Densely connected GCN model for motion prediction

Received: 2 July 2020 | Accepted: 15 July 2020

DOI: 10.1002/cav.1958

SPECIAL ISSUE PAPER

WILEY

Densely connected GCN model for motion prediction

Yanran Li[1] | Lingteng Qiu[2] | Li Wang[1] | Fangde Liu[3] | Zhao Wang[4] | Sebastian Iulian Poiana[5] | Xiaosong Yang[1] | Jianjun Zhang[1]

[1]National Centre for Computer Animation, Bournemouth University, Dorset, UK

[2]Harbin Institute of Technology, Shenzhen, China

[3]SurgicalAI.cn, Shenzhen, China

[4]China Horizon Robotics, Nanjing, China

[5]Bournemouth University, Dorset, UK

Correspondence
Xiaosong Yang, National Centre for Computer Animation, Bournemouth University, Dorset, UK.
Email: xyang@bournemouth.ac.uk

Funding information
EU H2020 under the REA grant agreement, Grant/Award Number: 691215; the Automation Fellow in the South West Creative Technology Network; the Key Laboratory of Agricultural Internet of Things, Ministry of Agriculture and Rural Affairs, Yangling, Shaanxi 712100, China, Grant/Award Number: 2018AIOT-09

Abstract

Human motion prediction is a fundamental problem in understanding human natural movements. This task is very challenging due to the complex human body constraints and diversity of action types. Due to the human body being a natural graph, graph convolutional network (GCN)-based models perform better than the traditional recurrent neural network (RNN)-based models on modeling the natural spatial and temporal dependencies lying in the motion data. In this paper, we develop the GCN-based models further by adding densely connected links to increase their feature utilizations and address oversmoothing problem. More specifically, the GCN block is used to learn the spatial relationships between the nodes and each feature map of the GCN block propagates directly to every following block as input rather than residual linked. In this way, the spatial dependency of human motion data is exploited more sufficiently and the features of different level of scale are fused more efficiently. Extensive experiments demonstrate our model achieving the state-of-the-art results on CMU dataset.

KEYWORDS

densely, GCN, motion prediction, Spatial temporal

1 | INTRODUCTION

Forecasting the future movements of human actions is a crucial topic in computer vision and computer graphics for its various practical applications in real life, such as surveillance,[1] pedestrian tracking,[2-4] interactive robotics,[5-7] and autonomous driving systems.[8] The data of human motions are usually captured by the Mocap system and represented in the format of the three-dimensional (3D) skeleton. In this paper, we address the problem of generating human action movements in the 3D skeleton format.

There were a lot of researchers attempting to propose various approaches for motion prediction. Most deep learning models treat motion prediction similar to machine translation problems and employ long short-term memory (LSTM)- or convolutional neural network (CNN)-based models.[9-16] However, different from machine translation, motion data has special human body constraints and is actually a spatial-temporal data rather than temporal data. The LSTMs are

Abbreviations: ANA, anti-nuclear antibodies, APC, antigen-presenting cells, IRF, interferon regulatory factor.

majorly designed for temporal data and they are not sufficient for capturing the dependency between joints on the human body. Several existing works[17,18] consider that the special hierarchical structure of the human body can enhance the performance to capture spatial information.

Recently, Graph Neural Networks (GNN) attracts increasing attention and achieved a significant margin of improvement on human motion tasks.[19-21] The reason is the human body is a natural graph structure, that is, the joints are nodes of the graph and connectivity is defined by the limbs. The GCN-based work[21] significantly outperforms those of the LSTM or CNN-based models for motion prediction. However, the existing GCN-based models are limited on feature utilization and under research 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, that is, 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, we propose an advanced GCN based framework 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, our model formulates the graph as the 3D skeleton of the human body. 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 Reference 22, we concatenate the different scale of features. Therefore, if our model has N GCN blocks, and the first GCN block has an output feature with size C, then the input features for the last GCN blocks will have size $C \times (N-1)$ and $N(N+1)/2$ links between blocks are built in our model.

Compared to the aforementioned GCN model,[21] our 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, our model has a regularizing effect and LeakyRELU layers reduce the overfitting on training.

