Remembering Across Blocks: Topology-Conditioned Block-Progressive Memory for Skeleton-Based Action Recognition

Seonho Lee, SangHoon Han, Hyeok Nam, and Sung In Cho*

Department of Artificial Intelligence, Sogang University
35, Baekbeom-ro, Mapo-gu, Seoul 04107, Republic of Korea
{sunhozizi,leo4102,skagur10,csi2267}@sogang.ac.kr

Abstract. Skeleton-based action recognition classifies human actions from sequences of joint coordinates and is widely studied using graph convolutional networks (GCNs). GCNs typically rely on repeated graph node feature aggregation via one-hop message passing to capture long-range dependencies across entire joint sequences, while progressively refining representations from multiple perspectives through block-wise learned topology. However, this repeated aggregation can often degrade representations, making the node features in the final embedding increasingly homogenized and causing the final embedding to fail to capture local or fine-grained representations formed in early blocks. To address these issues, we propose a novel topology-conditioned block-progressive memory for skeleton GCNs. Along the GCN block axis, memory stores complementary block-wise representations and fuses the accumulated memory state into the final embedding, thereby mitigating representation degradation caused by repeated aggregation. At each block, a sample-specific memory state is updated by minimizing a topology-conditioned reconstruction loss, where the reconstruction error quantifies how well the previous memory explains the current block representation while capturing newly emerging, discriminative cues. We support two inference-time modes: memory-free, which uses memory only during training, and an optional memory-enabled mode, which also applies the same memory update rule at test time for modest gains. Both variants achieve state-of-the-art results on large-scale benchmarks, including NTU RGB+D, NTU RGB+D 120, and Kinetics-Skeleton.

Keywords: Skeleton-Based Action Recognition · Graph Convolutional Networks · Topology-Conditioned Block-Progressive Memory

1 Introduction

Skeleton-based human action recognition (HAR) is a recognition task that models spatiotemporal human motion and classifies action categories using sequences of human joint coordinates over time [29,37,39]. Compared to RGB videos [4,36],

* Corresponding author.

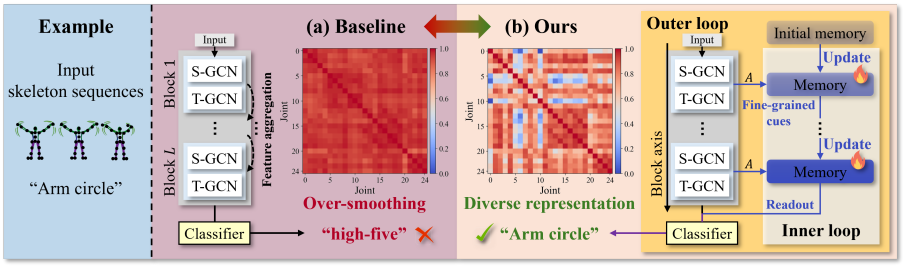

Fig. 1: Illustration of the overview and motivation of our topology-conditioned block-progressive memory. (a) Baseline: repeated aggregation across GCN blocks leads to over-smoothing of joint-node features. (b) Ours: our memory is updated block-by-block (inner loop) and injects fine-grained cues into the final embedding, preserving node-wise representational diversity. Here, A denotes the adjacency matrix.

skeleton data is relatively robust to changes in background, illumination, and clothing texture, and its compact representation makes it suitable for deployment in embedded systems, edge, and real-time environments. In addition, since it often does not include directly identifiable cues such as faces or appearance, it is frequently used in applications where privacy-friendly representations are required [13,25,47]. Owing to these properties, it has been widely adopted across various domains, including video understanding, video recognition, video surveillance, human computer interaction (HCI), VR/AR, and robotics. Skeleton-based HAR research has largely evolved around graph convolutional network (GCN)-based methods [8, 11, 12, 22, 24, 33, 46, 48, 53]. GCN-based methods model the human body as a graph by representing skeleton joints as nodes and the human kinematic topology as edges. The inductive bias provided by this graph structure encourages the model to learn features under the assumption of the topology, thereby enabling stable aggregation of local motions along edge-connected nodes.

ST-GCN [46], a representative early GCN-based method, constructs a skeleton sequence as a spatial-temporal graph, and the network is designed by sequentially stacking multiple ST-GCN blocks. As shown in Figure 1(a), each block serves as a basic unit that combines a spatial graph convolution network (S-GCN) and a temporal graph convolution network (T-GCN), aggregating spatial relational information and then integrating information along the temporal axis. By repeatedly applying these blocks, ST-GCN hierarchically updates (aggregates) node features. However, such message passing-based GCNs [8, 11, 12, 22, 24, 33, 46, 48, 53] inherently rely on one-hop neighborhood aggregation. Therefore, to capture dependencies between distant joints (weakly connected by edges) or dynamics over long temporal ranges, multiple blocks must be stacked deeply so that information propagates block-by-block (hop-by-hop). This structure (i) creates a propagation bottleneck that requires many layers to integrate long-range information, and (ii) remains strongly tied to a fixed skeleton prior (fixed adjacency), making it difficult to flexibly reflect joint relations that are important for each sample or action.

To alleviate these limitations, recent studies refine the adjacency matrix in a sample-wise manner by leveraging correlations in skeleton features [8, 22, 53] and are designed to dynamically learn different connection structures across blocks. By introducing such dynamic topology, shortcut-like connections can be formed even between distant joints within a given block, which facilitates modeling long-range inter-joint interactions that were limited in ST-GCN. Furthermore, as the network becomes deeper, performance has improved in the direction of forming more discriminative representations by progressively refining features through the accumulation of diverse block-wise representations.

However, these methods primarily focus on making the connectivity structure more flexible and do not replace the core operating principle of GCNs—repeated aggregation based on one-hop message passing. That is, dynamic-topology methods [8,22,33,53] selectively strengthen important connections for a given sample, but they still rely on a structure in which information is progressively propagated and integrated through multiple blocks. In this process, the following issues can occur. First, repeated neighborhood aggregation may cause over-smoothing (i.e., node representation homogenization) [3, 6, 21, 27, 35, 50]. Moreover, recent work reports the possibility of over-parameterization and structural redundancy [44], which may also be associated with reduced inter-block representational diversity. Second, spatially local cues and temporally local key dynamics (over short temporal intervals) captured in early blocks can be diluted through repeated feature mixing, which may as a result reduce the opportunity to learn diverse representations during training.

