GCNext: Towards the Unity of Graph Convolutions for Human Motion Prediction

Xinshun Wang[1,2], **Qiongjie Cui**[3], **Chen Chen**[4], **Mengyuan Liu**[2*]

[1]School of Intelligent Systems Engineering, Sun Yat-sen University
[2]National Key Laboratory of General Artificial Intelligence, Peking University, Shenzhen Graduate School
[3]Xiaohongshu Inc.
[4]Center for Research in Computer Vision, University of Central Florida

Abstract

The past few years has witnessed the dominance of Graph Convolutional Networks (GCNs) over human motion prediction. Various styles of graph convolutions have been proposed, with each one meticulously designed and incorporated into a carefully-crafted network architecture. This paper breaks the limits of existing knowledge by proposing **Universal Graph Convolution** (**UniGC**), a novel graph convolution concept that re-conceptualizes different graph convolutions as its special cases. Leveraging UniGC on network-level, we propose **GCNext**, a novel GCN-building paradigm that dynamically determines the best-fitting graph convolutions both sample-wise and layer-wise. GCNext offers multiple use cases, including training a new GCN from scratch or refining a preexisting GCN. Experiments on Human3.6M, AMASS, and 3DPW datasets show that, by incorporating unique module-to-network designs, GCNext yields up to $9\times$ lower computational cost than existing GCN methods, on top of achieving state-of-the-art performance. Our code is available at https://github.com/BradleyWang0416/GCNext.

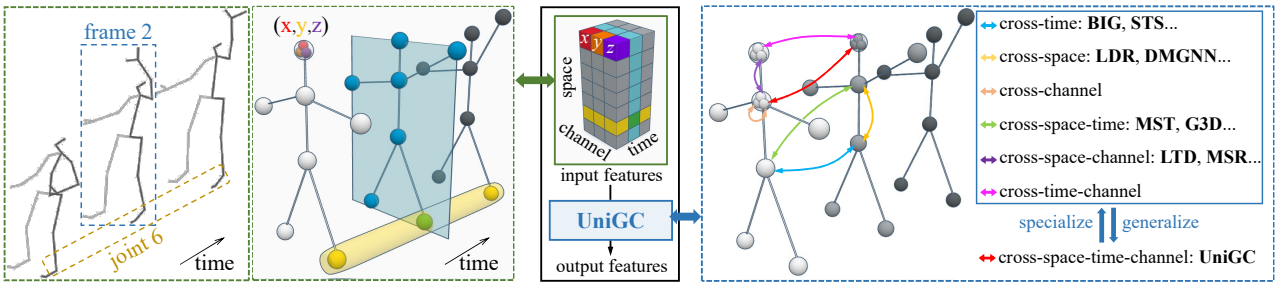
Introduction

The idea of predictable human motion has attracted much research interest over the years across a wide range of applications such as human-robot interaction and autonomous driving. To generate plausible predictions, early successes were achieved with RNNs (Fragkiadaki et al. 2015; Ghosh et al. 2017) and CNNs (Butepage et al. 2017; Li et al. 2018). At present, human motion prediction is dominated almost exclusively by Graph Convolutional Networks (GCNs) (Kipf and Welling 2016), due to the innate graph-like nature of human body joints and bones. Human motion data are often in the form of 3D skeleton sequences, where each dimension corresponds to a different aspect—time, space, or channel, as shown in Fig. 1. To extract features, various graph convolutions are proposed. They either focus on aggregating information in a single dimension (Li et al. 2020; Cui, Sun, and Yang 2020), or combine aggregations in two dimensions without considering the remaining one (Wang et al. 2024; Ma et al. 2022; Li et al. 2021; Mao et al. 2019; Sofianos et al. 2021; Liu et al. 2020). There are yet a number of other graph convolution types unexplored by existing works. One is naturally motivated to ask: *What makes an ideal type of graph convolution for human motion prediction*?

To answer this question, we propose **Universal Graph Convolution (UniGC)**, a novel graph convolution concept that pushes the boundaries of existing knowledge by re-conceptualizing different graph convolutions, both existing and unexplored, as its special cases. In the most general case, UniGC uses 6D global adjacency to encode inter-relationships within and across space, time, and channels, which can be categorized into 7 types as shown in Fig. 1. The entry at position $[i, k, m, j, \ell, n]$ of the 6D adjacency represents the relationship between channel-m of joint-k at time-i and channel-n of joint-ℓ at time-j. Using two techniques—adjacency masking and sub-adjacency tying, UniGC can be specialized into various different special cases, which focus on different subsets of all the 7 types of relationships. Existing graph convolutions are either the vanilla form or variations of these special cases, such as spatial (Cui, Sun, and Yang 2020; Li et al. 2020; Ma et al. 2022), temporal (Ma et al. 2022), spatial-temporal (Li et al. 2021), dense (Ma et al. 2022), trajectory (Mao et al. 2019; Mao, Liu, and Salzmann 2020; Dang et al. 2021), space-time-separable (STS) (Sofianos et al. 2021), and unified (G3D) (Liu et al. 2020) graph convolutions. This gives rise to the second question: *What is the best way for UniGC to be leveraged network-wise to address the task*?

An intuitive answer based on common practices would be to stack multiple layers of UniGC. However, its 6D global adjacency would require too many parameters, making computation less efficient and optimization more challenging. This paper gives a better answer by proposing **GCNext**, which exploits the versatility and adaptability of UniGC on network-level. In contrast to existing approaches which typically involve manually choosing one from all possible types of graph convolutions, stacking it multiple layers, and evaluating the resulting network, GCNext is structured to make the choice-making process learnable and sample-specific, where each layer dynamically chooses the optimal graph convolution type based on each different sample. Specifically, each layer in GCNext contains several graph convolutional blocks of different types in parallel and a light-weight selector block tasked with choosing the best-fitting one for each sample. Other network designs include a rein-

*Corresponding author: nkliuyifang@gmail.com



(a) A motion sample (b) Its simplified 3D sketch (c) UniGC (d) Connection between our UniGC and existing works

