application. Therefore, Szegedy et al. [2015] have introduced an inception network (GoogLeNet) with 22 layers, which increase depth and performance while simultaneously maintaining the computational budget. Crucially, they changed the layers' cascading structure to parallel connection. Multi-

scale convolutional filters were used at the same layer and the output features were then concatenated. Furthermore, they used a 1×1 convolutional filter to reduce the parameter numbers as well. Finally, their model only has 1/12 of the parameter size of AlexNet. This was achieved by significantly reducing the fully connected layers.

Researchers observed that a model's fitting ability does not always increase in tandem with depth. This is called the degraded problem. ResNet (He et al. [2016]) have addressed this problem perfectly, achieving the best performance yet in the tasks of ImageNet detection and localisation, COCO detection, and COCO segmentation. In comparison with the existing network, which learns the unreferenced functions $H(x)$ directly, ResNet learns the residual function $F(x)$, in which $H(x) = F(x) + x$. This residual link allows for identity mapping and maintains the parameter size. This simple idea has proven very effective in improving performance and addressing the gradient vanishing problem.

In a study of ResNet, Huang et al. [2017] have come up with Densely Net, which densely connects all layers so as to increase the use of feature maps. In this way, the model reuses feature maps efficiently and is able to reduce its parameter size too. Inspired by their work, we have also introduced dense links in our framework for motion prediction as a means of enhancing performance.

Summary. This section presents the history of CNNs, reviewing the most representative CNN models to establish references for our work. Chapter 3 describes how we have invented a CNN-based encoder to capture the constraints of the human body and temporal relationships simultaneously. In addressing the motion prediction problem in Chapter 4, we get inspiration from one of the most advance CNN models, the DenseNet.

2.4.2 Generative Adversarial Networks

Generative Adversarial Networks (GAN) play an important role in the forms of generation involved in computer vision, such as image generation and translation, music generation, and time series synthesis. Motion

prediction is a typical synthesis task. Accordingly, researchers have borrowed ideas from GAN models to generate plausible motion sequences.

Goodfellow et al. [2014] have introduced GAN as a means of image generation. In using an adversarial approach as a means of generation, they proposed a network with two components. The first subnetwork is a generator, G, which synthesises fake images. Another subnetwork is a discriminator, D, which is designed to distinguish between fake and real images. During the training process, generator G is trained to fool discriminator D, while the discriminator D continues improving its ability to discern fake from real images. Optimising the two components together forces the generator's distribution, P_g , to become similar to the real data's distribution, P_r. As a form of synthesis, this adversarial method has been hugely successful in the processing fields of both images and natural languages. This has prompted an increasing number of researchers to propose various GAN models in recent years. Some of these variants include a remedy strategy designed to address the instability problem during the GAN training process. Some deployed GAN in real-life applications. We summarise the most important of these variant GAN models in the following.

Radford et al. [2015] have investigated how constraints and convolutional layers can be used for stable GAN training. Their model is called DC-GAN (Deep convolutional GANs). By introducing a convolutional generator, they have caused this GAN framework to learn more effective representations of images. As a result, it significantly improved performance in the task of image classification. Furthermore, Radford et al. visualised the features during GAN learning so as to understand the inner mechanism of the GAN framework in more detail. However, given its limited capacities, their DCGAN model is only effective when it comes to low-resolution images belonging to a fairly limited range of types.

In some classes, the DCGANs perform well. They do significantly worse, however, on multi-class datasets such as ImageNet. To address this problem, Zhang et al. [2019] have introduced a Self-Attention Generative Adversarial Network (SAGAN), which combines the advantages

of different attention mechanisms to capture long-range dependency information. The DCGANs' limitations in relation to ImageNet, in their opinion, derive from the convolution operator, which only captures local information in a receptive field of a limited size. The self-attention strategy can work, therefore, as a complement to the conditional GAN framework. This makes it possible to extend the model's ability to capture long-term dependencies. Finally, SAGAN outperformed existing methods by a significant margin on various comparison experiments.

One of the biggest disadvantages to GANs is that they are very unstable and hard to converge during training. Usually, they need to be carefully set to achieve a balance between the generator and discriminator in training the model. Arjovsky et al. [2017] have noticed this drawback and proposed a Wasserstein GAN framework to address the problem. They introduced the Wasserstein distance and designed a new loss function for the original GAN. Elaborating the new network drastically reduced the model dropping phenomenon (which is characteristic of GANs) and made GAN training easier and more stable.

While recent approaches have achieved excellent performance in generating images, the creation of high-resolution and multi-category images has been held back – that is, until Brock et al. [2018] proposed a BigGAN framework. Their work is based on modifying SAGAN and introducing the interesting "truncation trick". By expanding ideas, this research has surpassed all state-of-the-art models on ImageNet. Moreover, the "truncation trick" offers an explicit way of controlling the trade-off between fidelity and variety in image generation. BigGAN's performance also greatly benefits from extensive training, an increased batch size and more channels.

Beside the stability of training processes and fidelity in generation, another prominent issue in GAN development is that of how to control the generation results in an explicit way. Usually, GAN has a random noisy z as input for the generators so as to produce various results. For example, the synthesised digital numbers from the Modified National Institute of Standards and Technology (MNIST) dataset can have dif-

ferent writing styles and angles. Owing to the random latent vector z, the output style and angles are random too. The human user, however, might hope to control number's output style or angles by setting certain parameters. To achieve this, Chen et al. [2016] have proposed an InfoGAN, which is able to learn interpretable disentangled representations in completing unsupervised tasks.

This GAN's benefits have made the introduction of many image generation applications possible. One of the most interesting of the GAN's applications is that of the image style transfer. A major problem in image style transfers is a lack of paired training data. Therefore, Zhu et al. [2017] have proposed a CycleGAN that addresses this issue. The idea at the heart of their approach is simple. Assuming that the mapping function from domain X to domain Y is G and the inverse mapping function denoted as F, Zhu et al. introduced a cycle consistency loss to enforce $F(G(X)) = X$. This strategy enhanced the quality of generated images and can be trained with unpaired datasets.

Summary. This section reviews the most typical GAN frameworks. GAN has been developed to make it more stable, easy to train, and ensures high resolution. GAN is one of the most important deep learning frameworks. As such, it is widely employed to address the synthesis problem. In Chapter 3 of this thesis, discriminators are used to supervise the fidelity of synthesised motion sequences.

2.4.3 Graph Neural Networks

Recently, researchers concerning with GNNs have become increasingly active, on account of their marked ability to tackle irregularly shaped data. Convolutional operations on graphs have also been investigated. Researchers designed Graph Convolutional Networks (GCN) from spectral (Henaff et al. [2015]; Duvenaud et al. [2015]; Li et al. [2016]; Berg et al. [2017]) and spatial (Denton et al. [2014]; Niepert et al. [2016]) perspectives. The human body can be regarded as a natural graph. Accordingly, recent approaches have been successful in using GCN to approach skeleton-related problems. Here, we focus on reviewing existing

GCN-based work for problems relating to human poses.

Yan et al. [2018a] introduce a Spatial-Temporal Graph Convolutional Network (ST-GCN) to improve motion classification accuracy and address its problems. As compared with existing CNN-based models for motion, their spatial-temporal graph is more capable of exploiting relationships between joints. Two types of edges have been designed for this framework, the "spatial edge" and the "temporal edge". The graph convolution is used to integrate the spatial and temporal information for feature extraction. On this basis, Yan et al.'s ST-GCN has reached a new level in terms of performance on heavily benchmarked datasets.

Furthermore, Li et al. [2019] have considered the relationship between edges and human joints, as well as enhanced the classification accuracy. In comparison with ST-GCN, their work extends the idea to exploit more relationships among joints. In addition to the spatial and temporal edges, they considered the implicit "action links" and "structure links" in the model. For example, hands and feet are not connected by a spatial edge. In the walking scenarios, however, hand and feet movements are quite related to each other. They are therefore connected by "action links". In this way, this model contributes toward the task of recognising motions. Li et al.'s proposed network, the Actional-structural Graph Convolution Network (AS-GCN), outperforms all existing approaches by a large margin.

Unlike the indirect graph used for the skeletons, Shi et al. [2019] have introduced a direct graph for the human body so as to exploit dependencies between bones and joints. The indirect graph usually models information relating to joints and bones separately, ignoring directional relationships between joints. Therefore, Shi et al. proposed a directed graph neural network (DGNN) to exploit the dependencies more fully. Their two-steam framework is significantly better than existing work.

GNNs have not only achieved significant success in motion recognition; they have also inspired a new task: that of human pose regression. Zhao et al. [2019] have designed a SemGCN operation for 3D human pose regression. Existing GCN operations lack the ability to capture global

information, for their receptive fields are usually very small. To address this problem, Zhao et al. have introduced a channel-wise weighting vector to the original graph convolution operation. The new SemGCN operation captures a more implicit local and global relationship of the human body to enhance the accuracy on evaluation.

Mao et al. [2019] have proposed a GCN-based approach to motion prediction. This approach has predicted future sequences within the trajectory space instead of the pose space, outperforming all existing RNN-based, FCN-based, and CNN-based approaches. Drastically different in relation to previous work, Moa et al. have formulated the spatial-temporal problem in a new way. They first use a discrete cosine transform (DCT) to encode the trajectory of each joint into feature vectors. The feature vectors are then regarded as node features on the graph. They designed a GCN framework with residual links so as to learn the output feature vectors. Finally, the DCT transforms the output features back into trajectories. This novel framework has significantly outperformed state-of-the-art works.

Summary. This section reviews important recent approaches to GCN. To further investigate the GCN-based model for motion prediction, in this thesis, we describe how we constructed a densely connected GCN-based model. In comparison to the work of Mao et al. [2019], our model is more effective in learning feature maps and less likely to overfit and experience gradient vanishing problems.

Chapter 3

Efficient Convolutional Hierarchical Autoencoder

3.1 Motivation

Inspired from the striking breakthrough of deep learning technology, the researchers introduced neural networks to model motion dynamics. Recently, Fragkiadaki et al. [2015] and Martinez et al. [2017] regard motion prediction as a sequence-to-sequence problem [Sutskever et al. [2014];Gehring et al. [2017]] similar to machine translation. Therefore, the researchers propose recurrent neural networks (RNN) and variants, to address motion prediction problems. Although these RNN models achieve promising results, they have intractable limitations such as: high computational complexity, less effective for the aperiodic motions, and error accumulation. Essentially, the basic assumption of RNN models ignored the major difference between motion data and language data: motion data contains not only the temporal information but also spatial information, which resembles the complicated human body structure and mechanical restrictions.

Li et al. [2018] proposed a CNN based autoencoder system which tries to capture both the temporal dynamics and the human body structure constraints. However, the 2D convolutional kernel is not able to precisely capture the human body hierarchical structure information as

well. To be more specific, convolutional kernel is designed for images because a rectangular patch of images usually represents a meaningful signal. However, the human body has a completely different spatial constraints, which the joints are linked in an articulated tree structure. Bütepage et al. [2017] demonstrate in their experiments that the convolution structure is not as effective as the fully connected structure and the hierarchical layers can improve the performance significantly. However, their work has limitations where the limbs' length may change in the predicted motion. Meanwhile, the fully connected networks have high computational complexity and are prone to overfitting. Moreover, their decoder results in a shaky and noisy output sequence because the logical dependency between adjacent frames is completely lost.

Considering the limitations above, a novel Convolution Hierarchical Autoencoder (CHA) framework is proposed to address the motion prediction problem. A new encoder network incorporated Hierarchical structure with 1D convolution layers is designed. Therefore, it captures the tree structures of the human body and temporal information at the same time. Compared to RNN, FCN and CNN networks, it has much lower computational complexity and a very small size but converges faster and more effective. This convolution hierarchical module is adopted for motion prediction and classification task. Extensive experiments are conducted in CMU data and HM3.6 datasets to demonstrate the CHA model's ability to improve the motion prediction performance and break the long-term error accumulation limitation.

This framework contributes to the field in four-fold: (1) A new Adversarial Hierarchical Autoencoder (AHA) model is proposed with a much lower computational complexity to address the motion prediction problem. (2) A convolution Hierarchical module is proposed which captures the human body tree structure and temporal information at the same time. (2) A novel D-loss function is introduced to address the mean pose and error accumulation problem. (3) Two hierarchical discriminators are proposed for this framework. (4) One general model is trained for diverse actions contained in CMU/ HM3.6m and generates high fidelity predicted motion sequences both for short-term and long-term predic-

tion.

3.2 Overview of Methodology

An overview of the Convolutional Hierarchical Autoencoder (CHA) model is shown in Figure 3.1. The seed motion clip (input) will be propagated in an autoencoder system to generate the future frames. In the autoencoder, two encoders with Convolutional Hierarchical modules are designed, which consist of three hierarchical layers and one fully connected layer, to extract both the temporal and spatial information in the human dynamics. One long-term encoder is used to extract the information of the whole input sequence, while the short-term encoder is used to extract the information of C ($C = 20$ in this experiment) neighbouring frames close to the current frame. The deep features generated by the two encoders will be concatenated into one feature. The decoder utilized two fully connected layers to restore the deep feature to a single human pose. In this decoder, a residual link is also incorporated to avoid the gradient vanishing problem. Therefore, the decoder will produce the output sequences recursively. To address the mean pose problem and accelerate convergence, a D-loss function is designed to assign a series of diminished weights to the frames.

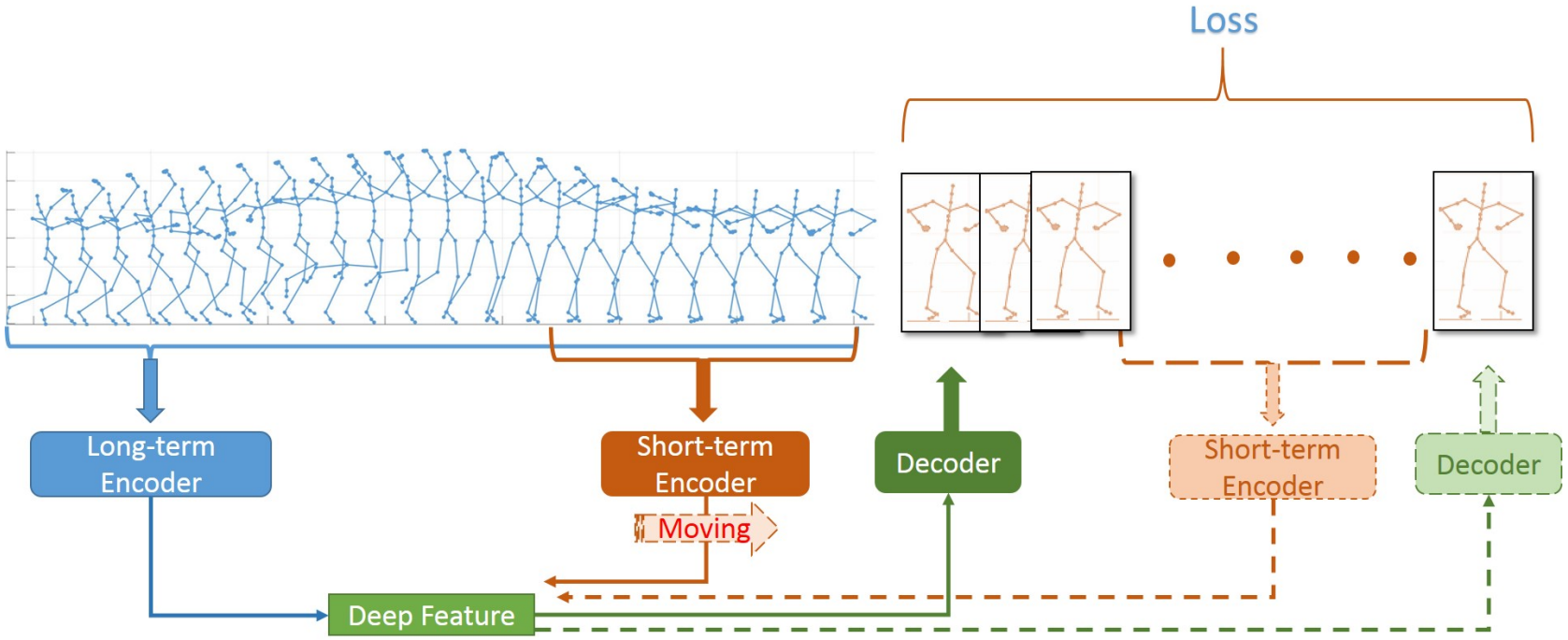

Figure 3.1: *The architecture of the Convolutional Hierarchical Autoencoder Model. The orange and green solid boxes are the initial state of the short-term encoder and decoder. They will produce the future frame recursively. The orange and green dashed boxes are the final stage of short-term encoder and decoder. They will stop moving after predicting all the frames.*