As shown in Figure 1, our method introduces a new GCN architecture equipped with a topology-conditioned block-progressive memory, which is updated block by block. This design mitigates node representation homogenization and the progressive attenuation of fine-grained spatiotemporal cues observed in early blocks during block progression in skeleton GCNs. The core idea of our method is to write (i.e., memorize) the distinct spatiotemporal cues provided by each block into a sample-specific memory and update it online as the blocks progress. In particular, when a "surprise" component that is not well explained emerges while predicting the current block feature based on the previous block representation, the memory adaptively accumulates this surprise and injects it into the final representation, enhancing the final embedding by preserving cues that tend to be diluted in deeper blocks. Specifically, as shown in Figure 1(b), in the outer loop, we optimize the parameters of the backbone, classifier, and memory update rule so that the memory learns **"what/how to memorize" (from a meta-learning perspective)**, while in the inner loop we update the memory state for each sample in a self-supervised manner, block by block, online along block progression. During training, the memory accumulated along block progression is fused with the final embedding right before the classifier to compensate for cues diluted by repeated aggregation and to support backbone representation shaping. As a result, the training benefits are internalized into the backbone, so that even without memory at inference time, our method retains improved performance over the baseline. We also provide an optional memory-enabled mode

that applies the same update rule at inference, serving as an auxiliary mechanism to refine sample-specific representations at test time. The main contributions are as follows:

- We propose a **topology-conditioned block-progressive memory module** that mitigates over-smoothing and preserves fine-grained spatiotemporal cues from early blocks.
- **Topology-queried surprise-driven inner loop**: During GCN block progression, we propose a topology-query inner-loop update that uses each block's adjacency as queries and spatiotemporal features as values, and writes to memory only when a "surprise" occurs—measured by the reconstruction loss of the current block representation from the previous memory—capturing newly emerging cues.
- **Training-time representation shaping**: During training, we use the memory module to enrich the final embedding with fine-grained spatiotemporal cues distilled from early-block embeddings that are prone to attenuation, guiding the backbone to learn diverse block-wise representations. As a result, even without memory at inference time, the model still delivers substantial improvements over the baseline. We also support an optional memory-enabled variant, where additional gains become more pronounced in settings with greater headroom for improvement.
- **Strong empirical performance on large-scale skeleton action benchmarks**: On large-scale benchmarks including NTU-RGB+D, NTU-RGB+D 120, and Kinetics-Skeleton, our method achieves state-of-the-art or comparably competitive performance compared to recent skeleton-based action recognition methods.

2 Related Work

2.1 GCN-based methods

ST-GCN [46] presents a standard pipeline that constructs a skeleton sequence as a spatial-temporal graph. Subsequent studies [8, 12, 22, 33, 53] extend toward dynamic topology, which adaptively learns relational structures depending on the input, layer (block), and channel. 2s-AGCN [33] introduces an adaptive graph that captures joint correlations and modulates feature aggregation conditioned on the input sample and network layer (block), and CTR-GCN [8] further refines modeling via channel-wise topology refinement by constructing a dynamic adjacency matrix that reflects channel-level correlations. From a representation learning perspective, InfoGCN [12] leverages information theory to refine graph-based representations and is designed to learn more structured latent representations. Building on this line, BlockGCN [53] strengthens topology awareness through block-level topological encoding and efficient refinement. More recently, ProtoGCN [22] enhances discriminative cues from a prototypical perspective by emphasizing fine-grained motion details to address the difficulty of distinguishing similar actions. Nevertheless, these dynamic-topology learning approaches

still remain closer to making inter-joint connectivity more flexible rather than fundamentally replacing one-hop message passing itself, leaving the limitation that increasing block depth can accelerate representation homogenization (over-smoothing) [3, 6, 21, 27, 35, 50] and the progressive attenuation of early-block fine-grained spatiotemporal cues.

2.2 Transformer-based methods

Transformer-based methods [14, 16, 30] treat joints or frames as tokens and directly capture long-range dependencies via self-attention, thereby escaping the structural constraints imposed by the one-hop aggregation of GCNs [8, 12, 15, 22, 33, 46, 53]. However, since global attention incurs high computational and memory costs, there is a strong trend toward jointly studying token/relation partitioning or more efficient attention designs. STTFormer [30] employs self-attention over spatiotemporal tuples to structurally model relations within neighboring frame and combines this self-attention mechanism with feature aggregation to better distinguish similar actions. Subsequently, SkateFormer [14] partitions body joints and frames according to skeletal-temporal relation types and performs attention within each partition, presenting a direction that improves efficiency while preserving long-range dependencies.

2.3 Titans

Neural memory designs for efficiently leveraging long-term context in long sequences have primarily evolved in natural language processing and Transformer-based sequence modeling, motivated by alleviating the representational limitations of fixed-length hidden states and the computational cost of self-attention [2]. To address these bottlenecks, Titans [1] handles the current context using sliding-window attention [2] while selectively memorizing information from earlier context via a surprise-based signal (test-time memorization), with the memory state updated online to maintain and exploit long-term context. Inspired by Titans, we propose a memory mechanism that reads/writes the skeleton joint topology learned at each block and the spatiotemporal features generated by it. Titans-like approaches mainly focus on storing and leveraging long-term context along the temporal axis (sequence axis), whereas our study, as shown in Figure 1(b), designs a memory that stores and updates complementary block-wise representations along the GCN block axis. Specifically, we use the block-wise topology as a query to retrieve (read) a memory prediction of the current block representation, and update the memory by updating (writing) the unexplained information from the current block, which captures newly emerging cues. We then inject the information stored in the memory across blocks into the final representation to compensate for fine-grained spatiotemporal cues and high-resolution relational cues that are prone to dilution through repeated aggregation. Therefore, rather than a general long-context neural memory, we propose a topology-conditioned block-progressive memory tailored to mitigating representation degradation in the hierarchical progression of skeleton GCNs.

3 Proposed Method

In this section, we briefly summarize the notation for GCN-based representations widely used in skeleton-based action recognition, and then present our topology-conditioned block-progressive memory in detail.

3.1 Preliminaries

A human skeleton sequence is commonly represented as a spatial-temporal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_\mathcal{S}, \mathcal{E}_\mathcal{T})$. We define the node (joint) set as $\mathcal{V} = \{v_{n,t} \mid n = 1, \ldots, N,\ t = 1, \ldots, T\}$, where N is the number of joints and T is the number of frames. $\mathcal{E}_\mathcal{S}$ denotes the spatial edge set (spatial topology) defined by inter-joint connections within the same frame, representing intra-body connections based on the human kinematic structure. This can be expressed by an adjacency matrix $A^{(l)} \in \mathbb{R}^{N \times N \times C}$, and each block uses it as a learnable parameter to adaptively model such spatial connectivity. The superscript $(l) \in \{1, \ldots, L\}$ indicates the index of a specific block among the total L blocks (typically $L = 10$), and C is the channel dimension. $\mathcal{E}_\mathcal{T}$ denotes the temporal graph topology defined by adjacent temporal edges between the same joint across neighboring frames, representing inter-body connections.

GCN-based methods [8,11,12,22,24,33,46,48,53] generally construct a single GCN block by combining spatial-GCN (S-GCN) and temporal-GCN (T-GCN). The input skeleton sequence is represented as a spatiotemporal feature $H^{(0)} \in \mathbb{R}^{N \times T \times C^{(0)}}$, and $C^{(0)} \in \{2, 3\}$ denotes 2D or 3D keypoints. S-GCN updates node features by aggregating neighboring node features connected by the spatial edges $\mathcal{E}_\mathcal{S}$ using a graph $\mathcal{G}_\mathcal{S} = (\mathcal{V}, \mathcal{E}_\mathcal{S})$. The equation is as follows:

$$H_\mathcal{S}^{(l)} = \mathrm{ReLU}\Big(A^{(l)} H^{(l-1)} W^{(l)}\Big).$$

Here, $H_\mathcal{S}^{(l)}$ is the S-GCN output feature of the l-th block, and $\mathrm{ReLU}(\cdot)$ is the activation function [26]. $W^{(l)} \in \mathbb{R}^{C^{(l-1)} \times C^{(l)}}$ is a learnable parameter in the l-th block for projecting the feature channels from $C^{(l-1)}$ to $C^{(l)}$.