Figure 1: Our proposed Universal Graph Convolution (UniGC) explores all aspects of human motion: space, time, and channel. (a) A motion sample. (b) A simplified 3D sketch of the sample. (c) Our proposed UniGC. (d) Graph convolutions for skeleton data in existing works such as BIG (Ma et al. 2022), STS (Sofianos et al. 2021), LDR (Cui, Sun, and Yang 2020), DMGNN (Li et al. 2020), MST (Li et al. 2021), G3D (Liu et al. 2020), LTD (Mao et al. 2019), MSR (Dang et al. 2021), DDGCN (Wang et al. 2024) focus on only a subset of all human motion aspects, and can be seen as variants of UniGC's special cases.

vention of update operations, and a replacement of batch normalization with layer normalization, both of which lead to performance improvement and efficiency boost. Combining these designs, GCNext can yield up to 9× reduction in computational cost compared to existing GCNs with comparable model sizes. GCNext offers multiple use cases. First, it can be trained from scratch to obtain a GCN combining the strengths of different graph convolutions both layer-wise and sample-wise. Second, it can be used to refine a preexisting GCN with a predetermined type of graph convolution.

Our proposed UniGC and GCNext combined represent a module-to-network paradigm package, providing an effective approach that better addresses human motion prediction. In summary, our contributions are three-fold:

- On concept-level, we propose UniGC, a novel graph convolution concept that re-conceptualizes different graph convolutions as its special cases. The specialization is achieved through two techniques: adjacency masking and sub-adjacency tying. UniGC represents not only a new concept of graph convolution, but also a new paradigm of designing graph convolutions.
- On network-level, we propose GCNext, a novel GCN framework that dynamically determines the optimal GCN architecture both sample-wise and layer-wise. GCNext offers multiple use cases, including training a new GCN from scratch or refining a preexisting GCN.
- We conduct extensive experiments on three benchmark datasets, Human3.6M, AMASS, and 3DPW, which show that GCNext can yield up to 9× lower computational cost than existing GCNs with comparable model sizes, besides achieving state-of-the-art performance.

Related Work

GCNs For Human Motion Data. Graph Convolutional Networks (GCNs) (Kipf and Welling 2016), a popular type of graph neural network (Scarselli et al. 2008; Hamilton, Ying, and Leskovec 2017; Zaheer et al. 2017), have wide-ranging applications in human motion prediction, action recognition (Yan, Xiong, and Lin 2018; Chen et al. 2021; Shi et al. 2021; Liu et al. 2023; Liu, Meng, and Liang 2022) and so on. Human motion data typically have three dimensions: space, time, and channel. Spatial graph convolutions are employed to aggregate information in the spatial domain (Cui, Sun, and Yang 2020; Li et al. 2020). Temporal graph convolutions aggregate temporal information across different frames (Ma et al. 2022; Li et al. 2021; Sofianos et al. 2021). Spatial-channel graph convolutions treat each trajectory as a node (Mao et al. 2019). Spatial-temporal graph convolutions capture both spatial and temporal relationships (Yan, Xiong, and Lin 2018; Liu et al. 2020; Zhong et al. 2022; Li et al. 2021). A related work to ours is G3D (Liu et al. 2020), which proposes a unified spatial-temporal graph convolution to aggregate cross-space-time features. However, the cross-channel relationships, known to be important for achieving superior performance (Mao et al. 2019; Mao, Liu, and Salzmann 2020; Dang et al. 2021; Guo et al. 2023; Mao et al. 2021), is not addressed in G3D. G3D can be seen as a variant of a special case in our UniGC. Therefore, our UniGC is a more inclusive and unified concept, taking into account richer information with and across different spatial, temporal, and channel aspects of human motion.

GCN-based Human Motion Prediction. LTD (Mao et al. 2019) considers each joint coordinate as a node in the graph. Follow-up works incorporate techniques that better address the task, such as attention (Mao, Liu, and Salzmann 2020; Mao et al. 2021), and multi-scale graph (Dang et al. 2021; Li et al. 2020). LDR (Cui, Sun, and Yang 2020) employ spatial graph convolutions to exploit skeleton structure, with temporal CNNs to learn temporal information. Later works combine spatial and temporal graph convolutions to learn spatial-temporal dependencies (Li et al. 2021; Ma et al. 2022; Sofianos et al. 2021). DDGCN (Wang et al. 2024) proposes a dynamic dense graph convolution that unifies spatial and temporal domains and dynamically learns sample-specific dependencies. In contrast to current approaches focusing on a subset of human motion aspects, our UniGC can provide both global modeling and more specific focus on local patterns among space, time, and channels, whose strengths can be combined by our GCNext.

Operations		
Notation	Variables	Description
$\mathbf{Y} = \mathcal{R}_{AB,CD}(\mathbf{X})$	$\mathbf{X} : (A, B, C, D)$ $\mathbf{Y} : (AB, CD)$	array reshape
$\mathbf{Y} = G(\mathbf{X}; \mathbf{A})$	$\mathbf{X}$, $\mathbf{Y}$: features $\mathbf{A}$: adjacency	graph convolution depending on $\mathbf{A}$
$\mathbf{X} \odot \mathbf{Y}$	$\mathbf{X}$, $\mathbf{Y}$: arrays of equal shape	element-wise multiply
Symbols		
Notation	Shape	Description
$\mathbb{X}$	(T, J, C)	input sample
$\mathbf{X}_t = \mathbb{X}[t, :, :]$	(J, C)	pose features
$\mathbf{x} = \mathcal{R}_{TJC}(\mathbb{X})$	$(TJC, 1)$	flattened $\mathbb{X}$
$\mathbf{x}_t = \mathcal{R}_{JC}(\mathbf{X}_t)$	$(JC, 1)$	flattened $\mathbf{X}_t$
$\mathbb{A}$	(T, J, C, T, J, C)	global adjacency
$\mathbb{M}$	(T, J, C, T, J, C)	adjacency mask

Table 1: Notations of operations and symbols.

Universal Graph Convolution (UniGC)

Our goal is to predict the future motion sequence given the history sequence. Suppose that a motion sequence consists of T frames, J joints and C coordinates (channels). Each sample is represented by a 3D array $\mathbb{X}$ of shape (T, J, C). For better readability, we list the notations of operations and symbols used in the paper in Table 1.