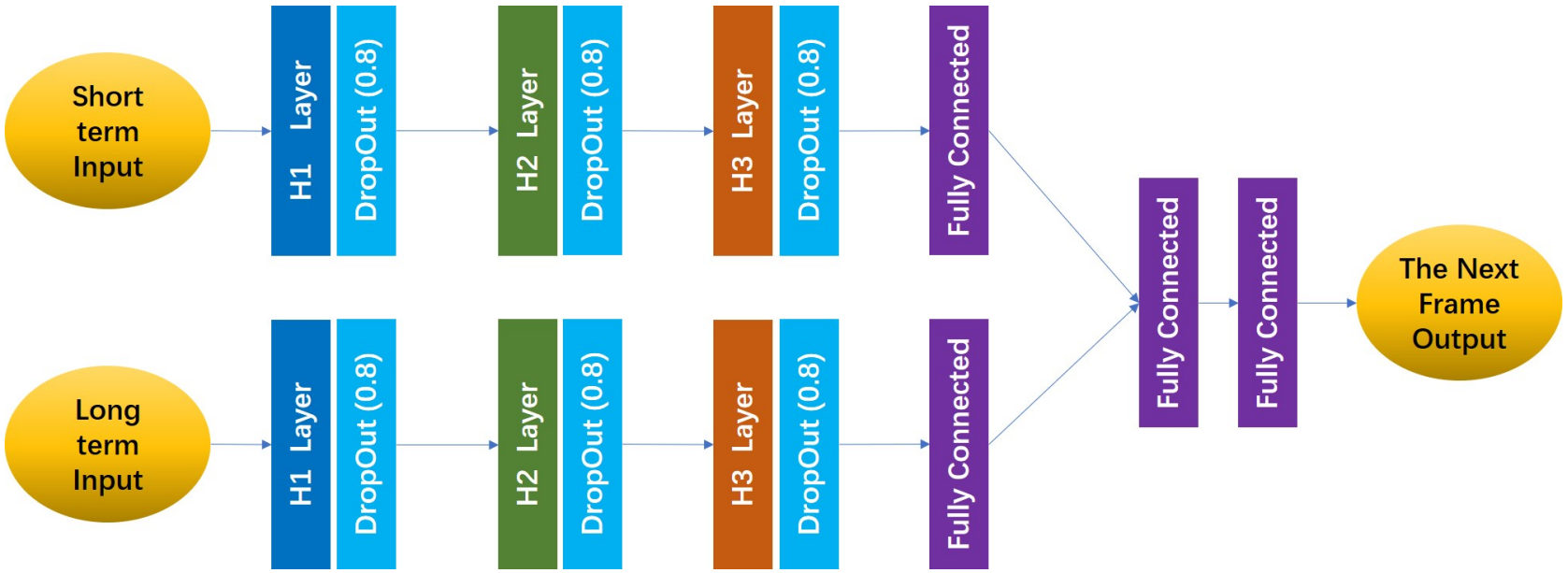

Figure 3.2: *The diagram of the network architecture.*

3.3 The mathematical formulation

The human motion data in this chapter refers to the mocap 3D skeleton data with joints. A sequence of motion data can be written as $X = \{f_1, f_2, ..., f_t, ..., f_n\}$. $f_t = (a_1, a_2, ..., a_K) \in \mathbf{R}^{3K}$ denoted the frame at time t, K is the number of joints and a_i is the exponential map representation [Bregler & Malik [1998]] of joints. Similar to the standard procedure [Gui et al. [2018a]; Jain et al. [2016]; Li et al. [2018]; Martinez et al. [2017]], the exponential map is normalized so that a_i only contains the relative joints' rotations without the global rotation and translation. Therefore, the motion prediction problem can be formulated in a mathematical way. A set of motion clips are denoted as $\mathfrak{A} = \{X^i, i = 1, ..., N\}$. In the training stage, if the input motion $X^i = \{f_1, f_2, ..., f_t, ..., f_n\} \in \mathfrak{A}$ has a length n, the m future ground truth frames are denoted as $X_f = \{f_{n+1}, f_{n+2}, ..., f_{n+m}\}$. This algorithm aims to generate the future frames $\hat{X} = \{\widehat{f_{n+1}}, \widehat{f_{n+2}}, ..., \widehat{f_{n+m}}\}$, which makes the distance function $D(X_f, \hat{X})$ as small as possible.

3.3.1 The representations of the rotations of limbs

3D rotation matrix

In the 2D rotation situation, every rotation can be denoted by a real 2×2 special orthogonal matrix, which is given as follows:

$$\begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \tag{3.1}$$

Similarly, every 3D rotation can be represented as a real 3×3 special orthogonal matrix. For example, a counterclockwise rotation around the $z-$ axis with degree θ can be formed as:

$$R_z(\theta) = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

Therefore, every rotation in the three dimensional space can be denoted as a 3×3 rotation matrix $R(\hat{n}, \theta)$. $\hat{n}$ is the axis of rotation, and θ is the counterclockwise rotation degree. All the 3D rotation matrixes consist of a Lie group $SO(3)$.

Then the rotations of each limbs in the motion sequences can be derived out. A frame is determined by the coordinate of K joint points, that is $F_t = \{J_t^1, J_t^2, ..., J_t^K\} \quad \forall t \in \{1, ..., T\}$. For every limb r in F_t, it can be regarded as a vector $\hat{r}_t = (x_t, y_t, z_t) = J_t^i - J_t^j$ in $\Re^3$, where i is the parent joint and j is the child joint. So in the $t+1$-th frame, the limb's position can be written in the form (Bregler & Malik [1998]) by homogeneous coordinates:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \\ 1 \end{bmatrix} = R_t \cdot \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix} \tag{3.3}$$

Therefore, the limb vector $\hat{r}_{t+1}$in F_{t+1} can be determined:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \\ 1 \end{bmatrix} = R_t R_{t-1} ... R_1 \cdot \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{3.4}$$

Euler Angle

Every 3D rotation can be one rotation θ from an axis $\hat{n}$, or obtained from a three times rotation, which uses $x-$axis, $y-$axis, $z-$axis in turns. Therefore, the following formulation can be deduced:

$$R_t = R_x(\alpha_t) R_y(\beta_t) R_z(\gamma_t) \tag{3.5}$$

Similar to the 2D rotation, positive rotation was identified when $\alpha, \beta, \gamma > 0$, and negative rotation was identified when $\alpha, \beta, \gamma < 0$. For a human skeleton, their limbs are rotated with restriction so that the interval of

α, β, γ is much less than the 360 which is freedom. The α, β, γ are called *Euler Angles* to represent a 3D rotation. Note that for one rotation, the Euler Angle coefficients are not unique. But one tuple of Euler Angle can determine a unique 3D rotation.

The unique rotation axis $\hat{n}$ has relationships to the Euler Angle α, β, γ:

$$\hat{n} = \pm(sin(\beta/2)cos\alpha, sin(\beta/2)sin\alpha, cos(\beta/2)) \tag{3.6}$$

The Euler Angle exists by mathematic proof, which can be referred to [$http://scipp.ucsc.edu/\ haber/ph216/rotation_12.pdf$].

Quaternion & Exponential Maps

In 2D rotation case, xy plane can be also regarded as a complex plane. Therefore, every rotation is accosiated with a exponential expression:

$$z = x + iy = |z|(cos\theta + isin\theta) = |z|e^{i\theta} \tag{3.7}$$

Similarly, every 3D rotation matrix can be expressed in the exponential way. The exponential map on $SO(3)$ refers to the following mapping function:

$$\begin{cases} exp : \mathbf{so}(3) \to SO(3) \\ A \mapsto e^A = \Sigma_{k=0}^{\infty} \frac{1}{k!} A^k = I + A + \frac{1}{2}A^2 + ... \end{cases} \tag{3.8}$$

In the equation, $\mathbf{so}(3)$ is a group of skew-symmetric matrixes. Every 3D rotation can be defined by the exponential formulation uniquely. Assuming that there is a unit vector $u =< x, y, z >$ in R^3. The 3D rotation around unit u by the angle θ can be formulated as:

$$exp\left(\theta \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}\right) \tag{3.9}$$

The above equation can be computed to obtain the final 3D rotation

matrix. (Please ref to 3DRotationWIKI)

$$\begin{bmatrix} 2(x^2-1)s^2+1 & 2xys^2-2zcs & 2xzs^2+2ycs \\ 2xys^2+2zcs & 2(y^2-1)s^2+1 & 2yzs^2-2xcs \\ 2xzs^2-2ycs & 2yzs^2+2xcs & 2(z^2-1)s^2+1 \end{bmatrix} \tag{3.10}$$

Where $c = cos\frac{\theta}{2}$ and $s = sin\frac{\theta}{2}$.

The benefits of using the exponential map instead of the Euler Angle is that the exponential map expression is unique. Usually, R^3 maps to $SO(3)$ is always with singularities. However, these kind of inevitable singularities in the exponential map are often avoidable.

3.4 The convolutional hierarchical module (CHM)

Different from the existing work, this network contains neither a typical 2D CNN nor RNN components but a Convolutional Hierarchical Module (CHM), which is particularly designed for human motion data (Figure 3.3). This module has a network topology similar to the human body tree structure, and every node in the network is a 1D convolutional layer. The network topology is better at preserving the special human body hierarchical constrains in deep features. The RNN did not exploit the spatial information in motion data enough and the human body has more sparse nodes and an articulated structure compared to images. Therefore, this CHM module is more capable to capture both the local and global features of motion data than RNN and CNN. 1D convolutional layers are utilized to capture the temporal information along with the spatial constraints and reduce the model complexity at the same time.

The convolution hierarchical module consisted of four layers, three convolutional hierarchical layers $H1$, $H2$ and $H3$, which are illustrated in Figure 3.3, and a fully connected layer. Intuitively, $H1$ stands for the information of each joint link in the human body. And $H2$ extracts

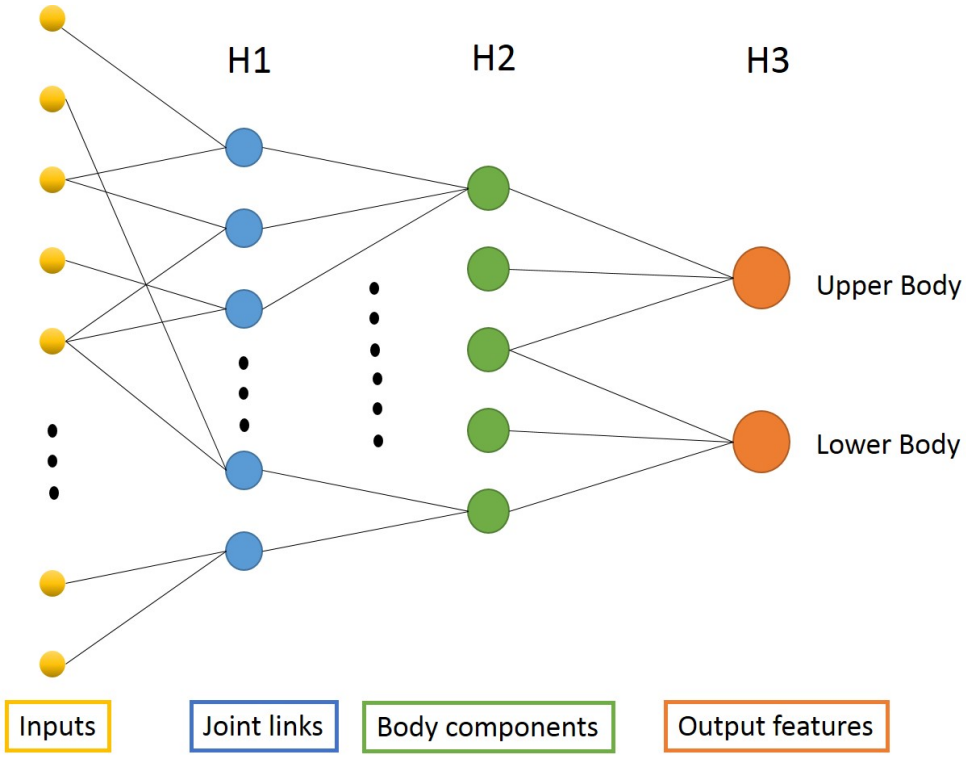

Figure 3.3: *The convolution hierarchical layers in this framework. The first layer contains the same number of neurons M_2 as the input frame feature dimension. Then the neurons from adjacent joints are linked together to one neuron in the secondary layer. Two neurons are linked together if and only if the related a_i and a_j represent the data from two adjacent joints in the human skeleton. After that, the two neurons input feature will be concatenated as a sub matrix and operated by 1D conv. Therefore, the output of each neuron will be $1 \times M_1$ (M_1 is the input frame number). In the same way, the output of H_1 will be sent in to the H_2 layer. H_2 consists of five neurons and H_3 consists of two neurons. All the nodes' outputs have explainable semantic meanings.*

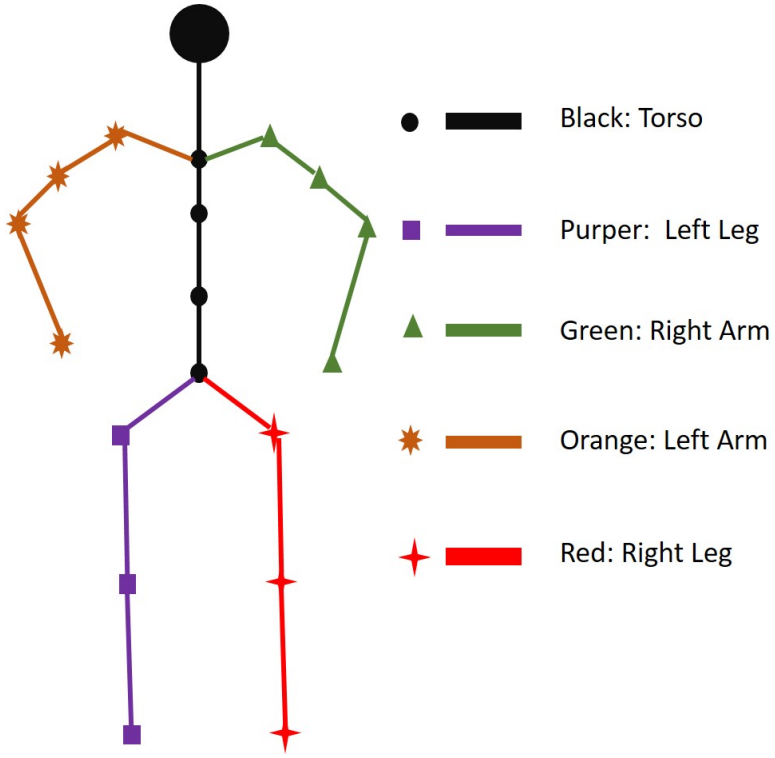

Figure 3.4: *The five body components. Note that the joints in the figure do not equal to the same number of joints used in experiments because different datasets have different skeletons. The skeleton of CMU and H3.6M are demonstrated in the following section.*

information separately for five parts of the human body, which is illustrated in Figure 3.4. $H3$ extracts the deep representations for two parts of the human body, Upper and Lower. Therefore, the deep features in the layers have semantic explanations as well.

The input sequence is a matrix of dimension $M_1 \times M_2$, referring to M_1 frames and each frame is represented by a feature of M_2 dimension. Each element from this feature comes from an exponential representation of a joint. The input matrix will be reconstructed by separating the connected joints into a group. For example, joints J_1 and J_2 are connected by one bone, then the related input feature $M_1 \times j_1$ and $M_1 \times j_2$ are combined together as a sub matrix $M_1 \times (j_1 + j_2)$. A 1D convolution kernel will operate each sub matrix along the time axis M_1. If there is L joint links in the human body tree, then the $H1$ layer consists of L nodes. The output feature of each node represents each joint link.

$H2$ and $H3$ are built in the same way. In the layer of $H2$, the features of joint links are concatenated into five sub matrixes related to five body parts (Figure 3.4). The sub matrixes are operated by the 1D-convolution as well. $H3$ only has two nodes, which represent the upper body action and lower body action respectively. The setting details are shown in Table 3.1.

Table 3.1: *The architecture of the Adversarial Hierarchical Autoencoder*

Scope	Layer	NodesNumber	NodeType	KernelSize	FilterNum	Stride	Pad
Encoder	$H1$	20(22 for CMU data)	Conv1D	5	64	1	'Same'
			LeakyRelu(0.2)	-	-	-	-
	Dropout(0.8)	-	-	-	-	-	-
	$H2$	5	Conv1D	5	256	1	'Same'
			LeakyRelu(0.2)	-	-	-	-
	Dropout(0.8)	-	-	-	-	-	-
	$H3$	2	Conv1D	5	320	1	'Same'
			LeakyRelu(0.2)	-	-	-	-
	Dropout(0.8)	-	-	-	-	-	-
	FullyConnect	256	-	-	-	-	-
Decoder	FullyConnect	256	-	-	-	-	-
	FullyConnect	54(70 for CMU data)	-	-	-	-	-

This design structure brings two benefits. Firstly, the hierarchical network structures blend the human body constrains in feature extraction. Compared to the CNN model [Li et al. [2018]], the CHA model captures more precise spatial structures and generates more meaningful deep features. Since the semantic meaning of deep feature is a very important research question, the CHA model will contribute to the researches by producing more explainable deep features. Secondly, the 1D convolutional layers can capture the temporal information in a much more efficient way than RNN models and CNN models. The less complex model can prevent overfitting problems better. It is the most efficient network for motion modelling based on our knowledge. The model complexity will be discussed in detail in Section 3.12.

3.5 The autoencoder framework

A deep sparse autoencoder system [Li et al. [2018]] is widely used for synthesis problems. In this work, an autoencoder system is used as the generator to produce $\hat{X}$.

