Context-based Interpretable Spatio-Temporal Graph Convolutional Network for Human Motion Forecasting

Edgar Medina, Leyong Loh, Namrata Gurung, Kyung Hun Oh, Niels Heller,
QualityMinds GmbH
{edgar.medina, leyong.loh, namrata.gurung, kyung-hun.oh, niels.heller}@qualityminds.de

Abstract

Human motion prediction is still an open problem extremely important for autonomous driving and safety applications. Due to the complex spatiotemporal relation of motion sequences, this remains a challenging problem not only for movement prediction but also to perform a preliminary interpretation of the joint connections. In this work, we present a Context-based Interpretable Spatio-Temporal Graph Convolutional Network (CIST-GCN), as an efficient 3D human pose forecasting model based on GCNs that encompasses specific layers, aiding model interpretability and providing information that might be useful when analyzing motion distribution and body behavior. Our architecture extracts meaningful information from pose sequences, aggregates displacements and accelerations into the input model, and finally predicts the output displacements. Extensive experiments on Human 3.6M, AMASS, 3DPW, and ExPI datasets demonstrate that CIST-GCN outperforms previous methods in human motion prediction and robustness. Since the idea of enhancing interpretability for motion prediction has its merits, we showcase experiments towards it and provide preliminary evaluations of such insights here. [1]

1. Introduction

Human motion prediction plays a critical role in autonomous driving, robotics, and safety applications. In the recent past, several methods for human motion prediction and modeling have led to significant results with the use of neural networks [24]. Recently, the main approaches to tackle this task have been by means of Graph Convolutional Networks (GCN) [15,16,25,27,28,34,39,42,44], Recurrent Networks (RNN) [9,22,28,30,35,37] and GANs [24]. Although in the last years, RNN-based models were the most effective methods, they come with the drawback of vanishing or exploding gradients. Recent approaches mix more sophisticated architectures such as Gated recurrent units (GRU) [35] or transformers [1,24] with feature extraction using CNN or a gate system in the hidden states. Alternately, the GAN-based approach [24] is another methodology for generating the sequence from a hidden vector, but this approach neglects the kinematic dependencies between pose joints and ignores the temporal correlation between frames. Instead, GCNs have received increasing attention because this architecture can find a temporal relation between poses and can understand relationships among joints.

A second branch in this work uses interpretability concepts, initial stages were inspired by class-activation [43] or saliency maps. However, the authors using saliency maps only performed a visual analysis without providing formal statistical evaluations. [18,41,43]. Nowadays, more sophisticated methods quantify the error or even measure the uncertainty level of movement predictions [31,32]. Despite the great advances in interpretability of CNNs in classification tasks, GCNs are not yet properly covered [4,12], especially for regression tasks (such as motion prediction) and not classification tasks.

The motivation for designing this architecture is to close the gap between motion prediction and interpretability and apply it to real-world problems to gain meaningful insights into why the model predicted a specific output. In our proposed Context-based Interpretable Spatio-Temporal GCN (CIST-GCN) architecture, we embed GCN layers which provide sample-specific adjacency matrices and importance vectors to explain motion forecasting. The matrices are composed of learnable parameters while the importance vectors are generated at output layers a mix of CNN and MLP layers. It stands to reason that these features are human-interpretable. While we provide some of such interpretations in this paper. To the best of our knowledge, this is the first work that drives a GCN-based architecture in this direction. Finally, data augmentation has been added to speed up the training and also make the system more robust to possible data glitches that may occur in production use-case such as faulty 3D-transformations or falsy reconstructed 3d poses. We conducted experiments to study the robustness of our model against out-of-distribution (OOD) data samples, for example, rotations, glitches in poses.

[1] available code: qualityminds.cistgcn

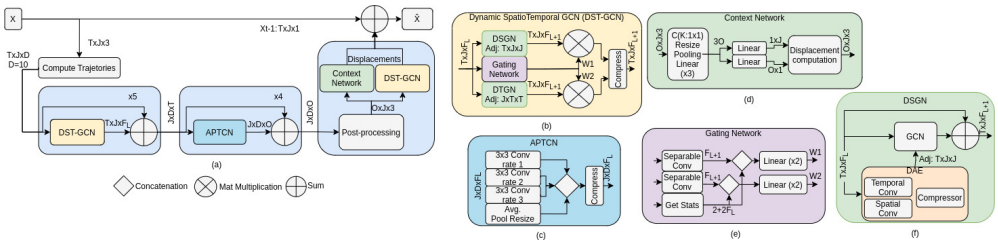

Figure 1. Illustration of our method. (a) Overview of the proposed CIST-GCN. X and $\hat{X}$ are the input and output respectively. (b) The basic block of DST-GCN, (c) the Atraus Pyramid TCN, and (d) Context Network. More detailed, (e) Gating network weights the output of DSGN and DTGN, and (f) Dynamic Adjacency Encoder (DAE) to compute the adjacency matrices.

Our approach consistently obtains comparable results to the previous results on short- and long-term motion prediction by training a single unified model for both settings. Specifically, we achieve superior performance in 6 out of 15 actions on the Human3.6M benchmark, while remaining comparable in the other motion predictions. Our model also surpasses previous works in 3DPW and 12 out of 16 actions on ExPI datasets, while also achieving comparable results on the AMASS benchmark. The main contributions of our work can be summarized as follows: 1) we propose a new architecture that provides not only human motion prediction, but also interpretability to some extent given an input sample, 2) we perform extensive experiments on Human3.6M [11], AMASS [26], 3DPW [36], and ExPI [8] datasets, showing that quantitative and qualitative results are comparable to state-of-the-art (SOTA) models, 3) we discuss the different extents of interpretations such as relation matrices, and importance vectors, and 4) we perform robustness experiments that showcases our model to be better than existing SOTA models for ODD samples.

2. Related work

2.1. Motion Prediction

Initial research prove the strength of relations between joint connections in pose sequences both in the temporal and spatial domains [24]. Subsequent work [16, 19, 28, 39, 44] research deeper into this approach by grouping the input joints in several ways or merging GCN, CNN, and GRU layers to learn the graph connectivity (forming a spatio-temporal graph). In [34], STS-GCN receives the 3D coordinates as input but uses two GCNs to encode sequentially temporal and spatial data in every encoding layer to feed the decoder. This decoder is composed of a 4-block Temporal Convolution Network (TCN) [27] that converts input frames into output frames. Overall, this work requires a lower number of parameters than previous approaches and inspired recent architectures.

Another newest branch, such as MotionMixer [2], employs linear and feature mix layers to merge information. Although the results are promising, there are problems in understanding which joints or frames may be relevant for further analysis such as prediction reliability or action clustering. Since the architecture behaves as a black box model, we cannot obtain correlation or relationship matrices coming from the model. Also, experiments such as the application of 3D transformations can illustrate its limitations.

