Progressively Generating Better Initial Guesses Towards Next Stages for High-Quality Human Motion Prediction

Anonymous CVPR submission

Paper ID 4504

Abstract

This paper presents a high-quality human motion prediction method that accurately predicts future human poses given observed ones. Our method is mainly based on the observation that a good initial guess of the future pose sequence, such as the mean of future poses, is very helpful to improve the forecasting accuracy. This motivates us to design a novel two-stage prediction strategy, including an init-prediction network that just computes a good initial guess and a formal-prediction network that takes both the historical and initial poses to predict the target pose sequence. We extend this idea further and design a multi-stage prediction framework with each stage predicting initial guess for the next stage, which rewards us with significant performance gain. To fulfill the prediction task at each stage, we propose a network comprising Spatial Dense Graph Convolutional Networks (S-DGCN) and Temporal Dense Graph Convolutional Networks (T-DGCN). Sequentially executing the two networks can extract spatiotemporal features over the global receptive field of the whole pose sequence effectively. All the above design choices cooperating together make our method outperform previous approaches by a large margin (6%-7% on Human3.6M, 5%-10% on CMU-MoCap, 13%-16% on 3DPW).

1. Introduction

Human motion prediction is a fundamental research topic that benefits many other applications such as intelligent security, autonomous driving, human-robot interaction and so on. Early works employed nonlinear Markov models [22], Gaussian Process dynamical models [41], and Restricted Boltzmann Machine [38] to tackle this problem, while recently a large number of methods based on deep learning have emerged, showing significant merits.

In the camp of deep learning, human motion prediction has been tackled with Recurrent Neural Networks (RNN) [5, 6, 11–16, 20, 29, 32–34, 37], Convolutional Neural Networks (CNN) [3, 8, 24, 35], Graph Convolutional Networks (GCN) [2, 7, 9, 10, 21, 23, 25–27, 30, 31], and Transformer models [1, 4]. Most of the previous approaches directly output the future poses given observed ones. This is very challenging as the output space is extremely large. Recently, a small portion of works [9, 10, 30, 31] duplicate the last observed pose as many times as the length of the future pose sequence, and append the duplicated poses to the original observed sequence. With the extended sequence as input and the ground truth historical and future poses as output, their models learn the residuals between the future poses and the last observed one. Dang *et al*. [10] reported that this reformulation significantly improves the prediction accuracy and they ascribed this to the reason that learning residuals between poses is much easier than learning the poses themselves. In this paper, we view this phenomenon

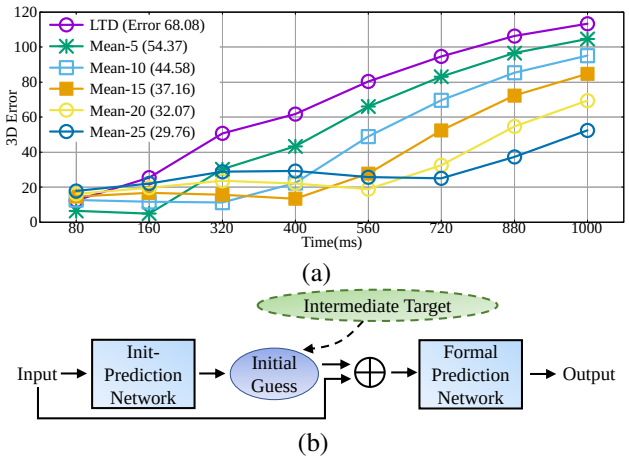


Figure 1. (a) Toy experiments showing importance of initial guess. Average prediction errors in the future from 80ms to 1000ms are plotted. LTD [31] uses the last observed pose as initial guess of future poses. We assume means of future 5, 10, 15, 20, and 25 frames are known and use them as initial guesses fed to LTD respectively. The better the initial guess (*e.g*., "Mean-25" is better than "Mean-15"), the lower the prediction error. (b) Two-stage initial guess and final pose prediction architecture. The intermediate target at each future time can be the mean pose of future poses.

from another perspective. Specifically, we treat the last observed pose as a good initial guess of the target poses, and ask the question: can we achieve better prediction accuracy when providing better initial guess?

The answer is "yes", and the toy experiments in Figure 1 (a) validate this. In these experiments, we append the mean of future x frames to the observed sequence, though these means are not known in practice. They are better than the last observed pose when used as initial guesses, and yield much lower prediction errors. This motivates us to design a two-stage prediction strategy as shown in Figure 1 (b). In the first stage, one can design an init-prediction network that just predicts an initial guess for the target. In the second stage, a formal prediction network then accepts both the historical and initial poses to predict the target poses.

More importantly, we extend the above two-stage prediction strategy to a multi-stage version. For this, we recursively apply a smoothing algorithm to the future pose sequence, obtaining a set of pose sequences at different smoothing scales. By a dedicated smoothing algorithm called Accumulated Average Smoothing (AAS), we achieve that the smoothing result at current scale is a good initial guess of that at the previous smoothing scale. By treating these smoothing results as intermediate targets, our multi-stage prediction framework can progressively predict better initial guesses for the final target pose sequence.

We shall emphasize that any existing human motion prediction model such as [24, 31, 32] can be used to accomplish the prediction task at each stage, among which GCN-based methods [9, 10, 31] are more effective than others. However in these works, GCNs are usually only used to extract spatial features while TCN is adopted to encode the temporal information [23] or the temporal features are represented in the frequency domain and the dependencies between them are neglected [10, 30, 31]. We propose to process both spatial and temporal features by GCNs. Specifically, we propose S-DGCN and T-DGCN. S-DGCN views each pose as a fully-connected graph and encodes global spatial dependencies in human pose, while T-DGCN views each joint trajectory as a fully-connected graph and encodes global temporal dependencies in motion trajectory. S-DGCN and T-DGCN together can extract global spatiotemporal features, better than ST-GCN [42] that is mostly used for local spatiotemporal feature extraction and Transformer models [1, 4] that are of high computation cost.

In summary, the main contributions of this paper are three-fold:

- We propose a novel multi-stage human motion prediction framework utilizing recursively smoothing results by AAS as intermediate targets, which progressively improves the guess for the final target future poses.
- We propose a network based on S-DGCN and T-DGCN that extract global spatiotemporal features effectively to fulfill the prediction task at each stage.
- We have conducted extensive experiments showing that our method outperforms previous approaches by a large margin.