Two encoders are employed in this autoencoder system. The first encoder network aims to map the whole n frame in the input sequence $X = \{f_1, f_2, ..., f_t, ..., f_n\}$ into a deep feature V_l which extracts the long term information such as the action type, the global tendency and the motion style. The second encoder aims to map the adjacent C frames $X_C^t = \{f_{t-C+1}, f_{t-C+2}, ..., f_t\}$ of the current frame f_t into another deep feature V_s^t which contains the short term information to predict the frame f_{t+1}. Finally, V_l and V_s^t is concatenated to be one feature V^t and propagated into a decoder. The two encoders are denoted as functions E_l and E_s respectively.

$$V_l = E_l(X|W_l), W_l \text{ is the parameters of } E_l \tag{3.11}$$

$$V_s^t = E_s(X_C^t|W_s), W_s \text{ is the parameters of } E_s \tag{3.12}$$

$$V^t = [V_l, V_s^t] \tag{3.13}$$

The predicted motion sequences are generated recursively. The X_C^n will encode the information to inference the first future frame $\widehat{f_{n+1}}$, Therefore, the window of the short term encoder can move to X_C^{n+1}, where

$X_C^{n+1} = \{f_{n-C+2}, f_{n-C+3}, ..., \widehat{f_{n+1}}\}$

So that the next frame $\widehat{f_{n+2}}$ of $\widehat{f_{n+1}}$ can be produced.

By following this methodology, all of the m future frames can be generated by this scheme. A decoder with two fully connected layers is used in the pipeline to restore the human pose from the low dimension representation V^t. The mapping function of the decoder is denoted as D, so that the $\widehat{f_{t+1}}$ can be written as follows:

$$\widehat{f_{t+1}} = D(V^t|W_D), W_D \text{ is the parameters of } D \tag{3.14}$$

In recursive cases, the residual link usually works better than producing the next status directly. So the decoder is designed with a residual link as well. The formula of equation 3.14 is rewritten as:

$$\widehat{f_{t+1}} = D(V^t|W_D) + \widehat{f}_t, W_D \text{ is the parameters of } D \tag{3.15}$$

Therefore, the three networks are combined together to encode the spatial-temporal information of long seed motions and relate the short neighbouring motions into a deep feature representation. Then the decoder maps the deep features back into the human body joints, relative to the rotation exponential map representations and produces the future frames iteratively.

3.6 The objective function

Inspired by Fragkiadaki et al. [2015], Jain et al. [2016], Martinez et al. [2017], Li et al. [2018], the l_2 loss function of two motions is used, which measures their difference by summing up the mean squares error (MSE) of the Euler angles of all frames.

The objective function of the Convolutional Hierarchical Autoencoder model is:

$$min\, \mathscr{L}_E(\hat{X}, X_f) + \lambda \|W\|_2 \tag{3.16}$$

The predicted motion is written as:

$$\hat{X} = \{\widehat{f_{n+1}}, \widehat{f_{n+2}}, ..., \widehat{f_{n+m}}\} \tag{3.17}$$

And the ground truth of the predicted motion represents the following:

$$X_f = \{f_{n+1}, f_{n+2}, ..., f_{n+t}, ..., f_{n+m}\} \tag{3.18}$$

The original Euclidean distance loss function is defined as:

$$\mathscr{L}_E(\hat{X}, X_f) = \|\hat{X} - X_f\| = \sum_{t=n+1}^{n+m} \|\hat{f}_t - f_t\|_2 \tag{3.19}$$

So the objective function can also be written in a frame level:

$$min \sum_{t=n+1}^{n+m} \|\hat{f}_t - f_t\|_2 + \lambda(\|W_E\|_2 + \|W_D\|_2) \tag{3.20}$$

Where the λ controls the balance of different loss sources. The W_E and W_D are the parameters of the encoder and decoder in the Hierarchical Convolutional Model. The increase in pramameters can improve the fitting accuracy, but the accuracy on the test dataset may decrease. Therefore the term $\lambda(\|W_E\|_2 + \|W_D\|_2)$ is introduced with an l_2 regularizer to prevent overfitting.

Remarks

A D-loss function is designed in this thesis to accelerate convergence. The idea is to assign gradually diminishing weights for each frame in the sequences. From the experiments, this D-loss function does not affect the result, but it prevents to produce the mean pose. We designed a diminishing weight $0 < \eta < 1$. The error between the generated frame and the ground truth frame is denoted as $\|\hat{f}_t - f_t\|_2$. We assumed that the importance of the error will decrease when the predicted frame goes further since the initial error will propagate during iteration. Therefore, the error of each frames will be assigned different weights and be summed up, then normalized to obtain the final loss:

$$\mathscr{L}_A(\hat{X}, X_f) = (\sum_{t=n+1}^{n+m} \eta^{t-n} \|\hat{f}_t - f_t\|_2)/(\sum_{t=n+1}^{n+m} \eta^{t-n}) \tag{3.21}$$

where $\eta \in (0, 1)$ is a parameter very close to 1. During the training process, the networks will put more efforts to decrease the error of earlier generated frames since they have more iteration steps which will amplify the initial error. The frame level distance $\|\hat{f}_t - f_t\|_2$ can use l_2, l_1 or the geodesic loss [Gui et al. [2018a]] as well.

3.7 Implementation Details

The input sequence has a length of 50 frames so that the first hierarchical layer has an input vector of 50×54 (50×70 for CMU) and the predicted sequences have a length of 25. The short-term encoder is set to have an input length of 20. A small batch size 16 and a learning rate of $5e^{-5}$ is used. For the parameter η, which aims to control the error generation during the experiment, 0.9 is found to be a good value during experiment experience. A NVIDIA GPU 1080Ti is used and the full model is trained in Tensorflow [Abadi et al. [2016]].

3.8 Evaluations

To validate the CHA model, extensive experiments of motion prediction are conducted on the existing benchmarks- H3.6M and CMU motion

dataset. Generally, the previous motion prediction research follows the same standard of experiments, which tests their models for short-term prediction and long-term prediction. The state-of-the-art baselines of motion predictions are included as the comparison. The experiments' results demonstrated that the prediction accuracy of the CHA model beats down the state-of-art baselines on diverse actions of H3.6M and CMU. Meanwhile, it is illustrated that the CHA model also produces more plausible human-like movements than baselines. Besides, the efficiency of the CHA model is also discussed in terms of computational complexity and parameters in Section 3.12.

3.9 Benchmarks

Following the standard comparison [Li et al. [2018]], motion prediction experiments are provided on the two widely used motion benchmarks, H3.6M [Ionescu et al. [2014]] dataset and CMU motion dataset.

H3.6M- This dataset is the largest motion dataset which provides 3.6 million 3D human poses and corresponding images in three kinds of formats. The 3D skeleton format is used, which has 32 joints in total to represent the human body structure. In the data process, each frame is recorded as the relative rotation of each joint, which is mathematically converted into an exponential map. There are performances of 11 professional actors in certain scenarios such as discussion, smoking, etc. Six actors' trails and all the 15 types of actions are selected in the experiment. Each type of action has two performance trials among 3000-5000 frames. Therefore, 180 trials of H3.6M are selected in total. The train set is five times to the test set as the previous work. Four types of actions, walking, smoking, eating and discussion are the most widely evaluated in these motion prediction mechanisms. The experiments are provided not only on these four but also on all the other actions as well. In the Figure 3.5, the skeleton structure of H3.6M is presented.

CMU- This dataset has a wider range of action types than H3.6M. It gives out a large amount of skeleton based MoCap data around 2605 sequences which represent 6 categories and 23 subcategories (actions).

Contrast to H3.6M, each action type contains different amounts of trials and their 3D-skeletons consist of 38 joints (The skeleton used in Li et al. [2018]). There are various sports and physical activities included, such as basketball, soccer and jumping. The same subsets are selected by Li et al. [2018] under the prescriptions, in which the action type should be a single type and should contain enough training trials. Finally, eight action types are selected and each of them contains more than five trials. Besides walking, all the other action types have 5 trials for training and 1 trial for testing. In the Figure 3.6, the skeleton structure of CMU (used in Li et al. [2018] and this thesis) is presented.

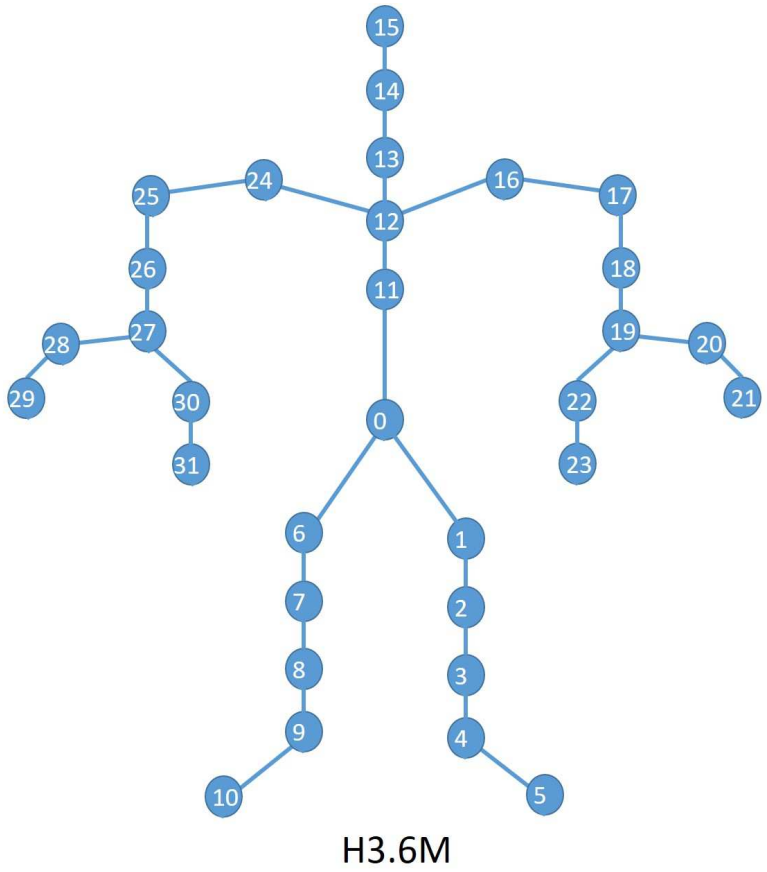

Figure 3.5: *The whole body of the H3.6M. The dimension will be reduced after normalization.*

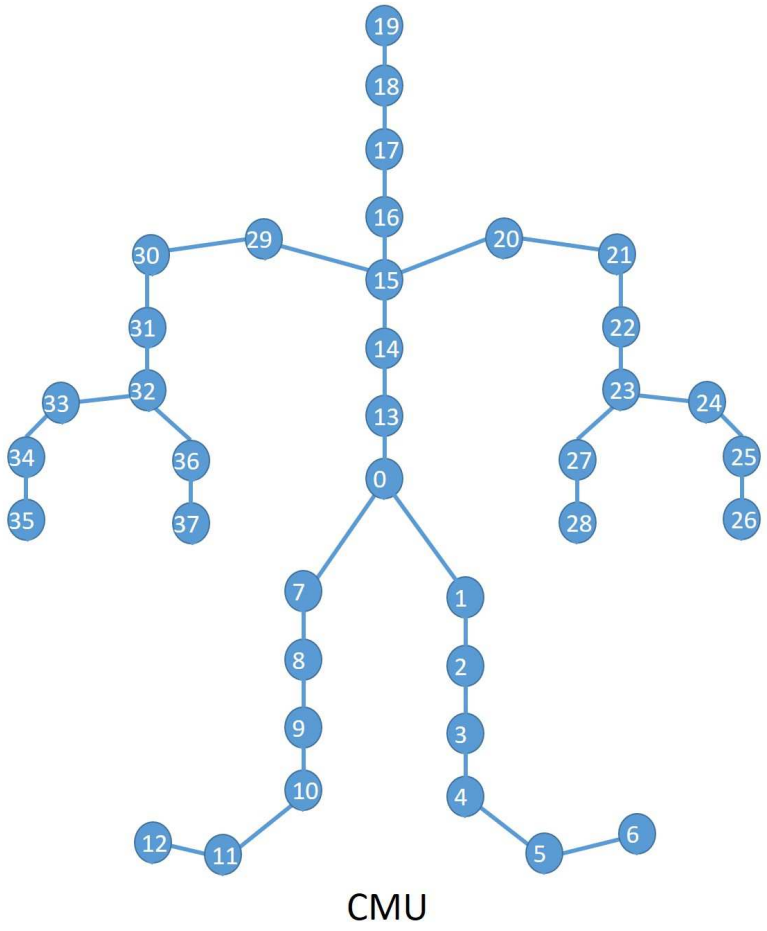

Figure 3.6: *The whole body of the CMU. The dimension will be reduced after normalization.*

The two datasets are pre-processed in the same way. Due to data normalization being an important factor affecting the network's performance, all the human pose data is normalized into mean values resulting to 0, and standard deviation resulting to 1. Therefore, the root point of all poses are set at the same point and the global orientation of the whole body is fixed. After that, every normalized human pose is represented by a 54 dimension feature for H3.6M and a 70 dimension feature for CMU. All the trials are down sample to 25Hz, so that every seed motion clip has 50 frames, equivalent to 2 seconds of information. The seed motion clip will be imported in the long-term encoder, and the neighboured 20 frames of the currently predicted frame will be imported in the short-term encoder. Note that, although all the models present their evaluation results by Euler Angles, their loss functions do not calculate Euclidean differences of Euler Angles directly. For example, Martinez et al. [2017] and Gui et al. [2018a] use losses which calculate the exponential maps and the orientation groups' difference between the predicted motion and the ground truth respectively. In this thesis, the Euclidean difference of the exponential maps in the loss function are considered as well.

3.10 Baselines

Five of the-state-of-the-art deep learning models are included in the comparative evaluation. Those are the following:

1. Encoder-Recurrent-Decoder model for human motion recognition and prediction (ERD) [Fragkiadaki et al. [2015]]
2. Dubbed the Dropout Autoencoder LSTM (DAELSTM) [Ghosh et al. [2017]]
3. Residual Recurrent sequence to sequence Model for human motion modelling (RRNN) [Martinez et al. [2017]]
4. Convolutional Sequence to Sequence Model for Human Dynamics (CNNHD) [Li et al. [2018]]

The experiments of CNNHD and RRNN from their public implementa-

tion on GitHub are reproduced. However, the remaining models do not public their implementation code. Therefore, their results are quoted from the existing publications [Martinez et al. [2017]] for error comparison. The quality of the predicted sequences with CNNHD are also compared. For the short-term prediction, the results reported from CNNHD [Li et al. [2018]] are quoted. For the long-term prediction, the CNNHD model from their public code are implemented and their model is trained with the same settings in Li et al. [2018]. Because all these models follow the standard procedure of data processing and the same evaluation, this comparison is impartial.

3.11 Evaluation Methods

From a practical perspective, the predicted motion should be accurate and look plausible at the same time. Therefore, the CHA model is evaluated in three aspects: complexity, qualitative and quantitative performance. All the short-term results of diverse action types are presented and also the more challenging long-term prediction accuracy.

1. To evaluate the efficiency of the CHA model, the discussions in Section 3.12 are conducted. Usually, RNN models have a higher complexity compared to CNN models. The CHM's complexity and size with the-state-of-the-art CNN based model are compared. Li et al. [2018] proposed a CNN based Autoencoder system with a convolutional encoding module (CEM) which outperformed all the RNN models.

2. To evaluate the prediction performance, the average mean square error (MSE) of the Euler angles between the predicted motion and ground truth are provided. Short term prediction for different durations: 80ms (2 frames), 160ms (4 frames), 320ms (8 frames), 400ms (10 frames) are conducted. In addition, the long term predictions for four durations: 560ms (14 frames), 720ms (18 frames), 840ms (21 frames), 1000ms (25 frames) are also conducted. The results and analysis are shown in section 3.13 and 3.14 with Table 3.2-3.7.

3. To evaluate the quantitative performance of the CHA model, the representative prediction sequences are illustrated in Figure 3.7–3.9 following the same settings in Bütepage et al. [2017], Gui et al. [2018a], Li et al. [2018], Martinez et al. [2017].

3.12 Size and Speed

CNN has the lowest model complexity compared to the FCN and RNN structures. But the design of the convolution hierarchical network greatly reduced the model complexity compared to CNN. The complexity of CEM and the CHM are calculated here.

For a normal convolutional layer with a kernel size $K_1 \times K_2$, it takes an input $M_{1in} \times M_{2in} \times C_{in}$ and generates an output of $M_{1out} \times M_{2out} \times C_{out}$. Here, M_{1in} and M_{2in} represent the width and height of the input tensor. The same for M_{1out} and M_{2out}. C_{in} and C_{out} are the numbers of input and output channels respectively. For simplification, it is considered that all the convolutional layers have the same output size of input and strides equal to 1. Therefore, $M_1 \times M_2$ both for input and output are denoted.

CEM consists of three 2D convolution layers and one fully connected layer. CHM consists of three hierarchical layers and one fully connected layer. The computational complexity A and the parameters' number B for each layer will be calculated separately and then summed them up.

3.12.1 Comparison of computational complexity

The computational cost of a 2D convolutional layer in CEM is calculated as:

$$A = M_1 \times M_2 \times C_{out} \times ((K_1 \times K_2 + 1) \times C_{in} + (C_{in} - 1)) \qquad (3.22)$$

The number of nodes are denoted in each hierarchical layer as h_i, then the computation cost of one H_i layer in CHM is calculated as:

$$A_i = h_i \times M_1 \times C_{out} \times ((K_1 \times K_2 + 1) \times C_{in} + (C_{in} - 1)) \qquad (3.23)$$

In the equations, $(K_1 \times K_2 + 1)$ is the calculation in one patch of convolution. Here, the bias weight are considered. $((K_1 \times K_2 + 1) \times C_{in} + (C_{in} - 1))$ means the total calculation to produce a point on the output. $M_1 \times M_2 \times C_{out}$ means the number of output points.