In [7], several modifications to DSTD-GCN are proposed such as dynamic spatial and temporal graph convolutions are presented as separate units, allowing features to be learned independently. The authors suggest using constrained training with different strategies. Later, they show that relations can be acquired in unshared sample-specific forms, reducing MPJPE significantly. The impact of this approach on metrics inspired us to incorporate learnable adjacency matrices in all our GCN layers, removing the need for duplicating dynamic spatial GCN. This results in fewer parameters compared to DSTD-GCN. Also, in Section 5, we explore the potential application of this as an interpretable output for individual samples. In [42], a gating network is proposed to generate blending coefficients that weighs the most meaningful features of the adjacency matrices from GCNs. Number of weight vectors for the temporal and spatial GCNs are equal. Also, the authors suggest that these acquired vectors could aid in action grouping and emphasize the most important features in motion prediction. Motivated by this approach, we use weighting vectors in a similar manner, adding layers with interpretable variables, but reducing the number of channels in every layer required for motion prediction. We show how similar movements have similar interpretable patterns in later sections.

2.2. Model Interpretability

The application of the CAM [33, 41, 43] methods is limited to GNN structures due to they have special requirements and assume heuristically that the final node embedding can reflect the input importance. This assumption may be wrong [40]. While saliency map techniques are commonly employed for model interpretability, they are primarily designed for images. Many of these methods were evaluated solely in classification tasks, applied as independent post-training steps, or require subsequent visual val-

idation. Given the nature of our problem, graph interpretation is required [40]. In [12], a model capable of incorporating interpretable attention is proposed. Later, applications started to use interpretable GCNs (IGCN) [4, 12, 13]. Other approaches [5, 17, 23, 38] need extra heavy computation to obtain interpretable features. A new method called GraphLIME [10] proposed a generic GNN-model explanation framework consisting of a local interpretable model explanation. Our model has similarities with IGCN and GraphLIME methods. However, we distinguish ourselves by explaining the relationship between frame-to-frame and joint-to-joint predictions, which deals with a subset of graph data. Specifically, GraphLIME [10] offers prediction explanations for GNN architectures, whereas our model provides interpretation output alongside motion predictions in a specific data structure. While IGCN [12] has shown combining prediction and interpretation on classification tasks with diverse situations, it only was tested on classification tasks and different data structures. In [5], self-explainable GNN can find K-nearest labeled nodes for the unlabeled nodes to explain the classification output. However, this approach only was tested with a synthetic dataset on classification tasks not related to motion movements. Also, it is well-known that post-hoc explanations can suffer bias and misrepresentation due to interpretations are not directly obtained from the GNNs [5]. In contrast, our model predicts not only motion but also its interpretations by means of adjacency matrices and weighting vectors, solely utilizing GCN layers without requiring external post-processing methods while being applied to regression tasks.

3. Methodology

3.1. Problem Formalization

We define the body motion as a sequence of poses $X \in \mathbb{R}^{T\times J\times D}$ where T and J define a number of frames and joints from the sequence, and D parameterizes each body joint dimension for angles or 3D coordinates. Our model receives a historical input poses $X_{in} = X_{0:t_1} = x_0, x_1, ..., x_{t_1-1} \in \mathbb{R}^{t_1\times J\times D}$ and predict a sequence of poses $\hat{X} = X_{t_1:t_2} = \hat{x}_{t_1}, \hat{x}_{t_1+1}, ..., \hat{x}_{t_1+t_2} \in \mathbb{R}^{t_2\times J\times D}$. The math representation for an adjacency matrix is given by $A \in \mathbb{R}^{P\times P}$ where $P \in \mathbb{R}^n$ is the node representation.

3.2. Review of GCN

Graph Convolutional Networks Graph Convolutions (GCs) are suitable for non-grid data, where data is represented by a set of nodes (e.g. x,y,z coordinates) carrying n-dimensional information. When GCs are stacked sequentially then they together become a GCN. In this work, such a set of nodes is called a pose, and an adjacency matrix shows the connection between pairs of nodes from the whole graph. Formally, let $H^l \in \mathbb{R}^{P\times F^l}$, $A^l \in \mathbb{R}^{P\times P}$ be the input and the adjacency matrix at the current layer l, whereas the trainable parameters at current layer l are represented by $W^l \in \mathbb{R}^{F^l\times F^{l+1}}$. F is the number of channels of this layer. We show the mathematical operation in Eq. (1) where σ is the activation function and $H^{l+1} \in \mathbb{R}^{P\times F^{l+1}}$ is the GC output.

$$H^{l+1} = \sigma(A^l H^l W^l) \tag{1}$$

Spatio-Temporal GCN Given that our problem contains a temporal factor in the data, we thus have a spatial-temporal graph. To process this graph we use two graph convolution operations, for the spatial and temporal domains, just like the STS-GCN model [34], which takes the interactions of the temporal evolution and the spatial joints. Spatial and temporal graph convolutions are presented in Eq. (2). Where W_s^l and $W_t^l \in \mathbb{R}^{F^l\times F^{l+1}}$ are trainable parameters. Graph convolutions are separable if we operate independently and stack later, as argued later in [7].

$$H^{l+1} = \sigma(A_t^l A_s^l H^l W^l) = \sigma(A_t^l (A_s^l H^l W_s^l) W_t^l) \tag{2}$$

We perform a similar approach and demonstrate via experimentation this operation is stable in Eq. (3). Where $W_D^l \in \mathbb{R}^{F^l\times F^{l+1}}$ are trainable parameters, and D represents the temporal or spatial domain.

$$H_D^{l+1} = \sigma(A_D^l H_D^l W_D^l) \tag{3}$$

3.3. Model architecture

Motivated by the interpretability of the feature importance of random forest, we built a model not only for pose sequence prediction but also for output understanding via feature importance and connectivity matrices similar to previous works [7, 34, 42]. This architecture is shown in Fig. 1. We argue that our results generate feature maps that can be used to observe and figure out unexpected behaviors in certain OOD data. More concretely, Fig. 1a uses an encoder-decoder architecture but splits the temporal and spatial GCNs. Inspired by DeepLabv3+ [3], we propose to replace the original TCN described in [34] with a new Atrous Pyramid TCN (APTCN). Also, we propose the Context Network (ConNet) and the Dynamic Spatio-Temporal Graph Convolutions Network (DST-GCN) placed in parallel that sum the results with a global residual connection to obtain the final pose sequence. Additionally, following previous works that use trajectory representation successfully as inputs [20, 21, 35]. We propose to use an overall of 10 input dimensions, 3 for joint positions, 6 for joint instant velocities and accelerations in x,y and z, and 1 L_2-norm vector of the instant velocities. We support the idea that the last two layers from the model, DST-GCN and ConNet, can

Table 1. Performance comparison for motion prediction using MPJPE in every action from the Human3.6M dataset. (*) implies metric is computed by us using our pipeline and the standard metric. (†) metric takes the average MPJPE over all previous frames.