2. Related Work

Due to the serialized nature of human motion data, most previous works adopt RNN as backbone [5, 6, 11–16, 20, 29, 32–34, 37]. For example, ERD [11] improves the recurrent layer of LSTM [17] by placing an encoder before it and a decoder after it. Jain *et al*. [20] organized RNNs according to the spatiotemporal structure of human pose, proposing Structural-RNN. Martinez *et al*. [32] used sequence to sequence architecture instead of the recurrent models of [11, 20]. However, RNNs are hard to train and cannot effectively capture spatial relationships between joints, yielding problems of error accumulation and discontinuity between the last observed pose and the first predicted one.

To mitigate this problem, Shu *et al*. [35] compensated RNN with skeleton-joint co-attention mechanism. From another perspective, the works of [3, 24, 28] had investigated the use of CNNs to model the spatiotemporal relationships in pose sequence. However, CNNs work in a local-to-global manner and cannot directly model the interaction between any pair of joints.

Representing human pose by a graph, recent works have popularly adopted GCNs for human motion prediction [2, 7, 9, 10, 21, 23, 25–27, 30, 31]. Although Aksan *et al*. [2] did not directly use GCN, they adopted a very similar idea that relies on many small networks to exchange features between adjacent joints. The works of [21, 25, 26] used Encoder-Decoder architecture. They either used GCN in the encoder [25, 26] or in the decoder [21]. Differently, the works of [9, 10, 30, 31] are totally based on GCN. Mao *et al*. [31] viewed a pose as a fully-connected graph and used GCN to discover the relationship between any pair of joints. To encode temporal information, they represented the joint trajectories by their Discrete Cosine Transform coefficients. Based on [31], Cui *et al*. [9] additionally learned the weights between joints that are naturally connected along human kinematic chain. Dang *et al*. [10] extended [31] to a multi-scale version across the abstraction levels of human pose. We also use GCN as the basic buildingblock, and one of our contribution is designing S-DGCN and T-DGCN that extract global spatiotemporal features, better than [10, 30, 31] that just extract spatial features. Although the method of [10] also works on multiple scales, the abstracted poses are very different from each other and the pose with fewer joints cannot be regarded as a good initial guess for the pose containing more joints.

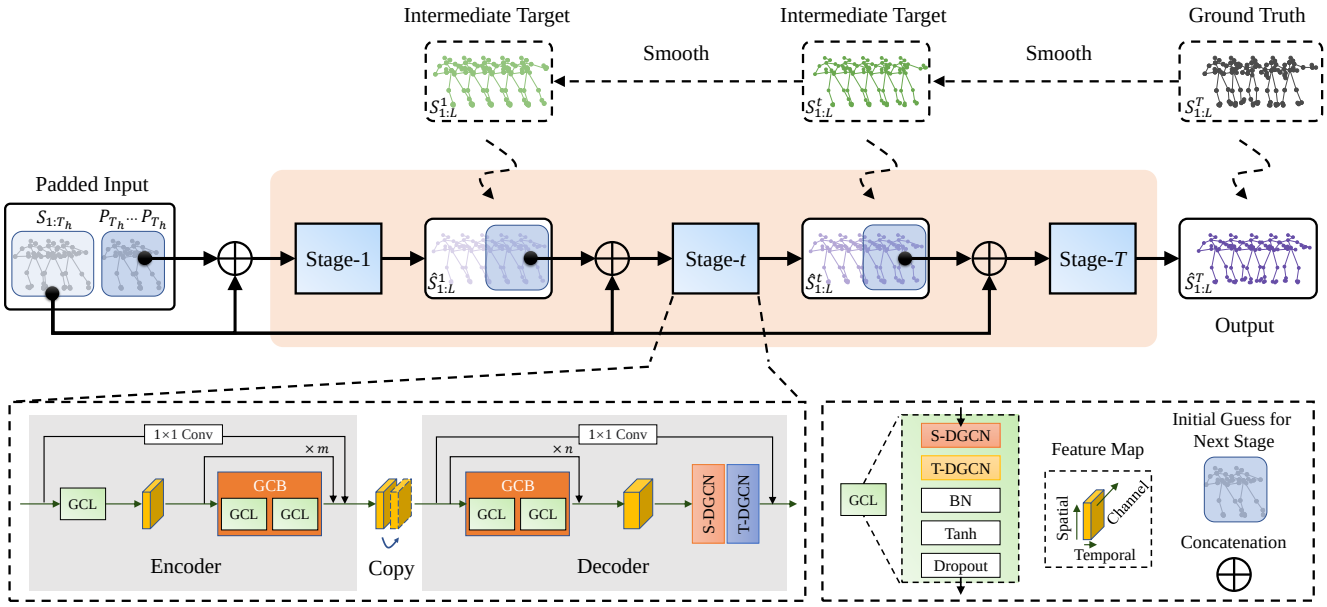


Figure 2. Overview of our progressive human motion prediction framework containing T stages. Each stage takes the original observed sequence $S_{1:T_h}$ and an initial guess for the future as input. For the first stage, the initial guess is composed of the repeated last observed poses. For all the other stages, the initial guess is the future part of the output of previous stage. The last stage is guided by the ground truth, while all the other stages are guided by the corresponding recursively smoothed results of the ground truth. All the stages use the same Encoder-Copy-Decoder prediction network. Please refer to the main text for more details.

Transformer [39] is now very popular in both Natural Language Processing and Vision communities, and has also been adapted to tackle the problem of human motion prediction [1, 4]. It is worth noting that the self-attention layers in Transformers work on fully-connected graph structures. In this paper, instead of using Transformers, we treat pose sequences as graphs and use S-DGCN and T-DGCN to process them, which is effective enough and very efficient.

3. Methodology

Let $S_{1:T_h} = \{P_1, P_2, \cdots, P_{T_h}\}$ denote an observed pose sequence of length T_h where P_i is a pose at time i, and $S_{T_h+1:T_h+T_f}$ be the future pose sequence of length T_f. Instead of directly mapping from $S_{1:T_h}$ to $S_{T_h+1:T_h+T_f}$, we follow [10, 30, 31] to repeat the last observed pose P_{T_h}, T_f times and append them to $S_{1:T_h}$, obtaining the padded input sequence $[S_{1:T_h}; P_{T_h}, \cdots, P_{T_h}]$ of length L with $L = T_h + T_f$. Then our aim becomes to find a mapping from the padded sequence to its ground truth $S_{1:L} = [S_{1:T_h}; S_{T_h+1:T_h+T_f}]$.