If the same kernel value for two models are set, the ratio of computational complexity of each convolution layer can be deduced from the equations above :

$$A_i^{CEM} : A_i^{CHM} = M_2 : h_i \tag{3.24}$$

For the last fully connected layer, the bias for simplicity is not included. CHM is more efficient than CEM as well:

$$\begin{aligned} A_F^{CEM} &= M_1 \times M_2 \times C_{out} \times F \\ A_F^{CHM} &= M_1 \times C_{out} \times F \\ A_F^{CEM} &: A_F^{CHM} = M_2 \end{aligned} \tag{3.25}$$

Since h_i is much smaller than M_2, the CHM module has much lower computational complexity under the same settings. For experiment convenience, a CHM with slightly different parameters is used, which are written in Table 3.1. The computational cost ratio of two networks will be obtained:

$$\frac{\text{Computational cost of } CEM}{\text{Computational cost of } CHM} \approx \frac{1{,}240M}{302M} \approx 4.1 \tag{3.26}$$

The computational cost of the model affects the total running time of the model. Usually, researchers and data engineers need to train a deep learning model dozens or hundreds of times to modulate it. Therefore, the time efficiency of the CHA model allows them to implement ideas and tasks faster.

Table 3.2: *The short-term prediction error of four action types on H3.6M dataset*

	Walking				Eating				Smoking				Discussion			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
ERD Fragkiadaki et al. [2015]	1.30	1.56	1.84	N/A	1.66	1.93	2.28	N/A	2.34	2.74	3.73	N/A	2.67	2.97	3.23	N/A
DAELSTM Ghosh et al. [2017]	1.00	1.11	1.39	N/A	1.31	1.49	1.86	N/A	0.92	1.03	1.15	N/A	1.11	1.20	1.38	N/A
RRNN Martinez et al. [2017]	0.33	0.56	0.78	0.85	0.26	0.43	0.66	0.81	0.35	0.64	1.03	1.15	0.37	0.77	1.06	1.10
CNNHD Li et al. [2018]	0.33	0.54	0.68	**0.73**	0.22	0.36	0.58	0.71	0.26	0.49	0.96	**0.92**	0.32	0.67	0.94	1.01
CHA	**0.27**	**0.45**	**0.65**	0.74	**0.20**	**0.34**	**0.53**	**0.66**	**0.26**	**0.48**	**0.89**	0.93	**0.28**	**0.62**	**0.85**	**0.91**

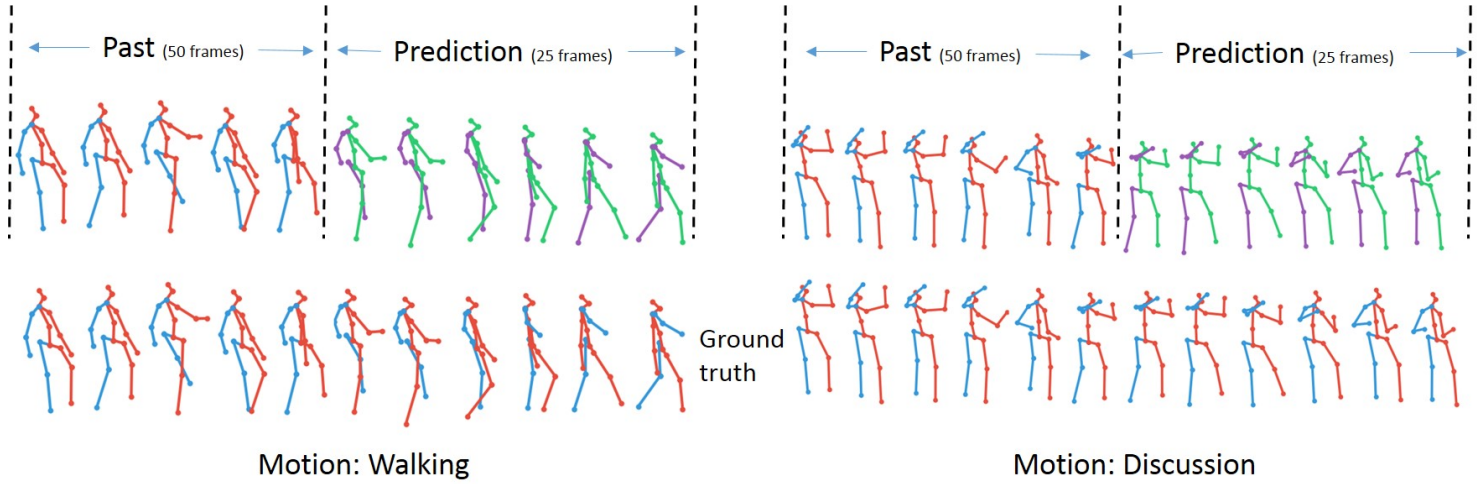

Figure 3.7: *The illustrate of the prediction result of 1000ms on H3.6M dataset. The blue and red skeletons are groundtruth frames. The green and purple skeleton on the top is the prediction result of CHA model.*

Table 3.3: *The short-term prediction error of 12 action types on H3.6M dataset*

	Directions				Greeting				Phoning				Posing			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
RRNN	0.44	0.70	0.86	0.97	0.55	0.90	1.34	1.51	0.62	1.10	1.54	1.70	0.40	0.76	1.37	1.62
CNNHD	**0.39**	**0.60**	0.80	0.91	**0.51**	**0.82**	**1.21**	**1.38**	**0.59**	1.13	1.51	1.65	0.29	0.60	**1.12**	**1.37**
CHA	0.40	0.62	**0.79**	**0.88**	0.53	0.87	1.28	1.44	0.60	**1.12**	**1.51**	**1.64**	**0.27**	**0.56**	1.16	1.41
	Purchases				Sitting				Sittingdown				Takingphoto			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
RRNN	0.59	0.83	1.22	1.30	0.47	0.80	1.30	1.53	0.50	0.96	1.50	1.72	0.32	0.63	0.98	1.12
CNNHD	0.63	0.91	1.19	1.29	0.39	**0.61**	**1.02**	**1.18**	0.41	**0.78**	1.16	1.31	**0.23**	**0.49**	0.88	1.06
CHA	**0.60**	**0.84**	**1.10**	**1.15**	**0.40**	0.64	1.03	1.21	**0.41**	0.79	**1.15**	**1.30**	0.26	0.51	**0.80**	**0.93**
	Waiting				Walkingdog				Walkingtogether				**Average**			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
RRNN	0.35	0.68	1.14	1.34	0.55	0.91	1.23	1.35	0.29	0.59	0.86	0.92	0.43	0.75	1.12	1.27
CNNHD	**0.30**	**0.62**	**1.09**	**1.30**	0.59	1.00	1.32	1.44	0.27	**0.52**	**0.71**	**0.74**	0.38	0.68	1.01	1.13
CHA	0.32	0.63	1.12	1.31	**0.54**	**0.90**	**1.24**	**1.38**	**0.26**	0.53	0.75	0.80	**0.37**	**0.66**	**0.99**	**1.11**

3.12.2 Comparison of the parameters

The number of parameters of a convolutional layer is calculated as:

$$B = K_1 \times K_2 \times C_{in} \times C_{out} \tag{3.27}$$

The parameter number of an H_i layer is calculated as:

$$B_i = K_1 \times K_2 \times C_{in} \times C_{out} \times h_i \tag{3.28}$$

The parameter number of the fully connected layers are:

$$\begin{aligned} B_F^{CEM} &= M_1 \times M_2 \times C_{out} \times F \\ B_F^{CHM} &= M_1 \times C_{out} \times F \\ B_F^{CEM} &: B_F^{CHM} = M_2 \end{aligned} \tag{3.29}$$

The total cost of a network is to sum up the cost of each layer:

$$\text{Total parameter} = \textstyle\sum_{l=1}^{L} B_l \tag{3.30}$$

Because B and B_i are significantly smaller than B_F^{CEM} and B_F^{CHM}, the CHM will have a smaller size under the same settings as well. To be more specific, the precise number of parameters are calculated with the setting in Table 3.1:

$$\frac{\text{The parameters in } CEM}{\text{The parameters in } CHM} \approx \frac{177M}{11M} \approx 15.1 \tag{3.31}$$

It is an intractable problem in deep learning models that more data is required when the model has more parameters. Due to motion data being expensive and inconvenient to obtain, the samples for each type of action is limited. It is observed that complex models are prone to overfitting on the small amount of motion data. Therefore, the CHA model alleviates this problem and it is more suitable for small sample learning.

3.13 H3.6M experiment results

There are four types of actions: walking, eating, smoking and discussion which are commonly used as benchmarks in comparison. Therefore the CHA model's results of the short term prediction of these four actions are presented in Table 3.2. The accuracy of other baselines is compared in this table as well. Note that all the results come from the model which is trained generally for the loss of 15 actions in the long-term. The CHA model beats down all the results of four actions in terms of 80ms, 160ms, 320ms and 400ms, except one. For the 400ms walking prediction, the CHA model actually achieved 0.735 which is almost the same to 0.73 of the CNNHD model. For the 80ms and 160ms walking prediction, the CHA model improved 0.06 significantly. Even for the aperiodic action discussion, the CHA model outperforms all the other baselines completely with a maximum of 0.8 Euler Angle error reduction.

For a more general comparison, the 12 remaining actions' results are displayed in Table 3.3. Compared to ERD, DAELSTM and RRNN the CHA model almost outperforms on every action. However, compared to CNNHD, the CHA model shows a different preference of actions. Half of the actions improved, but half decreased. Therefore, the average error is calculated to demonstrate a fair comparison. It shows that the CHA model achieved the best average error in terms of all prediction lengths. The general model CHA even beats the other action specific models like ERD.

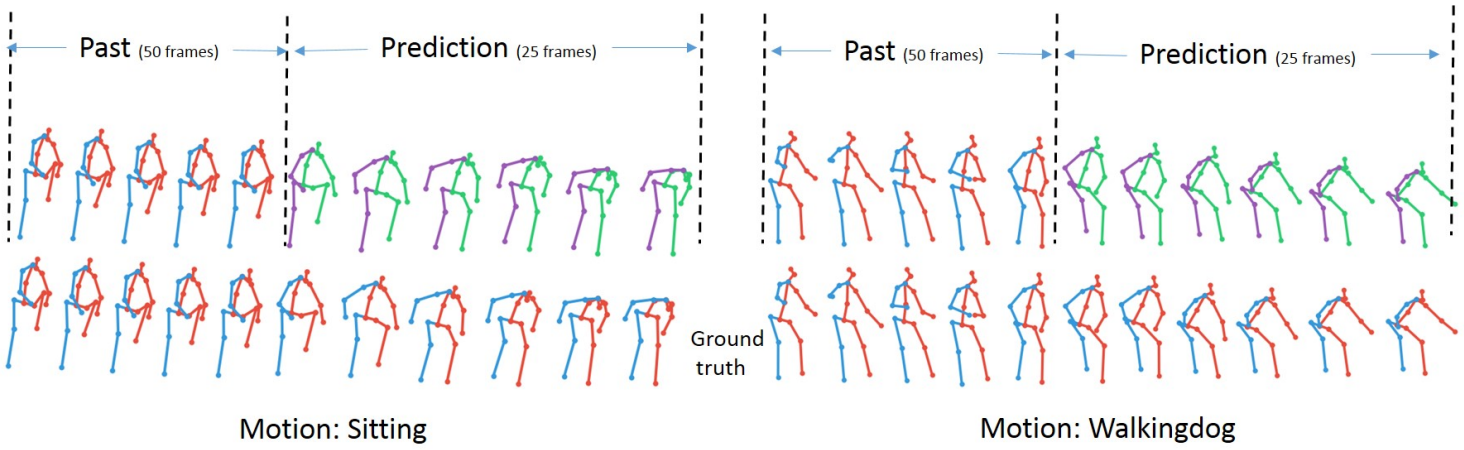

Figure 3.8: *The illustration of the prediction result of 1000ms on H3.6M dataset. The blue and red skeletons are ground truth frames. The green and purple skeletons on the top are the prediction result of the CHA model.*

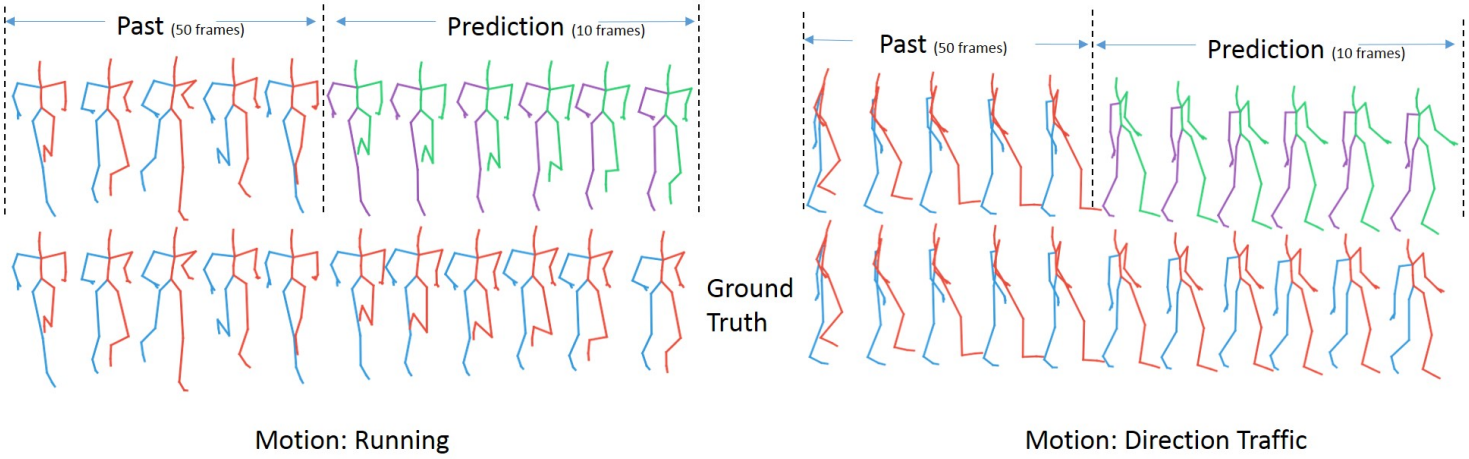

Figure 3.9: *The illustration of the prediction result of 400ms on CMU dataset. The blue and red skeletons are ground truth frames. The green and purple skeletons on top represents the prediction results of the CHA model.*

Table 3.4: *The long-term prediction error of 15 action types on H3.6M dataset*

	Walking			Eating			Smoking			Discussion		
milliseconds	560	840	1000	560	840	1000	560	840	1000	560	840	1000
CNNHD	0.86	0.92	0.97	0.86	1.09	1.33	1.04	1.38	1.70	1.34	1.71	1.79
CHA	**0.84**	**0.91**	**0.92**	**0.82**	**1.02**	**1.21**	**1.02**	**1.35**	**1.66**	**1.29**	**1.65**	**1.72**
	Directions			Greeting			Phoning			Posing		
milliseconds	560	840	1000	560	840	1000	560	840	1000	560	840	1000
CNNHD	**0.96**	**1.33**	1.40	**1.69**	1.82	1.84	1.59	**1.82**	**1.86**	1.89	2.30	2.50
CHA	0.97	1.34	**1.40**	1.72	**1.82**	**1.83**	**1.58**	1.91	2.03	**1.72**	**2.15**	**2.40**
	Purchases			Sitting			Sittingdown			Takingphoto		
milliseconds	560	840	1000	560	840	1000	560	840	1000	560	840	1000
CNNHD	1.61	1.88	2.39	**1.30**	1.60	1.71	1.58	2.02	2.19	1.08	1.24	1.32
CHA	**1.55**	**1.84**	**2.33**	1.34	**1.59**	**1.67**	**1.50**	**1.89**	**2.05**	**1.06**	**1.19**	**1.27**
	Waiting			Walkingdog			Walkingtogether			**Average**		
milliseconds	560	840	1000	560	840	1000	560	840	1000	560	840	1000
CNNHD	1.66	2.24	2.36	1.71	1.87	1.91	**0.86**	1.00	1.36	1.33	1.61	1.77
CHA	**1.65**	**2.23**	**2.34**	**1.65**	**1.86**	**1.90**	0.88	**0.99**	**1.32**	**1.31**	**1.58**	**1.74**

Table 3.5: *The short-term prediction error of 8 action types on the CMU dataset*

	Basketball				Basketball Signal				Directing Traffic				Jumping			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
RRNN	0.50	0.80	1.27	1.45	0.41	0.76	1.32	1.54	0.33	0.59	0.93	1.10	0.56	0.88	1.77	2.02
CNNHD	0.37	0.62	1.07	1.18	0.32	0.59	1.04	1.24	0.25	0.56	0.89	1.00	**0.39**	**0.60**	**1.36**	**1.56**
CHA(H)	**0.37**	**0.61**	**0.97**	**1.07**	**0.27**	**0.50**	**0.89**	**1.05**	**0.24**	**0.49**	**0.79**	**0.92**	0.41	0.66	1.46	1.66
	Running				Soccer				Walking				Washwindow			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
RRNN	0.33	0.50	0.66	0.75	0.29	0.51	0.88	0.99	0.35	0.47	0.60	0.65	0.30	0.46	**0.72**	**0.91**
CNNHD	**0.28**	**0.41**	**0.52**	0.57	0.26	0.44	**0.75**	**0.87**	0.35	0.44	0.45	0.50	0.30	0.47	0.80	1.01
CHA(H)	0.29	0.44	0.53	**0.56**	**0.23**	**0.42**	0.81	0.95	**0.33**	**0.43**	**0.45**	**0.50**	**0.28**	**0.44**	0.74	0.94

Table 3.6: *The long-term prediction error of 8 action types on the CMU dataset*

	Basketball				Basketball Signal				Directing Traffic				Jumping			
milliseconds	560	720	840	1000	560	720	840	1000	560	720	840	1000	560	720	840	1000
CNNHD	1.75	2.20	2.39	2.51	1.47	1.62	1.65	1.67	1.51	**1.68**	**1.78**	**1.93**	1.93	**1.92**	2.20	2.10
CHA	**1.21**	**1.37**	**1.47**	**1.54**	**1.30**	**1.44**	**1.47**	**1.52**	**1.49**	1.70	1.80	1.97	**1.91**	1.94	**2.20**	**2.10**
	Running				Soccer				Walking				Washwindow			
milliseconds	560	720	840	1000	560	720	840	1000	560	720	840	1000	560	720	840	1000
CNNHD	**0.51**	**0.48**	**0.54**	0.59	**1.11**	**1.27**	**1.32**	**1.46**	**0.55**	**0.65**	**0.75**	0.79	1.17	1.23	1.40	1.36
CHA	0.56	0.59	0.57	**0.55**	1.19	1.36	1.39	1.47	0.58	0.68	0.77	**0.78**	**1.17**	**1.21**	**1.37**	**1.32**

The long-term prediction result is shown in Table 3.4. Because the ERD, DAELSTM and RRNN models did not provide their long-term accuracy and CNNHD achieved the best performance of them all, it is only compared here using the CNNHD model. The results regarding the long term prediction of CNNHD model are obtained from their public implementation and use the same setting in their paper. From Table 3.4, the CHA model improves all the performance of diverse actions especially in the long term. In the most challenging long term task, motion prediction for 1000ms, the CHA model outperforms significantly on almost every action.