Time (ms)	Walking 80	160	320	400	560	1000	Eating 80	160	320	400	560	1000	Smoking 80	160	320	400	560	1000	Discussion 80	160	320	400	560	1000
ConvSeq2Seq [14]	17.7	33.5	56.3	63.6	72.2	82.3	11.0	22.4	40.7	48.4	61.3	87.1	11.6	22.8	41.3	48.9	60.0	81.7	17.1	34.5	64.8	77.6	98.1	129.3
Traj-CNN [19]	11.9	22.5	38.7	45.7	54.5	62.	8.4	16.6	32.4	39.8	53.5	78.4	8.4	16.2	31.1	37.6	49.3	72.3	11.7	26.3	57.3	70.4	91.4	122.7
STS-GCN [34]*	12.0	23.0	41.5	48.5	56.6	62.7	7.9	16.8	33.4	40.7	53.1	76.7	7.5	15.7	31.3	38.4	50.7	73.1	11.4	26.4	57.8	71.1	91.2	120.8
MSR-GCN [6]	12.2	22.6	38.6	45.2	52.7	63.0	8.4	17.0	33.0	40.4	52.5	77.1	8.0	16.3	31.3	38.2	49.4	71.6	12.0	26.8	57.1	69.7	88.6	117.6
MultiAttention [29]	**9.9**	**19.3**	**33.7**	**39.0**	**46.2**	**57.1**	7.9	17.5	37.4	45.2	**48.6**	73.7	7.0	14.3	**25.4**	**29.0**	**46.5**	68.7	**8.6**	22.8	**51.0**	**64.0**	**85.2**	117.5
DSTD-GCN [7]	11.0	22.4	38.8	45.2	52.7	59.8	7.0	15.5	31.7	39.2	51.9	76.2	**6.6**	14.8	29.8	36.7	48.1	71.2	10.0	24.4	54.5	67.4	87.0	116.3
MotionMixer [2]	10.8	22.4	36.5	42.4	-	59.9	7.7	**14.0**	27.3	**36.1**	-	76.6	7.1	**14.0**	29.1	36.8	-	**68.5**	10.2	**22.5**	**51.0**	64.1	-	117.4
PGBIG [25]	10.2	19.8	34.5	40.3	48.1	56.4	7.0	15.1	30.6	38.1	51.1	76.0	**6.6**	14.1	28.2	34.7	**46.5**	69.5	10.0	23.8	53.6	66.7	87.1	118.2
M16	12.0	23.6	41.0	46.8	54.3	61.9	6.9	15.1	30.6	37.8	50.8	75.3	7.5	15.7	31.5	38.4	49.7	71.5	10.3	24.1	52.8	65.8	85.9	**115.1**
M32	11.8	23.4	40.5	46.5	54.1	61.3	**6.7**	14.8	**29.8**	36.8	49.8	**74.7**	7.3	15.6	31.0	38.0	49.4	70.7	10.2	23.7	52.3	65.3	86.1	115.9

Time (ms)	Directions 80	160	320	400	560	1000	Greeting 80	160	320	400	560	1000	Phoning 80	160	320	400	560	1000	Posing 80	160	320	400	560	1000
ConvSeq2Seq [14]	13.5	29.0	57.6	69.7	86.6	115.8	22.0	45.0	82.0	96.0	116.9	147.3	13.5	26.6	49.9	59.9	77.1	114.0	16.9	36.7	75.7	92.9	122.5	187.4
Traj-CNN [19]	8.7	19.3	43.6	54.4	74.6	109.4	15.8	35.1	73.6	88.9	110.8	149.6	10.1	20.5	42.0	51.9	69.3	104.4	12.1	26.9	62.4	79.3	108.4	170.9
STS-GCN [34]*	7.8	18.7	42.6	53.2	71.0	102.1	15.3	35.0	73.4	89.1	112.2	143.9	9.5	20.4	41.6	51.1	68.3	103.7	11.6	27.6	63.8	81.2	111.7	168.4
MSR-GCN [6]	8.6	19.6	43.3	53.8	71.2	100.6	16.5	37.0	77.3	93.4	116.3	147.3	10.1	20.7	41.5	51.3	68.3	104.3	12.8	29.4	67.0	85.0	116.3	174.3
MultiAttention [29]	11.3	22.9	50.6	62.6	72.4	105.7	12.9	26.6	68.2	85.4	**100.5**	136.7	11.2	19.6	37.7	**44.1**	66.5	104.6	9.8	23.7	62.2	78.7	**105.8**	172.9
DSTD-GCN [7]	**6.9**	**17.4**	41.0	51.7	**69.0**	**99.0**	14.3	33.5	72.2	87.3	108.7	142.3	8.5	19.2	40.3	49.9	66.7	**102.2**	10.1	25.4	60.6	77.3	106.5	**163.3**
MotionMixer [2]	8.3	18.1	43.8	53.4	-	105.4	**12.8**	33.4	**62.3**	82.2	-	136.5	10.0	20.1	**37.4**	51.1	-	104.4	11.7	**23.3**	62.4	79.5	-	174.9
PGBIG [25]	7.2	17.6	**40.9**	**51.5**	69.3	100.4	15.2	34.1	71.6	87.1	110.2	143.5	**8.3**	**18.3**	38.7	48.4	**65.9**	102.7	10.7	25.7	60.0	76.6	106.1	164.8
M16	7.5	18.7	44.8	56.4	73.6	105.2	13.8	31.1	66.7	80.8	102.6	**133.9**	8.6	18.5	39.5	49.4	67.3	103.6	10.0	24.1	58.9	76.2	107.2	169.3
M32	7.3	18.1	43.6	55.3	72.8	105.5	13.7	**31.0**	65.7	**79.9**	101.4	135.7	8.6	18.5	39.3	49.6	67.4	103.5	**9.6**	23.7	**57.7**	**75.0**	**105.8**	168.7

Time (ms)	Purchases 80	160	320	400	560	1000	Sitting 80	160	320	400	560	1000	Sitting Down 80	160	320	400	560	1000	Taking Photo 80	160	320	400	560	1000
ConvSeq2Seq [14]	20.3	41.8	76.5	89.9	111.3	151.5	13.5	27.0	52.0	63.1	82.4	120.7	20.7	40.6	70.4	82.7	106.5	150.3	12.7	26.0	52.1	63.6	84.4	128.1
Traj-CNN [19]	14.5	31.9	66.6	80.8	103.6	141.0	11.0	21.2	45.5	57.5	79.0	120.1	16.1	29.6	58.7	72.6	97.0	147.0	10.4	20.6	44.4	55.8	76.8	120.1
STS-GCN [34]*	13.9	31.7	66.0	80.0	102.5	142.5	9.6	20.6	45.2	57.3	79.0	122.0	15.0	29.6	59.4	73.6	98.8	149.5	9.2	19.9	43.4	55.0	76.2	118.8
MSR-GCN [6]	14.8	32.4	66.1	79.6	101.6	139.2	10.5	22.0	46.3	57.8	78.2	120.0	16.1	31.6	62.4	76.8	102.8	155.5	9.9	21.0	44.6	56.3	78.0	121.9
MultiAttention [29]	18.1	36.8	58.4	67.9	**94.5**	**133.1**	9.9	24.3	53.8	66.3	75.8	115.0	**10.4**	**26.6**	**54.6**	**66.3**	96.0	141.8	5.9	**14.8**	38.0	49.4	**71.8**	115.2
DSTD-GCN [7]	12.7	29.6	62.3	75.8	97.5	137.8	**8.8**	19.3	42.9	54.3	74.9	117.8	14.1	28.0	57.3	71.2	96.1	147.2	8.4	18.8	42.0	53.5	74.5	117.9
MotionMixer [2]	14.6	31.3	62.8	76.1	-	135.1	10.0	20.9	43.7	54.5	-	115.7	12.0	31.4	61.4	74.5	-	141.1	9.0	18.9	41.0	**51.6**	-	**114.6**
PGBIG [25]	**12.5**	**28.7**	**60.1**	**73.3**	95.3	133.3	**8.8**	**19.2**	42.4	53.8	74.4	116.1	13.9	27.9	57.4	71.5	96.7	147.8	8.4	18.9	42.0	53.3	74.3	118.6
M16	13.0	30.3	62.8	76.7	97.9	136.2	8.9	19.4	42.7	53.9	74.2	113.4	14.4	30.2	58.5	71.3	95.7	141.6	8.5	18.6	41.0	51.7	72.9	116.4
M32	13.3	30.2	63.0	77.3	97.7	134.8	8.9	19.4	**42.3**	**53.6**	**73.9**	**113.0**	14.1	29.8	57.3	69.8	**94.3**	**140.2**	**8.2**	18.4	**40.6**	51.8	73.0	116.6

