The Visual Computer (2019) 35:1143–1156
https://doi.org/10.1007/s00371-019-01692-9

ORIGINAL ARTICLE

# Efficient convolutional hierarchical autoencoder for human motion prediction

Yanran Li[1] · Zhao Wang[2,3] · Xiaosong Yang[1] · Meili Wang[4] · Sebastian Iulian Poiana[1] · Ehtzaz Chaudhry[1] · Jianjun Zhang[1]

Published online: 11 May 2019


**Abstract**
Human motion prediction is a challenging problem due to the complicated human body constraints and high-dimensional dynamics. Recent deep learning approaches adopt RNN, CNN or fully connected networks to learn the motion features which do not fully exploit the hierarchical structure of human anatomy. To address this problem, we propose a convolutional hierarchical autoencoder model for motion prediction with a novel encoder which incorporates 1D convolutional layers and hierarchical topology. The new network is more efficient compared to the existing deep learning models with respect to size and speed. We train the generic model on Human3.6M and CMU benchmark and conduct extensive experiments. The qualitative and quantitative results show that our model outperforms the state-of-the-art methods in both short-term prediction and long-term prediction.

**Keywords** Motion prediction · Deep learning · Autoencoder · Hierarchical networks

## 1 Introduction

Forecasting the future movements of human behaviours is one of the most fundamental problems in understanding human motion, and it has various practical applications in computer animation [13,35], human interaction Robots [12,25], computer vision [5,23] and computer graphics [15,22]. Especially for the human interaction robotics [21] or virtual characters [13] in computer games, they are supposed to not only respond to the opponents' movements but also have a preemptively ability to predict future movements. For example, an intelligent agent should anticipate human athletes' actions accurately and rapidly from historical data in adversarial sports like badminton and fencing, or in collaboration scenarios like paired figure skating and Waltz dancing. For animation production, motion prediction techniques can be utilized to generate new motion data automatically so that the animators can avoid a lot of manual work and time [13,14,22].

Forecasting human motion is a natural intelligence for human beings; however, it still remains challenging for computers because human motion data is high dimensional and has complicated bio-mechanical constraints. Over the last decades, there were a number of models [2,3,5,7,10,11,13,17,23,24,33] introduced to address motion prediction problems. Inspired from the striking breakthrough of deep learning technology, they introduced neural networks to model motion dynamics as well. Recently, researchers [7,24] typically regard motion prediction as a sequence-to-sequence problem [9,29] similar to machine translation. Therefore, they propose recurrent neural networks (RNN) and variants, to address motion prediction problems. Although these RNN models achieve better 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 spatial information which resembles the compli-

**Electronic supplementary material** The online version of this article (https://doi.org/10.1007/s00371-019-01692-9) contains supplementary material, which is available to authorized users.

✉ Xiaosong Yang
xyang@bournemouth.ac.uk

1 Bournemouth University, Bournemouth, UK

2 Nanjing Institute of Advanced Artificial Intelligence, Nanjing, China

3 Horizon Robotics, Nanjing, China

4 Northwest A&F University, Shaanxi, China

cated human body structure and mechanical restrictions but also the temporal information.

Li et al. [23] propose 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. Butepage et al. [5] demonstrate in experiments that the convolution structure is not as effective as the fully connected structure and hierarchical layers can improve the performance significantly. However, their work has limitations that 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, we proposed a novel convolutional hierarchical autoencoder (CHA) framework to address the motion prediction problem. We designed a new encoder network incorporated Hierarchical structure with 1D convolution layers to capture the tree structures of the human body and its temporal information at the same time. Compared to RNN, FCN and CNN networks, it has a much lower computational complexity and very small memory size but converges faster and more effective. We adopt this convolution hierarchical module for motion prediction tasks. Extensive experiments are conducted to demonstrate our CHA model's ability to improve the performance in CMU data and Human3.6M (HM3.6) data in both short-term and long-term motion prediction.

Our contribution consists of four fold: (1) We propose a new convolutional hierarchical autoencoder model to address motion prediction problem and outperform state-of-the-art results. (2) Our model is running significantly more efficiently compared to the CNN model. (3) Being mindful of the motion data characteristics, we incorporated the 1D convolutional layers with hierarchical structures to exploit the human body constrains. (4) Our model can generate high fidelity motion sequences for both short-term prediction and long-term prediction on CMU dataset and H3.6M dataset.

## 2 Related work

Motion data is a typical time series, so the traditional work follows the statistic model of the hidden Markovian model (HMM) [3] which is widely used in machine translation and speech recognition, such as the Gaussian processes model [33,34] and the conditional restricted Boltzmann machine [28,30–32]. But all of them are hard to generalize for more diverse and complicated actions.

Recently, there are increasing amount of researchers dedicated to developing a deep learning framework for motion prediction. They significantly improved the performance both quantitatively and qualitatively. We summarized the recent deep learning approaches of motion prediction as follows:

Most existing network architectures adopt the RNN module to solve the time series problems like machine translation [18,19] and stock forecasting [20,26]. The encoder–recurrent–decoder (ERD) [7] is a typical encoder–decoder model which incorporates representation learning and temporal dynamics learning together by installing an encoder and a decoder network before and after the recurrent layers. The cyclic way to predict frames iterates errors and generates unrealistic poses. They also considered a noise schedule to tackle these problems. However, the noise schedule is very inconvenient in the practical case. Jain et al. [17] propose an SRNN model which combines multiple RNN structures under the spatiotemporal graphs to improve the performance. However, these multiple RNN models are very time consuming to train. ERD and SRNN are action-specific models which are trained separately for each action type so that these models do not explore the real strength of deep learning techniques to benefit from a large dataset. Furthermore, other researchers develop multi-action models to predict motions for diverse types of actions together. Ghost et al. [10] introduce a dropout autoencoder network (DAE) to learn the inherent human body structures along with a 3-layer LSTM to learn the temporal dynamics. Martinez et al. [24] achieve higher accuracy both at short-term and long-term motion prediction by applying three changes on the typical RNN model. They introduce the sequence to sequence and Residual architecture, as well as a sampling-based loss plan for robustness to avoid hyper-parameter tuning. Following Martinez's work, Gui et al. [11] added two discriminators on their system to improve the quality of the predicted motion and alleviate the error accumulation problem. These RNN architectures beat down the traditional Markov methods, but they still have high complexity and limited performance on aperiodic actions. Moreover, they are prone to generate mean poses in long-term prediction.