The Most General Case. We first present the most general case of UniGC. Given a sample $\mathbb{X}$, we represent it as a global graph with TJC nodes whose relationships are stored in global adjacency $\mathbb{A}$ of shape (T, J, C, T, J, C). The most general case of UniGC is defined as:

$$\mathbb{Y} = G(\mathbb{X}; \mathbb{A})$$
$$\begin{cases} \mathbf{x} = \mathcal{R}_{TJC}(\mathbb{X}); \\ \mathbf{A} = \mathcal{R}_{TJC,TJC}(\mathbb{A}); \\ \mathbf{y} \leftarrow \mathbf{A}\mathbf{x}; \\ \mathbb{Y} = \mathcal{R}_{T,J,C}(\mathbf{y}), \end{cases} \tag{1}$$

where the UniGC operation takes $\mathbb{X}$ as input and outputs $\mathbb{Y}$. Without any specialization, global adjacency $\mathbb{A}$ is fully parameterized, defining a global aggregation of node features over a fully-connected graph where every node aggregates information from all the nodes including itself.

Consistently with the space, time, and channel dimensions and their combinations, UniGC can be specialized into $C_3^1 + C_3^2 = 6$ graph convolution types: spatial-temporal (G^{st}), spatial-channel (G^{sc}), temporal-channel (G^{tc}), spatial (G^{s}), temporal (G^{t}), and channel (G^{c}) graph convolutions.

Special Case 1: Spatial-Channel Graph Convolution. Based on the most general case, an spatial-channel adjacency mask $\mathbb{M}^{\text{sc}}$ is introduced as an extra input to UniGC:

$$\mathbb{M}^{\text{sc}}[t_1, :, :, t_2, :, :] = \begin{cases} \mathbf{1} & \text{if } t_1 = t_2; \\ \mathbf{0} & \text{if } t_1 \neq t_2. \end{cases} \tag{2}$$

The mask $\mathbb{M}^{\text{sc}}$ is applied upon the global adjacency $\mathbb{A}$, giving the definition of *spatial-channel graph convolution*:

$$\mathbb{Y} = G^{\text{sc}}(\mathbb{X}; \mathbb{A}, \mathbb{M}^{\text{sc}})$$
$$\begin{cases} \mathbf{x} = \mathcal{R}_{TJC}(\mathbb{X}); \\ \mathbf{A} = \mathcal{R}_{TJC,TJC}(\mathbb{A}); \\ \mathbf{M}^{\text{sc}} = \mathcal{R}_{TJC,TJC}(\mathbb{M}^{\text{sc}}); \\ \mathbf{y} \leftarrow (\mathbf{A} \odot \mathbf{M}^{\text{sc}})\mathbf{x}; \\ \mathbb{Y} = \mathcal{R}_{T,J,C}(\mathbf{y}). \end{cases} \tag{3}$$

Component-wise, the effect of $\mathbf{M}^{\text{sc}}$ is to keep only the main-diagonal blocks of shape (JC, JC) in $\mathbf{A}$ such that:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_T \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_{TT} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \tag{4}$$

where $\mathbf{A}_{tt} = \mathcal{R}_{JC,JC}(\mathbb{A}[t, :, :, t, :, :])$. The mask $\mathbb{M}^{\text{sc}}$ nullifies cross-frame relationships, restricting the network to aggregating features only within each frame independently of other frames. For any time step t, we have:

$$\mathbf{y}_t \leftarrow \mathbf{A}_{tt}\mathbf{x}_t. \tag{5}$$

Such frame-wise independence allows one to apply graph convolution frame-by-frame, and then concatenate the outputs of each frame together for the final output.

The above graph convolution can be more constrained, by tying the sub-adjacencies of different frames:

$$\mathbf{A}_{\text{share}} = \mathbf{A}_{11} = \mathbf{A}_{22} = \cdots = \mathbf{A}_{TT}. \tag{6}$$

Using this cross-frame sub-adjacency tying, we can derive *temporally-tied spatial-channel graph convolution*:

$$\mathbb{Y} = G^{\text{sc}*}(\mathbb{X}; \mathbb{A})$$
$$\begin{cases} \mathbf{X} = \mathcal{R}_{JC,T}(\mathbb{X}); \\ \mathbf{A}_{\text{share}} = \mathcal{R}_{JC,JC}(\mathbb{A}[1, :, :, 1, :, :]); \\ \mathbf{Y} \leftarrow \mathbf{A}_{\text{share}}\mathbf{X}; \\ \mathbb{Y} = \mathcal{R}_{T,J,C}(\mathbf{Y}). \end{cases} \tag{7}$$

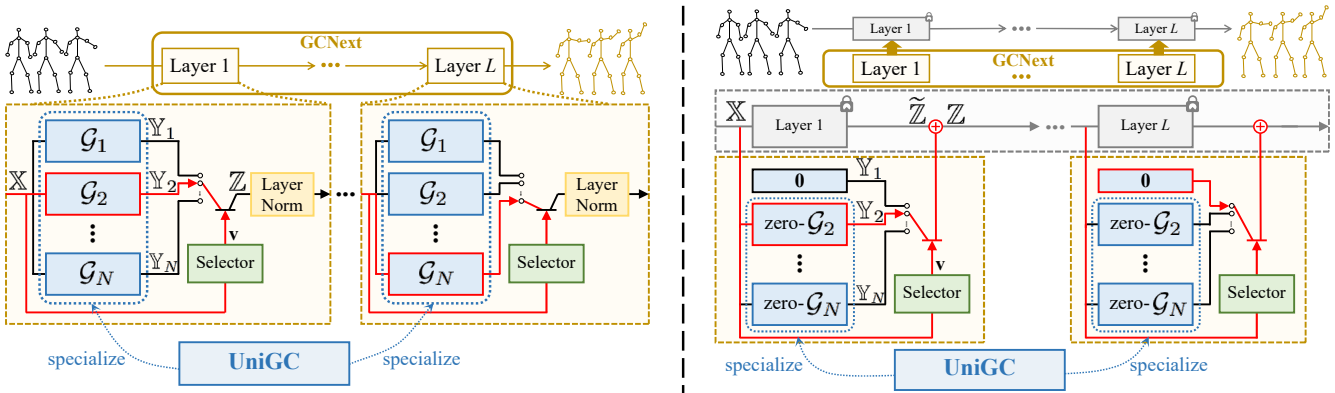
Similar to Eq. 4 and Eq. 5, the matrix multiplication step in Eq. 7 has component-wise representation as:

$$\begin{gathered} [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_T] \leftarrow \mathbf{A}_{\text{share}} [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T]; \\ \mathbf{y}_t \leftarrow \mathbf{A}_{\text{share}}\mathbf{x}_t. \end{gathered} \tag{8}$$