Time (ms)	Waiting 80	160	320	400	560	1000	Walking Dog 80	160	320	400	560	1000	Walking Together 80	160	320	400	560	1000	Average 80	160	320	400	560	1000
ConvSeq2Seq [14]	14.6	29.7	58.1	69.7	87.3	117.7	27.7	53.6	90.7	103.3	122.4	162.4	15.3	30.4	53.1	61.2	72.0	87.4	16.6	33.5	62.0	73.5	92.1	126.8
Traj-CNN [19]	10.5	21.8	45.8	56.3	73.4	104.5	21.3	43.3	80.8	94.5	115.6	153.5	10.3	21.1	38.5	44.8	54.8	68.0	12.1	24.9	50.7	62.0	80.8	114.9
STS-GCN [34]*	10.0	21.9	47.0	58.2	76.4	107.7	20.8	43.6	81.8	95.2	114.4	151.9	10.1	20.7	39.1	46.0	54.9	62.9	11.4	24.8	51.2	62.6	81.1	113.8
MSR-GCN [6]	10.7	23.1	48.2	59.2	76.3	106.3	20.6	42.9	80.4	93.3	111.9	148.2	10.6	20.9	37.4	43.8	52.9	65.9	12.1	25.6	51.6	62.9	81.1	114.2
MultiAttention [29]	9.0	22.5	55.7	71.1	72.7	105.1	29.5	54.8	100.3	105.1	119.0	141.4	**8.0**	**17.6**	**33.2**	42.0	51.2	63.2	11.0	23.6	49.2	60.0	**75.9**	**110.1**
DSTD-GCN [7]	8.7	20.2	44.3	55.2	73.2	105.7	19.6	41.8	77.6	90.2	109.8	147.7	9.1	19.8	36.3	42.7	**50.5**	**61.2**	10.4	23.3	48.8	59.8	77.8	111.0
MotionMixer [2]	10.2	21.1	45.2	56.4	-	107.7	20.5	42.8	75.6	87.8	-	142.2	10.5	20.6	38.7	43.5	-	65.4	11.0	23.6	47.8	59.3	-	111.0
PGBIG [25]	8.9	20.1	43.6	**54.3**	**72.2**	**103.4**	**18.8**	**39.3**	**73.7**	86.4	104.7	**139.8**	8.7	18.6	34.4	**41.0**	51.9	64.3	**10.3**	**22.7**	**47.4**	**58.5**	76.9	110.3
M16	8.7	19.5	43.6	54.6	73.2	104.6	19.9	40.9	74.1	86.6	106.6	149.2	9.7	20.4	38.5	45.8	55.2	64.7	10.6	23.3	48.5	59.5	77.8	110.8
M32	**8.6**	**19.4**	**43.5**	54.8	73.6	105.4	20.0	41.4	**73.7**	**85.1**	**103.8**	143.2	9.6	20.3	38.2	45.6	55.4	64.6	10.5	23.2	47.9	59.0	77.2	110.3

Table 2. Performance comparison on different architectures using MPJPE for common action split from the ExPI dataset.

Time (ms)	A1 0.2	0.4	0.6	0.8	1.0	A2 0.2	0.4	0.6	0.8	1.0	A3 0.2	0.4	0.6	0.8	1.0	A4 0.2	0.4	0.6	0.8	1.0	A5 0.2	0.4	0.6	0.8	1.0	A6 0.2	0.4	0.6	0.8	1.0	A7 0.2	0.4	0.6	0.8	1.0	AVG 0.2	0.4	0.6	0.8	1.0
LTD [28]	70	125	157	-	189	131	242	321	-	426	102	194	260	-	357	62	117	155	-	197	72	131	173	-	231	81	151	200	-	280	112	223	315	-	442	90	169	226	-	303
HRI [27]	52	103	139	-	188	96	186	256	-	349	57	118	167	-	240	45	93	131	-	180	51	105	149	-	214	61	125	176	-	252	71	150	222	-	333	62	126	177	-	251
MSR-GCN [6]	56	100	132	-	175	102	187	256	-	365	65	120	166	-	244	50	95	127	-	172	54	100	138	-	202	70	132	182	-	258	82	154	218	-	321	69	127	174	-	248
XIA [8]	49	98	140	-	192	84	166	234	-	346	51	105	154	-	234	41	84	120	-	161	**43**	90	132	-	197	**55**	113	163	-	**242**	**62**	**130**	**192**	-	**291**	**55**	112	162	-	**238**
M16	50	92	124	149	173	89	171	241	296	350	54	109	157	195	235	40	80	115	141	163	47	95	138	174	208	58	115	166	212	255	72	157	236	313	376	58	117	168	211	251
M32	**47**	**89**	**120**	**145**	**169**	**82**	**159**	**228**	**282**	**338**	**49**	**100**	**147**	**183**	**218**	**38**	**76**	**109**	**136**	**159**	45	**89**	**128**	**162**	**194**	**55**	**110**	**159**	**203**	245	70	155	238	**313**	378	**55**	**111**	**161**	**203**	243

satisfactorily obtain interpretability from the output pose sequence whereas the DST-GCN blocks in the input can extract relevant information regarding the input sequence.

Interpretable-GCN layer. We implement DST-GCN split into the Dynamic Spatial Graph Network (DSGN) and Dynamic Temporal Graph Network (DTGN) that are controlled by a Gating Network (GaNet) as shown in 1b, F is set given the model size and is detailed in Section 4.2. The interpretable information from DST-GCN is located in the adjacency matrices of the GCNs. In contrast to most GCNs, we consider learning sample-specific connections in the adjacency matrix has a more meaningful representation and

Table 3. Performance comparison on different architectures using MPJPE for unseen action split from the ExPI dataset.