Due to RNN architectures' lack of ability to extract spatial information and their tendency to produce noisy data, researchers propose various ideas as supplement, such as co-training, hand-crafted spatial graphs and adversarial networks. By these strategies, the limitations of RNN are alleviated but not completely solved. Therefore, some recent approaches consider different encoder layers to replace the recurrent layers entirely. Li et al. [23] propose a first convolutional sequence-to-sequence model for motion prediction

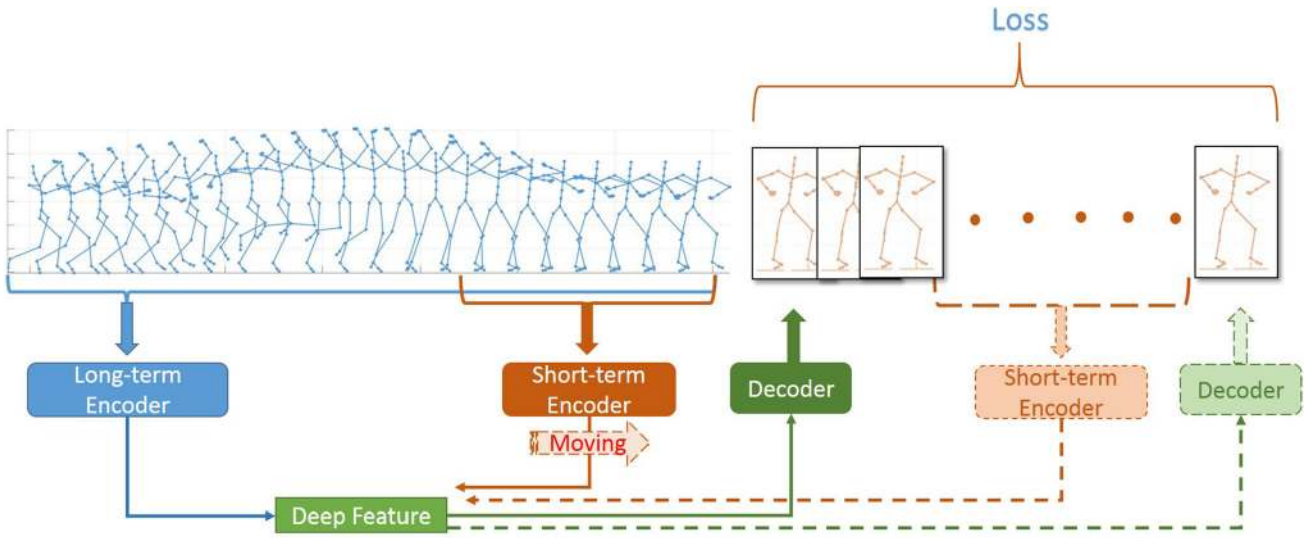


**Fig. 1** The architecture of our convolutional hierarchical autoencoder model. The orange and green solid boxes are the initial state of the short-term encoder and decoder. They will produce the future frame recursively. The orange and green dashed boxes are the final stage of short-term encoder and decoder. They will stop moving after predicting all the frames

and validate the effectiveness, on both the HM3.6M and CMU dataset. Different from the typical encode–decode model, they design a long-term and a short-term encoder for the entire input frames and short-term neighbouring frames, respectively. Besides, they add a simple two-layer fully connected network as a discriminator to enhance the quality. However, Butepage et al. [5] argue that convolutional layers are less effective than hierarchical layers since the 2D convolutional computation does not fit with the hierarchical human body structure. They prove that the fully connected hierarchical layer works significantly better in the prediction tasks. But the deep fully connected networks have high computational complexity and are prone to overfitting. Moreover, their framework sacrifices qualitative performance to improve the quantitative performance due to the decoder's design. The limbs' length varies during the training and prediction, and the output sequence may shake with noise for the reason that the framework lost the logical coherent information between frames in the decoder. Researchers [6,8,27] also considered the human hierarchical structures with five body parts. However, their hierarchical structure increased the computation complexity. Therefore, we propose a new model which contains the more reasonable convolutional hierarchical encoder structure that has much lower complexity than all the other modules and achieves the state-of-the-art performance both quantitatively and qualitatively.

## 3 Methodology

An overview of our convolutional hierarchical autoencoder (CHA) model is shown in Fig. 1. The seed motion clip (input) will be propagated in an autoencoder system to generate the future frames. In the autoencoder, we designed two encoders with convolutional hierarchical modules, which consisted 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 $C$ 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, we also incorporated a residual link to avoid the gradient vanishing problem. Therefore, the decoder will produce the output sequences recursively. To address the mean pose problem and accelerate convergence, we designed a D-loss function that assigns a series of diminished weights to the frames.