The *spatial-channel adjacency masking* and the *cross-frame sub-adjacency tying* techniques combined allow the module to capture frame-specific cross-spatial-channel information while benefiting from shared insights from similar patterns in different frames. Also, the size of adjacency parameters is reduced from $T \times J \times C \times T \times J \times C$ (as in $\mathbb{A}$) to a minimum of $JC \times JC$ (as in $\mathbf{A}_{\text{share}}$).

Special Case 2: Spatial-Temporal Graph Convolution. Similar to special case 1, we introduce spatial-temporal adjacency mask $\mathbb{M}^{\text{st}}$, which is defined as:

$$\mathbb{M}^{\text{st}}[:, :, c_1, :, :, c_2] = \begin{cases} \mathbf{1} & \text{if } c_1 = c_2; \\ \mathbf{0} & \text{if } c_1 \neq c_2. \end{cases} \tag{9}$$



(a) GCNext. Use case 1: training from scratch

(b) GCNext. Use case 2: refining a preexisting GCN

Figure 2: GCNext and its two use cases. (a) GCNext trained from scratch to dynamically determine the best-fitting GCN architecture at each layer for each sample. (b) GCNext as a plug-in to refine a preexisting GCN. By leveraging a novel "zero-graph convolution" scheme, GCNext preserves the qualities of the original GCN, and gradually refines it during training.

The *spatial-temporal graph convolution* G^{st} can be obtained by simply replacing the mask term $\mathbb{M}^{\text{sc}}$ in Eq. 3 with $\mathbb{M}^{\text{st}}$. Also similar to special case 1, it can be further constrained by cross-channel sub-adjacency tying, becoming *channel-tied spatial-temporal graph convolution*:

$$\begin{aligned} &\mathbb{Y} = G^{\text{st}*}(\mathbb{X}; \mathbb{A}) \\ &\begin{cases} \mathbf{X} = \mathcal{R}_{TJ,C}(\mathbb{X}); \\ \mathbf{A}_{\text{share}} = \mathcal{R}_{TJ,TJ}(\mathbb{A}[:,:,1,:,:,1]); \\ \mathbf{Y} \leftarrow \mathbf{A}_{\text{share}}\mathbf{X}; \\ \mathbb{Y} = \mathcal{R}_{T,J,C}(\mathbf{Y}). \end{cases} \end{aligned} \tag{10}$$

Special Case 3: Spatial Graph Convolution. Spatial graph convolution $G^{\text{s}}(\mathbb{X}; \mathbb{A})$ uses the spatial adjacency mask $\mathbb{M}^{\text{s}} = \mathbb{M}^{\text{sc}} \odot \mathbb{M}^{\text{st}}$, where only the spatial relationships within the same frame and the same channel are aggregated.

The rest of the special cases, including temporal-channel (G^{tc}), temporal (G^{t}), and channel (G^{c}) graph convolutions, can all be derived in ways similar to special case 1–3.

Connections with Existing Graph Convolutions. Graph convolutions proposed by existing works largely are the vanilla form or are variants of UniGC's special cases. For example, the *temporally-tied spatial-channel graph convolution* (Eq. 7) is used in LTD (Mao et al. 2019), while its variants are used in MSR-GCN (Dang et al. 2021) and HisRep (Mao, Liu, and Salzmann 2020). The well-accepted concept of spatial-temporal graph convolutions in the community (Ma et al. 2022; Li et al. 2021; Sofianos et al. 2021; Liu et al. 2020) is actually the variation of the *channel-tied spatial-temporal graph convolution* (Eq. 10), while the channel-untied case is unexplored by existing work.

Further Discussion. It is worth noting that, while the formulas for UniGC appear complex in matrix form, they can be easily implemented using tensor operations, with the help of Einstein notation.

GCNext Framework

The most general case of UniGC requires a large size of parameters, making computation less efficient and optimization more challenging. To balance effectiveness and efficiency, we propose GCNext, which leverages the adaptability of UniGC, and offers multiple use cases.

Use Case 1: Training from Scratch. As shown in Fig. 2 (a), GCNext contains L dynamic layers, where each layer has N candidate graph convolution operations, a selector, and a layer normalization. The selector is tasked with determining the optimal graph convolution operation for this layer based on the layer input $\mathbb{X}$. Each of the N candidates perform a different type of graph convolution:

$$\mathbb{Y}_i = G_i(\mathbb{X}; \mathbb{A}, \mathbb{M}_i, \theta_i), \forall i \in [1, N], \tag{11}$$

where $G_i \in \{G^{\text{st}}, G^{\text{sc}}, G^{\text{tc}}, G^{\text{s}}, G^{\text{t}}, G^{\text{c}}\}$, θ_i is the parameters that parameterize $\mathbb{A}$, and $\mathbb{M}_i$ is the adjacency mask required for specializing to G_i. The layer input $\mathbb{X}$ is also passed through the selector, which relies on a learnable process S followed by Gumbel Softmax (Jang, Gu, and Poole 2016):

$$\mathbf{v} = \mathit{GumbelSoftmax}\big(\mathcal{S}(\mathbb{X}; \theta_{\text{s}})\big), \tag{12}$$

where $\mathbf{v} = [v_1, v_2, \cdots, v_N] \in \{0, 1\}^N$ is a one-hot vector that indicates the optimal block, and $\mathcal{S} : \mathbb{R}^{T \times J \times C} \to \mathbb{R}^N$ is implemented with an average pooling followed by an MLP, which learns to assign higher values to the best-matched candidate. The graph convolution output $\mathbb{Z}$ is obtained by $\mathbb{Z} = \sum_{i=1}^{N} v_i \mathbb{Y}_i$, followed by layer normalization to produce the final output of this layer. During inference, the network directly passes the input through the optimal graph convolution block based on the index of the one-hot vector, thereby reducing computational cost.