	A8					A9					A10					A11					A12					A13					A14					A15					A16					AVG				
Time (ms)	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
LTD [28]	-	239	324	394	-	-	175	226	259	-	-	148	191	220	-	-	176	240	286	-	-	143	178	192	-	-	146	193	226	-	-	252	333	387	-	-	174	228	264	-	-	139	184	217	-	-	177	233	272	-
HRI [27]	-	195	283	358	-	-	121	169	206	-	-	92	129	160	-	-	129	193	245	-	-	80	104	121	-	-	112	154	187	-	-	157	219	257	-	-	134	190	233	-	-	96	146	187	-	-	124	176	218	-
MSR-GCN [6]	-	230	289	335	-	-	188	245	290	-	-	148	198	248	-	-	234	319	384	-	-	176	232	278	-	-	162	218	266	-	-	177	239	295	-	-	143	179	213	-	-	157	222	281	-	-	179	238	288	-
XIA [8]	-	191	287	377	-	-	118	165	203	-	-	**91**	**129**	**162**	-	-	**122**	**183**	**232**	-	-	**81**	**107**	**128**	-	-	106	150	185	-	-	**156**	**216**	**256**	-	-	126	175	213	-	-	96	152	205	-	-	121	174	218	-
M16	54	115	170	220	257	55	90	110	128	153	52	104	147	185	219	80	156	213	256	293	65	130	183	226	260	43	88	**129**	171	212	94	192	282	356	410	63	126	172	210	258	50	98	135	166	203	62	122	171	213	252
M32	**50**	**112**	169	219	**257**	**51**	**86**	**105**	**121**	**145**	**52**	107	151	190	**225**	**77**	150	203	243	**278**	**61**	123	174	215	**246**	**43**	**87**	**129**	**169**	**210**	**87**	185	277	354	**413**	**58**	**120**	**166**	**208**	**252**	**46**	**89**	**120**	**148**	**187**	**58**	**118**	**166**	**207**	**246**

Table 4. (left) (a) Performance comparison between different architectures using MPJPE for the AMASS and 3DPW datasets. (*) metric is computed by us using our pipeline and the standard metric. (†) metric takes the average MPJPE over all previous frames. (right) (b) Comparison summary of average MPJPE (using only 80, 160, 320, 400, and 1000ms), number of parameters and FLOPs.

Time (ms)	AMASS-BMLrub 80	160	320	400	560	720	880	1000	3DPW 80	160	320	400	560	720	880	1000
GAGCN [42]†	10.0	11.9	20.1	24.0	30.4	-	-	43.1	8.4	11.9	18.7	23.6	29.1	-	-	39.9
HRI [27]	11.3	20.7	35.7	42.0	51.7	58.6	63.4	67.2	12.6	23.1	39.0	45.4	56.0	63.6	69.7	73.7
STS-GCN [34]*	11.2	20.6	36.5	43.1	52.5	59.2	64.3	68.7	11.7	20.7	35.0	40.3	48.7	55.0	59.4	62.4
MultiAttention [29]	11.0	20.3	35.0	41.2	50.7	57.4	61.9	65.8	12.4	22.6	38.1	44.4	54.7	62.1	67.9	71.8
MotionMixer [2]*	10.1	**18.4**	**32.7**	**38.9**	**48.3**	**55.0**	60.4	64.2	10.9	19.4	33.3	39.0	48.4	55.2	60.0	63.6
M16	9.9	18.9	34.1	40.4	50.2	56.9	61.3	64.9	10.6	19.6	33.4	39.0	48.0	54.1	58.8	62.0
M32	**9.8**	18.6	33.6	39.8	49.2	56.0	**60.3**	**63.6**	**10.4**	**19.3**	**33.2**	**38.7**	**47.6**	**54.0**	**58.5**	**61.7**

Model	Human3.6M MPJPE	Params	≈FLOPs
MultiAttention [29]	50.8	3.42M	142.3M
MSR-GCN [6]	53.3	6.3M	192.4M
STS-GCN [34]	52.8	57.5k	7.1M
MotionMixer [2]	50.5	**30.2K**	**2.1M**
DSTD-GCN [7]	50.6	0.18M	-
PGBIG [25]	49.8	1.74M	55.8M
M8	50.8	115.6K	19.5M
M16	50.5	164.0K	21.3M
M32	50.2	345.6K	27.5M
M64	**49.6**	1.048M	49.7M

also helps to interpret graph connections. To do this, we replace the adjacency matrix with a Dynamic Adjacency Encoder (DAE) that provides a feature map of the same size, described below.

Atrous Pyramid TCN. TCN architecture is widely used as a decoder for the output sequences [7, 34, 42]. However, as explained above, we use a larger input dimension and can increase the complexity of the feature search in the output sequence. Given the outstanding results obtained by DeepLabv3+ in image segmentation, we modified the TCN to be pyramidal and used different dilation rates to later concatenate and compress the output, as shown in Fig. 1c.

Context Network. We propose a network to collect statistics and generate feature importance vectors, detailed in Fig. 1d. We think every pooling extracts different feature information, as observed in point clouds [11, 20]. Specifically, we use 3 different pooling operations for the same input. Conv+BN+PReLU blocks and linear layers are applied before and after each pooling. Later, every output with size $o \in \mathbb{R}^T$ is merged in one unique vector with size $O \in \mathbb{R}^{3T}$. Assuming we code the context information, we could extract two feature importance from this vector for spatial and temporal domains: displacement and joint features.

Gating Network (GaNet). After obtaining the output from DSGN and DTGN, we presume that not all feature maps contribute equally, similar to [42], we weight the feature maps. But our GaNet blocks generate two vectors $W1$ and $W2$ with F length but with a different fusion mode. GaNet, as shown in Fig. 1e, implements a lightweight architecture with the use of separable convolutions that reduce the number of parameters. Also, the aggregation of statistic values computed from the same input makes the representation more meaningful to the weighting vectors.

Dynamic Adjacency Encoder (DAE). This block, as shown in Fig. 1f, is responsible to compute the adjacency matrix used in every GCN. This efficient architecture uses convolutional layers not only to map an input feature map into a matrix adjacency shape but also to generate this matrix with a lower number of parameters than other GCNs.

4. Experimental Evaluation

4.1. Datasets

Using a unique model to evaluate short- and long-predictions, same as [27, 29], we conducted experiments on widely used datasets: Human 3.6M [11], AMASS [26], and 3DPW [36]. Additionally, we analyzed multi-pose motion prediction using the ExPI dataset [8].

Human 3.6M. Consists of 15 different actions performed by 7 different actors per action. Following [27] and [7], we downsample the frame rate to 25Hz and use 22 joints from the overall subjects 1,6,7,8,9 for training, subject 11 for validation, and subject 5 for testing.

AMASS. Consists of a gathering of 18 existing datasets. We perform a frame rate down-sampling to 25Hz as in Human 3.6M. Then, following [34] and [2], we select 8, 4, and 1 (BMLrub) datasets for training, validation, and testing respectively. For each body pose, hand joints are discarded and we consider 18 joints for training from the 22 body joints, skipping 5 frames instead of 1.

3DPW. Consists of both indoor and outdoor actions, containing 51,000 frames captured at 30Hz. Following [34] and [2], we only use 3DPW to test the generalization of the models trained on AMASS.

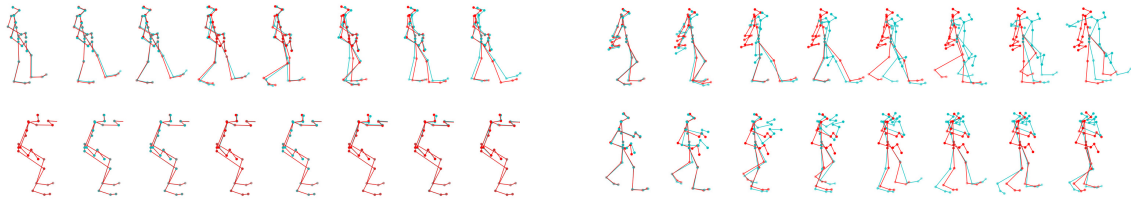

Figure 2. Motion prediction results on "walking" (top) and "eating" (bottom) motion classes from H3.6M dataset. Sorted by the lowest (left) and the largest errors (right). Solid lines are ground truth. Dashed lines are predictions from the $M32$ model. Blue color of the poses represents ground truth while the red color of the poses represents the predicted ones.