3.1. Multi-Stage Progressive Prediction Framework

For the above purpose, we design a multi-stage progressive prediction framework as shown in Figure 2 (the two-stage framework shown in Figure 1 (b) is a special case of the multi-stage framework), which contains T stages represented by $\Phi^1, \Phi^2, \cdots, \Phi^T$ respectively. These stages perform the following subtasks step by step:

$$\begin{aligned} \hat{S}^1_{1:L} &= \Phi^1([S_{1:T_h}; P_{T_h}, \cdots, P_{T_h}]), \\ \hat{S}^i_{1:L} &= \Phi^i([S_{1:T_h}; \hat{S}^{i-1}_{T_h+1:L}]), i = 2, 3, \cdots, T, \end{aligned} \tag{1}$$

in which $\hat{S}^i_{1:L}$ is the output of stage i.

The input to every stage is composed of two parts: the observed poses $S_{1:T_h}$ and the initial guess for the future. For the first stage, the initial guess is $P_{T_h}, \cdots, P_{T_h}$. For any other stage i, the initial guess is $\hat{S}^{i-1}_{T_h+1:L}$ which is the future part of the output of the previous stage $i-1$.

Let $S^T_{1:L} = S_{1:L}$. In the next subsection, we will introduce a way to generate $S^1_{1:L}, S^2_{1:L}, \cdots, S^{T-1}_{1:L}$ based on $S^T_{1:L}$, and use them as the intermediate targets of the corresponding stage networks $\Phi^1, \Phi^2, \cdots, \Phi^T$ go guide the generation of $\hat{S}^1_{1:L}, \hat{S}^2_{1:L}, \cdots, \hat{S}^T_{1:L}$, respectively. We demand that $P_{T_h}, \cdots, P_{T_h}$ is a good guess of $S^1_{1:L}$ and any $S^i_{1:L}$ is a good guess of $S^{i+1}_{1:L}$. That is because only in this way can the output $\hat{S}^{i-1}_{T_h+1:L}$ of previous stage be a good start point to predict the target $S^i_{1:L}$ at the current stage. For example, since $P_{T_h}, \cdots, P_{T_h}$ approaches $S^1_{1:L}$, Φ^1 can easily accomplish its task and output $\hat{S}^1_{1:L}$ of high-quality, *i.e.*, $\hat{S}^1_{1:L}$ will be very similar to $S^1_{1:L}$. According to the fact that $S^1_{1:L}$ is a good guess of $S^2_{1:L}$, therefore $\hat{S}^1_{1:L}$ will be an eligible initial guess for predicting $S^2_{1:L}$ too. This benefit is then passed forward stage by stage until the final result is generated based on a good initial condition.

3.2. Accumulated Average Smoothing

For the two-stage prediction framework as shown in Figure 1 (b), the mean pose of future poses could be a good intermediate target. However, we cannot recursively compute mean of data. Therefore this method cannot be used to prepare intermediate targets for our multi-stage framework. We instead resort to smoothing $S_{1:L}^{T}$ recursively to obtain $S_{1:L}^{T-1}, S_{1:L}^{T-2}, \cdots, S_{1:L}^{1}$ step by step. The adopted smoothing algorithm is Accumulated Average Smoothing (AAS) which is introduced in the following.

Let each pose have M joints, and each joint be a point in the 3D space (we do not use Euler angle based representation). For a pose sequence $S_{1:L}^{T}$, we have $M \times 3$ trajectories: $\{T_j | j \in [1, M \times 3]\}$, and each trajectory T_j is composed of the same coordinate across all the poses: $T_j = \{x_j^i | i \in [1, L]\}$. Since all of the trajectories are smoothed by the same method, we omit the subscript j in the following without loss of generality.

Note that the trajectory contains two parts: the historical part $\{x^i | i \in [1, T_h]\}$ and the future part $\{x^i | i \in [T_h + 1, T_h + T_f]\}$. We just need to smooth the future part and keep the historical part unchanged. The algorithm of AAS is defined as:

$$\bar{x}^i = \frac{1}{i - T_h} \sum_{k=T_h+1}^{i} x^k, \forall i \in [T_h + 1, T_h + T_f]. \quad (2)$$

That is, the smoothed value of a point on a curve is the average of all the previous points along the curve. We apply AAS to $S_{1:L}^{T}$ recursively, obtaining $S_{1:L}^{T-1}, S_{1:L}^{T-2}, \cdots, S_{1:L}^{1}$.

Remark 1. Figure 3 shows the results by AAS and compares them with those by a Gaussian filter (kernel size 21 and bandwidth 10). In these figures, the grey curve is historical, the black is the ground truth in the future, while the dashed line is obtained by padding the last observed data. From dark to light blue are the recursively smoothed results. Compared with Gaussian filter, AAS has two apparent advantages. Firstly, AAS preserves the continuity between the historical and the smoothed future trajectories according to its definition, while the Gaussian filter yields jumps at the boundary. Secondly, AAS has much stronger smoothing ability than the Gaussian filter. Because of this, the smoothed results by AAS evenly and steadily approach the dashed line. The padded line is a good guess of the smoothest curve of AAS. Meanwhile each smoothed curve by AAS is a good guess of that at the previous smoothing scale. Therefore, AAS is very suitable for preparing intermediate targets for our multi-stage framework, while the Gaussian filter is inferior.

Remark 2. The most recent work of Dang *et al.* [10] has proposed a method that gradually predicts future poses from abstracted poses with less joints to detailed poses with more joints. However, a pose with much fewer joints is not a good guess of the pose with more joints. Experiments demonstrate that our method is much better than theirs.

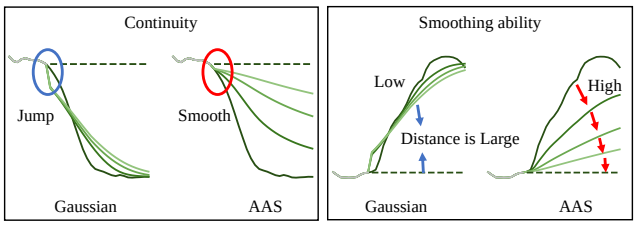


Figure 3. Two examples showing the superiority of AAS over Gaussian filter.

3.3. Encoder-Copy-Decoder Stage Prediction Network Comprising S-DGCN and T-DGCN

In this section, we introduce our network that fulfills the prediction task at each stage, the overview of which is illustrated at the bottom-left of Figure 2. Our network is totally based on GCNs. Specifically, we propose S-DGCN and T-DGCN that extract global spatial and temporal interactions between joints. Based on S-DGCN and T-DGCN, we build an Encoder-Copy-Decoder prediction network. In the following, we introduce them one by one.