T-GCN updates each node (joint) feature along the temporal axis by aggregating features from frames that are connected by $\mathcal{E}_\mathcal{T}$ between the same joints, based on a graph $\mathcal{G}_\mathcal{T} = (\mathcal{V}, \mathcal{E}_\mathcal{T})$. Given the S-GCN output feature of the l-th block,

$$H_\mathcal{S}^{(l)} = \left\{ \left(H_\mathcal{S}^{(l)}\right)_{n,t} \in \mathbb{R}^{C^{(l)}} \;\middle|\; n = 1, \ldots, N,\ t = 1, \ldots, T \right\},$$

the temporal update equation can be defined as follows:

$$\left(H_\mathcal{T}^{(l)}\right)_{n,t} = \sum_{k=-K}^{K} \left(w_k * \left(H_\mathcal{S}^{(l)}\right)_{n,t+k} \right), \quad n = 1, \ldots, N,\ t = 1, \ldots, T.$$

Here, $\left(H_\mathcal{T}^{(l)}\right)_{n,t}$ is the feature of the n-th joint at the t-th frame, $2K + 1$ is the window size of the temporal convolution, and $w = \{w_k\}$ is a learnable temporal kernel parameter.

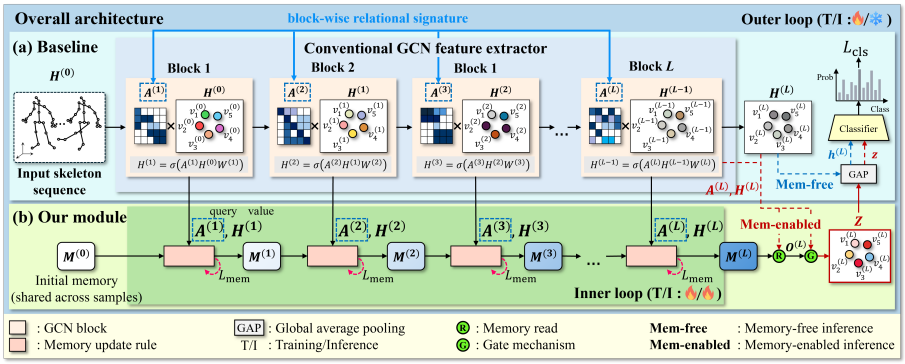

Fig. 2: Overall architecture of the proposed method. (a) Baseline with a conventional GCN feature extractor. (b) Our topology-conditioned block-progressive memory module. The outer loop covers the entire architecture except the inner-loop memory update.

As shown in Figure 2(a), for the final L-th block output feature $H^{(L)} \in \mathbb{R}^{N\times T\times C^{(L)}}$, we obtain an embedding vector $h^{(L)} \in \mathbb{R}^{C^{(L)}}$ by applying global average pooling (GAP) over the spatial and temporal axes. Then, $h^{(L)}$ is used as the input to the classifier to predict the action class of each skeleton sample. During training, for each skeleton sample x and its ground-truth label y, the GCN model is trained to minimize a classification loss L_{cls}, such as cross-entropy [32].

3.2 Topology-conditioned block-progressive memory

Overview. As shown in Figures 1 and 2, this work proposes a topology-conditioned block-progressive memory to alleviate representation homogenization (over-smoothing) [3, 6, 21, 27, 35, 50] and the progressive attenuation of early-block fine-grained spatiotemporal cues, which arise from repeated node feature aggregation during GCN block progression. The key observation is that the topology $A^{(l)}$ learned at each block is a block-wise relational signature that summarizes inter-joint interaction patterns for the current sample. Since the GCN block output $H^{(l)}$ is inherently the result of message passing defined by this topology $A^{(l)}$, we update the memory to reconstruct $H^{(l)}$ conditioned on $A^{(l)}$. This topology-conditioned reconstruction encourages the memory to consistently accumulate cues that are necessary from the perspective of the current block, rather than arbitrarily accumulating information. To this end, as shown

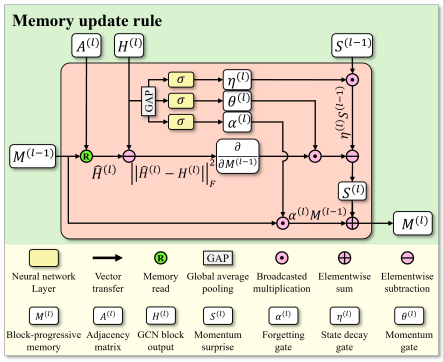

Fig. 3: An enlarged view of the memory update rule part in Figure 2.

in the inner loop of Figure 2(b), the memory performs an online, block-by-block update of a sample-wise memory state by using $A^{(l)}$ as a query and $H^{(l)}$ as a value. The update strength is determined by **"how well the previous memory $M^{(l-1)}$ explains the current block representation $H^{(l)}$,"** and the reconstruction error serves as a signal that quantifies the surprise—i.e., spatiotemporal cues that are not explained by the previous memory and newly emerge (or become more discriminative) at the current block. As a result, the memory cumulatively compensates for cues that are prone to dilution as blocks become deeper, helping the final representation form a more discriminative embedding. In the following, we describe the components of the topology-conditioned block progressive memory.

Memory state parameterization. Inspired by the neural memory design of Titans [1], we use two memory matrices M_{key} and M_{val} that are updated in the inner loop, together with the ReLU activation function, to satisfy a nonlinear mapping conditioned on the block-wise topology. M_{key} is an encoder matrix that projects the topology query A into a hidden space, determining which hidden components are activated by the topology of the current block. M_{val} is a decoder matrix that expands the resulting hidden code into $N \times T$ spatiotemporal positions to generate the predicted feature. This design induces current topology query A to attend to the stored past key M_{key} and retrieve (i.e., read) the corresponding value M_{val}. The memory state at the l-th block is defined as follows:

$$M^{(l)} \coloneqq \left\{ M_{\text{key}}^{(l)},\, M_{\text{val}}^{(l)} \right\}, \qquad M_{\text{key}}^{(l)} \in \mathbb{R}^{N^2 \times D}, \quad M_{\text{val}}^{(l)} \in \mathbb{R}^{D \times NT},$$

where D is the memory hidden dimension.

Topology-conditioned read. We define a memory read operation that predicts the current block feature $H^{(l)}$ from the previous memory $M^{(l-1)}$ conditioned on the query $A^{(l)}$. First, we reshape $A^{(l)} \in \mathbb{R}^{N \times N \times C^{(l)}}$ to obtain $\tilde{A}^{(l)} \in \mathbb{R}^{N^2 \times C^{(l)}}$. Then, the predicted feature $\hat{H}^{(l)}$ is given by:

$$\hat{H}^{(l)} = \text{Read}\Big(\tilde{A}^{(l)};\, M^{(l-1)}\Big),$$

$$\text{Read}\Big(\tilde{A}^{(l)};\, M\Big) = \left(\text{ReLU}\left(\left(\tilde{A}^{(l)}\right)^{\top} M_{\text{key}}\right) M_{\text{val}}\right)^{\top}.$$

The output of Read$(\cdot)$ is $\hat{H}^{(l)} \in \mathbb{R}^{NT \times C^{(l)}}$, which is reshaped to $\mathbb{R}^{N \times T \times C^{(l)}}$ for reconstruction loss computation, yielding the same tensor form as $H^{(l)}$.