Use Case 2: Refining Preexisting GCNs. Alternatively, GCNext can be used to refine a preexisting GCN. Suppose that the preexisting GCN is built on graph convolution G_τ. Fig. 2 (b) shows the case of $\tau = 1$. At each layer, the network decides whether or not it needs to be refined, and if

Model	Venue	Mean Per Joint Position Error (mm)							
		80ms	160ms	320ms	400ms	560ms	720ms	880ms	1000ms
LTD (Mao et al. 2019)	ICCV'19	11.2	23.4	47.9	58.9	78.3	93.3	106.0	114.0
HisRep (Mao, Liu, and Salzmann 2020)	ECCV'20	10.4	22.6	47.1	58.3	77.3	91.8	104.1	112.1
MSR-GCN (Dang et al. 2021)	ICCV'21	11.3	24.3	50.8	61.9	80.0	-	-	112.9
PGBIG (Ma et al. 2022)	CVPR'22	10.6	23.1	47.1	57.9	76.3	90.7	102.4	109.7
siMLPe (Guo et al. 2023)	WACV'23	9.6	21.7	46.3	57.3	75.7	90.1	101.8	109.4
Ours		**9.3**	**21.5**	**45.5**	**56.4**	**74.7**	**88.9**	**100.8**	**108.7**
STSGCN (Sofianos et al. 2021)†	ICCV'21	10.1	17.1	33.1	38.3	50.8	60.1	68.9	75.6
GAGCN (Zhong et al. 2022)†	CVPR'22	10.1	16.9	32.5	38.5	50.0	-	-	72.9
MotionMixer (Bouazizi et al. 2022)†	IJCAI'22	9.0	13.2	26.9	33.6	46.1	56.5	65.7	71.6
Ours†		**6.9**	**12.7**	**24.7**	**30.5**	**41.3**	**50.7**	**59.0**	**64.7**

Table 2: Results on Human3.6M in terms of mean per joint position error (MPJPE). † indicates methods that compute the average error over all frames, whose results (except ours) are taken from the paper by Bouazizi et al. (2022). Otherwise, the methods are evaluated at each frame, whose results (except ours) are taken from the paper by Guo et al. (2023).

Dataset	AMASS-BMLrub								3DPW							
milliseconds	80	160	320	400	560	720	880	1000	80	160	320	400	560	720	880	1000
LTD	11.0	20.7	37.8	45.3	57.2	65.7	71.3	75.2	12.6	23.2	39.7	46.6	57.9	65.8	71.5	75.5
HisRep	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
siMLPe	10.8	19.6	34.3	40.5	50.5	57.3	62.4	65.7	12.1	22.1	38.1	44.5	54.9	62.4	68.2	72.2
Ours	**10.2**	**19.3**	**34.1**	**40.3**	50.6	**57.3**	**62.0**	**65.3**	**11.8**	**22.0**	**37.9**	**44.2**	55.1	**62.1**	**67.8**	**72.0**

Table 3: Results on AMASS and 3DPW in terms of mean per joint position error (MPJPE).

so, which graph convolution operation to use to refine it. Let $\widetilde{\mathbb{Z}} = G_\tau(\mathbb{X}; \mathbb{A}, \mathbb{M}_\tau)$ denote the locked original output of this layer in the preexisting GCN. We re-define Eq. 11 as:

$$\mathbb{Y}_i = \begin{cases} \mathbf{0}, & \text{if } i = \tau; \\ G_i(\mathbb{X}; \mathbb{A}, \mathbb{M}_i, \theta_i), & \text{if } i \neq \tau. \end{cases} \quad \forall i \in [1, N] \tag{13}$$

The graph convolution output $\mathbb{Z}$ is obtained as:

$$\mathbb{Z} = \widetilde{\mathbb{Z}} + \sum_{i=1}^{N} v_i \mathbb{Y}_i. \tag{14}$$

During training, the adjacencies $\mathbb{A}$ are initialized at zeros. Therefore, in the first train step, we have:

$$\begin{cases} \theta_i = \mathbf{0}, \forall i \in [1, N], i \neq \tau; \\ G_i(\mathbb{X}; \mathbb{A}, \mathbb{M}_i, \theta_i) = \mathbf{0}, \forall i \in [1, N], i \neq \tau; \\ \mathbb{Z} = \widetilde{\mathbb{Z}}. \end{cases} \tag{15}$$

It indicates that, before optimization, the network is the same as the preexisting GCN as if GCNext did not exist. The qualities of that GCN are perfectly preserved. Therefore, further optimization takes effect in a fine tuning manner.

Other Network Designs. We adopt a few other network designs to balance effectiveness and efficiency.

1) We reinvent the *update* operations in traditional GCNs. The update operation usually uses a weight matrix to transform input features into high-dimensional hidden features, which is the main source of computation cost in GCNs. At each layer of GCNext, we employ one *dimension-preserving update* operation following the selection step, which saves a lot of computation cost, compared to updating high-dimensional features in every graph convolution.

2) We replace *batch normalization*, commonly applied in traditional GCNs, with *layer normalization*, which requires fewer parameters and computes more efficiently.

Experiments

We report the Mean Per Joint Position Error (MPJPE), which is the preferred metric in human motion prediction. The lower the MPJPE, the better the performance.

Implementation Details

The model was trained on RTX 3080 Ti GPU, and the training consumed about 3GB memory and took about 3 hours with batch size of 256. Testing took about 13 seconds. For H3.6M dataset, the model is trained for 85k iterations. The learning rate starts with 0.0006, and drops to 0.000005 after 75k iterations. For AMASS dataset, the model is trained for 115k iterations. The learning rate starts with 0.0003, and drops to 0.000001 after 100k iterations.