S-DGCN. By Dense GCN, *i.e.* DGCN, we mean the processed graph is fully connected. S-DGCN defines a spatially dense graph convolution applied to a pose, and the graph convolution is shared by all the poses of a pose sequence. Let $X \in \mathbb{R}^{L \times M \times F}$ be a pose sequence where L is the length of the sequence, M is the number of joints of a pose, and F indicates the number of features of a joint. Defining a learnable adjacency matrix $A^s \in \mathbb{R}^{M \times M}$ the elements of which measure relationships between pairs of joints of a pose, S-DGCN computes:

$$X' = \text{S-DGCN}(X) = A^s \cdot X \cdot W^s, \quad (3)$$

where $W^s \in \mathbb{R}^{F \times F'}$ indicates the learnable parameters of S-DGCN, and $X' \in \mathbb{R}^{L \times M \times F'}$ is the output of S-DGCN.

T-DGCN. T-DGCN defines a temporal graph convolution applied to a joint trajectory, and the graph convolution is shared by all the trajectories. We first transpose the first two dimensions of X' to obtain $Y \in \mathbb{R}^{M \times L \times F'}$. Defining a learnable adjacency matrix $A^t \in \mathbb{R}^{L \times L}$ measuring weights between pairs of joints of a trajectory, T-DGCN computes:

$$Y' = \text{T-DGCN}(Y) = A^t \cdot Y \cdot W^t, \quad (4)$$

where $W^t \in \mathbb{R}^{F' \times F'}$ is the learnable parameters of T-DGCN, and $Y' \in \mathbb{R}^{M \times L \times F'}$. Finally, we transpose the first two dimensions back to make $Y' \in \mathbb{R}^{L \times M \times F'}$.

GCL. As shown at the bottom-right of Figure 2, we define a Graph Convolutional Layer (GCL) as a unit that sequentially executes S-DGCN, T-DGCN, batch normalization [18], tanh, and dropout [36]. By executing S-DGCN

and T-DGCN sequentially, we can extract spatiotemporal features over the global receptive field of the whole pose sequence.

Encoder. As shown in Figure 2, the encoder is a residual block containing a GCL and multiple Graph Convolutional Blocks (GCB). The GCL projects the input from the pose space of $\mathbb{R}^{L\times M\times 3}$ to the feature space of $\mathbb{R}^{L\times M\times F}$. We set $F = 16$ in this paper. Each GCB itself is a residual block containing two GCLs. They always work in the feature space. In order to add the global residual connection for the encoder, we employ a 1×1 convolutional layer with 16 kernels that directly maps the input into the space of $\mathbb{R}^{L\times M\times F}$ which is then added to the output of the GCBs.

Copy. The encoder outputs a feature map in the space of $\mathbb{R}^{L\times M\times F}$. We duplicate it and concatenate the copy with the original feature map along the trajectory direction, obtaining a feature map of size $\mathbb{R}^{2L\times M\times F}$ which is then fed into the decoder. We find in practice that the "copy" operator enhances the capability of the decoder.

Decoder. The decoder is a residual block containing multiple GCBs and a pair of S-DGCN and T-DGCN. The GCBs work in the feature space of $F = 16$, while the S-DGCN and T-DGCN project the features back into the pose space. Since the input to the decoder is of length $2L$, the adjacency matrix A^t of all the T-DGCNs, including those in the GCBs, are of size $\mathbb{R}^{2L\times 2L}$. In order to add the residual connection for the decoder, a 1×1 convolutional layer with 3 kernels is used. The result of the decoder is of length $2L$, we just retain the front L poses.

3.4. Loss Function

We have two kinds of losses: L_{gt} that enforces the supervision from the ground truth, and L_{int} that enforces supervisions from the intermediate targets.

$$\begin{aligned} L =& L_{gt} + L_{int} \\ =& \|\hat{S}^T_{1:L} - S^T_{1:L}\|^2 + \sum_{i=1}^{T-1} \|\hat{S}^i_{1:L} - S^i_{1:L}\|^2. \end{aligned} \tag{5}$$

4. Experiments

4.1. Datasets

Human3.6M [19] owns 15 types of actions performed by 7 actors (S1, S5, S6, S7, S8, S9, and S11). Each pose has 32 joints in the format of exponential map. Following [10], we convert them to 3D coordinates and discard 10 redundant joints. The global rotations and translations of poses are excluded. The frame rate is downsampled from 50fps to 25fps. S5 and S11 are used as testing and validation datasets respectively, while the remaining are used for training.

CMU-MoCap owns 8 human action categories. Each pose contains 38 joints in the format of exponential map which are also converted to 3D coordinates. The global rotations and translations of the poses are excluded too. Following [10, 31], we keep 25 joints and discard the others. The division between training and testing datasets is also the same as [10, 31].

3DPW [40] is a challenging dataset containing human motion captured from both indoor and outdoor scenes. The poses in this dataset are already in the 3D space. Each pose contains 26 joints and 23 of them are used (the other 3 are redundant).

4.2. Comparison Settings

Evaluation only in 3D space. Besides 3D coordinates, an exponential map can also be converted to Euler angles. We only evaluate in the 3D space since (1) it is more meaningful to represent a trajectory in the 3D space, and (2) Euler angles suffer from ambiguities, *i.e*. two different sets of angles may represent the exact same pose [31].

Evaluation metric. We use the Mean Per Joint Position Error (MPJPE) as our evaluation metric which is widely adopted by previous approaches [4, 10, 31].

Test on the whole test dataset. We have noted that the works of [26, 31, 32] randomly take 8 samples per action to test, Mao *et al*. [30] randomly take 256 samples per action to test, and Dang *et al*. [10] take all the samples to test. In this paper, we follow Dang *et al*. [10] to test on the whole test dataset. In the supplemental material, the comparison results on the random 8 and 256 test sets are also provided.

Input and output length. Following [10], the input are 10 poses and the output are 25 poses for Human3.6M and CMU-MoCap, respectively. Following [31], the input are 10 poses and the output are 30 poses for 3DPW.