Self-supervised reconstruction objective. We define the inner-loop self-supervised loss for measuring the surprise between the prediction $\hat{H}^{(l)}$ and the block output $H^{(l)}$ as follows:

$$L_{\text{mem}}^{(l)} = \left\| \hat{H}^{(l)} - H^{(l)} \right\|_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The loss $L_{\text{mem}}^{(l)}$ measures how well the previous memory $M^{(l-1)}$ explains $H^{(l)}$ from the perspective of the current topology $\tilde{A}^{(l)}$, and quantifies the unexplained error as surprise.

The memory can be updated by gradient descent on the reconstruction loss:

$$M^{(l)} = M^{(l-1)} - \nabla_{M^{(l-1)}} L_{\text{mem}}^{(l)}.$$

Therefore, if $M^{(l-1)}$ can reconstruct $H^{(l)}$ well via $\tilde{A}^{(l)}$, then $L_{\text{mem}}^{(l)}$ and its gradient become small, limiting the update. Conversely, if the reconstruction error is large, it indicates that new structural information emerges at the current block, producing a larger correction signal, and the memory is updated to better explain the current block representation.

Stabilized update with momentum and adaptive forgetting. However, as noted in Titans [1], performing only immediate updates as above may excessively optimize for the current block signal, making it difficult to sufficiently reflect different block-wise representations observed in later blocks. Therefore, we adopt the momentum-based update and adaptive forgetting from Titans in our method, and use a momentum surprise $S^{(l)}$ and a forgetting gate. As shown in Figure 3, the update rule of M by using S is defined as follows:

$$S^{(l)} = \eta^{(l)} \odot S^{(l-1)} - \theta^{(l)} \odot \nabla_{M^{(l-1)}} L_{\text{mem}}^{(l)},$$
$$M^{(l)} = \alpha^{(l)} \odot M^{(l-1)} + S^{(l)},$$

where $\odot$ denotes element-wise multiplication with broadcasting. Here, $S^{(l)}$ has the same structure as the memory and is given by

$$S^{(l)} \coloneqq \left\{ S_{\text{key}}^{(l)}, S_{\text{val}}^{(l)} \right\}, \qquad S_{\text{key}}^{(l)} \in \mathbb{R}^{N^2 \times D}, \quad S_{\text{val}}^{(l)} \in \mathbb{R}^{D \times NT}.$$

$\alpha^{(l)} \in [0,1]$ is a gate that controls forgetting, $\eta^{(l)} \in [0,1]$ is a state decay factor, and $\theta^{(l)} \in [0,1]$ is a momentum step factor, where each gate is predicted from the block output feature H. We obtain

$$h^{(l)} = \text{GAP}\left(H^{(l)}\right) \in \mathbb{R}^{C^{(l)}}$$

via global average pooling (GAP), and then define the gates using a one-layer fully-connected layer and a sigmoid function as:

$$\alpha^{(l)} = \sigma\left(\text{FC}_{\alpha^{(l)}}\left(h^{(l)}\right)\right), \quad \eta^{(l)} = \sigma\left(\text{FC}_{\eta^{(l)}}\left(h^{(l)}\right)\right), \quad \theta^{(l)} = \sigma\left(\text{FC}_{\theta^{(l)}}\left(h^{(l)}\right)\right),$$

where $\sigma(\cdot)$ is the sigmoid function.

The initial memory $M^{(0)}$ is a shared initialization (learnable parameter) that is learned in the outer loop, and $S^{(0)}$ is initialized to zero for each sample. Therefore, the inner loop updates $M^{(l)}$ and $S^{(l)}$ online, block by block, for each sample, and in this process, a large reconstruction gradient encourages the memory to incorporate new cues from the current block that are difficult to capture solely from the perspectives of previous blocks.

Gated fusion of memory and GCN features. After completing the sample-wise memory update, we construct the final representation Z by fusing the memory readout $O^{(L)}$ at the final stage L and the last GCN block output $H^{(L)}$ using a gate mechanism. As shown in Figure 2, we obtain:

$$O^{(L)} = \mathrm{Read}\Big(A^{(L)};\, M^{(L)}\Big), \quad g = \sigma\Big(O^{(L)} w_1 + H^{(L)} w_2 + b\Big),$$
$$Z = g \odot O^{(L)} + (1 - g) \odot H^{(L)},$$

where $g \in [0,1]^{N \times T \times C^{(L)}}$ is a gate that controls the contributions of the memory output and the block output. Then, we obtain an embedding vector $z \in \mathbb{R}^{C^{(L)}}$ by applying GAP to Z, and use z as the input to the classifier to predict the action class.

As shown in Figures 2 and 3, in the outer loop, in addition to the GCN backbone and classifier parameters, we optimize the update-rule parameters $\{\mathrm{FC}_\alpha, \mathrm{FC}_\eta, \mathrm{FC}_\theta\}$, the shared initial memory $M^{(0)}$ $(l = 0)$, and the gate parameters (w_1, w_2, b) via a classification loss such as cross-entropy [32]. In contrast, in the inner loop, for the current sample, we update $M^{(l)}$ and $S^{(l)}$ $(l \neq 0)$ online using the block-wise self-supervised reconstruction objective.

4 Experiments

4.1 Datasets

We evaluate our method on three widely used skeleton-based action recognition benchmarks: NTU RGB+D (NTU-60) [31], NTU RGB+D 120 (NTU-120) [23], and Kinetics-Skeleton (Kinetics) [19]. NTU-60 contains 56,880 sequences from 40 subjects over 60 classes, while NTU-120 expands it to 114,480 sequences from 106 subjects over 120 classes. We follow the standard evaluation protocols provided by each dataset. Kinetics-Skeleton provides pose-estimated skeletons for Kinetics-400, and we use the publicly available skeleton annotations released by PYSKL [15], reporting Top-1/Top-5 accuracies. Further dataset details and protocol descriptions are provided in the Supp. A.

4.2 Implementation details

In our experiments, we implement our method based on the official code release of PYSKL [15], and adopt a baseline that dynamically learns the connectivity structure using skeleton feature correlations (motion topology enhancement module [22]), which is widely used in recent CTR-GCN-based methods [8, 12, 22, 53]. All models are trained for 150 epochs with a batch size of 64 [22]. We use stochastic gradient descent for optimization, with Nesterov momentum [38] set to 0.9 and weight decay set to 5×10^{-4}. The initial learning rate is set to 0.05, and we apply a cosine decay scheduler. We use the commonly adopted settings of $L = 10$ blocks [8, 12, 15, 22, 53] and $T = 100$ frames [15, 22]. The hidden dimension of the memory module is set to $D = 384$. For training stability, we apply

Table 1: Performance comparison with state-of-the-art methods. E2/E4/E6 denote 2-/4-/6-stream ensembles, respectively. "Mem-free" and "Mem-enabled" indicate whether the memory module is disabled or enabled at inference; all our models are trained with memory. Best results are highlighted in blue (bold), and second-best in red (bold).