The visualization is shown in Figure 3.7 and 3.8. Both for periodic motions like walking and aperiodic motions like discussion, the CHA model produces plausible and high fidelity predictions which looks very similar to the ground truth. Besides, the CHA model avoids generating mean poses in long term prediction like RNN models (Figure 3.8).

3.14 CMU experiment results

In order to demonstrate the CHA model's generalization ability, this model is trained from CMU dataset which includes eight actions as well. Only one existing work, CNNHD, provides their results on the CMU dataset. The prediction error of these actions are given out in Table 3.5 and Table 3.6. The ability of short term prediction is compared firstly. It shows that almost all the action improved their accuracy in some terms. More than 60% of the CHA model's results outperform the CNNHD. In the 80ms and 320ms of Running, the errors are 0.285 and 0.525 precisely, which are very close to CNNHD. The average error of these eight actions are also calculated, and they demonstrate the CHA model achieved a better performance than CNNHD.

For long term prediction, the CHA model outperforms more than half of the results of the CNNHD model. Similar to the H3.6M dataset, the CHA model produces an unbalanced improvement. Therefore, the average error of the 8 actions are provided. The results in Table 3.7 demonstrate that the CHA model outperforms the CNNHD model in

terms of all length of prediction. It improves significantly for long term prediction around 0.10. The experiments show that the CHA model has the ability to alleviate the error accumulation in long term.

The visualization of the data is shown in Figure 3.9. Compared to the Human3.6M dataset, the prediction results of the CMU are not always completely similar to the ground truth. The purple skeleton data at bottom is the result of CNNHD model. This model predicts the left arm and left foot more accurately than the CNNHD model.

3.15 Summary

A novel convolutional hierarchical module which combines 1D convolutional layers in a tree structure is designed. This module is utilized as an encoder and built up as an autoencoder system. The CHA model can extract the temporal and spatial information effectively and greatly reduce the model's computational complexity and size. It is demonstrated that the CHA model outperforms the state of the art accuracy on the Human3.6M and CMU benchmark by extensive experiments. In the experiments, the CMU prediction is not completely similar to the ground truth and the CHA model demonstrated an unbalanced preference of actions. In the future, this research plans to explore the data augmentation method on the CMU and introduce more expressive concatenated features of the three H layers.

Table 3.7: *The Average error of all types of actions in the CMU dataset*

milliseconds	80	160	320	400	560	720	840	1000
CNNHD	0.32	0.52	0.88	0.99	1.25	1.38	1.50	1.55
CHA	**0.30**	**0.50**	**0.83**	**0.96**	**1.18**	**1.29**	**1.38**	**1.41**

Chapter 4

Densely Connected GCN model

In the last chapter, it explored how to design a model to reduce the computational complexity and model size while retaining the performance of prediction at the same time. During the PhD research period, Graph Neural Networks have been proposed for modelling human poses and achieved trenmendous success since they are usually light weight and effective. Following the trend, this chapter develop the idea to use GCN models for motion prediction problem and further improved the state-of-the-art work.

4.1 Motivation

Recently, Graph Neural Networks(GNN) attracts increasing attention and achieved a significant margin of improvement on human motion tasks [Yan et al. [2018a]; Li et al. [2019]; Mao et al. [2019]]. The reason is because the human body is a natural graph, i.e. the joints are nodes of the graph and connectivity is defined by the limbs. The existing GCN based work [Mao et al. [2019]] significantly outperforms those of the LSTM or CNN based models for motion prediction. However, the existing GCN based models are limited in performance as an emerging topic. When the GCN layers go deeper, the gradient is prone to vanish.

Moreover, the features extracted from earlier layers contain the different scales of graph information. The GCN layers usually have a receptive field with size 1, i.e. only operating on the 1-nearest node. However, the impact of the 1-nearest feature will diminish when the layers go deeper. On the contrary, the nearest joints on the human body are vitally important for prediction movement.

To address these limitations in a simple but effective way, a new GCN based framework is proposed for motion prediction which connects all the GCN blocks directly. Therefore, the output feature of each GCN block skips the middle layers and jumps to the final layers. In details, the DGCN model formulates the graph as the 3D skeleton of the human body by following the previous work. Then the trajectory of each joint is encoded and fed as node input features. Each GCN block consists of two GCN layers and LeakyRELU layers. The first layer is used to preserve the feature map size. Similar to Huang et al. [2017], the different scale of features are concatenated. Therefore, if the DGCN model has N GCN blocks, and the first GCN block has an output feature with size F, then the input features for the last GCN blocks will have size $F \times (N-1)$ and $\frac{N(N+1)}{2}$ links between blocks are built in total in the DGCN model.

Compared to the aforementioned GCN model proposed by Mao et al. [2019], the DGCN model requires almost the same level of parameters. But it significantly enlarged the feature maps utilization and increase the impact of earlier layers' feature map. Moreover, this densely connected structure makes the model for motion prediction easier to train and able to go deeper. Another important factor decreasing the performance of motion prediction is that the models are prone to be overfitting due to the motion data amount being small. However, the DGCN model has a regularizing effect and LeakyRELU layers reduce the overfitting on training.

In conclusion, the contribution of this work comes from two aspects:

1. A new Densely GCN based model (DGCN) is proposed to address the problem of motion prediction. It reuses the multi-scale feature maps from every block to enlarge the receptive field and reduced

the oversmoothing problem.

2. Extensive comparison experiments are conducted on the standard benchmarks for motion prediction, which are Human3.6M, the CMU motion capture dataset and 3DPW. The model is evaluated both from the angle and 3D position aspect, and it surpasses the state-of-the-art performance on all datasets.

4.2 Methodology

In this section, the details of the methodology are given out to address the motion prediction task. Firstly, the mathematic formulation of the motion prediction is described. Then the definition of the DGCN model's graph networks is depicted. After that, the structure of the DGCN networks and how it has been densely connected are shown in Fig 4.1.

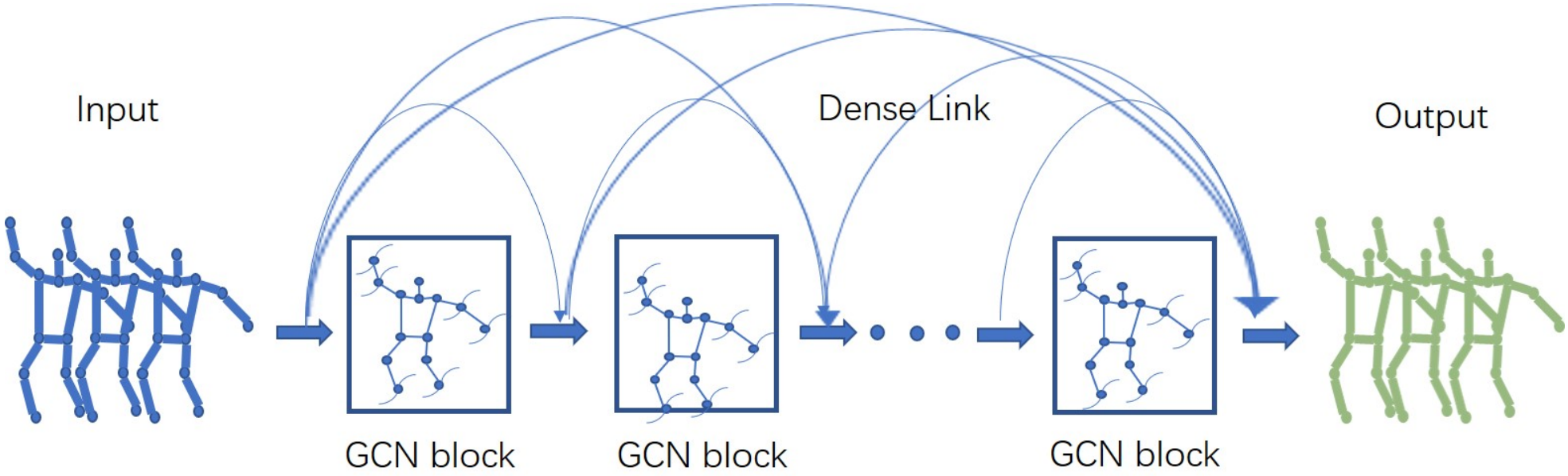

Figure 4.1: *The overview of the DGCN model. Dense link shows how the feature maps propagate. The input of each node of the graph is the DCT of the trajectory.*

4.3 Problem Formulation

In this section, the mathematic formulation details of the motion prediction problem are provided. Assuming that the sequence of the 3D skeleton is $X_{1:T} = [x_1, x_2, ..., x_t, ..., x_T]$, t is the time step that ranges from 1 to T so that x_t is the related pose at time t. Every pose x_t contains K joints and each joint can be represented as a 3D position (x, y, z) or (α, γ, β) in angle space (the data usually removes the global rotation and translation). Therefore, $x_t \in \mathbf{R}^{3\times K}$ are denoted.

This task aims to predict the future sequences $X_{T+1:T'}$. The ground truth future sequences are denoted as $GX_{T+1:T'}$ and the synthesised sequences from the model are denoted as $SX_{T+1:T'}$. Therefore, the objective of the problem is to minimize the error of $\|GX_{T+1:T'} - SX_{T+1:T'}\|$ and also make the $SX_{T+1:T'}$ look realistic like the real human actions. Most of the traditional methods predict the future sequences by generating poses recursively. However, this type of method suffers from great error accumulation. For example, if the generated x'_t has an error, the next pose $x_{t'+1}$ is generated based on the information of x'_t, so it will accumulate the error as well.

Therefore, this research follows the recent approach [Mao et al. [2019]], predicts the future sequences in the trajectory space and generates the whole trajectory of each joint at once. Moreover, a padding strategy is employed to predict the residue of the sequences rather than the absolute value of the sequences because zero-velocity [Martinez et al. [2017]] is proved to have better performance. Specifically, a padding sequence is obtained by repeating the last pose x_T in the sequence $(T' - T)$ times, which can be written as $P_{T+1:T'} = [x_T, ..., x_T, ..., x_T]$. The input sequences for the DGCN model are the concatenation of $X_{1:T}$ and $P_{T+1:T'}$, which is denoted as $Input_{1:T'} = [x_1, x_2, ..., x_T, ..., x_T]$. The target sequences of the DGCN model are the concatenation of $X_{1:T}$ and $GX_{T+1:T'}$, which can be denoted as $Target_{1:T'} = [x_1, x_2, ..., x_T, ..., x_{T'}]$. Therefore, the DGCN model is designed to take the $Input_{1:T'}$ and produce a synthesis sequence $Output_{1:T'}$. Then the objective function of the DGCN model measures the error between $Output_{1:T'}$ and $Target_{1:T'}$.

4.4 Graph Neural Networks

In this approach, a GCN-based model is proposed to predict future movements in the trajectory space. As the aforementioned denotations, a human pose contains K joints and every joint is represented as the 3D position or Euler Rotation Angle.

4.4.1 Graph Formulation

The human body is a natural graph. Graph Neural Network-based methods achieved remarkable success on a lot of human pose related tasks in recent years because of their ability to exploit the implicit dependencies between joints. Therefore, this graph model is formed intuitively. Recalling that the human pose has K joints and the graph is defined as $G = (V, E)$. Here the node-set V contains K joints $\{J_0, J_1, ..., J_{k+1}\}$ and the edge set E contains the graph edges which correspond to the limbs on the human skeleton. Usually, the adjacent matrix of this graph G is denoted as A. In the matrix A, element a_{ij} on ith row jth column has the value 1 if and only if V_i and V_j are connected on this graph or $i = j$. In the experiment, this model is used to predict the graph connectivity, in other words, the edge is set E rather than give out the predefined E. Due to that, the model has a stronger fitting ability by the flexibility. Moreover, every joint is treated as three nodes on the graph, for they have the position $< x, y, z >$. So the node set V is actually $\{J_{0x}, J_{0y}, J_{0z}, J_{1x}, J_{1y}, J_{1z}, ..., J_{kx}, J_{ky}, J_{kz}\}$ for practical use.

In the DGCN model, the input feature F_{ix} for node J_{ix} is obtained from the trajectory data of J_{ix} from time period 1 to T'. It is a one-dimensional continuous function. Following Mao et al. [2019], this trajectory is transformed into a series of Discrete Cosine Transform (DCT) representations which are compact in the space to benefit the training. The DCT method uses cosine trajectories as a basis to represent the original trajectory. Any continuous trajectory can be represented by a series of the linear combinations of these bases uniquely. Therefore, every trajectory can be represented by the DCT representation's coefficients. In other words, the input feature F_{ix} of the Graph model is the DCT

method's coefficients. The Graph model will output a feature F'_{ix}, which are the DCT coefficients as well. Then the F'_{ix} will be transformed back to the trajectory by the linear combination because the basis is predefined. Therefore, the Graph model takes in the trajectory information for every joint and then produces the output trajectory for every joint. As evident in literature and experiments, predicting the residual data rather than the exact data will greatly reduce the gradient vanishing problem and gradient explosion problem, therefore, achieving better performance. Based on this knowledge, the coefficients $F_{ix} + F'_{ix}$ as the final results are actually used and then use it to reconstruct the output trajectory. Therefore, not only the future movements are predicted from time $T+1$ to T' but also the movements are reconstructed from time 1 to T. The part of the generated sequence from 1 to T can be used as an identity regularizer to guide the model training. In the next section, the details of the layers used in obtaining output features F' from the input features F will be given out.

4.4.2 Graph Convolutional Layers

There have been various kinds of convolutional layers introduced for graph data. Here, the graph convolutional layer Kipf & Welling [2016] designed in the spectrum perspective is adopted.

Laplacian Matrix

Firstly, this section will introduce the concept of the Laplacian Matrix of graphs. For example, a graph G like the following is given:

The related Degree Matrix of this graph G is defined as:

$$D = \begin{Bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{Bmatrix} \tag{4.1}$$

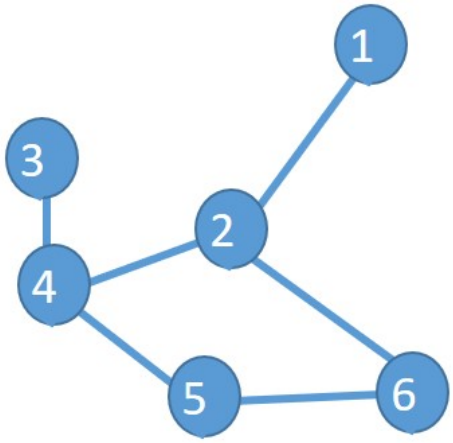

Figure 4.2: *The example of a graph.*

Where a_{ii} in this matrix means the number of edges which connected ith node.

The related Adjacency Matrix of this graph G is defined as:

$$W = \begin{Bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{Bmatrix} \tag{4.2}$$

In this matrix W, $a_{i,j} = 1$ if and only if the ith node and the jth node are connected by an edge.