In conclusion, our contribution in this paper are: We proposed a new Densely GCN-based model to address the problem of motion prediction. It reuses the multi-scale feature maps from every block to enlarge the receptive field and reduce overfitting problem. We conduct extensive comparison experiments on the standard benchmarks for motion prediction, which are Human3.6M, and the CMU motion capture dataset. The model is evaluated from both the angle and 3D position aspects, and it surpasses the state-of-the-art performance on CMU dataset.

2 | RELATED WORK

2.1 | Motion prediction

The mainstream of the existing deep learning models for motion prediction can be categorized as RNN-based, fully convolutional network (FCN)-based and CNN-based. In the early research, Fragkiadaki et al.[9] proposed a recurrent based Encoder-Recurrent Decoder model to address this problem. Following this trend, researchers put effort into designing various recurrent based models for this problem. Jain et al.[23] investigated the spatial-temporal structure of motion data and introduced the Structural-RNN model, which forms a spatial-temporal graph that trains a RNN on each node. Furthermore, inspired by the success of Seq2Seq model on machine translation, Martinez et al.[12] proposed a sequence to sequence model for motion prediction. They surprisingly find a simple zero-velocity baseline which outperforms all the existing complex models. All of the aforementioned strategies face a discontinuity problem and mean pose problem. To address the limitations, Gui et al.[11] built an adversarial model to distinguish the synthesised sequences and the real ones so that the performance of prediction can be lifted. Aside from these RNN-based models, researchers also attempt to solve the problem from other aspects. Instead of training the motion in angle space, Butepage et al.[17] proposed a new FCN-based model with three types of bottlenecks and learned the movements directly in the format of 3D positions. However, FCN-based models are easily overfitting and insufficient to capture the relationships between different body parts. To overcome this, Li et al.[16] designed a convolutional sequence to sequence model which used a long term and short term encoder to extract deep features.

2.2 | Graph neural network

Recently, researches concerning Graph Neural Networks have become increasingly active because of their superior ability to tackle irregular shaped data. Convolutional operations on the graphs have been investigated as well. The human body can be regarded as a natural graph, therefore recent approaches achieved distinctive success by introducing GCN for skeleton-related problems. A spatial-temporal graph[19] is introduced to address the motion classification problems and gains a considerable improvement. Furthermore, Li et al.[20] considered more relationship edges between human joints and enhanced the classification accuracy. In contrast to the undirect graph used for the skeletons, Shi et al.[24] introduced direct graph for the human body to exploit the dependencies between bones and joints. Zhao et al.[25] designed a SemGCN operation for 3D human pose regression. For motion prediction, Mao et al.[21] firstly proposed a GCN-based approach, which predicts the future sequences within the trajectory space instead of the pose space and outperformed all the existing RNN-, FCN-, and CNN-based approaches. To investigate further on the GCN-based model for motion prediction, we constructed a densely connected GCN-based model for motion prediction in this paper. Compared to the work of Reference 21, our model is more effective on learning feature maps and it is less likely to overfit and experience gradient vanishing problems.

3 | METHODOLOGY

In this section, we give out the details of our methodology to address the motion prediction problem. Firstly, the mathematic formulation of the motion prediction problem is described. Then the definition of our model's graph networks is depicted. After that, we show the structure of our networks and how it has been densely connected. The whole pipeline is shown in Figure 1.

3.1 | Problem formulation

In this section, we provide the mathematic formulation of the motion prediction problem. Assuming that the sequence of the 3D skeleton is $X_{1:T} = [x_1, x_2, \dots, x_t, \dots, x_T]$, t is the time step 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 orientations (α, γ, β) (the data usually removes the global rotation and translation). Therefore, we denote $x_t \in \mathbf{R}^{3\times K}$. This task aims to predict the future sequences $X_{T+1:T'}$. T' is the final frame. The ground truth future sequences are denoted as $GX_{T+1:T'}$ and the synthesized 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 $SX_{T+1:T'}$ looks plausible like the real human actions as well. Most of the traditional methods predict the future sequences by generating poses recursively. However, this type of method suffers from large 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, we follow the recent approach,[21] predict the future sequences in the trajectory space, and generate 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[12] is proved to have better performance. Specifically, we obtain a padding sequence by repeating the last pose x_T in the sequence $(T' - T)$ times, which can be written as $P_{T+1:T'} = [x_T, \dots, x_T, \dots, x_T]$. The input sequences for our model are the concatenation of $X_{1:T}$ and $P_{T+1:T'}$, which is denoted as $Input_{1:T'} = [x_1, x_2, \dots, x_T, \dots, x_T]$. The target sequences of our model are the concatenation of $X_{1:T}$ and $GX_{T+1:T'}$, which can be denoted as $Target_{1:T'} = [x_1, x_2, \dots, x_T, \dots, x_{T'}] = [X_{1:T}; GX_{T+1:T'}]$.