Category	Methods	Publication	NTU RGB+D (%)						NTU RGB+D 120 (%)						Kinetics-Skeleton (%)					
			X-Sub			X-View			X-Sub			X-Set			Top-1			Top-5		
			E2	E4	E6	E2	E4	E6	E2	E4	E6	E2	E4	E6	E2	E4	E6	E2	E4	E6
Transformer	STST [49]	ACM MM 2021	–	91.9	–	–	96.8	–	–	–	–	–	–	–	–	38.3	–	–	61.2	–
	SkeMixFormer [45]	ACM MM 2023	–	93.0	93.2	–	97.0	97.2	–	90.0	90.2	–	91.3	91.5	–	–	–	–	–	–
	JT-GraphFormer [51]	AAAI 2024	92.5	93.4	–	97.1	97.5	–	89.0	89.9	–	91.0	91.7	–	–	–	–	–	–	–
	FreqMixFormer [41]	ACM MM 2024	–	93.4	93.6	–	97.3	97.4	–	90.2	90.5	–	91.5	91.9	–	–	–	–	–	–
	SkateFormer [14]	ECCV 2024	93.0	93.3	–	**97.4**	**97.8**	–	89.4	89.8	–	91.0	91.4	–	–	–	–	–	–	–
	FreqMixFormerv2 [40]	IEEE FG 2025	–	92.9	–	–	96.9	–	–	90.0	–	–	91.1	–	–	–	–	–	–	–
GCN	ST-GCN [46]	AAAI 2018	81.5	–	–	88.3	–	–	70.7	–	–	73.2	–	–	30.7	–	–	52.8	–	–
	2s-AGCN [33]	CVPR 2019	88.5	–	–	95.1	–	–	82.5	–	–	84.2	–	–	36.1	–	–	58.7	–	–
	DC-GCN+ADG [10]	ECCV 2020	–	90.8	–	–	96.6	–	–	86.5	–	–	88.1	–	–	–	–	–	–	–
	MS-G3D [24]	CVPR 2020	–	91.5	–	–	96.2	–	–	86.9	–	–	88.4	–	–	38.0	–	–	60.9	–
	MST-GCN [9]	AAAI 2021	–	91.5	–	–	96.6	–	–	87.5	–	–	88.8	–	–	38.1	–	–	60.8	–
	CTR-GCN [8]	ICCV 2021	–	92.4	–	–	96.8	–	88.7	88.9	–	90.1	90.6	–	–	–	–	–	–	–
	STF [20]	AAAI 2022	92.5	–	–	96.9	–	–	88.9	–	–	89.9	–	–	39.9	–	–	–	–	–
	InfoGCN [12]	CVPR 2022	–	92.7	93.0	–	96.9	97.1	–	89.4	89.8	–	90.7	91.2	–	–	–	–	–	–
	ST-GCN++ [15]	ACM MM 2022	91.4	92.1	92.6	96.7	97.0	97.4	87.0	87.5	88.6	87.5	89.8	90.8	–	–	49.1	–	–	–
	SkeletonGCL [18]	ICLR 2023	–	92.8	–	–	97.1	–	–	89.8	–	–	91.2	–	–	–	–	–	–	–
	FR-Head [52]	CVPR 2023	92.3	92.8	–	96.4	96.8	–	–	89.5	–	–	90.9	–	–	–	–	–	–	–
	GAP [42]	ICCV 2023	–	92.9	–	–	97.0	–	–	89.9	–	–	91.1	–	–	–	–	–	–	–
	HD-GCN [48]	ICCV 2023	92.4	93.0	93.4	96.6	97.0	97.2	89.1	89.8	90.1	90.6	91.2	91.6	–	–	40.9	–	–	63.5
	DS-GCN [43]	AAAI 2024	–	93.1	–	–	97.5	–	–	89.2	–	–	91.1	–	–	50.6	–	–	–	–
	BlockGCN [53]	CVPR 2024	–	93.1	–	–	97.0	–	–	90.3	–	–	91.5	–	–	–	–	–	–	–
	Hilbert-Schmidt [7]	AAAI 2025	–	**93.7**	–	–	97.3	–	89.3	**90.6**	–	90.7	91.7	–	–	–	–	–	–	–
	HiOD [5]	ICCV 2025	–	93.6	93.8	–	97.4	97.6	–	90.2	90.4	–	91.6	91.9	–	–	–	–	–	–
	ProtoGCN [22]	CVPR 2025	93.0	93.5	93.8	97.2	97.5	**97.8**	**89.7**	90.4	90.9	**91.2**	**91.9**	**92.2**	49.9	51.3	51.9	**74.0**	**75.1**	**75.6**
	Ours (Mem-free)		**93.3**	**93.7**	**93.9**	**97.3**	**97.7**	**97.9**	**90.0**	**90.7**	**91.0**	**91.5**	**92.1**	**92.4**	**50.9**	**52.8**	**53.2**	**74.3**	**75.9**	**76.4**
	Ours (Mem-enabled)		**93.5**	**93.8**	**94.0**	**97.3**	**97.7**	**97.9**	**90.0**	**90.7**	**91.1**	**91.5**	**92.1**	**92.4**	**51.1**	**52.9**	**53.4**	**74.3**	**75.9**	**76.4**

gradient clipping [28] (max ℓ_2 norm 1.0) to the FC layers that predict (α, η, θ) and to $M_{\text{key}}^{(0)}, M_{\text{val}}^{(0)}$. We apply weight decay of 5×10^{-5} only to the parameters of the FC layers corresponding to (α, η, θ). Learnable matrix parameters are randomly initialized, and data preprocessing follows the default pipeline provided by PYSKL [15]. All experiments are conducted on a single NVIDIA RTX PRO 6000 Blackwell GPU, with additional implementation details provided in Supp. C.

4.3 Comparison with the state-of-the-art methods

To improve recognition performance, multi-stream ensemble strategies are widely used in skeleton-based human action recognition. For a fair comparison, this work follows the six-stream ensemble setting of InfoGCN [12], which is widely adopted in prior methods. We compare the proposed method with recent state-of-the-art methods on large-scale benchmarks including NTU-60, NTU-120, and Kinetics. Table 1 shows that our method improves performance across all scenarios. We use the memory module during training, and even in a memory-free mode at inference, it maintains performance gains over prior state-of-the-art methods; in the memory-enabled setting that applies the same sample-wise update rule at inference, it yields additional modest improvements. As a result, our method achieves state-of-the-art performance across datasets and ensemble settings. On NTU-60, we obtain the second-best results on X-View for E2 and E4, while achieving the best performance for all other ensemble settings. Moreover, we achieve the best results across E2/E4/E6 on NTU-120 and Kinetics. Table 2 provides modality-wise performance comparisons with GCN-based methods. While

Table 2: Top-1 accuracy (%) comparison with state-of-the-art GCN-based methods on NTU RGB+D and NTU RGB+D 120. J, B, JM, and BM denote the joint, bone, joint-motion, and bone-motion modalities, respectively.