The Laplacian Matrix L of a graph G is defined as $L = D - W$. In this case, the Laplacian Matrix is:

$$L = D - W = \begin{Bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & 0 & -1 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 0 & 0 & -1 & 2 \end{Bmatrix} \tag{4.3}$$

Laplacian Matrix is *symmetric positive semidefinite* matrix. This kind of matrix has n non-negative eigen values. Therefore, the Laplacian Matrix can be eigen decomposition, which is called Spectral decomposition. U is the matrix which is composed of the eigenvectors of L.

$$L = U\Lambda U^{-1} = U\Lambda U^T = U \begin{bmatrix} \lambda_1 & & \\ & \dots & \\ & & \lambda_2 \end{bmatrix} U^T \tag{4.4}$$

Graph Convolution

The graph convolutional layer is designed based on the idea that the convolutional operation of the two signals x and g is actually the dot product of them in the Fourier domain. The following formulation can be formed.

$$x * g = F^{-1}(F(x) \odot F(g)) \tag{4.5}$$

Where x stands for the input signal on the graph and g is the convolutional kernel signal respectively. F is the Fourier transformation function to project the signal on the graph to their Fourier domain. Due to the existing knowledge, Fourier transforms $F(x)$ on the graph that can be written as $U^T x$, where U is a matrix obtained from the Laplacian matrix of the graph. Every row of U is the eigenvector of the Laplacian matrix. Therefore, the equation 4.5 can be rewritten as:

$$x * g = U(U^T x \odot U^T g) = U g_\theta U^T x \tag{4.6}$$

Where $g_\theta = diag(U^T g)$. Therefore the formulation can be written as

the following:

$$x*g = Ug_\theta U^T x = U \begin{pmatrix} \hat{g}(\lambda_1) & & \\ & \ldots & \\ & & \hat{g}(\lambda_n) \end{pmatrix} \begin{pmatrix} \hat{x}(\lambda_1) \\ \hat{x}(\lambda_2) \\ \vdots \\ \hat{x}(\lambda_n) \end{pmatrix} = U \begin{pmatrix} \hat{g}(\lambda_1) & & \\ & \ldots & \\ & & \hat{g}(\lambda_n) \end{pmatrix} U^T x \tag{4.7}$$

Therefore, the g_θ is the convolutional kernel of the graph convolution operations. Different types of graph convolutions use different methods to approximate g_θ. In this thesis, the model adopted one of the state-of-the-art GCN methods[Kipf & Welling [2016]] to approximate g_θ, which is described in the following.

Graph Convolution Kernel

In math theory, every smooth function can be approximated by the Chebyshev polynomials. The Chebyshev Polynomials are a series of functions defined as $T_0(x) = I, T_1(x) = x, T_{n+1}(x) = 2LT_n(x) - T_{n-1}(x)$.

The function g_θ can be approximated by a linear combination of Chebyshev Polynomials:

$$g_\theta(x) = \sum_{k=0}^{K} \beta_k T_k(x) = \beta_0 T_0(x) + \beta_1 T_1(x) + ... + \beta_K T_K(x)$$

In this way, the original convolution operation changes to be:

$$x * g = Ug_\theta U^T x = U \sum_{k=0}^{K} \beta_k T_k(\hat{\Lambda}) U^T x = \sum_{k=0}^{K} \beta_k T_k(U\hat{\Lambda}U^T)x \tag{4.8}$$

The function $U\hat{\Lambda}U^T$ are denoted as $\hat{L}$ in the following discussion.

ChebNet [Defferrard et al. [2016]] first proposed to use Chebyshev Polynomials to approximate g_θ. In this way, the graph convolutional

layer only needs to learn the $K+1$ parameters rather than the n parameters. It greatly reduced the network complexity. Moreover, the key benefits of this strategy is that the Laplacian Eigen decomposition is not necessary anymore, which greatly reduced the computational complexity.

Furthermore, Kipf & Welling [2016] simplified the formulation by only considering the first order of the Chebyshev Polynomials so that the learnable parameters of each convolutional operation is significantly reduced to one.

If only the first order of the Chebyshev polynomials are considered, the equations can be written as [Quote from Kipf & Welling [2016]]:

$$\begin{aligned} x * g &= \textstyle\sum_{k=0}^{1} \beta_k T_k(\hat{L})x = \beta_0 T_0(\hat{L})x + \beta_1 T_1(\hat{L})x \\ &= (\beta_0 + \beta_1(L - I_n))x \\ &= (\beta_0 - \beta_1(D^{-1/2}WD^{-1/2}))x \end{aligned} \tag{4.9}$$

This equation can be further simplified by assuming $\beta_0 = -\beta_1 = \theta$. Then the final graph convolution operation can be written as [Quote from Kipf & Welling [2016]]:

$$x * g = \theta(\hat{D}^{-1/2}\hat{W}\hat{D}^{-1/2})x \tag{4.10}$$

The Graph Convolutional Layers

In this thesis, the graph convolutional operations mentioned before are adopted. These graph convolutional layers can be described as the following:

L is a matrix related to the trainable parameters and the degree matrix of the graph. Finally, it is assumed that the input feature of lth GCN layer is $F^l \in \mathbf{R}^{K \times C}$ (C is the channel number of the input features) and the output feature is $F^{l+1} \in \mathbf{R}^{K \times C'}$. The trainable parameters of the neural network is denoted as $W \in \mathbf{R}^{C \times C'}$, the matrix related to the Lapalian Matrix of graph is denoted as $\tilde{Z} \in \mathbf{R}^{K \times K}$, then the

convolutional operation can be derived out:

$$F^{l+1} = \sigma(\tilde{Z}^l F^l W) \tag{4.11}$$

σ is an activation function. The $LeakRELU()$ as function σ is used here rather than $Tanh()$ used in previous work [Mao et al. [2019]]. In this experiment, $\tilde{Z}$ is set as trainable to improve the performance because it can reduce overfitting.

4.5 The Densely Connected Network Structure

After this problem formulation and layer operations are explained, the network architecture will be described in this section, which is the main contribution. The input feature for the model is F, then it will pass through N GCN blocks and generate the final output feature F^N.

Residual GCN Blocks. Each GCN block contains two GCN layers and every GCN layer will append a BatchNorm layer, a LeakRELU layer and a Dropout layer. Inside every GCN block, the residual part of features are estimated. Then the procedure passing through the lth GCN layer can be formulated as:

$$F^{l+1} = GCN(F^l) + F^l \tag{4.12}$$

Densely Connection. In the previous work Mao et al. [2019], the output of each GCN block is directly fed into the next block. It can be believed that the approach does not exploit the feature maps of each layer sufficiently. For example, the first layer offers feature maps obtained by operating convolution on the 1-nearest node. Then the next feature map has a receptive field 2 because every node's feature contains the 2-nearest information. The key idea is to produce a more informative input feature by fusing multi-scale feature maps with a different size of the receptive field. Instead of feeding F^l feature maps for the lth block,

we try to feed all the feature maps $F^0, F^1, ..., F^l$ into the lth block to enlarge the ability of layers to exploit the hidden dependencies between joints in a different level.

Therefore, the network structure is reconstructed by adding dense links on the network to increase its ability. Firstly, the GCN blocks do not have residual links anymore, because the input feature size is not matching the output feature size anymore.

Assuming the input feature map F^0 of the model has feature size C, then the output feature map of lth GCN block has size C as well. The input feature map for lth GCN block is actually not the same anymore, but the concatenation of the output feature of all the previous GCN blocks. Therefore, the input size of lth GCN block is $(l+1) \times C$. Consequently, the formulation of the lth GCN block can be described as:

$$F^{l+1} = GCN([F^0, F^1, ..., F^l]) \tag{4.13}$$

Compared to Huang et al. [2017], this is the first time a dense structure-based GCN network is proposed for the motion prediction task. In this way, the feature maps of each layer contribute more significantly to the final results since the final layer is getting backpropagation directly to all the other GCN blocks. Therefore, it is less likely to get a gradient vanishing and gradient explosion problem as well.

Another significant advantage of this kind of structure compared to Residual GCN blocks is that the residual GCN block requires size matching but the dense structure does not. For example, the channel size of every output feature map F^l can be actually set differently. Assuming their channel sizes are $C^0, C^1, C^2, ..., C^N$, then the input feature of lth GCN block has feature size $\sum_{k=0}^{l} C^k$. The benefit of this is that the network can use the same size of parameters but actually goes much deeper. For a special case, $C^1, C^2, ..., C^N$ can set to be the same size but narrower than C^0, such as half of C^0. Then the network can have twice the deeper layers than before and also keep the model size at the same level. Meanwhile, the narrower middle layers also help to reduce

the overfitting problem.

4.6 Evaluation

For evaluations, the proposed DGCN models' effectiveness can empirically be demonstrated on the widely used benchmark Human3.6M Ionescu et al. [2013], CMU-Mocap[1] and 3DPWvon Marcard et al. [2018]. Comprehensive experiments have been carried out to validate the superior ability of the DGCN model in accuracy. The error results are reported both in the aspect of Euler Angles and 3D coordinates. The comparison results to the state-of-the-artwork including RNN-based model RRNN [Martinez et al. [2017]], CNN-based model convSeq2Seq [Li et al. [2018]], and GCN-based model LearnTraj [Mao et al. [2019]] are reported. In the end, ablation evaluations are conducted to investigate the impact of the proposed strategy.

4.7 Implementation Details

For a fair comparison, the same set of other prior works [Mao et al. [2019]] are used in the experiments. The input feature size of the model is 15 and every GCN layer outputs a hidden feature with size 256. Every GCN block contains 2 GCN layers and 12 GCN blocks employed in total. The dropout rate of each layer is set to be 0.5. The learning rate is 0.0005 and batch size is 16. An Adam optimizer is used for training and the results are trained after 50 epochs. The whole framework is implemented in PyTorch and trained on an NVIDIA GeForce GTX 1080 Ti with 11GB memory. The approximate training hour is 50 hours.

4.8 Benchmarks

Human3.6M Almost all of the existing motion prediction models are evaluated on the benchmark Human3.6M since it provides the largest amount of human poses. Following the typical settings [Martinez et al.

[1] http://mocap.cs.cmu.edu/

[2017]; Li et al. [2018]; Mao et al. [2019]], 15 actions performed by six actors are selected for the experiment from Human3.6M(H3.6M). Three kinds of format are provided by H3.6M. Here the 3D skeleton format with 32 joints are used to represent the human structure. The global rotation and translations are removed and all sequences are downsampled to 25 HZ. The trials from five actors are used as training dataset and the rest trials of one actor are used as testing dataset.

CMU-Mocap This dataset is firstly introduced for motion prediction evaluation by Li et al. [2018]. It contains a wider range of action types than H3.6M. Similar to H3.6M, the global rotations are removed and translations as well and normalize it. In total, eight actions (such as basketball, soccer, jumping and etc.) are selected under the prescriptions [Li et al. [2018]]. Every action set contains more than 5 trials. For the experiments, the same dataset splitting strategy is used like Li et al. [2018].

3DPW The 3D Pose in the Wild (3DPW) dataset is a new large-scale dataset with more than 51, 000 poses proposed by von Marcard et al. [2018]. These poses are extracted from challenging sequences such as going up-stairs and taking the bus. Mao *et al.* firstly introduced this dataset for motion prediction evaluation. This dataset is downloaded from the public resources and follow the same settings as the previous work.

4.9 Evaluation baselines and metrics

Baselines. The existing approaches for motion prediction can be broadly categorized as RNN-based, CNN-based, and GCN-based methods. The models with the best performance are selected and public codes so far in these three domains accordingly, which are RRNN [Martinez et al. [2017]], convSeq2Seq [Li et al. [2018]] and LearnTraj [Mao et al. [2019]]. For the errors reported directly in the Euler Angle space and 3D coordinates, their results are quoted directly from papers. For the visualization comparison, only LearnTraj [Mao et al. [2019]] are compared and the re-

sults are obtained from the public code[2].

Metrics. Two kinds of evaluation protocols are used in the experiments. Firstly, the traditional Euler angle error [Martinez et al. [2017]; Li et al. [2018]; Mao et al. [2019]] which represents the input and prediction data in the Euler Angle space and measures their Euclidean distance. However, some researchers find this kind of loss incapable to completely reflect the visual similarity – the zero-velocity baseline has a smaller error but looks different at visual aspect. Therefore, another Mean Per Joint Position Error (MPJPE) [Ionescu et al. [2013]] is used as an evaluation metric in this work as well. This error calculates the displacement between the groundtruth and predicted sequences from the 3D coordinates representation.

[2]https://github.com/wei-mao-2019/LearnTrajDep

Table 4.1: *The short-term prediction error of four action types on H3.6M dataset*

	Walking				Eating				Smoking				Discussion			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
RRNN Martinez et al. [2017]	0.28	0.49	0.72	0.81	0.23	0.39	0.62	0.76	0.33	0.61	1.05	1.15	0.31	0.68	1.01	1.09
CNNHD Li et al. [2018]	0.33	0.54	0.68	0.73	0.22	0.36	0.58	0.71	0.26	0.49	0.96	0.92	0.32	0.67	0.94	1.01
LearnTraj Mao et al. [2019]	**0.18**	**0.31**	**0.49**	**0.56**	**0.16**	**0.29**	**0.50**	**0.62**	0.22	0.41	**0.86**	**0.80**	**0.20**	**0.51**	**0.77**	**0.85**
Ours	0.20	0.32	0.54	0.61	0.18	0.32	0.54	0.66	**0.22**	**0.41**	0.87	0.83	0.22	0.59	0.92	1.00
LearnTraj(3D)	8.9	15.7	29.2	33.4	8.8	18.9	39.4	47.2	7.8	14.9	25.3	**28.7**	**9.8**	**22.1**	**39.6**	**44.1**
Ours(3D)	**8.4**	**15.3**	**28.3**	**33.2**	**8.6**	**18.2**	**38.3**	**46.5**	**6.9**	**13.4**	**24.2**	29.0	10.1	23.3	43.0	49.6

Table 4.2: *The short-term prediction error of 12 action types on H3.6M dataset*

	Directions				Greeting				Phoning				Posing			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
RRNN	0.26	0.47	0.72	0.84	0.75	1.17	1.74	1.83	0.23	0.43	0.69	0.82	0.36	0.71	1.22	1.48
CNNHD	0.39	0.60	0.80	0.91	0.51	0.82	1.21	1.38	0.59	1.13	1.51	1.65	0.29	0.60	1.12	1.37
LearnTraj	**0.26**	**0.45**	**0.71**	**0.79**	**0.36**	**0.60**	**0.95**	**1.13**	**0.53**	**1.02**	**1.35**	**1.48**	**0.19**	**0.44**	**1.01**	**1.24**
Ours	0.38	0.80	1.35	1.49	0.37	0.65	1.10	1.30	0.57	1.04	1.46	1.59	0.39	1.03	1.87	2.19
LearnTraj(3D)	**12.6**	24.4	**48.2**	**58.4**	14.5	30.5	74.2	89.0	11.5	20.2	37.9	43.2	9.4	23.9	66.2	82.9
Ours(3D)	12.9	**24.2**	57.0	72.4	**14.0**	**29.8**	**71.9**	**87.4**	**11.3**	**19.1**	**35.8**	**40.4**	**8.0**	**22.8**	**65.5**	**82.1**
	Purchases				Sitting				Sittingdown				Takingphoto			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
RRNN	0.51	0.97	1.07	1.16	0.41	1.05	1.49	1.63	0.39	0.81	1.40	1.62	0.24	0.51	0.90	1.05
CNNHD	0.63	0.91	1.19	1.29	0.39	0.61	1.02	1.18	0.41	0.78	1.16	1.31	0.23	0.49	0.88	1.06
LearnTraj	**0.43**	**0.65**	**1.05**	**1.13**	0.29	**0.45**	**0.80**	**0.97**	**0.30**	**0.61**	**0.90**	**1.00**	**0.14**	**0.34**	**0.58**	**0.70**
Ours	0.60	1.24	1.84	2.00	**0.29**	0.49	0.86	1.05	0.32	0.67	0.97	1.07	0.15	0.36	0.59	0.71
LearnTraj(3D)	**19.6**	**38.5**	**64.4**	**72.2**	10.7	**24.6**	**50.6**	**62.0**	11.4	27.6	56.4	67.6	6.8	**15.2**	**38.2**	**49.6**
Ours(3D)	21.6	40.5	67.1	78.0	**10.7**	25.0	53.2	66.6	**10.5**	**23.0**	**51.8**	**65.6**	**6.8**	15.6	40.7	52.9
	Waiting				Walkingdog				Walkingtogether				**Average**			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
RRNN	0.28	0.53	1.02	1.14	0.56	0.91	1.26	1.40	0.31	0.58	0.87	0.91	0.36	0.67	1.02	1.15
CNNHD	0.30	0.62	1.09	1.30	0.59	1.00	1.32	1.44	0.27	0.52	0.71	0.74	0.38	0.68	1.01	1.13
LearnTraj	0.23	0.50	0.91	1.14	0.46	0.79	1.12	1.29	0.15	0.34	0.52	**0.57**	**0.27**	**0.51**	**0.83**	**0.95**
Ours	**0.22**	**0.48**	**0.90**	**1.12**	**0.67**	**1.03**	**1.95**	**2.32**	**0.15**	**0.32**	**0.52**	0.60	0.33	0.65	1.09	1.24
LearnTraj(3D)	**9.5**	**22.0**	**57.5**	**73.9**	32.2	58.0	102.2	122.7	8.9	18.4	35.3	44.3	12.1	25.0	**51.0**	**61.3**
Ours(3D)	9.5	22.3	59.6	76.6	**22.8**	**48.3**	**95.8**	**116.3**	**8.3**	**18.2**	**34.2**	**43.1**	**11.36**	**23.93**	51.1	62.7

Table 4.3: *The short-term prediction error of 8 action types on the CMU dataset*

	Basketball				Basketball Signal				Directing Traffic				Jumping			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
LearnTrajMao et al. [2019]	0.33	0.52	0.89	1.06	0.11	0.20	0.41	0.53	**0.15**	**0.32**	0.52	0.60	**0.31**	**0.49**	**1.23**	**1.39**
Ours	**0.31**	**0.49**	**0.85**	**1.04**	**0.09**	**0.16**	**0.34**	**0.44**	0.17	0.33	**0.50**	**0.60**	0.33	0.70	1.74	1.63
LearnTraj(3D)Mao et al. [2019]	14.0	25.4	49.6	61.4	3.5	6.1	11.7	15.2	7.4	15.1	31.7	42.2	16.9	34.4	76.3	96.8
Ours(3D)	**10.5**	**19.0**	**38.9**	**49.0**	**2.3**	**4.4**	**10.1**	**13.9**	**6.0**	**12.4**	**30.1**	**38.8**	**12.0**	**27.0**	**70.7**	**94.6**
	Running				Soccer				Walking				Washwindow			
milliseconds	80	160	320	400	80	160	320	400	80	160	320	400	80	160	320	400
LearnTraj Mao et al. [2019]	0.33	0.55	0.73	0.74	0.18	**0.29**	**0.61**	**0.71**	0.33	0.45	0.49	0.53	0.22	0.33	**0.57**	**0.75**
Ours	**0.24**	**0.35**	**0.43**	**0.49**	**0.16**	0.30	0.70	0.84	**0.30**	**0.40**	**0.38**	**0.45**	**0.21**	**0.29**	0.60	0.78
LearnTraj(3D)Mao et al. [2019]	25.5	36.7	**39.3**	**39.9**	11.3	**21.5**	**44.2**	**55.8**	7.7	11.8	19.4	23.1	5.9	11.9	30.3	40.0
Ours(3D)	**20.5**	**32.3**	49.5	54.7	**9.8**	21.9	49.1	63.2	**5.7**	**10.2**	**18.4**	**21.1**	**4.9**	**10.1**	**27.9**	**37.2**

Table 4.4: *The Average error of all types of actions in the CMU dataset*

milliseconds	80	160	320	400
LearnTraj	0.25	0.39	**0.68**	0.79
Ours	**0.23**	**0.38**	0.69	**0.78**
LearnTraj(3D)	11.53	20.36	37.81	46.80
Ours(3D)	**8.96**	**17.16**	**36.84**	**46.56**

4.10 Results

In this section, the performance of the DGCN model is reported in these tables. The results of both the long-term (> 400ms) prediction and short-term (80ms, 160ms, 320ms, 400ms) prediction of each dataset are given out.