ExPI. The recent ExPI dataset contains 115 sequences of 2 professional couples performing 16 different dance actions. It is recorded in a multiview motion capture studio at 25 fps. Following [8], The experiments performed include two persons represented by 18 joints each one in all the 30K frames. We benchmark in both protocols for common and unseen action splits using the same settings.

4.2. Experimental results

Metric. The Mean Per Joint Position Error (MPJPE) is a widely adopted evaluation metric used in previous works [6, 19, 24] to compare two pose sequences and is described in Eq. (4). We evaluate our topology defined as M and the number of channels in the hidden DST-GCN layers for various experiments, i.e. $M8$, $M16$, $M32$, and $M64$.

$$\mathcal{L}_{\text{MPJPE}} = \frac{1}{J \times T} \sum_{t=1}^{T} \sum_{j=1}^{J} \|\hat{x}_{j,t} - x_{j,t}\|_2 \qquad (4)$$

Quantitative results. We first compare our method with the SOTA approaches on the four datasets. Since we realized that some papers used another metric, which calculates the mean error across preceding frames with diverse normalizations, resulting in a lower error compared to the standard metric. we attempted to reproduce these outcomes whenever feasible implementations were obtainable. but our replication efforts relied solely on the standard metric defined in Eq. (4). While running these experiments, we found other noteworthy peculiarities in the literature: Firstly, some authors did not use all motion classes for comparison. Secondly, some authors sampled 256 samples from each motion class while others used the complete test sets. We chose to use the 256-sample variation, as it is more common in the literature and made more sense for a balanced test set. We present in Tab. 1 our model results compared to the benchmark architectures for H3.6M. Regrettably, we did not find an implementation for GAGCN [42]. In general, we observe that more complex actions such as "Purchase", "Sitting Down", "Posing" and "Walking Dog" yield lower performances for all methods since these "spontaneous movements" have large motion variations appear after the input sequence. Although we outperformed most of the SOTA models, we also obtained slightly lower but still comparable performance to MultiAttention and PGBIG models. But, these models are larger in both the number of parameters and demanding complexity as detailed in section 4.3. The newest MotionMixer [2] architecture obtains comparable results using only a feature mix while keeping a lower number of parameters. We also experimented with OOD samples by introducing 3D transformations and noise as adversarial examples to evaluate their impact on the models. Our model demonstrates its robustness on the test set against random rotation attacks (Fig. 4a) when compared to other models, and, performs slightly better than other models, for random noise attacks (depicted in Fig. 4b). This shows the importance of data augmentation in the overall robustness of the model against "natural perturbations".

We also extend our experiments to the AMASS and 3DPW datasets as detailed in Tab. 4a. Our $M32$ and $M16$ models outperform previous works and $M16$ is comparable to MotionMixer. On AMASS, we observe that for some models, the error is similar for short-term predictions while differing more for long-term predictions. Interestingly, MotionMixer reduced significantly the short-term error but the long-term margin remains similar. We obtain a similar behavior to MotionMixer with our two models and assume that this is because of the nature of AMASS dataset which contains many complex samples that are not necessarily cyclic such as "Walking" action from H3.6M. On 3DPW dataset, we observe that long-term predictions differ more between models, and still $M32$ and $M16$ outperform previous approaches. We believe that this behavior happens due to spontaneous movements, even when we find some "Walking" motion in both datasets. GAGCN is not directly comparable to the other methods shown in the table.

Finally, we wanted to explore the multi-pose motion interaction in the new ExPI dataset and how CIST-GCN may behave in this situation. CIST-GCN was initially designed for single-pose predictions. However, it impressively reaches previous SOTA methods in both short-term and long-term. In Tabs. 2 and 3, our model outperforms other approaches by a short margin independent of the protocol type. But, our model complexity is only 0.76M on $M32$ compared to 8.5M from XIA model [8].

Qualitative results. In order to complement the quantitative results, we present in Fig. 2 the results of the $M32$

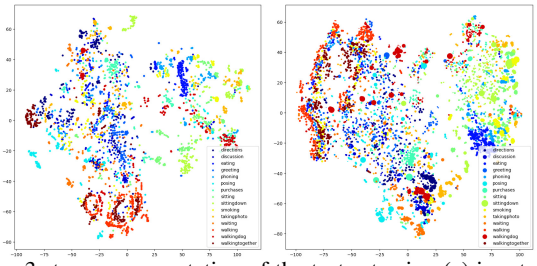

Figure 3. t-sne representation of the test set using (a) input poses (b) all feature importance from the model concatenated. MPJPE values are represented by scatter size.

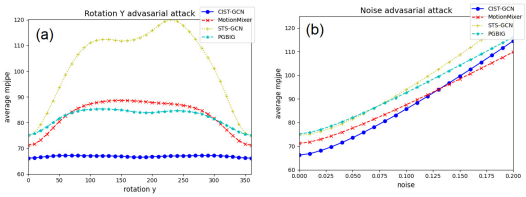

Figure 4. Augmentation effect on test set evaluated on our pipeline. Average MPJPE over the 25 output frames, with (a) rotations between 0-360 degrees, and (b) noise rate between 0.0-0.2.

model for input sequences of the "walking" and "eating" classes from the H3.6M test set. The plot shows the samples with the lowest (left) and largest (right) errors with predictions of 80, 160, 320, 400, 560, 720, and 1000 ms. As we can see, the right side sample (OOD or most difficult case) is arguably very hard to predict without context knowledge, for example, "walking and raising your hand after the input period". This might even be considered as unexpected behavior after the input sequence which is similar to an OOD.

4.3. Computational Complexity

We assess the trade-off between the performance and the approximated computational cost of SOTA architectures versus our model across 4 complexity configurations, as presented in Table 4b. We use the average MPJPE value as a performance reference next to a number of parameters and FLOPs approximation. We observe that our architecture has a lower number of parameters compared to most of the previous works and outperforms other approaches while being lightweight. However, due to the interpretable features implemented in our model, the number of FLOPs increased significantly due to the matrix multiplications and linear layers. Although $M64$ obtained the lowest error and overcomes previous works, it requires a large number of parameters and sometimes has convergence issues. We believe that the sum operator in the DST-GCN blocks can sometimes generate overflow or large gradients, making training unstable. Therefore, we focus on $M32$ for further analysis because of the accuracy-complexity trade-off. PGBIG and MotionMixer have the best MPJPE values from the SOTA models, however, the estimated complexity (FLOPs) for PGBIG is larger due to it requires a multi-stage architecture with intermediate targets whereas MotionMixer underperforms in terms of MPJPE by a small margin our model.