### 3.1 The mathematical formulation

The human motion data in this paper refers to the MoCap 3D skeleton data with joints. A sequence of motion data can be written as $X = \{f_1, f_2, \ldots, f_t, \ldots, f_n\}$. $f_t = (a_1, a_2, \ldots, a_K) \in \mathbf{R}^{3K}$ denoted the frame at current time $t$, $K$ is the number of joints, and $a_i$ is the exponential map representation [4] of joints. Similar to the standard procedure [11,17,23,24], we normalized the exponential map 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. We have a set of motion clips $\mathfrak{A} = \{X^i, i = 1, \ldots, N\}$. In the training

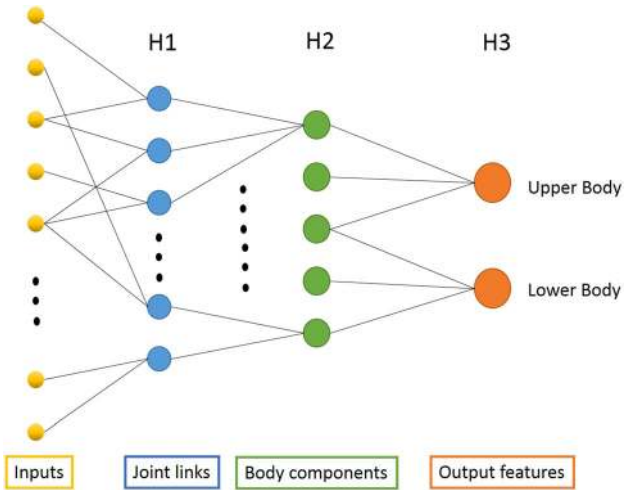


**Fig. 2** The convolution hierarchical layers in our framework. The first layer contains the same number of neurons $M_2$ as the input frame feature dimension. Then, the neurons from adjacent joints are linked together to one neuron in the secondary layer. Two neurons are linked together if and only if the related $a_i$ and $a_j$ represents 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 $H_2$ layer. $H_2$ consists of five neurons, and $H_3$ consists of two neurons. All the nodes' output have explainable semantic meanings

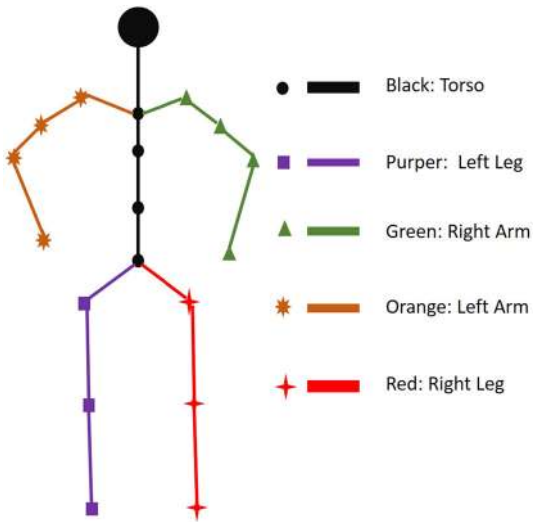


**Fig. 3** The five body components. Note that the joints in the figure do not equal to the same number of joints used in experiments

stage, if the input motion $X^i = \{f_1, f_2, \ldots, f_t, \ldots, f_n\} \in \mathfrak{A}$ has a length $n$, the $m$ future ground truth frames are denoted as $X_f = \{f_{n+1}, f_{n+2}, \ldots, f_{n+m}\}$. This algorithm aims to generate the future frames $\hat{X} = \{\widehat{f_{n+1}}, \widehat{f_{n+2}}, \ldots, \widehat{f_{n+m}}\}$, which makes the distance function $D(X_f, \hat{X})$ as small as possible.

### 3.2 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 (Fig. 2). 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 structures 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 sectional and holistic details of motion data than RNN and CNN. We utilize 1D convolutional layers to capture the temporal information along with the spatial constraints and reduce the model complexity at the same time.

Our convolution hierarchical module consisted of four layers, three convolutional hierarchical layers $H1$, $H2$ and $H3$, which are illustrated in Fig. 2, and a fully connected layer. Intuitively, $H1$ stands for the information of each joint link in the human body. And $H2$ extracts information separately for five parts of the human body, which is illustrated in Fig. 3. $H3$ extracts the deep representations for two parts of the human body, upper and lower. Therefore, the deep features of our 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 features $M_1 \times j_1$ 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 (Fig. 3). 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 1.

This design of structure brings two benefits. Firstly, the hierarchical network structures blend the human body constrains in feature extraction. Compared to the CNN model [23], our 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, our 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

**Table 1** The architecture of 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) | – | – | – | – | – |

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. We will discuss the model complexity on details in Sect. 4.4.

### 3.3 The autoencoder framework

A deep sparse autoencoder system [23] is widely used for synthesis problems. In this work, we used an autoencoder system as the generator to produce $\hat{X}$.

Two encoders are employed in this autoencoder system. The first encoder network aims to map the whole $n$ frames in the input sequence $X = \{f_1, f_2, \ldots, f_t, \ldots, f_n\}$ 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 $X_C^t = \{f_{t-C+1}, f_{t-C+2}, \ldots, f_t\}$ 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 $V_s^t$ are 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.

$$V_l = E_l(X|W_l),\ W_l \text{ is the parameters of } E_l \tag{1}$$
$$V_s^t = E_s(X_C^t|W_s),\ W_s \text{ is the parameters of } E_s \tag{2}$$
$$V^t = [V_l, V_s^t] \tag{3}$$

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

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

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-dimensional 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:

$$\widehat{f_{t+1}} = D(V^t|W_D),\ W_D \text{ is the parameters of } D \tag{4}$$

In recursive cases, the residual link usually works better than producing the next status directly. So we designed the decoder with a residual link as well. The formula of Eq. 4 is rewritten as:

$$\widehat{f_{t+1}} = D(V^t|W_D) + \widehat{f_t},\ W_D \text{ is the parameters of } D \tag{5}$$

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.