4.3. Implementation Details

Our multi-stage framework contains 4 stages, all using the proposed Encoder-Copy-Decoder prediction network. The encoder contains 1 GCB, and the decoder contains 2 GCBs. The framework contains 12 GCBs in total. We employ Adam as the solver. The learning rate is initialized as 0.005 and decayed by 0.96 every epoch. The model is trained by 50 epochs with batchsize 16. The devices we used are an NVIDIA RTX 2060 GPU and an AMD Ryzen 5 3600 CPU. For more implementation details, please refer to the supplemental material, and we will release our code in the future.

4.4. Comparisons with previous approaches

We compare our method with Res. Sup. [32], DMGNN [26], LTD [31], and MSR [10]. Res. Sup. is an early RNN based approach. DMGNN is also RNN based but uses GCN to extract features. LTD relies on GCN totally

Table 1. Comparisons of short-term prediction on Human3.6M. Results at 80ms, 160ms, 320ms, 400ms in the future are shown. The best results are highlighted in bold, and the second best are marked by underline.

scenarios	walking				eating				smoking				discussion			
millisecond	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms
Res. Sup.	29.4	50.8	76.0	81.5	16.8	30.6	56.9	68.7	23.0	42.6	70.1	82.7	32.9	61.2	90.9	96.2
DMGNN	17.3	30.7	54.6	65.2	11.0	21.4	36.2	43.9	9.0	17.6	32.1	40.3	17.3	34.8	61.0	69.8
LTD	12.3	23.0	39.8	46.1	8.4	16.9	33.2	40.7	7.9	16.2	31.9	38.9	12.5	27.4	58.5	71.7
MSR	12.2	22.7	38.6	45.2	8.4	17.1	33.0	40.4	8.0	16.3	31.3	38.2	12.0	26.8	57.1	69.7
Ours	**10.2**	**19.8**	**34.5**	**40.3**	**7.0**	**15.1**	**30.6**	**38.1**	**6.6**	**14.1**	**28.2**	**34.7**	**10.0**	**23.8**	**53.6**	**66.7**
scenarios	directions				greeting				phoning				posing			
millisecond	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms
Res. Sup.	35.4	57.3	76.3	87.7	34.5	63.4	124.6	142.5	38.0	69.3	115.0	126.7	36.1	69.1	130.5	157.1
DMGNN	13.1	24.6	64.7	81.9	23.3	50.3	107.3	132.1	12.5	25.8	48.1	58.3	15.3	29.3	71.5	96.7
LTD	9.0	19.9	43.4	53.7	18.7	38.7	77.7	93.4	10.2	21.0	42.5	52.3	13.7	29.9	66.6	84.1
MSR	8.6	19.7	43.3	53.8	16.5	37.0	77.3	93.4	10.1	20.7	41.5	51.3	12.8	29.4	67.0	85.0
Ours	**7.2**	**17.6**	**40.9**	**51.5**	**15.2**	**34.1**	**71.6**	**87.1**	**8.3**	**18.3**	**38.7**	**48.4**	**10.7**	**25.7**	**60.0**	**76.6**
scenarios	purchases				sitting				sittingdown				takingphoto			
millisecond	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms
Res. Sup.	36.3	60.3	86.5	95.9	42.6	81.4	134.7	151.8	47.3	86.0	145.8	168.9	26.1	47.6	81.4	94.7
DMGNN	21.4	38.7	75.7	92.7	11.9	25.1	44.6	**50.2**	15.0	32.9	77.1	93.0	13.6	29.0	46.0	58.8
LTD	15.6	32.8	65.7	79.3	10.6	21.9	46.3	57.9	16.1	31.1	61.5	75.5	9.9	20.9	45.0	56.6
MSR	14.8	32.4	66.1	79.6	10.5	22.0	46.3	57.8	16.1	31.6	62.5	76.8	9.9	21.0	44.6	56.3
Ours	**12.5**	**28.7**	**60.1**	**73.3**	**8.8**	**19.2**	**42.4**	53.8	**13.9**	**27.9**	**57.4**	**71.5**	**8.4**	**18.9**	**42.0**	**53.3**
scenarios	waiting				walkingdog				walkingtogether				average			
millisecond	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms	80ms	160ms	320ms	400ms
Res. Sup.	30.6	57.8	106.2	121.5	64.2	102.1	141.1	164.4	26.8	50.1	80.2	92.2	34.7	62.0	101.1	115.5
DMGNN	12.2	24.2	59.6	77.5	47.1	93.3	160.1	171.2	14.3	26.7	50.1	63.2	17.0	33.6	65.9	79.7
LTD	11.4	24.0	50.1	61.5	23.4	46.2	83.5	96.0	10.5	21.0	38.5	45.2	12.7	26.1	52.3	63.5
MSR	10.7	23.1	48.3	59.2	20.7	42.9	80.4	93.3	10.6	20.9	37.4	43.9	12.1	25.6	51.6	62.9
Ours	**8.9**	**20.1**	**43.6**	**54.3**	**18.8**	**39.3**	**73.7**	**86.4**	**8.7**	**18.6**	**34.4**	**41.0**	**10.3**	**22.7**	**47.4**	**58.5**

Table 2. Comparisons of long-term prediction on Human3.6M. Results at 560ms and 1000ms in the future are shown.

scenarios	walking		eating		smoking		discussion		directions		greeting		phoning		posing	
millisecond	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms
Res. Sup.	81.7	100.7	79.9	100.2	94.8	137.4	121.3	161.7	110.1	152.5	156.1	166.5	141.2	131.5	194.7	240.2
DMGNN	73.4	95.8	58.1	86.7	50.9	72.2	81.9	138.3	110.1	115.8	152.5	157.7	78.9	98.6	163.9	310.1
LTD	54.1	59.8	53.4	77.8	50.7	72.6	91.6	121.5	71.0	101.8	115.4	148.8	69.2	103.1	114.5	173.0
MSR	52.7	63.0	52.5	77.1	49.5	71.6	88.6	**117.6**	71.2	100.6	116.3	147.2	68.3	104.4	116.3	174.3
Ours	**48.1**	**56.4**	**51.1**	**76.0**	**46.5**	**69.5**	**87.1**	118.2	**69.3**	**100.4**	**110.2**	**143.5**	**65.9**	**102.7**	**106.1**	**164.8**
scenarios	purchases		sitting		sittingdown		takingphoto		waiting		walkingdog		walkingtogether		average	
millisecond	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms	560ms	1000ms
Res. Sup.	122.7	160.3	167.4	201.5	205.3	277.6	117.0	143.2	146.2	196.2	191.3	209.0	107.6	131.1	97.6	130.5
DMGNN	118.6	153.8	**60.1**	**104.9**	122.1	168.8	91.6	120.7	106.0	136.7	194.0	182.3	83.4	115.9	103.0	137.2
LTD	102.0	143.5	78.3	119.7	100.0	150.2	77.4	119.8	79.4	108.1	111.9	148.9	55.0	65.6	81.6	114.3
MSR	101.6	139.2	78.2	120.0	102.8	155.5	77.9	121.9	76.3	106.3	111.9	148.2	52.9	65.9	81.1	114.2
Ours	**95.3**	**133.3**	74.4	116.1	**96.7**	**147.8**	**74.3**	**118.6**	**72.2**	**103.4**	**104.7**	**139.8**	**51.9**	**64.3**	**76.9**	**110.3**

Table 3. CMU-MoCap: comparisons of average prediction errors.