Methods	Publication	NTU RGB+D								NTU RGB+D 120							
		X-Sub				X-View				X-Sub				X-Set			
		J	B	JM	BM	J	B	JM	BM	J	B	JM	BM	J	B	JM	BM
ST-GCN [46]	AAAI 2018	87.8	88.6	85.8	86.2	95.5	95.0	93.7	92.8	82.1	83.7	–	–	84.5	85.8	–	–
AAGCN [34]	IEEE TIP 2020	89.0	89.2	–	–	95.7	95.2	–	–	82.8	84.7	–	–	84.8	86.2	–	–
MS-G3D [24]	CVPR 2020	89.6	89.3	–	–	95.9	95.0	–	–	84.0	85.3	–	–	86.0	87.3	–	–
CTR-GCN [8]	ICCV 2021	89.6	90.0	88.0	87.5	95.6	95.4	94.4	93.6	84.0	85.9	–	–	85.9	87.4	–	–
ST-GCN++ [15]	ACM MM 2022	89.3	90.1	87.5	87.3	95.6	**95.5**	94.3	93.8	83.2	85.6	–	–	85.6	87.5	–	–
InfoGCN [12]	CVPR 2022	89.8	90.6	88.9	88.6	95.2	**95.5**	94.2	93.6	–	–	–	–	–	–	–	–
STA-GCN [17]	ACCV 2022	89.5	90.2	88.3	88.6	95.6	95.4	94.3	**94.1**	85.0	85.5	82.9	**83.1**	86.2	87.1	84.0	84.9
SkeletonGCL [18]	ICLR 2023	90.8	91.1	–	–	95.3	95.4	–	–	–	–	–	–	–	–	–	–
FR-Head [52]	CVPR 2023	90.3	91.1	88.7	87.6	95.3	95.0	93.6	92.6	–	–	–	–	–	–	–	–
HD-GCN [48]	ICCV 2023	90.6	90.9	–	–	95.7	95.1	–	–	85.2	86.4	–	–	87.2	88.4	–	–
BlockGCN [53]	CVPR 2024	90.9	91.3	88.7	88.3	95.4	95.3	93.3	92.6	**86.9**	88.1	82.7	83.0	**88.2**	89.3	84.6	84.8
Hilbert-Schmidt [7]	AAAI 2025	–	–	–	–	–	–	–	–	86.0	87.6	–	–	87.4	**88.9**	–	–
ProtoGCN [22]	CVPR 2025	**91.5**	**92.0**	**89.3**	**89.1**	**96.3**	**96.2**	**95.5**	94.0	85.3	**88.6**	**83.2**	**83.5**	**88.2**	**89.9**	**86.0**	**85.3**
Ours (Mem-enabled)		**91.9**	**92.3**	**89.6**	**89.2**	**96.6**	**96.2**	**95.6**	**94.6**	**86.5**	**88.9**	**83.9**	**83.5**	**88.5**	**90.1**	**86.4**	**85.6**

Table 3: Effect of training-time memory on two baselines on NTU RGB+D 120 (X-Sub), bone modality. Inference is memory-free; gains are in parentheses.

Methods	Training		Accuracy (%)
	w/o memory	w/ memory	
Baseline 1 [15]	✓		87.8
		✓	**88.4** (↑0.6)
Baseline 2 [22]	✓		88.1
		✓	**88.9** (↑0.8)

Table 4: Effect of test-time memory. Δ denotes the resulting absolute accuracy gain in percentage points, computed as (Mem-enabled − Mem-free).

Setting	NTU RGB+D				Kinetics-Skeleton	
	X-Sub		X-View			
	J	B	J	B	J	B
Mem-free	91.87	92.16	96.52	96.12	48.12	46.33
Mem-enabled	91.91	92.26	96.57	96.18	48.36	47.45
Δ	**+0.04**	**+0.10**	**+0.05**	**+0.06**	**+0.24**	**+0.12**

multi-stream ensemble strategies can achieve strong performance, they increase training/inference cost and system complexity because multiple modalities must be trained and deployed separately. Moreover, ensemble performance is sensitive to the choice of modality-wise ensemble weights; thus, for fair comparison and interpretation, it is necessary to examine the single-modality performance. The results in Table 2 are all measured under the memory-enabled setting that applies the same sample-wise update rule at inference. As a result, on NTU-120, we obtain the second-best result on the joint modality, and achieve the best accuracy on all other modalities on NTU-60 and NTU-120.

4.4 Ablation studies

Training-time effect of memory. Table 3 compares a baseline that does not use memory in two representative training pipelines with a setting that applies memory during training and uses memory-free inference. We use two baselines: (1) PYSKL [15] (re-trained using the latest officially released code), and (2) PYSKL equipped with the motion-topology enhancement module [22]. Note that the method referred to as "baseline" in [22] corresponds to our Baseline 2. By adding our memory to each baseline, we achieve performance improvements of 0.6% and 0.8%, respectively, showing that our memory contributes to representation shaping during training.

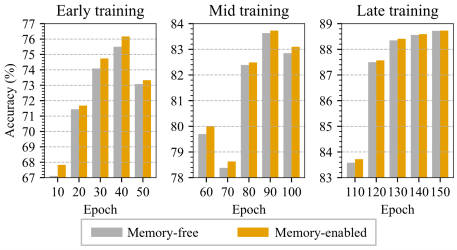

Fig. 4: Epoch-wise Top-1 accuracy (%) on NTU RGB+D 120 (X-Sub) for the bone modality, comparing memory-free inference and memory-enabled inference. Results are shown at epochs grouped into early (10–50 epochs), mid (60–100 epochs), and late training (110–150 epochs).

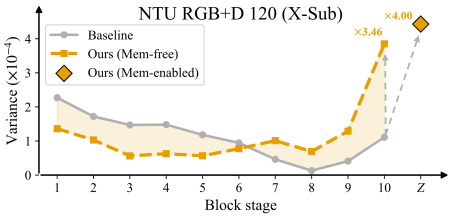

Fig. 5: Variance is computed per test sample across joints and averaged over the test set. Ours (Mem-free) is reported for stages 1–10, while Z denotes the gated fusion output produced only when memory is enabled. Annotated values indicate the variance ratio (Ours/Baseline) last stage.

Test-time memory: Memory-free vs. Memory-enabled inference. Figure 4 compares epoch-wise inference performance between memory-free and memory-enabled settings. In the early training stage (before performance converges/saturates), the memory-enabled setting shows larger gains over the memory-free setting, and the effect of memory is relatively more pronounced. As training proceeds and approaches convergence, the performance gap between the two settings decreases. This suggests that the proposed memory supports backbone representation shaping during training, and its benefits are gradually internalized into the backbone parameters. Table 4 compares memory-free and memory-enabled inference for two representative modalities, joint and bone, on NTU-60 and Kinetics. As shown in Table 6, memory-free inference retains the main benefit without additional overhead, whereas memory-enabled inference provides modest performance gains with about 0.4M additional parameters. Thus, the memory module can be selectively enabled depending on the trade-off between accuracy and deployment cost.

Effect of memory on representation homogenization (over-smoothing). Figure 5 quantifies the variability of joint-node feature magnitudes across block stages over the full NTU-120 X-Sub test set. Specifically, for each test sample, we compute the variance of feature magnitudes across joints, and then average it over the entire test set to compare stage-wise statistics. As a result, our method maintains larger variance values in later blocks compared to the baseline. Here, baseline refers to the case where memory is not used during both training and inference under the same training setting, and Mem-enabled refers to the case where memory is enabled during both training and inference. Maintaining larger inter-node variance in the final embedding indicates that excessive similarity among node representations (representation homogenization) is alleviated and node-wise representational diversity is improved, supporting reduced over-smoothing caused by repeated aggregation. Figure 6 visualizes, for

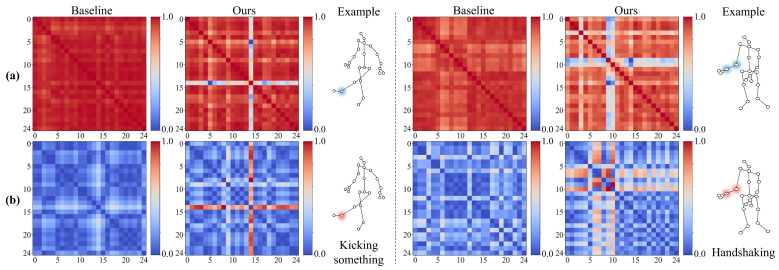

Fig. 6: Qualitative visualization of inter-joint relations for example sequences, comparing the baseline and our method. The horizontal and vertical axes denote joint nodes. (a) Cosine similarity of joint features; (b) magnitude differences of joint features.