### 3.4 The objective function

Inspired by [7,17,23,24], we used the $l_2$ loss function of two motions, which measures their difference by summing up the mean squares error (MSE) of the Euler angles of all frames.

The objective function of our convolutional hierarchical autoencoder model is:

$$min\ \mathscr{L}_E(\hat{X}, X_f) + \lambda \|W\|_2 \tag{6}$$

The predicted motion is written as:

$$\hat{X} = \{\widehat{f_{n+1}}, \widehat{f_{n+2}}, \ldots, \widehat{f_{n+m}}\} \tag{7}$$

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

$$X_f = \{f_{n+1}, f_{n+2}, \ldots, f_{n+t}, \ldots, f_{n+m}\} \tag{8}$$

The original Euclidean distance loss function is defined as:

$$\mathscr{L}_E(\hat{X}, X_f) = \|\hat{X} - X_f\| = \sum_{t=n+1}^{n+m} \|\hat{f}_t - f_t\|_2 \tag{9}$$

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

$$min \sum_{t=n+1}^{n+m} \|\hat{f}_t - f_t\|_2 + \lambda(\|W_E\|_2 + \|W_D\|_2) \tag{10}$$

where the $\lambda$ controls the balance of different loss sources. The $W_E$ and $W_D$ are the parameters of the encoders and decoder in the hierarchical convolutional model. The term $\lambda(\|W_E\|_2 + \|W_D\|_2)$ is a $l_2$ regularizer to prevent overfitting.

**Remarks**

We designed a D-loss function 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 prevents to produce the mean pose.

The iteration loss function with assigned diminishing weights for frames is:

$$\mathscr{L}_A(\hat{X}, X_f) = \left(\sum_{t=n+1}^{n+m} \eta^{t-n} \|\hat{f}_t - f_t\|_2\right) / \left(\sum_{t=n+1}^{n+m} \eta^{t-n}\right) \tag{11}$$

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 geodesic loss [11] as well.

### 3.5 Implementation Details

The input sequence has a length of 50 frames so that the first hierarchical layer has an input vector of $50 \times 54$ ($50 \times 70$ for CMU) and the predicted sequences have a length of 25. We set the short-term encode to have an input length of 20. We used a small batch size 16 and a learning rate of $5e^{-5}$. For the parameter $\eta$, which aims to control the error generation during the experiment, we found 0.9 is a good value during experiment experience. We used a NVIDIA GPU 1080Ti and trained our full model in Tensorflow [1].

## 4 Experiments

To validate our model, we conduct extensive experiments of motion prediction on the existing benchmarks—H3.6M dataset and CMU motion dataset. Generally, the previous motion prediction task follows the same standard of experiments, which tests their model for short-term prediction and long-term prediction. We included the state-of-the-art baselines of motion predictions as the comparison. The experiments results demonstrated that the prediction accuracy of our model beats down the state-of-the-art baselines on diverse actions of H3.6M and CMU. Meanwhile, we illustrate that our model also produces more plausible human-like movements than baselines. Besides, we also discuss the efficiency of our model in terms of computational complexity and parameters in Sect. 4.4.

### 4.1 Dataset

Following the standard comparison [23], we provide motion prediction experiments on the two widely used motion benchmarks—H3.6M [16] 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. We use the 3D skeleton format, which has 32 joints in total to represent the human body structure. In our data process, each frame is recorded as the relative rotation of each joint, which is mathematically converted into an exponential map. There is performance 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 our experiment. Each type of action has two performance trials among 3000–5000 frames. Therefore, we select 180 trials of H3.6M 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 most widely evaluated in these motion prediction mechanisms. We provide the experiments not only on these four but also on all the other actions as well.

**CMU** This dataset has a more wide range of action types than H3.6M. It gives out a large amount of skeleton-based MoCap data around 2605 sequences which represent six categories and 23 subcategories (actions). Contrast to H3.6M, each action type contains different amounts of trials and their 3D-skeletons consist of 31 joints. There are various sports and physical activities included, such as basketball, soccer

and jumping. We use the same subset selected by [23] 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 five trials for training and one trial for testing.

The two datasets are preprocessed in the same way. Due to data normalization being an important factor affecting the network's performance, we normalize all the human pose data into mean value resulting to 0, and standard deviation resulting to 1. Therefore, the root point of all poses is 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 25 Hz, 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. [24] and Gui et al. [11] use losses which calculate the exponential maps and the orientation groups' difference between the predicted motion and the ground truth, respectively. In this paper, we consider the Euclidean difference of exponential maps in the loss function as well.

### 4.2 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) [7]
2. Dubbed the dropout autoencoder LSTM (DAELSTM) [10]
3. Residual Recurrent sequence-to-sequence model for human motion modelling (RRNN) [24]
4. Convolutional sequence-to-sequence model for human Dynamics (CNNHD) [23]

We reproduce the experiments of CNNHD and RRNN from their public implementation on GitHub. However, the remaining models do not public their implementation code. Therefore, we quote their results from the existing publications [24] for error comparison. We also compare the quality of the predicted sequences with CNNHD. For the short-term prediction, we quote the results reported from CNNHD [23]. For the long-term prediction, we implement the CNNHD model from their public code and train their model with the same settings in [23]. Because all these models follow the standard procedure of data processing and the same evaluation, this comparison is impartial.