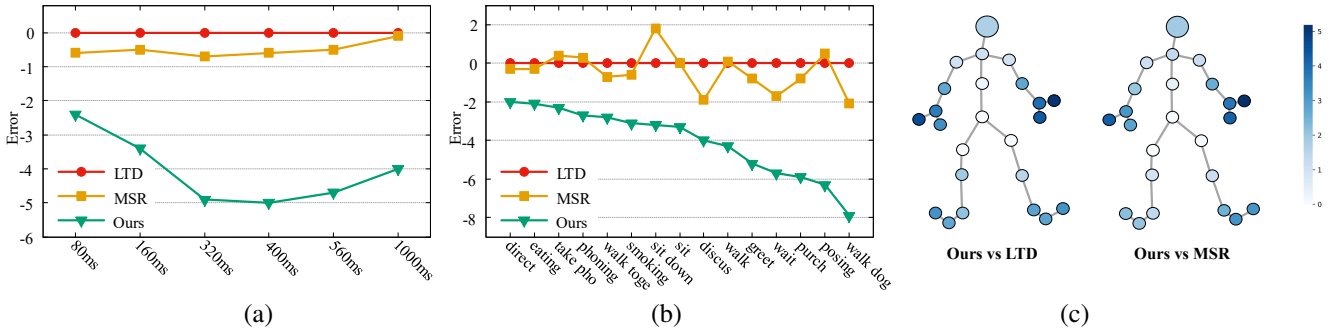
millisecond	80ms	160ms	320ms	400ms	560ms	1000ms
Res. Sup.	24.0	43.0	74.5	87.2	105.5	136.3
DMGNN	13.6	24.1	47.0	58.8	77.4	112.6
LTD	9.3	17.1	33.0	40.9	55.8	86.2
MSR	8.1	15.2	30.6	38.6	53.7	83.0
Ours	**7.6**	**14.3**	**29**	**36.6**	**50.9**	**80.1**

Table 4. Average prediction errors of different methods on 3DPW.

millisecond	200ms	400ms	600ms	800ms	1000ms
Res. Sup.	113.9	173.1	191.9	201.1	210.7
DMGNN	37.3	67.8	94.5	109.7	123.6
LTD	35.6	67.8	90.6	106.9	117.8
MSR	37.8	71.3	93.9	110.8	121.5
Ours	**29.3**	**58.3**	**79.8**	**94.4**	**104.1**

and performs the prediction in the frequency domain. MSR is a recent method executing LTD in a multiscale fashion. All these methods are previous state-of-the-arts which release their code publicly. For fair comparison, we use their pre-trained models or re-train the models using their default hyper-parameters.

Human3.6M. Table 1 shows the quantitative comparisons of short-term prediction (less than 400ms) on Human3.6M between our method and the above four approaches. Table 2 shows the comparisons of long-term prediction (more than 400ms but less than 1000ms) on Human3.6M. For almost all cases, our results are better than those of the compared methods. Moreover, our method outperforms the current state-of-the-arts by a large margin. This can be seen from Figure 4. In (a) and (b), we treat LTD as the baseline, and subtract the prediction errors of MSR and our method from those of LTD. In (a), the relative average prediction errors with respect to LTD at every future timestamp are plotted. As can be seen, MSR is better than LTD, while our method is much better than MSR. Our advantage is the most significant at 400ms. In (b), the rela-



(a) (b) (c)

Figure 4. Advantage analysis (Human3.6M). (a) The advantage of our method is most significant at 400ms. (b) The advantage is most significant for "walking dog". (c) The advantage per joint is illustrated. The darker the color, the larger the advantage of our method.

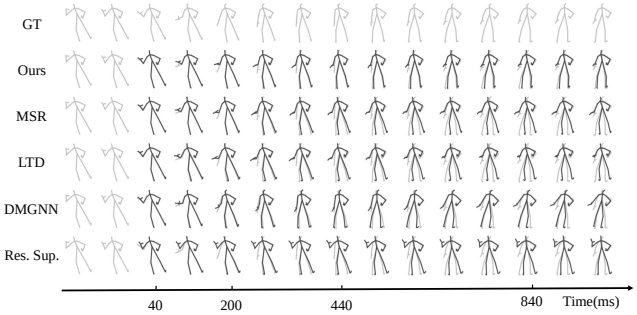


Figure 5. Visualization of predicted poses of different methods on a sample of Human3.6M.

tive average prediction errors with respect to LTD for every action category are plotted. Again, the advantage of our method compared with LTD and MSR is large, and for the action of "walking dog" the advantage is the most significant. In (c), we plot the advantage per joint of our method over LTD and MSR. The darker the color, the higher the advantage. As can be seen, our method achieves higher performance gain on limbs, especially on hands and feet. In Figure 5, we show an example of the predicted poses of different methods. With the increase of the forecast time, the result of our method becomes more and more better than those of the others.

CMU-MoCap and 3DPW. Table 3 and Table 4 show the comparisons on CMU-MoCap and 3DPW respectively. Due to space limit, we only show the average prediction errors at every timestamp. More detailed tables are provided in the supplementary material. On the two datasets, our method also outperforms all the compared approaches. Especially, for the very challenging dataset 3DPW, our advantage is very significant.

4.5. Ablation Analysis

In this section, we conduct ablation studies to analyze our method in depth. All experimental results are obtained on Human3.6M.

Architecture. Several design choices contribute to the effectiveness of our method: (1) the multi-stage progressive learning framework, (2) the intermediate supervisions offered by the recursively smoothed results of the ground truth, (2) the Encoder-Copy-Decoder prediction network at each stage, and (4) the "Copy" operator.