Table 5: Ablation of gate module and memory components.

Variant	Mem.	Gate	α	η	θ	Acc.
Baseline (NTU-60 X-Sub J)	×	×	×	×	×	91.3
w/o all memory components	✓	×	×	×	×	89.8
w/o forgetting factor (α)	✓	✓	×	✓	✓	90.4
w/o momentum factor (θ)	✓	✓	✓	✓	×	91.3
w/o decay factor (η)	✓	✓	✓	×	✓	91.4
w/o gate module	✓	×	✓	✓	✓	91.5
Ours	✓	✓	✓	✓	✓	**91.9**

Table 6: Training and inference cost under memory usage settings.

Setting	Baseline	Mem-free	Mem-enabled
Train Mem.	×	✓	✓
Infer Mem.	×	×	✓
Train time	9h	12h	12h
Throughput	70.2	70.2	27.9
Params (M)	3.95	3.95	4.36
GFLOPs	7.48	7.48	7.85

the same sample, the inter-joint representation similarity (cosine similarity) and magnitude differences in the final embedding. In the baseline, a homogenization (over-smoothing) tendency is observed, where joint representations become broadly similar. In contrast, the proposed method exhibits clearer regions with lower inter-joint similarity and more distinct magnitude differences. This suggests that our method alleviates the mixing of information across different joints caused by repeated aggregation, thereby maintaining more separated joint-wise roles/contributions in the final embedding.

Stabilized memory updates and representation shaping. As shown in Tab. 5, naive memory updates without stabilization components degrade performance by 1.5% compared with the baseline (highlighted in blue), indicating that direct surprise-based updates may overfit to high-frequency noise. Our forgetting factor α, momentum factor θ, and decay factor η mitigate this issue by stabilizing memory updates in a block-adaptive manner, suppressing noisy updates while preserving fine-grained cues.

We further analyze how the proposed memory module shapes block-wise representations. As shown in Fig. 7(a), the most-updated joints in M_{val} for a "cheer up" sample are concentrated on both hand tips (J22 and J24), suggesting that memory captures fine-grained semantic cues. (b) shows that memory updates are larger in early blocks than in later blocks across epochs, indicating that memory mainly incorporates early-stage local cues. Moreover, as shown in (c), the baseline exhibits larger gradients on the adjacency matrix A in later blocks

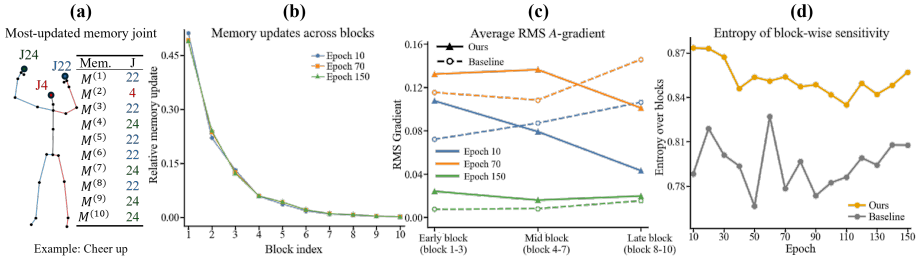

Fig. 7: Analysis of memory-induced representation shaping on NTU RGB+D 60. (a) Largest-updated joints in M_{val}. (b) Block-wise relative memory update, measured by the normalized RMS norm of $M^{(l)} - M^{(l-1)}$. (c) Normalized RMS gradient of the adjacency matrix $A^{(l)}$. (d) Entropy of block-wise feature sensitivity, where higher entropy indicates more evenly distributed sensitivity across blocks.

than in early/middle blocks, while our method shifts the relative gradient contribution toward early/middle blocks through the meta-gradient induced by L_{mem} (see Supp. D). Consistently, (d) shows that our method yields higher entropy of block-wise gradient contributions across blocks than the baseline, where each contribution is computed from $\|\partial L_{\text{cls}}/\partial H^{(l)}\|_2$, suggesting that the memory reduces reliance on a few dominant blocks while helping early-block information become internalized into the learned parameters.

5 Conclusion

In this paper, we present topology-conditioned block-progressive memory, a memory module tailored for skeleton-based action recognition with GCN backbones. The key idea is to store and update complementary block-wise representations along the GCN block axis and inject the accumulated memory into the final embedding, thereby alleviating progressive representation degradation caused by repeated aggregation—over-smoothing and the attenuation of early-block fine-grained spatiotemporal cues. The memory is updated via a topology-conditioned reconstruction loss, where the reconstruction error measures how well the previous memory explains the current block representation and captures newly emerging cues. We use the memory during training and can optionally enable it at test time. Under both inference options—memory-free and memory-enabled—we achieve state-of-the-art performance and consistently outperform baselines on large-scale benchmarks including NTU RGB+D, NTU RGB+D 120, and Kinetics-Skeleton.

Acknowledgements

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2026-25489602), the AI Seoul Tech Research Support Program of the Seoul Future Foundation, and the Artificial Intelligence Innovation Graduate School Program (Sogang University) grant funded by the Korea government (MSIT) (No. RS-2026-25547954).