Datasets

Human3.6M (H3.6M) (Ionescu et al. 2013) comprises 15 types of actions by 7 actors. Following siMLPe (Guo et al. 2023), the dataset is preprocessed and converted into 3D coordinates, and each pose contains 22 joints. For the validation and test sets, we respectively use subject 11 and subject 5. The remaining 5 subjects for training.

AMASS (Mahmood et al. 2019) combines multiple Mocap datasets unified by SMPL parameterization. Following siMLPe (Guo et al. 2023), we use AMASS-BMLrub as the test set and split the rest into training and validation sets.

3D Pose in the Wild (3DPW) (Von Marcard et al. 2018) includes activities captured from indoor and outdoor scenes. We evaluate 18 joints using the model trained on AMASS.

Comparison with State-of-the-Art Methods

Comparison on Human3.6M. Considering that the baseline methods follow two separate testing protocols: error computation at each time step (Mao, Liu, and Salzmann

Model	Walking				Eating				Smoking				Discussion			
	80	400	560	1000	80	400	560	1000	80	400	560	1000	80	400	560	1000
LTD	11.1	42.9	53.1	70.7	7.0	37.3	51.1	78.6	7.5	37.5	49.4	71.8	10.8	65.8	88.1	121.6
HisRep	10.0	39.8	47.4	58.1	6.4	36.2	50.0	75.7	7.0	36.4	47.6	69.5	10.2	65.4	86.6	119.8
MSR-GCN	10.8	42.4	53.3	63.7	6.9	36.0	50.8	75.4	7.5	37.5	50.5	72.1	10.4	65.0	87.0	116.8
PGBIG	11.2	42.8	49.6	58.9	6.5	36.8	50.0	74.9	7.3	37.5	48.8	69.9	10.2	64.4	86.1	116.9
siMLPe	9.9	39.6	46.8	55.7	5.9	36.1	49.6	74.5	6.5	36.3	47.2	69.3	9.4	64.3	85.7	116.3
Ours	**8.8**	**38.9**	**46.4**	**55.0**	**5.9**	**35.0**	**48.2**	**73.9**	**5.6**	**36.1**	**46.9**	**68.6**	**8.8**	**63.1**	**84.0**	**114.8**
STSGCN†	10.7	38.2	40.6	51.8	6.7	31.6	33.9	52.5	7.1	30.6	33.6	50.1	9.7	45.0	53.4	78.8
GAGCN†	10.3	32.4	39.9	51.1	6.4	25.2	31.8	51.4	7.1	24.3	31.1	48.7	9.7	38.9	53.1	76.9
MotionMixer†	7.3	28.6	-	49.2	4.3	20.9	-	47.4	4.7	21.4	-	45.4	6.4	35.5	-	78.0
Ours†	**6.8**	**24.3**	**30.3**	**40.3**	**4.3**	**18.9**	**25.9**	**42.6**	**4.5**	**19.8**	**26.5**	**41.2**	**6.2**	**32.9**	**45.4**	**70.8**

Model	Waiting				Walking Dog				Walking Together				Average			
	80	400	560	1000	80	400	560	1000	80	400	560	1000	80	400	560	1000
LTD	9.2	54.4	73.4	107.5	20.9	86.6	109.7	150.1	9.6	44.0	55.7	69.8	11.2	58.9	78.3	114.0
HisRep	8.7	54.9	74.5	108.2	20.1	86.3	108.2	146.9	8.9	41.9	52.7	64.9	10.4	58.3	77.3	112.1
MSR-GCN	10.4	62.4	74.8	105.5	24.9	112.9	107.7	145.7	9.2	43.2	56.2	69.5	11.3	61.9	80.0	112.9
PGBIG	8.7	53.6	71.6	103.7	20.4	84.6	105.7	145.9	8.9	43.8	54.4	64.6	10.6	57.9	76.3	109.7
siMLPe	7.8	53.2	71.6	104.6	18.2	83.6	105.6	141.2	8.4	41.2	50.8	61.5	9.6	57.3	75.7	109.4
Ours	**7.7**	**52.6**	**71.5**	**104.0**	18.4	**82.8**	**104.6**	142.8	**8.4**	**40.5**	**50.2**	**60.8**	**9.3**	**56.4**	**74.7**	**108.7**
STSGCN†	8.6	40.7	47.3	72.0	17.6	66.4	74.7	102.6	8.6	35.1	38.9	51.1	10.1	38.3	51.7	75.6
GAGCN†	8.5	33.8	45.9	69.3	17.0	59.4	70.1	91.3	-	-	-	-	10.1	38.5	50.0	72.9
MotionMixer†	5.4	30.0	-	68.2	13.4	54.1	-	99.6	5.9	27.4	-	50.4	9.0	33.6	-	71.6
Ours†	**5.4**	**27.4**	**38.0**	**61.3**	**13.3**	**49.0**	**62.6**	**89.8**	6.1	**23.4**	**30.1**	**41.6**	**6.9**	**30.5**	**41.3**	**64.7**

Table 4: Action-wise results on Human3.6M.

Model refined	80	560	1000
LTD	11.2	78.3	114.0
Ours+LTD	**9.9** (↓ 1.3)	**77.6** (↓ 0.7)	**110.5** (↓ 3.5)
MSR-GCN	11.3	80.0	112.9
Ours+MSR	**10.5** (↓ 0.8)	**78.1** (↓ 1.9)	**110.8** (↓ 2.1)
PGBIG	10.6	76.3	109.7
Ours+PGBIG	**9.7** (↓ 0.9)	**75.7** (↓ 0.6)	**108.8** (↓ 0.9)

Table 5: Results on use case 2: refining preexisting GCNs.

Model	FLOPs (M)		Infer. time (ms)	Mem.
	Train	Infer.		
MSR-GCN	1449.6	1449.6	3.9	5.1GB
PGBIG	224.8	224.8	2.1	2.5GB
HisRep	148.7	148.7	1.6	2.3GB
LTD	133.3	133.3	1.2	2.2GB
Ours	29.1	14.5	0.8	3.1GB

Table 6: Comparison of efficiency in terms of train and inference FLOPs, inference time, and memory.