Table 5 shows the ablation experiments on different variants of our full model. The full model has 4 stages each containing 3 GCBs, and there are 12 GCBs in total, achieving an average prediction error of 65.02. Firstly, to show the effectiveness of the progressive learning strategy, we test the case when there is only one Encoder-Copy-Decoder network with 12 GCBs (6 in encoder and 6 in decoder) which directly maps the padded input sequence to the final target sequence. The prediction error increases from 65.02 to 67.48 which is a very large performance drop. Secondly, we maintain the progressive learning strategy but remove the intermediate loss in Equation 5. The prediction error increases from 65.02 to 67.07, demonstrating the necessity of the intermediate supervisions. In the third experiment, we replace our designed Encoder-Copy-Decoder network at each stage by the prediction model of LTD [31]. The prediction error increases from 65.02 to 67.15. In the fourth experiment, we replace S-DGCN and T-DGCN by ST-GCN [42]. The prediction error drastically increases from 65.02 to 67.97, showing the importance of extracting global spatiotemporal relationships between any pair of joints. Finally, removing the "Copy" operator in the middle of the Encoder-Copy-Decoder network yields a slightly increase of the prediction error from 65.02 to 65.99.

Number of stages. In Figure 6, we conduct more ablations on the number of stages. Although these variants contain different number of stages, they all own 12 GCBs in total, evenly distributed in each stage. For example, if the stage number is 3, there will be 4 GCBs at each stage, 2 in the encoder and 2 in the decoder. The experiments tell that the best performance is obtained when there are 4 stages.

Direction and number of "Copy". In the default setting of the Encoder-Copy-Decoder network, we copy the output of the encoder just one time and paste it along the

Table 5. Ablation on architecture.

	80ms	160ms	320ms	400ms	560ms	720ms	880ms	1000ms	average
Single stage prediction	11.95	24.47	49.69	60.94	79.56	93.93	105.86	113.41	67.48
Without intermediate loss	11.42	24.02	49.73	60.94	79.49	93.45	105.12	112.42	67.07
Replacing Encoder-Copy-Decoder by LTD	11.11	24.01	49.48	60.67	79.34	93.91	105.55	113.10	67.15
Replacing S-DGCN, T-DGCN by ST-GCN	11.84	25.78	51.87	62.73	80.23	93.61	105.00	112.72	67.97
Without "Copy" operator	10.53	23.25	48.89	59.99	78.16	92.34	103.70	111.04	65.99
Full Model	**10.33**	**22.74**	**47.45**	**58.46**	**76.91**	**91.20**	**102.77**	**110.31**	**65.02**

Table 6. Ablation on "Copy" times and dimension.

Copy times	Error	Model size	Copy dimension	Error	Model size
No copy	65.99	1.06M	Copy in channel	65.75	1.67M
Copy once(Ours)	**65.02**	1.74M	Copy in spatial	65.21	1.69M
Copy three times	65.35	3.28M	Copy in temporal(Ours)	**65.02**	1.74M

Table 7. Ablation on using different intermediate targets in our multi-stage framework.

	80ms	160ms	320ms	400ms	560ms	1000ms	average
Gaussian-15	12.0	24.4	49.8	60.9	78.7	111.0	66.6
Gaussian-21	11.5	23.7	48.8	60.0	78.5	112.2	66.5
AAS	**10.3**	**22.7**	**47.4**	**58.5**	**76.9**	**110.3**	**65.0**

temporal direction. In Table 6, we conduct ablation studies on the number of copying and the direction of pasting. As can be seen, copying once or three times is better than not copying. But copying three times does not bring more performance gain than copying once. Copying once along the spatial dimension, the channel dimension and the temporal dimension are all better than not copying, while copying along the temporal dimension yields the best result.

Intermediate targets. In Figure 7, we compare between using different kinds of intermediate targets in the two-stage framework, for which the mean of future poses and the smoothing results (just smoothing once) by Gaussian filter and AAS can be used as the intermediate target. In the figure, "Mean-x" indicates the mean of future x poses. As can be seen, our two-stage method always outperforms LTD [31] no matter what kind of intermediate target is used. Among these intermediate targets, "Gaussian-21" performs the worst, and then "Mean-5" and "Mean-25", while "AAS" is the best. Our multi-stage model, indicated by "Progressive AAS", is much better than its two-stage variant of "AAS".

In Table 7, we compare between using different kinds of intermediate targets in our multi-stage framework, *i.e.*, those generated recursively by Gaussian filter and AAS. In the table, "Gaussian-x" means the kernel size of the filter is x. It can be seen that AAS yields much better prediction accuracy than the two Gaussian smoothing filters.

5. Conclusion and Future Work

We have presented a multi-stage human motion prediction framework. The key to the effectiveness of the framework is that we decompose the originally difficult prediction task into many subtasks, and ensure each subtask is simple enough. We achieve this by taking the recursively smoothed versions of the target pose sequence as the prediction targets of the subtasks. The adopted accumulated average smoothing strategy guarantees that the smoothest intermediate target approaches to the last observed data, and the intermediate targets of adjacent stages are good guess of each other. Besides that, we have proposed the novel Encoder-Copy-Decoder prediction network, the S-DGCN and T-DGCN of which can extract spatiotemporal features effectively while the "Copy" operator further enhance the capability of the decoder. We have conducted extensive experiments and analysis demonstrating the effectiveness and advantages of our method. An interesting future work is to investigate more effective forms of the intermediate targets. Designing better prediction network for the stages, *e.g.* through self-attention, is also promising.

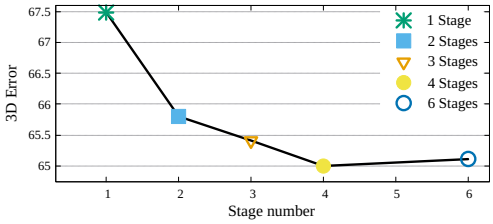


Figure 6. Ablation on the number of stages.

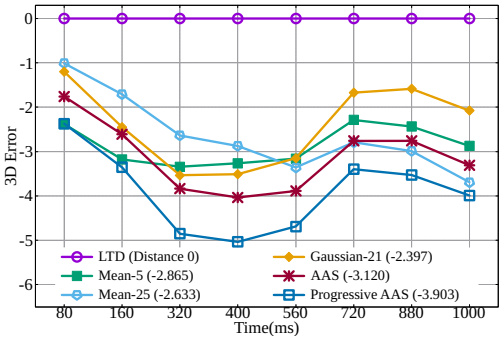


Figure 7. Comparison between using different intermediate targets in two-stage framework. LTD [31] is the baseline. "Mean-x" indicates using the mean pose of future x poses as intermediate target in the two-stage framework, while "Gaussian-21" (kernel size 21 bandwidth 10) and "AAS" mean using their smoothing once results. "Progressive AAS" plots the results of our full model.