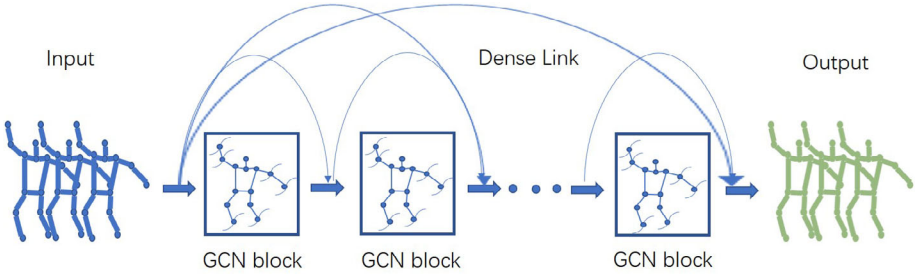


FIGURE 1 The overview of our model. Dense link shows how the feature maps propagate. Each GCN block shows the input of each node of the graph is the Discrete Cosine Transform of the trajectory

Therefore, our model is designed to take the $Input_{1:T'}$ and produce a synthesis sequence $Output_{1:T'}$. Then the objective function of our model measures the error between $Output_{1:T'}$ and $Target_{1:T'}$.

3.2 | Graph neural networks

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

3.2.1 | Graph formulation

The human body is of a natural graph structure. 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, we form our graph model 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 $\{J_0, J_1, \dots, J_K\}$ and the edge set E contains the graph edges which correspond to the limbs on the human skeleton. Usually, the adjacent matrix of our 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, we use this model to predict the graph connectivity, in other word, the edge set E is obtained from training rather than predefined. Moreover, we treat every joint as three nodes on the graph for they have the position $\langle x,y,z\rangle$. So our node set V is actually $\{J_{0x}, J_{0y}, J_{0z}, J_{1x}, J_{1y}, J_{1z}, \dots, J_{Kx}, J_{Ky}, J_{Kz}\}$ for practical use.

In our 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 Reference 21, we transform this trajectory 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 our Graph model is the DCT method's coefficients. The Graph model will output a feature F'_{ix}, which is the DCT coefficients as well. Then the F'_{ix} will be transformed back to the trajectory by the linear combination of the basis. Therefore, the Graph model takes in the trajectory information for every joint and then produces the output trajectory for every joint. As evident in the literature and experiments, predicting the residual data rather than the exact data will greatly reduce the gradient vanishing and gradient explosion problem, therefore, achieving better performance. We use the coefficients $F_{ix} + F'_{ix}$ as the final results and then use it to reconstruct the output trajectory. Therefore, we not only predict the future movements from time $T+1$ to T' but also reconstruct the movements 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, we will give out the layers used in obtaining output features F' from the input features F.

3.2.2 | Graph convolutional layers

There have been various kinds of convolutional layers introduced for graph data. Here, we adopt the graph convolutional layer[26] designed in the spectrum perspective.

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.

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

where x stands for the input signal to 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 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 $L = I - D^{-1/2}AD^{-1/2}$ which can be formulated as $L = U\Lambda U^T$.

Therefore, the Equation (1) can be rewritten as:

$$x * g = U(U^T x \odot U^T g) = U g_\theta U^T x. \tag{2}$$

Here, g_θ is the convolutional function operates on Λ. The Chebyshev polynomials are introduced to approximate g_θ. If only the first order of Chebyshev polynomials are considered, the equations can be written as:

$$x * g = \sum_{k=0}^{1} \beta_k T_k L x. \tag{3}$$

T_k is the kth order of Chebyshev polynomials. Finally, we assume 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 drived out:

$$F^{l+1} = \sigma(\tilde{Z}^l F^l W). \tag{4}$$

σ is an activation function. We used the $LeakRELU()$ as function σ here rather than $Tanh()$ used in previous work.[21] In our experiment, $\tilde{Z}$ is setting trainable to improve the performance because it can reduce overfitting.