2020) and error averaging (Sofianos et al. 2021), we test our method on both protocols. Table 2 shows the results on H3.6M using average Mean Per Joint Position Errors (MPJPEs) at different time steps for all actions. Our method surpasses all others under both testing protocols. Table 4 shows action-wise results for 7 exemplary actions in H3.6M. Table 5 reports the results of existing GCNs refined with GCNext, showing that refining with GCNext brings performance boost. Table 6 compares the efficiency of different GCNs, showing that GCNext is generally more efficient.

Comparison on AMASS and 3DPW. In Table 3, we present the results obtained from AMASS and 3DPW datasets. As there are two distinct evaluation protocols (Mao, Liu, and Salzmann 2020; Sofianos et al. 2021), we evaluate our method under both. The models are trained on AMASS and tested on AMASS-BMLrub and 3DPW.

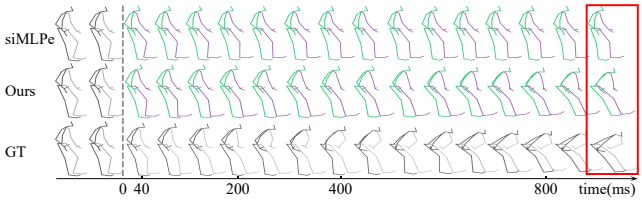


Figure 3: Qualitative results on H3.6M.

Ablation Studies

We conduct ablation studies to verify two factors: 1) the necessity of a dynamic network over static ones; 2) the best architecture design within the dynamic network. Factor 1 is studied in Table 7, and factor 2 in Figure 4, Table 8 & 9.

Necessity of Dynamic Network. Table 7 verifies the advantages of our dynamic network compared against static networks. Specifically, we design two types of static networks for ablation. The first type, similar to existing GCNs, is built using one single type of graph convolution, which can be one of the special cases or the most general case of UniGC, i.e., G^{st}, G^{sc}, G^{tc}, G^{s}, G^{t}, G^{c}, or G. The second type adopts the original GCNext architecture, but instead of letting the selector select one at each layer, we simply use all graph convolutions by averaging or summing their outputs. Table 7 shows that our GCNext (default) outperforms all static architectures, no matter how they are designed, verifying the necessity of a dynamic architecture for the task.

Ablation on Graph Convolution Options. At each layer, the selector is expected to select the best candidate graph

Architecture	80	1000	Architecture	80	1000
st	9.8	109.9	s	10.1	110.7
sc	9.6	109.9	t	10.1	111.0
tc	11.0	113.6	c	10.2	111.6
general	14.5	116.0	all average	9.7	109.6
default	**9.3**	**108.7**	all sum	10.1	110.9

Table 7: Ablation on dynamic architecture of GCNext. The dynamic architecture of GCNext (default) is compared against different static variations. st, sc, tc, s, t, and c represents a single-typed GCN. "general" means UniGC without any specialization. "all average" and "all sum" represents using multiple graph convolutions by averaging or summing their outputs, instead of selecting one.

GC option						80	400	560	1000
st	sc	tc	s	t	c				
✓	✓					9.8	57.1	75.2	109.5
✓			✓			9.9	58.0	75.8	110.1
	✓		✓			9.8	57.3	75.5	109.8
✓	✓		✓			9.8	57.2	75.3	109.7
✓		✓	✓		✓	9.7	57.0	75.4	109.8
✓	✓		✓		✓	9.4	56.7	75.2	109.5
✓	✓		✓	✓		9.5	56.9	75.2	109.1
✓	✓		✓		✓	**9.3**	**56.4**	**74.7**	**108.7**

Table 8: Ablation on graph convolution options. Each line represents a GCNext variant, whose option set is formed from all the graph convolutions marked with ✓. The best architecture is using {st,sc,s,c} as the option set.

convolution for each sample. While with more options comes higher representational capacity and more flexibility, training and optimization become more challenging. Therefore, we determine the optimal set of options through experiments. Specifically, we use different combinations of graph convolutions as different sets of options. Table 8 shows that the best architecture is providing the selector with 4 options: spatial-temporal (G^{st}), spatial-channel (G^{sc}), spatial (G^{s}), and channel (G^{c}) graph convolutions.

Ablation on Number of Layers. To determine the optimal model size, we study the influence of model size by decreasing or increasing the number of layers. Table 9 shows that GCNext with 48 layers yields the best performance.

Ablation on Dynamic Policy. We assess the impact of dynamic policy on GCNext in Fig. 4. Specifically, we compare 6 different policies, including unbiased (default), random, and biased towards different graph convolution types. GC 1–4 represent G^{st}, G^{sc}, G^{s}, and G^{c} respectively, which are used in default GCNext architecture. The biased policy is obtained by piling more blocks such that the selector will make biased decision towards that type. The red, yellow, green, and blue bars respectively represent the proportions of GC 1–4 being selected. Fig. 4 shows that GCNext achieves the best performance with the unbiased policy.

# Layers	80	560	1000	#	80	560	1000
12	9.8	76.6	110.7	64	9.5	75.0	109.5
24	9.6	75.3	109.8	80	9.5	75.3	109.6
48	**9.3**	**74.7**	**108.7**	96	10.1	75.9	110.2

Table 9: Ablation on the number of layers.

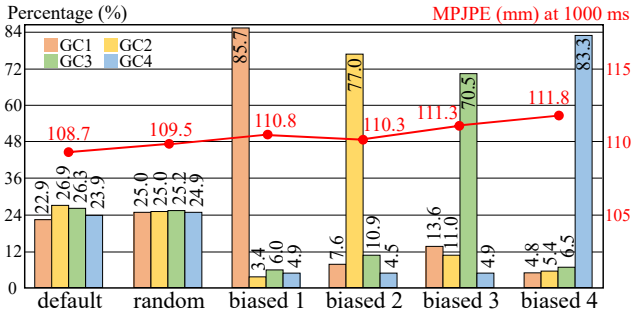


Figure 4: Ablation on dynamic policy. Six different policies are evaluated, including unbiased, random, and biased towards GC 1–4. Their corresponding MPJPEs at 1000ms (red line) are also reported. The unbiased (default) policy achieves the best performance.