Human3.6M Typically, walking, eating and smoking and discussion actions are most widely evaluated as they are basic and ubiquitous in daily life. Firstly, the results of these four types of actions are shown in Table 4.1. Four baseline performances in the Euler Angle spaces are given out in this table. It shows the DGCN model lifting the performance in that term. However, it has been found out that the Euler Angle error does not reflect the similarity visually [Mao et al. [2019]]. So the model in the 3D coordinate space is trained and their comparison results are reported. LearnTraj achieved the smallest error in terms of 3D errors compared to the existing trending work. However, it can be seen from Table4.1 that the DGCN model surpasses it with a significant gap. Moreover, the results of the rest 12 types of actions are reported in Table 4.2. The DGCN outperforms LearnTraj on all the action types and average which demonstrates the effectiveness of the DGCN model. Besides the short term prediction results given by Tabel 4.1 and 4.2. For the more challenging long term prediction task, the DGCN model surpasses the state-of-art method both in Euler Angle error and 3D error.

CMU-Mocap Similarly to H3.6M, both the short-term and long-term prediction results of the CMU-Mocap dataset are shown in Table 4.3 and 4.4. Firstly, the Euler Angle error and 3D error of eight actions are shown in Table 4.3. More than 80 % of the errors are smaller after

using the DGCN model, except for Jumping and Soccer. This might happen because of the unbalance of the dataset, i.e. some actions achieve their best value while other action may get overfitting. To investigate further, the average values of different time intervals are reported. Table 4.4 demonstrates that the DGCN model is generic for the CMU-mocap dataset as well.

4.11 Summary

In this work, a densely connected GCN based model is firstly introduced for the motion prediction task which enhanced the feature maps utilization and reduced the overfitting problem. Experiments on three databases validate the effectiveness of the DGCN model. The performance of 3D joints' representation is better than the representation in angle space. The DGCN model beats down the state-of-the-art methodologies, therefore, shows that the dense strategy has the ability to increase the feature utilization therefore enhance the performance of the prediction accuracy.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Human motion prediction has been a longstanding problem in the fields of computer vision and computer graphics. The model that can perform this task successfully will have significant impacts and various applications in the industry and commerce. It will allow for artificially intelligent robots and virtual agents, for example, that perform smoother interactive movements when engaged in forms of cooperation or sport with human users. This is because they will be forecasting users' actions more precisely. Moreover, this research is also important for autonomous driving systems. Autonomous cars can predict pedestrians' future movements much earlier than they otherwise would and more accurate in order to enhance their security level.

In this thesis, existing work on motion prediction and related deep learning methods have been extensively investigated. Two novel motion prediction frameworks have also been proposed, which have achieved state-of-the-art performance.

In Chapter 2, an overview of existing works related to four areas are presented : motion data, motion-related topics, motion prediction methods, and deep learning methods. The first section discussed previous feature engineering strategies concerning motion and the related prob-

lem of motion research. The second focused on current research progress on the motion prediction problem and pointed out its limitations. The research conducted as part of this PhD work is motivated by a desire to address these limitations. The last section summarised prevailing deep learning models, which provided a range of insights and ideas taken up in this research.

In Chapter 3, a novel efficient convolutional hierarchical autoencoder framework to address the efficiency problem in motion prediction is proposed. The majority of existing models put forward for motion prediction are either RNN- or CNN-based; as such, they are usually expensive and complex in terms of computation. In practice, however, this involves a large amount of motion data, reaching more than 200,000. a new encoder has therefore been designed, which combines 1D convolutional operations with a hierarchical tree structure. This new encoder captures information for motion prediction, greatly reducing parameter size and computational complexity. This new framework is evaluated on two existing heavily benchmarked datasets, H3.6M and CMU. Its performance is on par with state-of-the-art works, yet it uses much less memory and computes much faster. A visualisation of prediction results for qualitative comparison is also provided; it shows that this model generates smooth and realistic results for prediction.

Chapter 4 delves deeper into the motion prediction problem, following up on the recent success of GNN. To address the oversmoothing problem that attends the existing graph network in motion prediction, dense links are introduced into the framework to enhance its use of features. When compared with previous networks, the densely connected framework is of the same model size, but reuses feature maps much more extensively. The first layers of feature maps skip over the middle layers to propagate in the last layers. Therefore more useful information is captured. The experiments also analysed the relationship between performance, the number of blocks and size of dense links. Following protocol, this chapter also provided qualitative and quantitative comparison experiments with existing baselines. These confirm that introducing dense links into the framework can ensure performance that exceeds that of state-of-the-art

works, thus validating the effectiveness of the proposed model.

5.2 Future Work

Although the two models proposed in Chapter 3 and Chapter 4 address the challenges which are set out in the Introduction, there remain problems in the path of motion prediction. The following ideas are therefore proposed for addressing these limitations in future works.

5.2.1 Long term error accumulation problem

Various models have been proposed for motion prediction and promising results have been achieved [Fragkiadaki et al. [2015]; Jain et al. [2016]; Ghosh et al. [2017]; Gui et al. [2018a]; Li et al. [2018]]. Like most existing models, the ECHA model (described in Chapter 3) predicts future movements in a recursive way. Specifically, these models generate a first frame and use generated results to predict the next frame. In this way, the errors generated in synthesising the first frame will propagate during the synthesising process and affect the accuracy of the next frame. The longer the predicted sequences need to be, the more these errors will accumulate. For this reason, previous work reported very good performance when it comes to short-term prediction. When applied to long-term prediction, though, it will incur more errors. This problem, which is called the long-term error accumulation problem, afflicts most prediction tasks. Some research works [such as Bütepage et al. [2017]] have abandoned the recursive approach to synthesising output sequences. Although this move greatly reduces the error accumulation problem, it may also produce unrealistic prediction sequences, such as quivering or unnatural motions. This is because it ignores the temporal causal (which implicitly lies in the sequence). The challenging question of how researchers might address the error accumulation problem, therefore, is pivotal to motion prediction.

In the future, a Cycle Strategy Motion Prediction Model (CSMPM) is planned to be proposed, which would alleviate the error accumula-

tion problem by adding an identity regularisation function to the task of prediction. A cycle loop will be proposed to build through which the model can predict future sequences and then use these to infer the original input sequences. In addition to the generator G, therefore, an inverse generator G^* will be designed to map future sequences back onto the input domain. Assuming that $GG^* = I$, a cycle loss can be added to the objective function as a means of alleviating the error accumulation problem. In the future design, the G is adopted from the state-of-the-art motion prediction model. Although the G^* has the same network architecture as G , they have different learning difference parameters. The loss function consists of two parts, the mean square error of the predicted sequences and the reconstruction errors. To map input sequences onto future sequences, G would be trained to be the effective generator. Similarly, G^* would be trained by using the same dataset, but reversing the input and output domains. Finally, the whole model would be fine-tuned to train GG^* together. What is more, new Discriminators will be designed, D_a and D_b for this model to supervise the quality of the synthesised sequences.

This idea might be advanced through two potential innovations: (a) proposing a new CSMPM for motion prediction for the first time, so as to address the long term error accumulation problem, and (b) designing a cycle loss and two novel discriminators to strengthen the model's performance.

5.2.2 Multi-possible future of Motion prediction

In this thesis, just one possible solution for future prediction is provided. It is important to bear in mind, though, that the problem of human motion prediction might have a variety of possible solutions. This means the future developments might lead to multiple possible results. Existing work has used a predefined latent vector $z \sim (0, 1)$ to evoke different possible futures. When it comes to predicting the multimodal future, however, they are unable to fully assimilate complex prior spatial-temporal information. It can be believed that the latent factors affect future move-

ments in human intentions. The question of how to design a framework that considers human intentions in producing the multimodal future of human motion, then, remains problematic.

Aliakbarian et al. [2020] have proposed a conditional variational autoencoder block to force the model to take account of random noise. Their model is therefore more able to explore the possibility of stochastic futures. Li [2019] has proposed a novel imitative decision learning (IDL) approach for pedestrian trajectory prediction. Their whole framework uses a ConvGRU subnetwork to predict future movements and a Fully Convolutional subnetwork to infer the latent vector z, which represents human decisions about their next step. This work has a similar architecture to InfoGAN, which has been proposed by Chen et al. [2016]. InfoGAN also investigates the latent vector more deeply. Chen et al's latent vector is factorised, meaning that they obtained interpretable representations that disentangled writing style and digital shape during the MNIST generation task. Similarly, the goal for the prediction task is that of finding interpretable representations of z, as opposed to random noises.

In the future, this research will focus on addressing multiple solutions to the problem of human motion prediction. A possible solution is that of designing a new decision-based framework, which would incorporate human intentions with history sequences. A subnetwork will take the history sequences as inputs and produce a predicted latent Gaussion distribution $z \sim (x, y)$ as the human intention. Then the other prediction subnetwork will generate future movements based on the history information and a noise vector. The predicted latent distribution will be randomly sampled and fed into the motion generation subnetwork. Finally, the loss will be the minimised error of the multiple possible prediction results and ground truth. A discriminator or multiple discriminators will be designed to evaluate the level of realism of the generated sequences.

5.2.3 Oversmoothing problem on the Motion Prediction framework

As we have laid out in the preceding literature review, one emerging approach to the motion prediction problem is that of using GNNs. An existing approach proposed a residual graph neural network to model the spatial and temporal modalities of motion data and achieved significant progress. However, this field remains underexploited. One of the most challenging issue is the oversmoothing problem that attends GCN, which still limits the accuracy of motion prediction. A densely connected GCN network has been proposed; however, that can only alleviate the oversmoothing problem. In the future, this research will continue to focus on addressing the oversmoothing problem. One possible solution is that of building up a natural U-structure graph network for motion prediction, which can achieve better performance by addressing the oversmoothing problem.

In 2015, Ronneberger et al. [2015] proposed the idea of U-net, demonstrating that they had made surprising advances when it came to the ISBI cell tracking challenge. Their framework introduces a simple idea: that of using a symmetrical expanding path to reuse the feature maps. Numerous researchers have taken up this idea and achieved remarkable results in various applications. However, the key difference of the image and graph tasks is that the convolutional network has natural upsample and downsample operations, which are not trivial according to the graph data. To address this problem, Gao & Ji [2019] have proposed novel graph pooling (gPool) and unpooling (gUnpool) operations, which select the subset of the node in an adaptive way. Their experiments have demonstrated that their work outperforms state-of-the-art research by a large margin. Furthermore, Yu et al. [2019] have proposed a Spatio-Temporal U-Net (ST-UNet), which learns more global and local spatial and temporal information from the graph series data. Their framework outperforms existing work because ST-UNet captures features at multiple scales.

In the future, a new graph-based framework will be proposed for mo-

tion prediction to address the oversmoothing problem. It might have upsample and downsample structures. For example, five layers might be used to encode input motion sequences; accordingly five layers might be used as decoders to generate future motion sequences. Every layer will therefore produce a feature map, which would represent the original motion clips. The feature maps will be copied directly from the five layers in the encoder and sent to the related decoder layers. The whole process will be similar to a U-Net structure. In this way, existing feature maps would be increasingly use, thus reducing the oversmoothing problem. The key challenge is how to design a U-structure graph neural network for motion prediction. The network will be trained on existing benchmarks to learn the proper parameters and generate prediction results.

5.2.4 Loss Function for Motion Prediction

One key problem, which limits the practical uses of motion prediction, is that of loss function. Today, two loss functions are used in mainstream research. One is the 3D coordinates loss function. Another is the Euler Angle format loss function. The 3D coordinate loss function usually obtains more accurate results and produces results that are visually similar. The output skeleton may have limbs of slightly different lengths; however, for their length is not fixed by the 3D coordinate representation. This small difference might not be tolerated in practical applications, such as in robotics. Euler Angle representations are not hampered by these disadvantages because they use a fixed limb length and it is evaluated on angles. It has been found that this kind of representation, however, does not always lead to results that are more visually similar than those provided by alternative approaches [Martinez et al. [2017]]. The Euler Angle error may be small, but the visual difference can be significant. A new loss function should therefore be introduced so as to measure the distance between motions more accurately in the future.

Coskun et al. [2018] have observed that the L_2 distance of motion data cannot reflect the real similarity between two motion clips. For example,

even if the L_2 error is small, yet two clips may still look markedly dissimilar. Coskun et al. have therefore proposed a triplet RNN network that can learn a new metric to evaluate the similarity between two motion clips. Gui et al. [2018a] have put forward a framewise geodesic loss by projecting a motion clip as a curve on a Lie surface. Their loss function is more geometrically meaningful because it reflects the distance on a Lie manifold. In their experiments, the geodesic loss performed better than the Euclidean loss. Gopalakrishnan et al. [2019] have introduced a new loss function, Normalized Power Spectrum Similarity (NPSS), to help evaluate the model's capacity for long-term prediction. This metric contains a frequency shift and a phase shift design, which can complement the Euclidean loss in long-term evaluation.

In the future, this research will focus on developing a new loss function that better fits the properties of motion. Consider four ideas for the future development of this research. The first idea that we might explore is that of investigating the whole motion sequence in Lie Algebra Space and designing mathematical distance in the Lie manifold. The second possible idea is that of designing a hybrid representation of 3D coordinates and Euler angles to both avoid their drawbacks and take up their advantages. For example, a multi-task structure network with two branches can be designed. Each branch might use a different loss function, fusing them together in the last layer. The third idea is to use the trained deep metric of [Coskun et al. [2018]] for the task of motion prediction. The fourth is to design a new loss function, which would evaluate the error in the coodinated space but train the model in the space of Euler angles. Among these four ideas, the third one needs a lot of effort on building up new architectures and the first one cannot guarantee the leveraging of performance. The second idea lacks novelty in this domain. Therefore, the fourth one could be the first priority to try for this task.

Bibliography

(2020). 'MoCapWeb'. http://www.gamelook.com.cn/2016/02/244513.

(2020). 'WebDataset'. https://www.marketresearchfuture.com/reports/3d-motion-capture-system-market-3026.

M. Abadi, et al. (2016). 'Tensorflow: a system for large-scale machine learning.'. In *OSDI*, vol. 16, pp. 265–283.

S. Aliakbarian, et al. (2020). 'A Stochastic Conditioning Scheme for Diverse Human Motion Prediction'. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5223–5232.

M. Arjovsky, et al. (2017). 'Wasserstein gan'. *arXiv preprint arXiv:1701.07875* .

M. Baccouche, et al. (2011). 'Sequential deep learning for human action recognition'. In *International Workshop on Human Behavior Understanding*, pp. 29–39. Springer.

M. Barnachon, et al. (2012). 'Human actions recognition from streamed motion capture'. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 3807–3810. IEEE.

M. Barnachon, et al. (2014). 'Ongoing human action recognition with motion capture'. *Pattern Recognition* **47**(1):238–247.

E. Barsoum, et al. (2018). 'HP-GAN: Probabilistic 3D human motion prediction via GAN'. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1418–1427.

R. v. d. Berg, et al. (2017). 'Graph convolutional matrix completion'. *arXiv preprint arXiv:1706.02263* .

C. Böhm, et al. (2001). 'Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases'. *ACM Computing Surveys (CSUR)* **33**(3):322–373.