References

1. Behrouz, A., Zhong, P., Mirrokni, V.: Titans: Learning to memorize at test time (2024), arXiv:2501.00663
2. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
3. Cai, C., Wang, Y.: A note on over-smoothing for graph neural networks (2020), arXiv:2006.13318
4. Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: CVPR. pp. 7291–7299 (2017)
5. Chang, H., Ren, P., Zhang, H., Xie, L., Chen, H., Yin, E.: Hierarchical-aware orthogonal disentanglement framework for fine-grained skeleton-based action recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11252–11261 (2025)
6. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: AAAI. vol. 34, pp. 3438–3445 (2020)
7. Chen, H., Yang, Y., Lyu, Y.: Skeleton-based action recognition with non-linear dependency modeling and hilbert-schmidt independence criterion. In: AAAI. vol. 39, pp. 2043–2051 (2025)
8. Chen, Y., Zhang, Z., Yuan, C., Li, B., Deng, Y., Hu, W.: Channel-wise topology refinement graph convolution for skeleton-based action recognition. In: ICCV. pp. 13359–13368 (2021)
9. Chen, Z., Li, S., Yang, B., Li, Q., Liu, H.: Multi-scale spatial temporal graph convolutional network for skeleton-based action recognition. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 1113–1122 (2021)
10. Cheng, K., Zhang, Y., Cao, C., Shi, L., Cheng, J., Lu, H.: Decoupling gcn with dropgraph module for skeleton-based action recognition. In: European conference on computer vision. pp. 536–553. Springer (2020)
11. Cheng, K., Zhang, Y., He, X., Chen, W., Cheng, J., Lu, H.: Skeleton-based action recognition with shift graph convolutional network. In: CVPR. pp. 183–192 (2020)
12. gun Chi, H., Ha, M.H., Chi, S., Lee, S.W., Huang, Q., Ramani, K.: Infogcn: Representation learning for human skeleton-based action recognition. In: CVPR. pp. 20186–20196 (2022)
13. Cippitelli, E., et al.: A human activity recognition system using skeleton data from rgb-d sensors. Computational Intelligence and Neuroscience (2016)
14. Do, J., Kim, M.: Skateformer: skeletal-temporal transformer for human action recognition. In: ECCV. pp. 401–420. Springer (2024)
15. Duan, H., Wang, J., Chen, K., Lin, D.: Pyskl: Towards good practices for skeleton action recognition. In: ACM MM. pp. 7351–7354 (2022)
16. Han, S.H., Lee, S., Nam, H., Park, J.H., Cha, M.H., Kim, M.G., Lee, H., Ahn, S., Chae, M., Cho, S.I.: Doodle to detect: A goofy but powerful approach to skeleton-based hand gesture recognition. In: NeurIPS (2025)
17. Hang, R., Li, M.: Spatial-temporal adaptive graph convolutional network for skeleton-based action recognition. In: ACCV. pp. 1265–1281 (2022)
18. Huang, X., Zhou, H., Wang, J., Feng, H., Han, J., Ding, E., Wang, J., Wang, X., Liu, W., Feng, B.: Graph contrastive learning for skeleton-based action recognition. In: ICLR (2023)
19. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al.: The kinetics human action video dataset (2017), arXiv:1705.06950

20. Ke, L., Peng, K.C., Lyu, S.: Towards to-at spatio-temporal focus for skeleton-based action recognition. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36, pp. 1131–1139 (2022)
21. Keriven, N.: Not too little, not too much: a theoretical analysis of graph (over) smoothing. Advances in Neural Information Processing Systems **35**, 2268–2281 (2022)
22. Liu, H., Liu, Y., Ren, M., Wang, H., Wang, Y., Sun, Z.: Revealing key details to see differences: A novel prototypical perspective for skeleton-based action recognition. In: CVPR. pp. 29248–29257 (2025)
23. Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L.Y., Kot, A.C.: Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. IEEE TPAMI **42**(10), 2684–2701 (2019)
24. Liu, Z., Zhang, H., Chen, Z., Wang, Z., Ouyang, W.: Disentangling and unifying graph convolutions for skeleton-based action recognition. In: CVPR. pp. 143–152 (2020)
25. Moon, S., Kim, M., Qin, Z., Liu, Y., Kim, D.: Anonymization for skeleton action recognition. In: AAAI (2023)
26. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML. pp. 807–814 (2010)
27. Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification. In: International Conference on Learning Representations (ICLR) (2020)
28. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: ICML. pp. 1310–1318 (2013)
29. Presti, L.L., Cascia, M.L.: 3d skeleton-based human action classification: A survey. PR **53**, 130–147 (2016)
30. Qiu, H., Hou, B., Ren, B., Zhang, X.: Spatio-temporal tuples transformer for skeleton-based action recognition (2022), arXiv:2201.02849
31. Shahroudy, A., Liu, J., Ng, T.T., Wang, G.: Ntu rgb+d: A large scale dataset for 3d human activity analysis. In: CVPR. pp. 1010–1019 (2016)
32. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27**(3), 379–423 (1948)
33. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In: CVPR. pp. 12026–12035 (2019)
34. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Skeleton-based action recognition with multi-stream adaptive graph convolutional networks. IEEE TIP **29**, 9532–9545 (2020)
35. Song, Y.F., Zhang, Z., Shan, C., Wang, L.: Constructing stronger and faster baselines for skeleton-based action recognition. IEEE transactions on pattern analysis and machine intelligence **45**(2), 1474–1488 (2022)
36. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: CVPR. pp. 5693–5703 (2019)
37. Sun, Z., Ke, Q., Rahmani, H., Bennamoun, M., Wang, G., Liu, J.: Human action recognition from various data modalities: A review. IEEE TPAMI **45**(3), 3200–3225 (2022)
38. Sutskever, I., Martens, J., Dahl, G., Hinton, G.E.: On the importance of initialization and momentum in deep learning. In: ICML. pp. 1139–1147 (2013)
39. Wang, L., Huynh, D.Q., Koniusz, P.: A comparative review of recent kinect-based action recognition algorithms. IEEE TIP **29**, 15–28 (2019)
40. Wu, W., Wang, P., Chen, C., Lu, A.: Freqmixformerv2: Lightweight frequency-aware mixed transformer for human skeleton action recognition. In: 2025 IEEE

19th International Conference on Automatic Face and Gesture Recognition (FG). pp. 1–5. IEEE (2025)
41. Wu, W., Zheng, C., Yang, Z., Chen, C., Das, S., Lu, A.: Frequency guidance matters: Skeletal action recognition by frequency-aware mixed transformer. In: ACM MM. pp. 4660–4669 (2024)
42. Xiang, W., Li, C., Zhou, Y., Wang, B., Zhang, L.: Generative action description prompts for skeleton-based action recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10276–10285 (2023)
43. Xie, J., Meng, Y., Zhao, Y., Nguyen, A., Yang, X., Zheng, Y.: Dynamic semantic-based spatial graph convolution network for skeleton-based human action recognition. In: Proceedings of the AAAI conference on artificial intelligence. vol. 38, pp. 6225–6233 (2024)
44. Xie, J., Zhao, Y., Meng, Y., Zhao, H., Nguyen, A., Zheng, Y.: Are spatial-temporal graph convolution networks for human action recognition over-parameterized? In: CVPR. pp. 24309–24319 (2025)
45. Xin, W., Miao, Q., Liu, Y., Liu, R., Pun, C.M., Shi, C.: Skeleton mixformer: Multivariate topology representation for skeleton-based action recognition. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 2211–2220 (2023)
46. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: AAAI. vol. 32 (2018)
47. Yang, Y., et al.: Privacy-preserving human activity sensing: A survey. Array (2024)
48. Yang, Z., Li, K., Gan, H., Huang, Z., Shi, M.: Hd-gcn: A hybrid diffusion graph convolutional network (2023), arXiv:2303.17966
49. Zhang, Y., Wu, B., Li, W., Duan, L., Gan, C.: Stst: Spatial-temporal specialized transformer for skeleton-based action recognition. In: Proceedings of the 29th ACM international conference on multimedia. pp. 3229–3237 (2021)
50. Zhao, L., Akoglu, L.: Pairnorm: Tackling oversmoothing in gnns. In: International Conference on Learning Representations (ICLR) (2020)
51. Zheng, Y., Huang, H., Wang, X., Yan, X., Xu, L.: Spatio-temporal fusion for human action recognition via joint trajectory graph. In: AAAI. vol. 38, pp. 7579–7587 (2024)
52. Zhou, H., Liu, Q., Wang, Y.: Learning discriminative representations for skeleton based action recognition. In: CVPR. pp. 10608–10617 (2023)
53. Zhou, Y., Yan, X., Cheng, Z.Q., Yan, Y., Dai, Q., Hua, X.S.: Blockgcn: Redefine topology awareness for skeleton-based action recognition. In: CVPR. pp. 2049–2058 (2024)