4.4. Implementation details

Our model was trained end-to-end and in a fully supervised manner using Eq. (4) as the loss function for all the experiments. We used data augmentation composed of random noises, rotations, scales, and translations. Inspired by scalable modeling, we control the model size by two hyperparameters, complexity, and the number of layers. We stack the DST-GCN module five times for the input but only apply the DST-GCN module once for the output layer as shown in Fig. 1a. The complexity was set on 8, 16, 32, and 64 for simplicity (details in supplementary material and our code).

5. Discussion

In this section, we focus on the $M32$ model. Additional details can also be found in the supplementary materials.

5.1. Feature importance vectors

We explore the significance of interpretations learned from the model by comparing them to another data representation, as depicted in Fig. 3. We concatenate all features' importance obtained by every model layer. In contrast to the approach in GAGCN [42], where authors computed the average of 16 samples from 4 motion classes and plotted differences of the blending coefficients, we use the t-SNE algorithm to visualize the entire test set. This approach avoids interpretation bias, especially when certain classes share similar movements, as shown in Fig. 2. In Fig. 3, only pure 3D euclidean poses and corresponding feature importance representations are shown (see supplementary material for displacement representations). Our observation reveals that using pure input poses, as seen in Fig. 3a, results in a cluster-like distribution visualization for some motion sequences, while others exhibit less pronounced grouping and higher variance without distinct centroids. In comparison to other representations, Fig. 3b effectively shows well-grouped motion classes, while also unveiling instances of larger MPJPE located away from the centroids. This experiment found a similar grouping as in GAGCN [42] for "walking" and "sitting". A similar interpretation could be seen in other representations using other grouping strategies. Quantitatively, we measured distances from points to centroids and found that our data representation was more accurate. We can also utilize average vectors for both the temporal and spatial domains to conduct soft clustering on movements. Also, given our architecture uses a global residual connection, the model indeed is learning displacements from the last input. This is particularly useful in inference since movements can be ambiguous sometimes, leading to instances where a sequence may exhibit a blend of multiple motion actions such as "walking" and

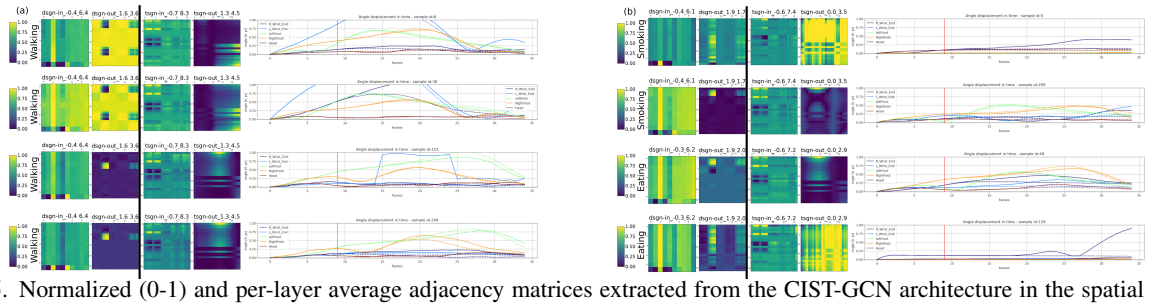

Figure 5. Normalized (0-1) and per-layer average adjacency matrices extracted from the CIST-GCN architecture in the spatial (left) and temporal (right) domains for (a) walking, and (b) other motion actions. The right parts display changes in the angles of movement.

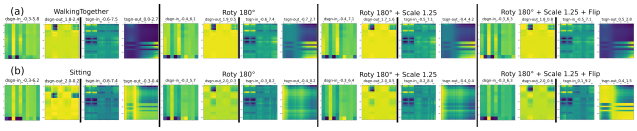

Figure 6. Normalized (0-1) and per-layer average adjacency matrices for (a) "walkingtogether" and (b) "sitting" motion actions.

"eating". Then, we could predict the future motion by also getting an idea about the kind of motion that is observed.

5.2. Feature Maps

After evaluating Fig. 3b, we observe that motion classes may behave as a mix of at least two motion classes. A qualitative analysis of the saliency maps shows us the motion behavior. Consistent with DSTD-GCN [7], the interpretable adjacency matrices are equivalent in some way to relation matrices. In Fig. 5a, we present the saliency maps, along with the spatial and temporal matrices, corresponding to walking actions, as well as the associated variations in movement angles (see supplementary materials for relative angle computation). We observe that when the input sequence comprises cyclic movements like walking, the temporal saliency maps ("tsgn-out" for the output) prominently feature values close to 0. On the other hand, the output spatial saliency maps ("dsgn-out") present lower values when the right foot starts the cyclic movement before the left foot, we observe the opposite behavior when these spatial maps are mostly 1. We have observed that when the input sequence comprises cyclic movements like walking, the output temporal saliency maps ("tsgn-out") prominently feature values close to 0. Conversely, the output spatial saliency maps (dsgn-out) exhibit lower values when the cyclic movement begins with the right foot before the left. By contrast, when the left foot begins the movement, the spatial maps predominantly display values near to 1. To broaden this analysis to additional motion classes, we observe in Fig. 5b that similar patterns persist in "tsgn-out" when the movement is cyclic, however, when the model predict static motions "tsgn-out" displays a prevalence of values near 1. Conversely, when we evaluate the averages of the input saliency maps ("tsgn-in" and "dsgn-in"), the patterns are faded in the hidden layers, making motion identification a challenging task. Sometimes, the saliency maps exhibit inconsistencies for the highest MPJPE values. The saliency maps should show the relation between frames and joints, however, it becomes apparent that the precise values are distributed across consecutive rows and columns. This phenomenon is evident when we present in Fig. 5 the saliency label alongside the map's mean and standard deviation. Similar patterns are observed when input sequences are rotated or slightly scaled as shown in Fig. 6. Essentially, this shows that the matrices exhibit similarity for similar predictions, enabling us to derive comparable interpretations irrespective of the object's position, orientation, or scale. Given our model forecasts displacements, the output layers contain more substantial and visually prominent information. Additional experiments are necessary to have a deeper comprehension of which layer(s) within our network transform input displacements into output displacements.

6. Conclusions and Future Outlook

We have introduced a novel architecture for human motion prediction using GCNs. The evaluations show that our approach obtains comparable and/or superior performance to SOTA models. As observed, our model is a robust approach not only for motion prediction but also for achieving a certain interpretability level. We discussed the effects of data augmentation and OOD data, also showed the robustness of our models against previous works. For future outlook, we plan to extend our study on adversarial attacks and OOD to have a deeper understanding of the feature maps and how the model complexity can be optimized.

Acknowledgement

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Climate Action within the project "ATTENTION – Artificial Intelligence for realtime injury prediction". The authors would like to thank the consortium for the successful cooperation.