M. Brand & A. Hertzmann (2000). 'Style machines'. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 183–192. ACM Press/Addison-Wesley Publishing Co.

C. Bregler & J. Malik (1998). 'Tracking people with twists and exponential maps'. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pp. 8–15. IEEE.

A. Brock, et al. (2018). 'Large scale gan training for high fidelity natural image synthesis'. *arXiv preprint arXiv:1809.11096* .

J. Bütepage, et al. (2017). 'Deep representation learning for human motion prediction and classification'. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 2017. IEEE.

J. Bütepage, et al. (2018). 'Anticipating many futures: Online human motion prediction and generation for human-robot interaction'. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1–9. IEEE.

X. Chen, et al. (2016). 'Infogan: Interpretable representation learning by information maximizing generative adversarial nets'. In *Advances in neural information processing systems*, pp. 2172–2180.

P. Climent-Pérez, et al. (2012). 'Optimal joint selection for skeletal data from RGB-D devices using a genetic algorithm'. In *Mexican International Conference on Artificial Intelligence*, pp. 163–174. Springer.

H. Coskun, et al. (2018). 'Human motion analysis with deep metric learning'. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 667–683.

M. Defferrard, et al. (2016). 'Convolutional neural networks on graphs

with fast localized spectral filtering'. In *Advances in neural information processing systems*, pp. 3844–3852.

E. L. Denton, et al. (2014). 'Exploiting linear structure within convolutional networks for efficient evaluation'. In *Advances in neural information processing systems*, pp. 1269–1277.

Y. Du, et al. (2015). 'Hierarchical recurrent neural network for skeleton based action recognition'. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1110–1118.

D. K. Duvenaud, et al. (2015). 'Convolutional networks on graphs for learning molecular fingerprints'. In *Advances in neural information processing systems*, pp. 2224–2232.

C. Ellis, et al. (2013). 'Exploring the trade-off between accuracy and observational latency in action recognition'. *International Journal of Computer Vision* **101**(3):420–436.

S. A. Etemad & A. Arya (2015). 'Correlation-optimized time warping for motion'. *The Visual Computer* **31**(12):1569–1586.

G. Evangelidis, et al. (2014). 'Skeletal quads: Human action recognition using joint quadruples'. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 4513–4518. IEEE.

A. Eweiwi, et al. (2014). 'Efficient pose-based action recognition'. In *Asian Conference on Computer Vision*, pp. 428–443. Springer.

Y. Feng, et al. (2014). 'Exploiting temporal stability and low-rank structure for motion capture data refinement'. *Information Sciences* **277**:777–793.

E. Fotiadou & N. Nikolaidis (2014). 'Activity-based methods for person recognition in motion capture sequences'. *Pattern Recognition Letters* **49**:48–54.

K. Fragkiadaki, et al. (2015). 'Recurrent network models for human dynamics'. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4346–4354.

J. Gall, et al. (2009). 'Motion capture using joint skeleton tracking and surface estimation'. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1746–1753. IEEE.

H. Gao & S. Ji (2019). 'Graph u-nets'. *arXiv preprint arXiv:1905.05178* .

M. Garbade & J. Gall (2016). 'Handcrafting vs deep learning: an evaluation of ntraj+ features for pose based action recognition'. In *Workshop on New Challenges in Neural Computation and Machine Learning (\(NC^ 2\))*, pp. 85–92.

J. Gehring, et al. (2017). 'Convolutional sequence to sequence learning'. *arXiv preprint arXiv:1705.03122* .

P. Ghosh, et al. (2017). 'Learning human motion models for long-term predictions'. In *3D Vision (3DV), 2017 International Conference on*, pp. 458–466. IEEE.

T. Golda, et al. (2019). 'What goes around comes around: Cycle-Consistency-based Short-Term Motion Prediction for Anomaly Detection using Generative Adversarial Networks'. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–8. IEEE.

I. Goodfellow, et al. (2014). 'Generative adversarial nets'. In *Advances in neural information processing systems*, pp. 2672–2680.

A. Gopalakrishnan, et al. (2019). 'A neural temporal model for human motion prediction'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12116–12125.

L.-Y. Gui, et al. (2018a). 'Adversarial geometry-aware human motion prediction'. In *ECCV*, pp. 823–842.

L.-Y. Gui, et al. (2018b). 'Teaching robots to predict human motion'. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 562–567. IEEE.

D. K. Hammond, et al. (2009). 'Wavelets on graphs via spectral graph theory'. *arXiv preprint arXiv:0912.3848* .

K. He, et al. (2016). 'Deep residual learning for image recognition'. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

M. Henaff, et al. (2015). 'Deep convolutional networks on graph-structured data'. *arXiv preprint arXiv:1506.05163* .

A. Hernandez, et al. (2019). 'Human motion prediction via spatio-temporal inpainting'. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7134–7143.

D. Holden, et al. (2017). 'Phase-functioned neural networks for character control'. *ACM Transactions on Graphics (TOG)* **36**(4):42.

D. Holden, et al. (2016). 'A deep learning framework for character motion synthesis and editing'. *ACM Transactions on Graphics (TOG)* **35**(4):138.

D. Holden, et al. (2015). 'Learning motion manifolds with convolutional autoencoders'. In *SIGGRAPH Asia 2015 Technical Briefs*, p. 18. ACM.

E. Hsu, et al. (2005). 'Style translation for human motion'. In *ACM Transactions on Graphics (TOG)*, vol. 24, pp. 1082–1089. ACM.

G. Huang, et al. (2017). 'Densely connected convolutional networks'. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

T. Huang, et al. (2012). 'Motion retrieval based on kinetic features in large motion database'. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pp. 209–216. ACM.

D. H. Hubel & T. N. Wiesel (1962). 'Receptive fields, binocular interaction and functional architecture in the cat's visual cortex'. *The Journal of physiology* **160**(1):106.

L. Ikemoto, et al. (2009). 'Generalizing motion edits with gaussian processes'. *ACM Transactions on Graphics (TOG)* **28**(1):1.

C. Ionescu, et al. (2013). 'Human3. 6m: Large scale datasets and pre-

dictive methods for 3d human sensing in natural environments'. *IEEE transactions on pattern analysis and machine intelligence* **36**(7):1325–1339.

C. Ionescu, et al. (2014). 'Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments'. *IEEE transactions on pattern analysis and machine intelligence* **36**(7):1325–1339.

A. Jain, et al. (2016). 'Structural-RNN: Deep learning on spatio-temporal graphs'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5308–5317.

S. Ji, et al. (2013). '3D convolutional neural networks for human action recognition'. *IEEE transactions on pattern analysis and machine intelligence* **35**(1):221–231.

N. Kalchbrenner & P. Blunsom (2013). 'Recurrent continuous translation models'. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1700–1709.

N. Kalchbrenner, et al. (2016). 'Neural machine translation in linear time'. *arXiv preprint arXiv:1610.10099* .

K.-i. Kamijo & T. Tanigawa (1990). 'Stock price pattern recognition-a recurrent neural network approach'. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pp. 215–221. IEEE.

T. Kerola, et al. (2014). 'Spectral graph skeletons for 3D action recognition'. In *Asian Conference on Computer Vision*, pp. 417–432. Springer.

T. N. Kipf & M. Welling (2016). 'Semi-supervised classification with graph convolutional networks'. *arXiv preprint arXiv:1609.02907* .

A. Krizhevsky, et al. (2012). 'Imagenet classification with deep convolutional neural networks'. In *Advances in neural information processing systems*, pp. 1097–1105.

B. Krüger, et al. (2010). 'Fast local and global similarity searches in large motion capture databases'. In *Proceedings of the 2010 ACM*

SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 1–10. Eurographics Association.

J. N. Kundu, et al. (2019). 'Bihmp-gan: Bidirectional 3d human motion prediction gan'. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 8553–8560.

Y. LeCun, et al. (1998). 'Gradient-based learning applied to document recognition'. *Proceedings of the IEEE* **86**(11):2278–2324.

G. Lefebvre, et al. (2013). 'BLSTM-RNN based 3D gesture classification'. In *International Conference on Artificial Neural Networks*, pp. 381–388. Springer.

A. M. Lehrmann, et al. (2014). 'Efficient nonlinear markov models for human motion'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1314–1321.

C. Li, et al. (2018). 'Convolutional Sequence to Sequence Model for Human Dynamics'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5226–5234.

J. Li, et al. (2016). 'A persona-based neural conversation model'. *arXiv preprint arXiv:1603.06155* .

M. Li, et al. (2019). 'Actional-structural graph convolutional networks for skeleton-based action recognition'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3595–3603.

S. Li & A. B. Chan (2014). '3d human pose estimation from monocular images with deep convolutional neural network'. In *Asian Conference on Computer Vision*, pp. 332–347. Springer.

W. Li, et al. (2010). 'Action recognition based on a bag of 3d points'. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 9–14. IEEE.

Y. Li (2019). 'Which way are you going? imitative decision learning for path forecasting in dynamic scenes'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 294–303.

J. Liang, et al. (2019). 'Peeking into the future: Predicting future person activities and locations in videos'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5725–5734.

Y. Liu, et al. (2011). 'Markerless motion capture of interacting characters using multi-view image segmentation' .

W. Ma, et al. (2010). 'Modeling style and variation in human motion'. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 21–30. Eurographics Association.

W. Mao, et al. (2019). 'Learning trajectory dependencies for human motion prediction'. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9489–9497.

J. Martinez, et al. (2017). 'On human motion prediction using recurrent neural networks'. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4674–4683. IEEE.

T. Mikolov, et al. (2010). 'Recurrent neural network based language model'. In *Eleventh Annual Conference of the International Speech Communication Association*.

J. Min, et al. (2010). 'Synthesis and editing of personalized stylistic human motion'. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pp. 39–46. ACM.

R. Mittelman, et al. (2014). 'Structured recurrent temporal restricted boltzmann machines'. In *International Conference on Machine Learning*, pp. 1647–1655.

T. B. Moeslund & E. Granum (2001). 'A survey of computer vision-based human motion capture'. *Computer vision and image understanding* **81**(3):231–268.

M. Müller, et al. (2009). 'Efficient and robust annotation of motion capture data'. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 17–26. ACM.

M. Müller & T. Röder (2006). 'Motion templates for automatic classification and retrieval of motion capture data'. In *Proceedings of the 2006*

ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 137–146. Eurographics Association.

M. Müller, et al. (2005). 'Efficient content-based retrieval of motion capture data'. In *ACM Transactions on Graphics (ToG)*, vol. 24, pp. 677–685. ACM.

M. Müller, et al. (2007). 'Documentation MoCap database hdm05' .

M. Niepert, et al. (2016). 'Learning convolutional neural networks for graphs'. In *International conference on machine learning*, pp. 2014–2023.

F. Ofli, et al. (2013). 'Berkeley mhad: A comprehensive multimodal human action database'. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pp. 53–60. IEEE.

F. Ofli, et al. (2012). 'Learn2Dance: Learning Statistical Music-to-Dance Mappings for Choreography Synthesis.'. *IEEE Trans. Multimedia* **14**(3-2):747–759.

C. Pérez-D'Arpino & J. A. Shah (2016). 'Fast Motion Prediction for Collaborative Robotics.'. In *IJCAI*, pp. 3988–3989.

L. Porzi, et al. (2017). 'Depth-aware convolutional neural networks for accurate 3D pose estimation in RGB-D images'. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 5777–5783. IEEE.

T. Qi, et al. (2013). 'A semantic feature for human motion retrieval'. *Computer animation and virtual worlds* **24**(3-4):399–407.

T. Qi, et al. (2014). 'Real-time motion data annotation via action string'. *Computer Animation and Virtual Worlds* **25**(3-4):291–300.

A. Radford, et al. (2015). 'Unsupervised representation learning with deep convolutional generative adversarial networks'. *arXiv preprint arXiv:1511.06434* .

A. M. Rather, et al. (2015). 'Recurrent neural network and a hybrid

model for prediction of stock returns'. *Expert Systems with Applications* **42**(6):3234–3241.

B. Ristic, et al. (2008). 'Statistical analysis of motion patterns in AIS data: Anomaly detection and motion prediction'. In *2008 11th International Conference on Information Fusion*, pp. 1–7. IEEE.

O. Ronneberger, et al. (2015). 'U-net: Convolutional networks for biomedical image segmentation'. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer.

J. Sedmidubsky, et al. (2018). 'Effective and efficient similarity searching in motion capture data'. *Multimedia Tools and Applications* **77**(10):12073–12094.

A. Shahroudy, et al. (2016). 'NTU RGB+ D: A large scale dataset for 3D human activity analysis'. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1010–1019.

Z. Shao & Y. Li (2015). 'Integral invariants for space motion trajectory matching and recognition'. *Pattern Recognition* **48**(8):2418–2432.

Z. Shao, et al. (2018). 'A Hierarchical Model for Human Action Recognition from Body-Parts'. *IEEE Transactions on Circuits and Systems for Video Technology* .

S. Shariat & V. Pavlovic (2011). 'Isotonic CCA for sequence alignment and activity recognition'. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2572–2578. IEEE.

L. Shi, et al. (2019). 'Skeleton-based action recognition with directed graph neural networks'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7912–7921.

K. Simonyan & A. Zisserman (2014). 'Very deep convolutional networks for large-scale image recognition'. *arXiv preprint arXiv:1409.1556* .

I. Sutskever, et al. (2009). 'The recurrent temporal restricted boltzmann machine'. In *Advances in neural information processing systems*, pp. 1601–1608.

I. Sutskever, et al. (2014). 'Sequence to sequence learning with neural networks'. In *Advances in neural information processing systems*, pp. 3104–3112.

C. Szegedy, et al. (2015). 'Going deeper with convolutions'. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Z. Tang, et al. (2014). 'Human motion retrieval based on freehand sketch'. *Computer Animation and Virtual Worlds* **25**(3-4):271–279.

G. W. Taylor & G. E. Hinton (2009). 'Factored conditional restricted Boltzmann machines for modeling motion style'. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1025–1032. ACM.

G. W. Taylor, et al. (2007). 'Modeling human motion using binary latent variables'. In *Advances in neural information processing systems*, pp. 1345–1352.

G. W. Taylor, et al. (2011). 'Two distributed-state models for generating high-dimensional time series'. *Journal of Machine Learning Research* **12**(Mar):1025–1068.

G. W. Taylor, et al. (2010). 'Dynamical binary latent variable models for 3d human pose tracking'. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 631–638. IEEE.

D. Tome, et al. (2017). 'Lifting from the deep: Convolutional 3d pose estimation from a single image'. *CVPR 2017 Proceedings* pp. 2500–2509.

R. Urtasun, et al. (2008). 'Topologically-constrained latent variable models'. In *Proceedings of the 25th international conference on Machine learning*, pp. 1080–1087. ACM.

R. Vemulapalli, et al. (2014). 'Human action recognition by representing 3d skeletons as points in a lie group'. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 588–595.

T. von Marcard, et al. (2018). 'Recovering accurate 3d human pose in the

wild using imus and a moving camera'. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 601–617.

J. Walker, et al. (2017). 'The pose knows: Video forecasting by generating pose futures'. In *Proceedings of the IEEE international conference on computer vision*, pp. 3332–3341.

J. Wan, et al. (2014). 'Deep learning for content-based image retrieval: A comprehensive study'. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 157–166. ACM.

J. Wang, et al. (2012). 'Mining actionlet ensemble for action recognition with depth cameras'. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1290–1297. IEEE.

J. M. Wang, et al. (2007). 'Multifactor Gaussian process models for style-content separation'. In *Proceedings of the 24th international conference on Machine learning*, pp. 975–982. ACM.

J. M. Wang, et al. (2008). 'Gaussian process dynamical models for human motion'. *IEEE transactions on pattern analysis and machine intelligence* **30**(2):283–298.

P. Wohlhart & V. Lepetit (2015). 'Learning descriptors for object recognition and 3d pose estimation'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3109–3118.

S. Xia, et al. (2015). 'Realtime style transfer for unlabeled heterogeneous human motion'. *ACM Transactions on Graphics (TOG)* **34**(4):119.

J. Xiao, et al. (2015). 'Sparse motion bases selection for human motion denoising'. *Signal Processing* **110**:108–122.

S. Yan, et al. (2018a). 'Spatial temporal graph convolutional networks for skeleton-based action recognition'. In *Thirty-second AAAI conference on artificial intelligence*.

X. Yan, et al. (2018b). 'Mt-vae: Learning motion transformations to generate multimodal human dynamics'. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 265–281.

H. Yasin, et al. (2016). 'A dual-source approach for 3D pose estimation from a single image'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4948–4956.

B. Yu, et al. (2019). 'ST-UNet: A spatio-temporal U-network for graph-structured time series modeling'. *arXiv preprint arXiv:1903.05631* .

Q. Yu, et al. (2015). 'Sketch-a-net that beats humans'. *arXiv preprint arXiv:1501.07873* .

H. Zhang, et al. (2019). 'Self-attention generative adversarial networks'. In *International Conference on Machine Learning*, pp. 7354–7363.

L. Zhao, et al. (2019). 'Semantic graph convolutional networks for 3D human pose regression'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3425–3435.

F. Zhou & F. Torre (2009). 'Canonical time warping for alignment of human behavior'. In *Advances in neural information processing systems*, pp. 2286–2294.

J.-Y. Zhu, et al. (2017). 'Unpaired image-to-image translation using cycle-consistent adversarial networks'. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232.

W. Zhu, et al. (2016). 'Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks.'. In *AAAI*, vol. 2, p. 6.