### 4.3 Evaluation methods

From a practical perspective, the predicted motion should be accurate and look plausible at the same time. Therefore, we evaluate our model in three aspects: complexity, qualitative and quantitative performance. We present all the short-term results of diverse action types and also the more challenging long-term prediction accuracy.

1. To evaluate the efficiency of our model, we conduct the discussion in Sect. 4.4. Usually, RNN models have a higher complexity compared to CNN models. We compared the our CHM's complexity and size with the state-of-the-art CNN-based model. Li et al. [23] proposed a CNN-based autoencoder system with convolutional encoding module (CEM) which outperformed all the RNN models.
2. To evaluate the prediction performance, we provide the average mean square error (MSE) of the Euler angles between the predicted motion and ground truth. We conduct short-term prediction for different durations: 80 ms (2 frames), 160 ms (4 frames), 320 ms (8 frames), 400 ms (10 frames). In addition, we also conduct the long-term predictions for four durations: 560 ms (14 frames), 720 ms (18 frames), 840 ms (21 frames), 1000 ms (25 frames). The results and analysis are shown in Sects. 4.5 and 4.6 with Table 2, 3, 4, 5, 6 and 7.
3. To evaluate the quantitative performance of our model, we illustrate the representative prediction sequences in Figs. 4, 5 and 6 following the same settings in [5,11,23, 24]. The attached demo shows the predicted results in video, which demonstrates the smoothness, fidelity and similarity of the proposed method.

### 4.4 Size and speed

CNN has the lowest model complexity compared to the FCN and RNN structures. But the design of our convolution hierarchical network even greatly reduced the model complexity compared to CNN. We calculate the complexity of CEM and our CHM here.

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

**Table 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 [7] | 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 [10] | 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 [24] | 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 [23] | 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** |

Bold values indicate the lowest MSE of Euler Angle

**Table 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** |

Bold values indicate the lowest MSE of Euler Angle

CEM consists of three 2D convolution layers and one fully connected layer. CHM consists of three hierarchical layers and one fully connected layer. We will calculate the computational complexity $A$ and the parameters' number $B$ for each layer separately and then sum them up.

#### 4.4.1 Comparison of computational complexity

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

$$A = M_1 \times M_2 \times C_{\text{out}} \times ((K_1 \times K_2 + 1) \times C_{\text{in}} + (C_{\text{in}} - 1)) \tag{12}$$

We denote the number of nodes in each hierarchical layer as $h_i$; then, the computation cost of one $H_i$ layer in CHM is calculated as:

$$A_i = h_i \times M_1 \times C_{\text{out}} \times ((K_1 \times K_2 + 1) \times C_{\text{in}} + (C_{\text{in}} - 1)) \tag{13}$$

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

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

$$A_i^{CEM} : A_i^{CHM} = M_2 : h_i \tag{14}$$

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

**Table 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** |

Bold values indicate the lowest MSE of Euler Angle

**Table 5** The short-term prediction error of eight 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 |

Bold values indicate the lowest MSE of Euler Angle

**Table 6** The long-term prediction error of eight 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** |

Bold values indicate the lowest MSE of Euler Angle

**Table 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** |

Bold values indicate the lowest MSE of Euler Angle

$$\begin{aligned} A_F^{CEM} &= M_1 \times M_2 \times C_{\text{out}} \times F \\ A_F^{CHM} &= M_1 \times C_{\text{out}} \times F \\ A_F^{CEM} &: A_F^{CHM} = M_2 \end{aligned} \tag{15}$$

Since $h_i$ is much smaller than $M_2$, our CHM model has much lower computational complexity under the same settings. For experiment convenience, we use a CHM with slightly different parameters which are written in Table 1. We will obtain the computational cost ratio of two networks:

$$\frac{\text{Computational cost of } CEM}{\text{Computational cost of } CHM} \approx \frac{1,240M}{302M} \approx 4.1 \tag{16}$$

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 our model allows them to implement ideas and tasks faster.

#### 4.4.2 Comparison of the parameters

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

$$B = K_1 \times K_2 \times C_{\text{in}} \times C_{\text{out}} \tag{17}$$

The parameter number of a $H_i$ layer is calculated as:

$$B_i = K_1 \times K_2 \times C_{\text{in}} \times C_{\text{out}} \times h_i \tag{18}$$

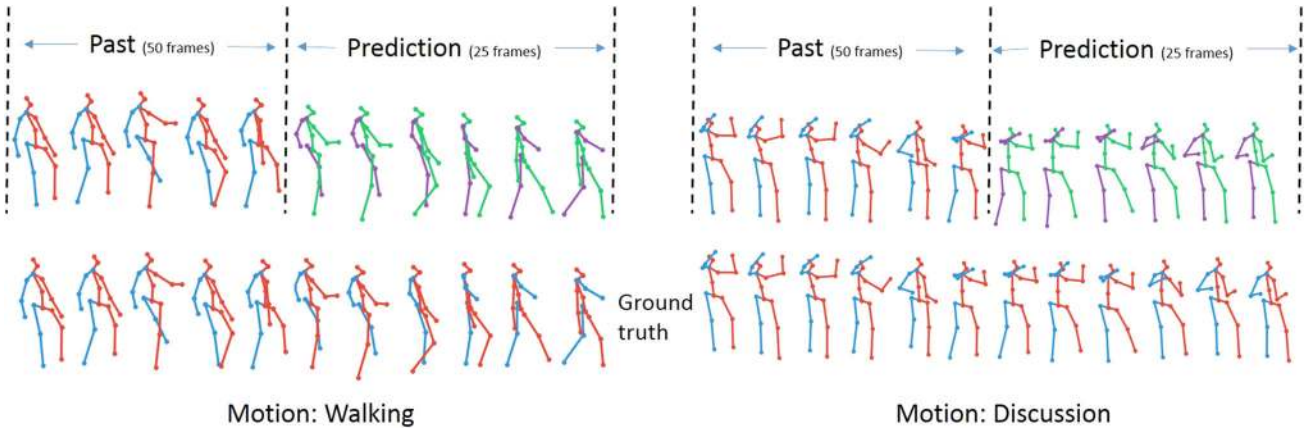


**Fig. 4** The illustration of the prediction result of 1000 ms on H3.6M dataset. The blue and red skeletons are ground truth frames. The green and purple skeletons on the top are the prediction results of CHA model