3.3 | The densely connected network structure

After we explained our problem formulation and layer operations, we will describe the network architecture in our work, which is our 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, we estimate the residual part of features as well. Then the procedure passing through the lth GCN layer can be formulated as:

$$F^{l+1} = GCN(F^l) + F^l. \tag{5}$$

Densely connection. In the previous work,[21] the output of each GCN block is directly feed into the next block. We believe that 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 feature contains the 2-nearest information. The key idea is to produce a more informative input feature by fusing multiscale 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 $F^0, F^1, \ldots, F^l$ into the lth block to enlarge the ability of layers to exploit the hidden dependencies between joints in a different levels.

Therefore, we reconstruct the network structure by adding dense links on the network to increase its ability. Firstly, our 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 $l \times C$. Consequently, the formulation of the lth GCN block can be described as:

$$F^{l+1} = GCN([F^0, F^1, \ldots, F^l]). \tag{6}$$

Compared to Huang et al.,[22] this is the first time 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 not. For example, we can actually set the channel size of every output feature map F^l differently. Assuming their channel sizes are $C^0, C^1, C^2, \ldots, C^N$, then the input feature of lth GCN block has feature size $\sum_{k=0}^{l} C^k$. The benefit of that is we can use the same size of parameters but actually design a much deeper network. For a special case, we can set $C^1, C^2, \ldots, C^N$ 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 keep the model size at the same level. Meanwhile, the more narrow middle layers also help to reduce the overfitting problem.

4 | EXPERIMENTS

For evaluations, we empirically demonstrate our proposed models' effectiveness on the widely used benchmark Human3.6M[27] and CMU-Mocap[1]. Comprehensive experiments have been carried out to validate the superior ability of our model. The error results are reported both in the aspect of Euler Angles and 3D coordinates. The comparison results to the state-of-the-art work including RNN-based model random recurrent neural networks (RRNN),[12] CNN-based model convSeq2Seq,[16] and GCN-based model LearnTraj[21] are reported. In the end, we conduct ablation evaluations to investigate the impact of the proposed strategy.

4.1 | Implementation details

For a fair comparison, we follow the same set of other prior works.[21] The input feature size of the model is 15 and every GCN layer output a hidden feature with size 256. Every GCN block contain 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.

4.2 | Datasets

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,[12,16,21] 15 actions performed by six actors are selected for our experiment from Human3.6M(H3.6M). Three kinds of format are provided by H3.6M. Here we used the 3D skeleton format with 32 joints 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..[16] It contains a wider range of action types than H3.6M. Similar to H3.6M, we remove the global rotations and translations as well and normalize it. In total, eight actions (such as basketball, soccer, jumping and etc.) are selected under the prescriptions.[16] Every action set contains more than five trials. For our experiments, we use the same dataset splitting strategy like Li et al.[16]

4.3 | Evaluation baselines and metrics

Baselines. The existing approaches for motion prediction can be broadly categorized as RNN-, CNN-, and GCN-based methods. We select the models with the best performance and public codes so far in these three domains accordingly, which are RRNN,[12] convSeq2Seq,[16] and LearnTraj.[21] For the errors reported directly in the Euler angle space and 3D coordinates, we quote their results directly from papers. For the visualization comparison, we only compare our method to LearnTraj[21] and obtain the results from the public code[2]. We trained our model both in 3D position and orientations space.

Metrics. Two kinds of evaluation protocols are used in the experiments. Firstly, the traditional Euler angle error[12,16,21] 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

[1] http://mocap.cs.cmu.edu/
[2] https://github.com/wei-mao-2019/LearnTrajDep

has a smaller error but looks different at visual aspect. Therefore, another Mean Per Joint Position Error (MPJPE)[27] 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.

4.4 | Results

In this section, we report our performance in the given tables. The results of short-term (80, 160, 320, 400 ms) 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, we show the results of these four types of actions in Table 1. Four baseline

TABLE 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[12]	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[16]	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[21]	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 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 3 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
LearnTraj[21]	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)[21]	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[21]	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)[21]	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 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**

performances in the Euler Angle spaces is given out in this table. However, it has been found out that the Euler Angle error does not reflect the similarity visually.[21] So we train the model in 3D coordinate space and report their comparison results. LearnTraj achieved the smallest error in terms of 3D errors compared to the existing trending work. However, it can be seen from Table1 that our model surpasses it with a gap on serval actions as well. Moreover, the results of the rest 12 types of actions are reported in Table 2. Our methods outperform LearnTraj on majority of the action types in 3D coordinate spaces which demonstrates the effectiveness of our method.