References

[1] Emre Aksan, Peng Cao, Manuel Kaufmann, and Otmar Hilliges. A spatio-temporal transformer for 3d human motion prediction. *arXiv preprint arXiv:2004.08692*, 2020. 1, 2, 3

[2] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. Structured prediction helps 3d human motion modelling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7144–7153, 2019. 1, 2

[3] Judith Butepage, Michael J Black, Danica Kragic, and Hedvig Kjellstrom. Deep representation learning for human motion prediction and classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6158–6166, 2017. 1, 2

[4] Yujun Cai, Lin Huang, Yiwei Wang, Tat-Jen Cham, Jianfei Cai, Junsong Yuan, Jun Liu, Xu Yang, Yiheng Zhu, Xiaohui Shen, et al. Learning progressive joint propagation for human motion prediction. In *European Conference on Computer Vision*, pages 226–242. Springer, 2020. 1, 2, 3, 5

[5] Hsu-kuang Chiu, Ehsan Adeli, Borui Wang, De-An Huang, and Juan Carlos Niebles. Action-agnostic human pose forecasting. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1423–1432. IEEE, 2019. 1, 2

[6] Enric Corona, Albert Pumarola, Guillem Alenya, and Francesc Moreno-Noguer. Context-aware human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6992–7001, 2020. 1, 2

[7] Qiongjie Cui and Huaijiang Sun. Towards accurate 3d human motion prediction from incomplete observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4801–4810, 2021. 1, 2

[8] Qiongjie Cui, Huaijiang Sun, Yue Kong, Xiaoqian Zhang, and Yanmeng Li. Efficient human motion prediction using temporal convolutional generative adversarial network. *Information Sciences*, 545:427–447, 2021. 1

[9] Qiongjie Cui, Huaijiang Sun, and Fei Yang. Learning dynamic relationships for 3d human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6519–6527, 2020. 1, 2

[10] Lingwei Dang, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. Msr-gcn: Multi-scale residual graph convolution networks for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11467–11476, 2021. 1, 2, 3, 4, 5

[11] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015. 1, 2

[12] Partha Ghosh, Jie Song, Emre Aksan, and Otmar Hilliges. Learning human motion models for long-term predictions. In *2017 International Conference on 3D Vision (3DV)*, pages 458–466. IEEE, 2017. 1, 2

[13] Anand Gopalakrishnan, Ankur Mali, Dan Kifer, Lee Giles, and Alexander G Ororbia. A neural temporal model for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12116–12125, 2019. 1, 2

[14] Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and José MF Moura. Adversarial geometry-aware human motion prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 786–803, 2018. 1, 2

[15] Liang-Yan Gui, Yu-Xiong Wang, Deva Ramanan, and José MF Moura. Few-shot human motion prediction via meta-learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 432–450, 2018. 1, 2

[16] Xiao Guo and Jongmoo Choi. Human motion prediction via learning local structure representations and temporal dependencies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2580–2587, 2019. 1, 2

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 4

[19] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013. 5

[20] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5308–5317, 2016. 1, 2

[21] Tim Lebailly, Sena Kiciroglu, Mathieu Salzmann, Pascal Fua, and Wei Wang. Motion prediction using temporal inception module. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 1, 2

[22] Andreas M Lehrmann, Peter V Gehler, and Sebastian Nowozin. Efficient nonlinear markov models for human motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1314–1321, 2014. 1

[23] Bin Li, Jian Tian, Zhongfei Zhang, Hailin Feng, and Xi Li. Multitask non-autoregressive model for human motion prediction. *IEEE Transactions on Image Processing*, 30:2562–2574, 2020. 1, 2

[24] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. Convolutional sequence to sequence model for human dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5226–5234, 2018. 1, 2

[25] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1, 2

[26] Maosen Li, Siheng Chen, Yangheng Zhao, Ya Zhang, Yanfeng Wang, and Qi Tian. Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 214–223, 2020. 1, 2, 5

[27] Jin Liu and Jianqin Yin. Multi-grained trajectory graph convolutional networks for habit-unrelated human motion prediction. *arXiv preprint arXiv:2012.12558*, 2020. 1, 2

[28] Xiaoli Liu, Jianqin Yin, Jin Liu, Pengxiang Ding, Jun Liu, and Huaping Liub. Trajectorycnn: a new spatio-temporal feature learning network for human motion prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020. 2

[29] Zhenguang Liu, Shuang Wu, Shuyuan Jin, Qi Liu, Shijian Lu, Roger Zimmermann, and Li Cheng. Towards natural and accurate future motion prediction of humans and animals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10004–10012, 2019. 1, 2

[30] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. History repeats itself: Human motion prediction via motion attention. In *European Conference on Computer Vision*, pages 474–489. Springer, 2020. 1, 2, 3, 5

[31] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9489–9497, 2019. 1, 2, 3, 5, 7, 8

[32] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017. 1, 2, 5

[33] Dario Pavllo, Christoph Feichtenhofer, Michael Auli, and David Grangier. Modeling human motion with quaternion-based neural networks. *International Journal of Computer Vision*, 128(4):855–872, 2020. 1, 2

[34] Hai-Feng Sang, Zi-Zhen Chen, and Da-Kuo He. Human motion prediction based on attention mechanism. *Multimedia Tools and Applications*, 79(9):5529–5544, 2020. 1, 2

[35] Xiangbo Shu, Liyan Zhang, Guo-Jun Qi, Wei Liu, and Jinhui Tang. Spatiotemporal co-attention recurrent neural networks for human-skeleton motion prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1, 2

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 4

[37] Yongyi Tang, Lin Ma, Wei Liu, and Weishi Zheng. Long-term human motion prediction by modeling motion context and enhancing motion dynamic. *arXiv preprint arXiv:1805.02513*, 2018. 1, 2

[38] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2007. 1

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3

[40] Timo von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 601–617, 2018. 5

[41] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models. In *NIPS*, volume 18, page 3. Citeseer, 2005. 1

[42] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018. 2, 7