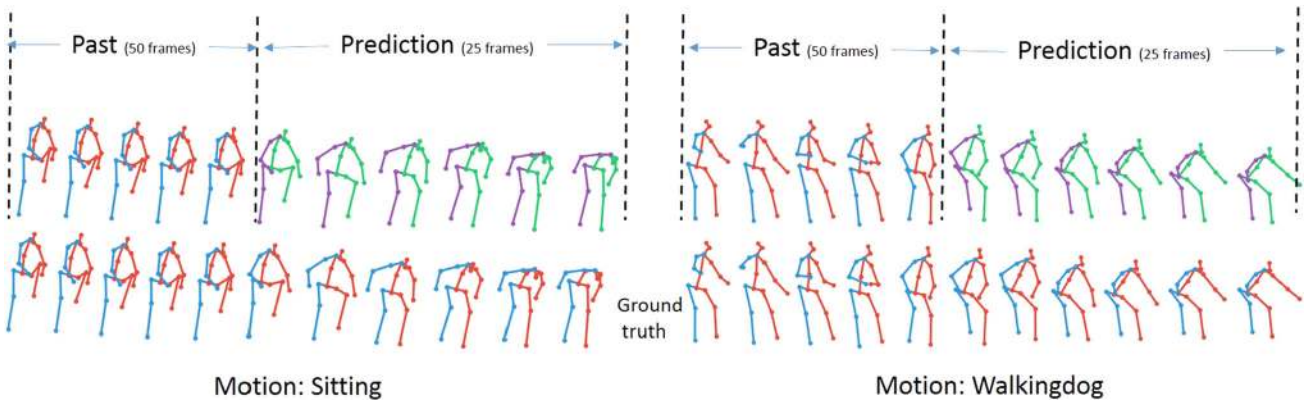


**Fig. 5** The illustration of the prediction result of 1000 ms on H3.6M dataset. The blue and red skeletons are ground truth frames. The green and purple skeletons on the top are the prediction results of the CHA model

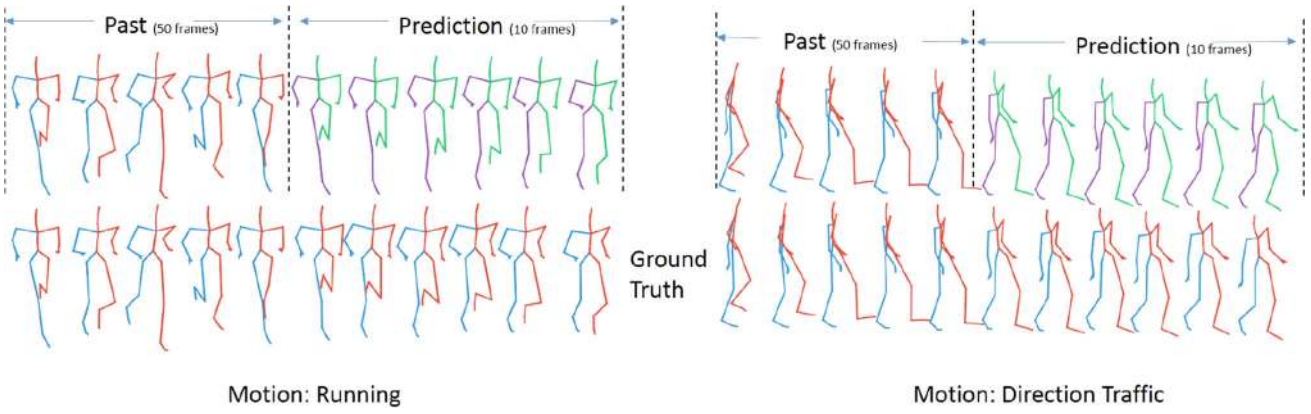


**Fig. 6** The illustration of the prediction result of 400 ms on CMU dataset. The blue and red skeletons are ground truth frames. The green and purple skeletons on top represent the prediction results of the CHA model

The parameter number of the fully connected layers is:

$$\begin{aligned} B_F^{CEM} &= M_1 \times M_2 \times C_{\text{out}} \times F \\ B_F^{CHM} &= M_1 \times C_{\text{out}} \times F \\ B_F^{CEM} &: B_F^{CHM} = M_2 \end{aligned} \tag{19}$$

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

$$\text{Total parameter} = \textstyle\sum_{l=1}^{L} B_l \tag{20}$$

Because $B$ and $B_i$ are significantly smaller than $B_F^{CEM}$ and $B_F^{CHM}$, our CHM will have a smaller size under the same settings as well. To be more specific, we calculate the precise number of parameters with the setting in Table 1:

$$\frac{\text{The parameters in } CEM}{\text{The parameters in } CHM} \approx \frac{177M}{11M} \approx 15.1 \tag{21}$$

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 are limited. We observed that complex models are prone to overfitting on the small amount of motion data. Therefore, our model alleviates this problem and it is more suitable for small sample learning.