Visualization

To put a finer point in the evaluation of our method, we present qualitative results, by visualizing the predicted motion sequence. Fig. 3 provides a sample of action "sitting" on H3.6M. We compare the results of ours and of the most recent state-of-the-art, siMLPe (Guo et al. 2023), along with the corresponding ground truth (GT). Our method generates more realistic and accurate body movements, as can be seen from the frame highlighted with red boxes.

Conclusion

This paper advances towards achieving unity of graph convolutions applied to 3D skeleton sequence data for human motion prediction. First, we propose Universal Graph Convolution (UniGC), which re-conceptualizing different graph convolutions, both existing and unexplored, as its special cases. This innovation brings significant potential for discovering novel graph convolutions and evaluating diverse graph convolutions fairly. Moreover, we propose GCNext framework, which dynamically builds the optimal GCN architecture for each sample. GCNext offers multiple use cases, such as to be trained from scratch into a new GCN that better addresses the specific task, or to be used to refine a preexisting GCN. We believe GCNext is a universal operator that can be applied to more than just human motion data. Exploiting its further application in more graph-structured spatial-temporal data types could be promising.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 62203476), Natural Science Foundation of Shenzhen (No. JCYJ20230807120801002).

References

Bouazizi, A.; Holzbock, A.; Kressel, U.; Dietmayer, K.; and Belagiannis, V. 2022. Motionmixer: mlp-based 3d human body pose forecasting. *arXiv preprint arXiv:2207.00499*.

Butepage, J.; Black, M. J.; Kragic, D.; and Kjellstrom, H. 2017. Deep representation learning for human motion prediction and classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6158–6166.

Chen, Y.; Zhang, Z.; Yuan, C.; Li, B.; Deng, Y.; and Hu, W. 2021. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 13359–13368.

Cui, Q.; Sun, H.; and Yang, F. 2020. Learning dynamic relationships for 3d human motion prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6519–6527.

Dang, L.; Nie, Y.; Long, C.; Zhang, Q.; and Li, G. 2021. MSR-GCN: Multi-scale residual graph convolution networks for human motion prediction. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 11467–11476.

Fragkiadaki, K.; Levine, S.; Felsen, P.; and Malik, J. 2015. Recurrent network models for human dynamics. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 4346–4354.

Ghosh, P.; Song, J.; Aksan, E.; and Hilliges, O. 2017. Learning human motion models for long-term predictions. In *International Conference on 3D Vision (3DV)*, 458–466.

Guo, W.; Du, Y.; Shen, X.; Lepetit, V.; Alameda-Pineda, X.; and Moreno-Noguer, F. 2023. Back to mlp: A simple baseline for human motion prediction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 4809–4819.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

Ionescu, C.; Papava, D.; Olaru, V.; and Sminchisescu, C. 2013. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(7): 1325–1339.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *ArXiv:1609.02907*.

Li, C.; Zhang, Z.; Lee, W. S.; and Lee, G. H. 2018. Convolutional sequence to sequence model for human dynamics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5226–5234.

Li, M.; Chen, S.; Zhao, Y.; Zhang, Y.; Wang, Y.; and Tian, Q. 2020. Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 214–223.

Li, M.; Chen, S.; Zhao, Y.; Zhang, Y.; Wang, Y.; and Tian, Q. 2021. Multiscale spatio-temporal graph neural networks for 3d skeleton-based motion prediction. *IEEE Transactions on Image Processing (TIP)*, 30: 7760–7775.

Liu, J.; Wang, X.; Wang, C.; Gao, Y.; and Liu, M. 2023. Temporal Decoupling Graph Convolutional Network for Skeleton-based Gesture Recognition. *IEEE Transactions on Multimedia*.

Liu, M.; Meng, F.; and Liang, Y. 2022. Generalized Pose Decoupled Network for Unsupervised 3d Skeleton Sequence-based Action Representation Learning. *Cyborg and Bionic Systems*.

Liu, Z.; Zhang, H.; Chen, Z.; Wang, Z.; and Ouyang, W. 2020. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 143–152.

Ma, T.; Nie, Y.; Long, C.; Zhang, Q.; and Li, G. 2022. Progressively generating better initial guesses towards next stages for high-quality human motion prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6437–6446.

Mahmood, N.; Ghorbani, N.; Troje, N. F.; Pons-Moll, G.; and Black, M. J. 2019. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, 5442–5451.

Mao, W.; Liu, M.; and Salzmann, M. 2020. History repeats itself: Human motion prediction via motion attention. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, 474–489. Springer.

Mao, W.; Liu, M.; Salzmann, M.; and Li, H. 2019. Learning trajectory dependencies for human motion prediction. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 9489–9497.

Mao, W.; Liu, M.; Salzmann, M.; and Li, H. 2021. Multi-level motion attention for human motion prediction. *International journal of computer vision*, 129(9): 2513–2535.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE Transactions on Neural Networks (ITNN)*, 20(1): 61–80.

Shi, L.; Zhang, Y.; Cheng, J.; and Lu, H. 2021. Adasgn: Adapting joint number and model size for efficient skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 13413–13422.

Sofianos, T.; Sampieri, A.; Franco, L.; and Galasso, F. 2021. Space-time-separable graph convolutional network for pose forecasting. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 11209–11218.

Von Marcard, T.; Henschel, R.; Black, M. J.; Rosenhahn, B.; and Pons-Moll, G. 2018. Recovering accurate 3d human pose in the wild using imus and a moving camera. In

Proceedings of the European conference on computer vision (ECCV), 601–617.

Wang, X.; Zhang, W.; Wang, C.; Gao, Y.; and Liu, M. 2024. Dynamic Dense Graph Convolutional Network for Skeleton-based Human Motion Prediction. *IEEE Transactions on Image Processing*.

Yan, S.; Xiong, Y.; and Lin, D. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; and Smola, A. J. 2017. Deep sets. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

Zhong, C.; Hu, L.; Zhang, Z.; Ye, Y.; and Xia, S. 2022. Spatio-temporal gating-adjacency GCN for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6447–6456.