CMU-Mocap: Similarly to H3.6M, the short-term results of the CMU-Mocap dataset are shown in Table 3 and 4. Firstly, the Euler Angle error and 3D error of eight actions are shown in Table 3. More than 80 % of the errors are smaller after using our method, 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, we report the average values of different time intervals. Table 4 demonstrates that our method achieved the state-of-the-art performance for the CMU-mocap dataset which volidate the effectiveness of our model.

5 | CONCLUSION

In this paper, we firstly introduced a densely connected GCN-based model for motion prediction task which enhance the feature maps utilization and reduced the overfitting problem. Experiments on heavily benchmarked databases validate the effectiveness of our model. The performance of 3D joints representation is better than the representation in angle space. The performance on CMU dataset is much better than on H3.6M datasets. Our methods beat down the state-of-the-art methodologies, therefore, shows the dense strategy is useful.

ACKNOWLEDGEMENTS

We would like to thank Yingyu Nie, Nan Xiang, Tao Jiang and Jinglu Zhang for their help of this research. 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.

CONFLICT OF INTEREST

J.Z. has received research grants from EU H2020. Meili Wang has received research grants from China (2018AIOT-09). Y.L. 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.

ORCID

Yanran Li https://orcid.org/0000-0003-1385-7604

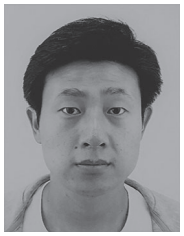
REFERENCES

1. Gaur U, Zhu Y, Song B, Roy-Chowdhury A. A εstring of feature graphsε model for recognition of complex activities in natural videos. Proceedings of the 2011 International Conference on Computer Vision. Barcelona, Spain: IEEE; 2011. p. 2595–2602.
2. Alahi A, Goel K, Ramanathan V, Robicquet A, Fei-Fei L, Savarese S. Social lstm: Human trajectory prediction in crowded spaces. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, Nevada; 2016. p. 961–971.
3. Gupta A, Martinez J, Little JJ, Woodham RJ. 3D pose from motion for cross-view action recognition via non-linear circulant temporal encoding. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Columbus, Ohio; 2014. p. 2601–2608.
4. Bhattacharyya A, Fritz M, Schiele B. Long-term on-board prediction of people in traffic scenes under uncertainty. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Salt Lake City, Utah; 2018. p. 4194–4202.
5. Koppula H, Saxena A. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. Proceedings of the International Conference on Machine Learning. Atlanta, USA; 2013. p. 792–800.
6. Koppula HS, Saxena A. Anticipating human activities using object affordances for reactive robotic response. IEEE Trans Pattern Anal Mach Intell. 2015;38(1):14–29.
7. Gui LY, Zhang K, Wang YX, Liang X, Moura JM, Veloso M. Teaching robots to predict human motion. Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Madrid, Spain: IEEE; 2018. p. 562–567.
8. Chen S, Liu B, Feng C, Vallespi-Gonzalez C, Wellington C. 3d point cloud processing and learning for autonomous driving; 2020. arXiv preprint arXiv:200300601.
9. Fragkiadaki K, Levine S, Felsen P, Malik J. Recurrent network models for human dynamics. Proceedings of the IEEE International Conference on Computer Vision. Santiago, Chile; 2015. p. 4346–4354.
10. Ghosh P, Song J, Aksan E, Hilliges O. Learning human motion models for long-term predictions. Proceedings of the 2017 International Conference on 3D Vision (3DV). Qingdao, China: IEEE; 2017. p. 458–466.
11. Gui LY, Wang YX, Liang X, Moura JM. Adversarial geometry-aware human motion prediction. Proceedings of the European Conference on Computer Vision (ECCV); 2018. p. 786–803.
12. Martinez J, Black MJ, Romero J. On human motion prediction using recurrent neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Hawaii, USA; 2017. p. 2891–2900.
13. Guo X, Choi J. Human motion prediction via learning local structure representations and temporal dependencies. Proceedings of the AAAI Conference on Artificial Intelligence. Hawaii, USA; 2019. vol. 33, p. 2580–2587.
14. Pavllo D, Grangier D, Auli M. Quaternet: A quaternion-based recurrent model for human motion; 2018. arXiv preprint arXiv:180506485.
15. Kundu JN, Gor M, Babu RV. Bihmp-gan: Bidirectional 3d human motion prediction gan. Proceedings of the AAAI Conference on Artificial Intelligence. Hawaii, USA; 2019. vol. 33, p. 8553–8560.
16. Li C, Zhang Z, Sun LW, Hee LG. Convolutional sequence to sequence model for human dynamics. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Utah, USA; 2018. p. 5226–5234.
17. Butepage J, Black MJ, Kragic D, Kjellstrom H. Deep representation learning for human motion prediction and classification. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Hawaii, USA; 2017. p. 6158–6166.
18. Li Y, Wang Z, Yang X, et al. Efficient convolutional hierarchical autoencoder for human motion prediction. Vis Comput. 2019;35(6-8):1143–1156.
19. Yan S, Xiong Y, Lin D. Spatial temporal graph convolutional networks for skeleton-based action recognition. Proceedings of the 32nd AAAI Conference on Artificial Intelligence. Louisiana, USA; 2018.
20. Li M, Chen S, Chen X, Zhang Y, Wang Y, Tian Q. Actional-structural graph convolutional networks for skeleton-based action recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. California, USA; 2019. p. 3595–3603.
21. Mao W, Liu M, Salzmann M, Li H. Learning trajectory dependencies for human motion prediction. Proceedings of the IEEE International Conference on Computer Vision. Seoul, Korea; 2019. p. 9489–9497.
22. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Hawaii, USA; 2017. p. 4700–4708.
23. Jain A, Zamir AR, Savarese S, Saxena A. Structural-rnn: Deep learning on spatio-temporal graphs. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Nevada, USA; 2016. p. 5308–5317.
24. Shi L, Zhang Y, Cheng J, Lu H. Skeleton-based action recognition with directed graph neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Long Beach, California, USA; 2019. p. 7912–7921.
25. Zhao L, Peng X, Tian Y, Kapadia M, Metaxas DN. Semantic graph convolutional networks for 3D human pose regression. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Long Beach, California, USA; 2019. p. 3425–3435.

26. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks; 2016. arXiv preprint arXiv:160902907.
27. 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. 2013;36(7):1325–1339.

AUTHOR BIOGRAPHIES

Yanran Li Yanran Li is currently a PhD 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 Xi'an Jiaotong University. Her research interests include Computer Graphics, Computer Vision, Deep Learning, Motions and Images.



Li Wang Li Wang is currently a PhD candidate at National Centre for Computer Animation, Bournemouth University, UK. He received his BS and ME degree in Computer Science from Jilin Universty (China) in 2013 and 2016, respectively. His research interests include deep learning, computer vision and computer graphics.

Fangde Liu Fagnde Liu is current CEO of SurgicalAI, a start up building AI for Surgical Robots in Hangzhou, China. He received PhD (2012) from National Centre for Computer Animation in Bournemouth University. He worked as Research Associate (2012-2016) at Imperial College London, on Surgical Robotics for neurosurgery and Data Science for Neurology. His research interest include image guided cardiac surgery, surgical robotics, motion synthesis for computer animation and motor control system for cooperative robotics.

Zhao Wang 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 Imageneering Research & Development (2017) and Zhejiang Lab (2018). His research interests include 3D Computer Vision, Federated Learning, Model Compression and Acceleration and Computer Animation.

Sebastian Iulian Poiana 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.

Xiaosong Yang 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 PostDoc (2000–2002) 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.

Jianjun Zhang 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 modeling, medical simulation and visualization.

Lingteng Qiu Lingteng Qiu received the B.S. degree from the Institute of Electrical and Automation Engineering, Fuzhou University, Fuzhou, China, in 2017. He is currently pursuing the M.Sc. degree with the Harbin Institute of Technology, Shenzhen, China. His research interests include machine learning, pattern recognition, and computer vision.

How to cite this article: Li Y, Qiu L, Wang L, et al. Densely connected GCN model for motion prediction. *Comput Anim Virtual Worlds*. 2020;31:e1958. https://doi.org/10.1002/cav.1958