### 4.5 H3.6M experiment results

There are four types of actions, walking, eating, smoking and discussion, which are commonly used as benchmarks in comparison. Therefore, we present our results of the short-term prediction of these four actions in Table 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. Our model beats down all the results of four actions in terms of 80 ms, 160 ms, 320 ms and 400 ms, except one. For the 400 ms walking prediction, our model actually achieved 0.735 which is almost the same to 0.73 of the CNNHD model. For the 80 ms and 160 ms walking prediction, our model improved 0.06 significantly. Even for the aperiodic action discussion, our model outperforms all the other baselines completely with a maximum of 0.8 Euler angle error reduction.

For a more general comparison, we display the 12 remaining actions' results in Table 3. Compared to ERD, DAELSTM and RRNN, our model almost outperforms on every action. However, compared to CNNHD, our model shows a different preference of actions. Half of actions improved but half decreased. Therefore, we calculate the average error to demonstrate a fair comparison. It shows that our 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.

The long-term prediction result is shown in Table 4. Because the ERD, DAELSTM and RRNN models did not provide about their long-term accuracy and CNNHD achieved best performance of them, we 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 4, our model improves all the performance of diverse actions especially in the long term. In the most challenging long-term task, motion prediction for 1000 ms, our model outperforms significantly on almost every action.

The visualization is shown in Figs. 4 and 5. Both for periodic motions like walking and aperiodic motions like discussion, our model produces plausible and high fidelity predictions which are very similar to the ground truth. Besides, our model avoids generating mean poses in long-term prediction like RNN models (Fig. 5).

### 4.6 CMU experiment results

In order to demonstrate our model's generalization ability, we trained our model for eight actions from CMU data as well. Only one existing work, CNNHD, provides their results on CMU dataset. We give out the prediction error of these actions in Table 5 and Table 6. We compared the short-term prediction ability first. It shows that almost all the action improved their accuracy in some terms. More than 60% of our results outperform the CNNHD. In the 80 ms and 320 ms of running, our errors are 0.285 and 0.525 precisely, which are very close to CNNHD. We also calculate the average error of these eight actions, and they demonstrate our model achieved a better performance than CNNHD.

For long-term prediction, our model outperforms more than half of the result of the CNNHD model. Similar to the H3.6M dataset, our model produces an unbalanced improvement. Therefore, we provide the average error of the eight actions. The results in Table 7 demonstrate that our 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 our model has ability to alleviate the error accumulation in long term.

The visualization of our data is shown in Fig. 6. 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. We can see our model predict the left arm and left foot more accurate than the CNNHD model.

## 5 Conclusion

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

**Acknowledgements** We would like to thank Li Wang, Nan Xiang and Ming Zhang for their help of this research.

**Funding** This study was funded by EU H2020 under the REA grant agreement (Grant Number 691215) and the Key Laboratory of Agricultural Internet of Things, Ministry of Agriculture and Rural Affairs, Yangling, Shaanxi 712100, China (2018AIOT-09) and the Automation Fellow in the South West Creative Technology Network.

### Compliance with ethical standards

**Conflict of interest** Jianjun Zhang has received research grants from EU H2020. Meili Wang has received research grants from China (2018AIOT-09). Yanran Li has received research grants from the Automation Fellow in the South West Creative Technology Network. The rest authors declare that they have no conflict of interest.



## References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. OSDI **16**, 265–283 (2016)
2. Akhter, I., Simon, T., Khan, S., Matthews, I., Sheikh, Y.: Bilinear spatiotemporal basis models. ACM Trans. Graph. **31**(2), 17 (2012)
3. Brand, M., Hertzmann, A.: Style machines. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 183–192. ACM Press/Addison-Wesley Publishing Co. (2000)
4. Bregler, C., Malik, J.: Tracking people with twists and exponential maps. In: 1998 Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1998, pp. 8–15. IEEE (1998)
5. Bütepage, J., Black, M.J., Kragic, D., Kjellström, H.: Deep representation learning for human motion prediction and classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p. 2017. IEEE (2017)
6. Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1110–1118 (2015)
7. Fragkiadaki, K., Levine, S., Felsen, P., Malik, J.: Recurrent network models for human dynamics. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4346–4354 (2015)
8. Garbade, M., Gall, J.: 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 (2016)
9. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122 (2017)
10. Ghosh, P., Song, J., Aksan, E., Hilliges, O.: Learning human motion models for long-term predictions. In: 2017 International Conference on 3D Vision (3DV), pp. 458–466. IEEE (2017)
11. Gui, L.Y., Wang, Y.X., Liang, X., Moura, J.M.: Adversarial geometry-aware human motion prediction. In: ECCV, pp. 823–842 (2018)
12. Gui, L.Y., Zhang, K., Wang, Y.X., Liang, X., Moura, J.M., Veloso, M.: Teaching robots to predict human motion. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 562–567. IEEE (2018)