References

[1] Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. A Spatio-temporal Transformer for 3D Human Motion Prediction. apr 2020. 1

[2] Arij Bouazizi, Adrian Holzbock, Ulrich Kressel, Klaus Dietmayer, and Vasileios Belagiannis. MotionMixer: MLP-based 3D Human Body Pose Forecasting. jul 2022. 2, 4, 5, 6

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. feb 2018. 3

[4] Hejie Cui, Wei Dai, Yanqiao Zhu, Xiaoxiao Li, Lifang He, and Carl Yang. Interpretable Graph Neural Networks for Connectome-Based Brain Disorder Analysis. jun 2022. 1, 3

[5] Enyan Dai and Suhang Wang. Towards Self-Explainable Graph Neural Network. aug 2021. 3

[6] Lingwei Dang, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. MSR-GCN: Multi-Scale Residual Graph Convolution Networks for Human Motion Prediction. aug 2021. 4, 5, 6

[7] Jiajun Fu, Fuxing Yang, Xiaoli Liu, and Jianqin Yin. Learning Constrained Dynamic Correlations in Spatiotemporal Graphs for Motion Prediction. apr 2022. 2, 3, 4, 5, 8

[8] Wen Guo, Xiaoyu Bie, Xavier Alameda-Pineda, and Francesc Moreno-Noguer. Multi-Person Extreme Motion Prediction. may 2021. 2, 4, 5, 6

[9] Xiao Guo and Jongmoo Choi. Human Motion Prediction via Learning Local Structure Representations and Temporal Dependencies. feb 2019. 1

[10] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–6, 2022. 3

[11] 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, jul 2014. 2, 5

[12] Anees Kazi, Soroush Farghadani, and Nassir Navab. IA-GCN: Interpretable Attention based Graph Convolutional Network for Disease prediction. mar 2021. 1, 3

[13] Anees Kazi, S. Arvind Krishna, Shayan Shekarforoush, Karsten Kortuem, Shadi Albarqouni, and Nassir Navab. Self-Attention Equipped Graph Convolutions for Disease Prediction. dec 2018. 3

[14] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. Convolutional Sequence to Sequence Model for Human Dynamics. may 2018. 4

[15] 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. mar 2020. 1

[16] Maosen Li, Siheng Chen, Yangheng Zhao, Ya Zhang, Yanfeng Wang, and Qi Tian. Multiscale Spatio-Temporal Graph Neural Networks for 3D Skeleton-Based Motion Prediction. *IEEE Transactions on Image Processing*, 30:7760–7775, aug 2021. 1, 2

[17] Yuan Li, Li Liu, Guoyin Wang, Yong Du, and Penggang Chen. EGNN: Constructing explainable graph neural networks via knowledge distillation. *Knowledge-Based Systems*, 241:108345, apr 2022. 3

[18] Lu Liu, Yibo Cao, and Yuhan Dong. Attention-Based Multiple Graph Convolutional Recurrent Network for Traffic Forecasting. *Sustainability*, 15(6):4697, mar 2023. 1

[19] Xiaoli Liu, Jianqin Yin, Jin Liu, Pengxiang Ding, Jun Liu, and Huaping Liu. TrajectoryCNN: A New Spatio-Temporal Feature Learning Network for Human Motion Prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(6):2133–2146, jun 2021. 2, 4, 6

[20] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-Shape Convolutional Neural Network for Point Cloud Analysis. apr 2019. 3, 5

[21] Zhenguang Liu, Pengxiang Su, Shuang Wu, Xuanjing Shen, Haipeng Chen, Yanbin Hao, and Meng Wang. Motion Prediction using Trajectory Cues. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13279–13288. IEEE, oct 2021. 3

[22] Zhenguang Liu, Shuang Wu, Shuyuan Jin, Shouling Ji, Qi Liu, Shijian Lu, and Li Cheng. Investigating Pose Representations and Motion Contexts Modeling for 3D Motion Prediction. dec 2021. 1

[23] Antonio Longa, Steve Azzolin, Gabriele Santin, Giulia Cencetti, Pietro Liò, Bruno Lepri, and Andrea Passerini. Explaining the Explainers in Graph Neural Networks: a Comparative Study. oct 2022. 3

[24] Kedi Lyu, Haipeng Chen, Zhenguang Liu, Beiqi Zhang, and Ruili Wang. 3D Human Motion Prediction: A Survey. mar 2022. 1, 2, 6

[25] Tiezheng Ma, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. Progressively Generating Better Initial Guesses Towards Next Stages for High-Quality Human Motion Prediction. mar 2022. 1, 4, 5

[26] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael Black. AMASS: Archive of Motion Capture As Surface Shapes. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5441–5450. IEEE, oct 2019. 2, 5

[27] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. History Repeats Itself: Human Motion Prediction via Motion Attention. jul 2020. 1, 2, 4, 5

[28] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning Trajectory Dependencies for Human Motion Prediction. aug 2019. 1, 2, 4, 5

[29] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Multi-level Motion Attention for Human Motion Prediction. *International Journal of Computer Vision*, 129(9):2513–2535, sep 2021. 4, 5

[30] Julieta Martinez, Michael J. Black, and Javier Romero. On human motion prediction using recurrent neural networks. may 2017. 1

[31] Felipe Moreno-Vera, Edgar Medina, and Jorge Poco. WSAM: Visual Explanations from Style Augmentation as Adversarial Attacker and Their Influence in Image Classification. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 830–837. SCITEPRESS - Science and Technology Publications, 2023. 1

[32] Jishnu Mukhoti, Joost van Amersfoort, Philip H. S. Torr, and Yarin Gal. Deep Deterministic Uncertainty for Semantic Segmentation. oct 2021. 1

[33] Zhenyue Qin, Dongwoo Kim, and Tom Gedeon. Informative Class Activation Maps. jun 2021. 2

[34] Theodoros Sofianos, Alessio Sampieri, Luca Franco, and Fabio Galasso. Space-Time-Separable Graph Convolutional Network for Pose Forecasting. oct 2021. 1, 2, 3, 4, 5

[35] Pengxiang Su, Zhenguang Liu, Shuang Wu, Lei Zhu, Yifang Yin, and Xuanjing Shen. Motion Prediction via Joint Dependency Modeling in Phase Space. jan 2022. 1, 3

[36] 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. pages 614–631. 2018. 2, 5

[37] Hongsong Wang, Jian Dong, Bin Cheng, and Jiashi Feng. PVRED: A Position-Velocity Recurrent Encoder-Decoder for Human Motion Prediction. *IEEE Transactions on Image Processing*, 30:6096–6106, 2021. 1

[38] Jiaxuan Xie, Yezi Liu, and Yanning Shen. Explaining Dynamic Graph Neural Networks via Relevance Backpropagation. jul 2022. 3

[39] Zigeng Yan, Di-Hua Zhai, and Yuanqing Xia. DMS-GCN: Dynamic Mutiscale Spatiotemporal Graph Convolutional Networks for Human Motion Prediction. dec 2021. 1, 2

[40] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in Graph Neural Networks: A Taxonomic Survey. dec 2020. 2, 3

[41] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A Survey on Neural Network Interpretability. dec 2020. 1, 2

[42] Chongyang Zhong, Lei Hu, Zihao Zhang, Yongjing Ye, and Shihong Xia. Spatio-Temporal Gating-Adjacency GCN for Human Motion Prediction. mar 2022. 1, 2, 3, 5, 6, 7

[43] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. dec 2015. 1, 2

[44] Honghong Zhou, Caili Guo, Hao Zhang, and Yanjun Wang. Learning Multiscale Correlations for Human Motion Prediction. In *2021 IEEE International Conference on Development and Learning (ICDL)*, pages 1–7. IEEE, aug 2021. 1, 2