13. Holden, D., Komura, T., Saito, J.: Phase-functioned neural networks for character control. ACM Trans. Graph. **36**(4), 42 (2017)
14. Holden, D., Saito, J., Komura, T.: A deep learning framework for character motion synthesis and editing. ACM Trans. Graph. **35**(4), 138 (2016)
15. Holden, D., Saito, J., Komura, T., Joyce, T.: Learning motion manifolds with convolutional autoencoders. In: SIGGRAPH Asia 2015 Technical Briefs, p. 18. ACM (2015)
16. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3. 6m: large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Trans. Pattern Anal. Mach. Intell. **36**(7), 1325–1339 (2014)
17. Jain, A., Zamir, A.R., Savarese, S., Saxena, A.: Structural-rnn: Deep learning on spatio-temporal graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5308–5317 (2016)
18. Kalchbrenner, N., Blunsom, P.: Recurrent continuous translation models. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1700–1709 (2013)
19. Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A.v.d., Graves, A., Kavukcuoglu, K.: Neural machine translation in linear time. arXiv preprint arXiv:1610.10099 (2016)
20. Kamijo, K.i., Tanigawa, T.: Stock price pattern recognition—a recurrent neural network approach. In: 1990 IJCNN International Joint Conference on Neural Networks, 1990, pp. 215–221. IEEE (1990)
21. Koppula, H.S., Saxena, A.: Anticipating human activities using object affordances for reactive robotic response. IEEE Trans. Pattern Anal. Mach. Intell. **38**(1), 14–29 (2016)
22. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: ACM SIGGRAPH 2008 classes, p. 51. ACM (2008)
23. Li, C., Zhang, Z., Lee, W.S., Lee, G.H.: Convolutional sequence to sequence model for human dynamics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5226–5234 (2018)
24. Martinez, J., Black, M.J., Romero, J.: On human motion prediction using recurrent neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4674–4683. IEEE (2017)
25. Pérez-D'Arpino, C., Shah, J.A.: Fast motion prediction for collaborative robotics. In: IJCAI, pp. 3988–3989 (2016)
26. Rather, A.M., Agarwal, A., Sastry, V.: Recurrent neural network and a hybrid model for prediction of stock returns. Expert Syst. Appl. **42**(6), 3234–3241 (2015)
27. Shao, Z., Li, Y., Guo, Y., Zhou, X., Wang, Z., Yang, J., Chen, S.: A hierarchical model for human action recognition from body-parts. IEEE Trans. Circuits Syst. Video Technol. (2018)
28. Sutskever, I., Hinton, G.E., Taylor, G.W.: The recurrent temporal restricted Boltzmann machine. In: Advances in Neural Information Processing Systems, pp. 1601–1608 (2009)
29. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
30. Taylor, G.W., Hinton, G.E.: Factored conditional restricted Boltzmann machines for modeling motion style. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1025–1032. ACM (2009)
31. Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. In: Advances in Neural Information Processing Systems, pp. 1345–1352 (2007)
32. Taylor, G.W., Sigal, L., Fleet, D.J., Hinton, G.E.: Dynamical binary latent variable models for 3d human pose tracking. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 631–638. IEEE (2010)
33. Urtasun, R., Fleet, D.J., Geiger, A., Popović, J., Darrell, T.J., Lawrence, N.D.: Topologically-constrained latent variable models. In: Proceedings of the 25th International Conference on Machine Learning, pp. 1080–1087. ACM (2008)
34. Wang, J.M., Fleet, D.J., Hertzmann, A.: Gaussian process dynamical models for human motion. IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 283–298 (2008)
35. Wang, Y., Che, W., Xu, B.: Encoder-decoder recurrent network model for interactive character animation generation. Vis. Comput. **33**(6–8), 971–980 (2017)



**Yanran Li** is currently a Ph.D Researcher in the National Centre for Computer Animation, Bournemouth University, UK. She received her MSc degree in Mathematics from the University of Science and Technology of China and Bachelor degree in Applied Mathematics from Xian Jiaotong University. Her research interests include computer graphics, computer vision, deep learning, motions and images.

**Zhao Wang** is currently a researcher in Institute of Advanced Artificial Intelligent in Nanjing. He received his Ph.D (2018) in National Centre for Computer Animation in Bournemouth University, UK. He has worked as Research Assistant in Walt Disney Imagineering Research & Development (2017) and Zhejiang Lab (2018). His research interests include 3D Computer Vision, Federated Learning, Model Compression and Acceleration and Computer Animation.

**Xiaosong Yang** is currently an Associate Professor in the National Centre for Computer Animation, Bournemouth University, UK. He received his bachelor (1993) and master degree (1996) in computer science from Zhejiang University (P. R. China) and Ph.D (2000) in computing mechanics from Dalian University of Technology (P. R. China). He worked as Post-Doc (2000C2002) in the Department of Computer Science and Technology of Tsinghua University for two years and as Research Assistant (2001–2002) at Chinese University of Hong Kong. His research interests include deep learning, computer vision, computer animation, motion capture and synthesis, VR&AR, special effects and game development, digital health, data mining, medical visualization. He has published over 70 papers in journals and refereed conferences.

**Meili Wang** is an associate professor at College of Information Engineering, Northwest A&F University. She received her Ph.D degree in computer animation in 2011 at the National Centre for Computer Animation, Bourne mouth University. Her research interests include computer graphics, geometric modelling, image processing, visualization and virtual reality.

**Sebastian Iulian Poiana** is currently a Masters Degree student in the National Centre for Internet of Things (IoT) with Cyber Security, Bournemouth University, UK. He received his BSc Undergraduate degree in Software Engineering also from Bournemouth University, Bournemouth, UK. His research interests include cyber security, Internet of things (IoT), computer animation, programming languages (Java, Python and C#) and Games.

**Ehtzaz Chaudhry** is currently a Postdoctoral Researcher in the National Centre for Computer Animation, Bournemouth University, UK. He received his MSc degree in Computer Game Graphics & Animation from the University of Westminster, London, UK and Ph.D in Computer Animation from Bournemouth University, UK. His research interests include computer graphics, computer animation, 3D modelling, VR/AR and games.

**Jian Jun Zhang** is Professor of Computer Graphics at the National Centre for Computer Animation, Bournemouth University. He is Head of the National Research Centre for Computer Animation. His research interests include computer graphics, computer animation, physically based simulation, geometric modelling, medical simulation and visualization.