GCN-DevLSTM: Path Development for Skeleton-Based Action Recognition

Anonymous authors
Paper under double-blind review

Abstract

Skeleton-based action recognition (SAR) in videos is an important but challenging task in computer vision. The recent state-of-the-art (SOTA) models for SAR are primarily based on graph convolutional neural networks (GCNs), which are powerful in extracting the spatial information from skeleton data. However, their ability to capture temporal dynamics remains limited. To address this, we propose the G-Dev layer, which leverages path development—a principled and parsimonious representation for sequential data based on Lie group structures—to enhance temporal modeling. By integrating the G-Dev layer, the proposed DevLSTM module summarizes local temporal dynamics, reducing the time dimension while retaining high-frequency information. It can be conveniently applied to any temporal graph data, complementing existing advanced GCN-based models. Our empirical studies on the NTU-60, NTU-120 and Chalearn2013 datasets demonstrate that our proposed GCN-DevLSTM network consistently improves the strong GCN baseline models and achieves competitive performance. [1]

1 Introduction

Action recognition has a wide range of applications in diverse fields such as human-computer interaction, performance assessment in athletics, virtual reality, and video monitoring (Wang & Mohamed, 2026). Compared to video-based action recognition, skeleton-based action recognition (SAR) exhibits stronger robustness to varying lighting conditions, improved efficiency in computation and storage (Chen et al., 2021), and better preservation of data privacy. Skeleton data can be obtained either by localization of 2D/3D human joints using depth cameras (Wang & Mohamed, 2026) or pose estimation algorithms from videos (Lal et al., 2025).

Despite extensive research in SAR, designing effective spatio-temporal representations of skeleton sequences remains a key challenge. Previously, Recurrent Neural Networks (RNNs) were commonly used due to their ability to capture temporal dynamics. The advent of Spatial-Temporal Graph Convolutional Networks (STGCNs) (Yan et al., 2018) has shifted the focus towards Graph Convolutional Networks (GCNs), which incorporate both spatial graph convolution and temporal convolution. The majority of subsequent GCN-based approaches (Chen et al., 2021; Shi et al., 2020; Cheng et al., 2020b), following the STGCN framework, focus on the improvement of the spatial graph convolution, with limited attention to temporal modules. These methods utilize 1-D temporal convolution operations to capture temporal features. However, whether temporal convolution is the most effective approach for modeling complex temporal dynamics remains an open question.

On the other hand, Rough path theory (Lyons, 1998) originated as a branch of stochastic analysis to make sense of controlled differential equations driven by highly oscillatory paths. More recently, the applications of rough path theory in sequential data analysis are emerging. In particular, the path signature, as a fundamental object of Rough path theory, has demonstrated strong capability in capturing temporal structure across various domains, including SAR (Yang et al., 2022; Shi et al., 2024). By embedding time series into the continuous path space, the signature offers a unified treatment to handle variable length and irregular sampling. However, the path signature may face challenges such as the curse of dimensionality due to high

[1] The camera-ready version will contain a link to the code repository to ensure reproducibility.

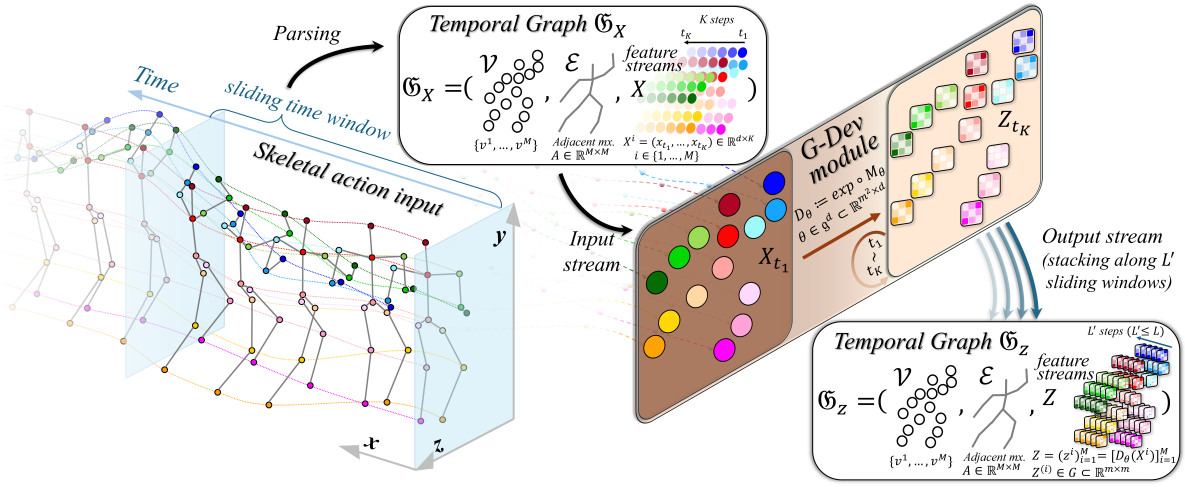


Figure 1: The work flow of G-Dev layer. The G-Dev layer takes a temporal graph (e.g., the skeleton sequence) as an input and outputs a temporal graph with a potentially significantly reduced time dimension. For each sub-time interval, we apply the G-Dev module (see Section 3.3) to generate a static graph representing the features within that window. We obtain the final temporal graph output by stacking these static graphs along the time dimension.

path dimensionality and lack of adaptability due to its deterministic nature (Lou et al., 2024a). To address these, Lou et al. (2024a) propose the path development from rough path theory, as a trainable alternative to path signature. The path development enjoys the desirable properties of the signature while reducing feature dimension significantly. Lou et al. (2024a) show that combining the path development under the appropriate Lie group with LSTM (Hochreiter & Schmidhuber, 1997) alleviates the gradient vanishing, leading to significant enhancement across various sequential tasks.

Motivated by these observations, this paper focuses on improving temporal modeling within GCN-based frameworks. To this end, we introduce a novel G-Dev layer (as shown in Figure 1) on the temporal graph, where each node is associated with time-series features. This layer utilizes a sliding window to summarize local temporal information via the principled path development, offering time dimension benefit and robustness to missing data and irregular sampling. Building on this, we propose the GCN-DevLSTM network, which retains the effectiveness of graph convolution for modeling spatial relationships among joints while leveraging the hybrid capacity of the G-Dev layer and LSTM to capture temporal dynamics. To further enrich structural modeling, we additionally incorporate a line graph stream to capture complementary higher-order relationships between skeletal connections. Our GCN-DevLSTM is flexible and can be integrated with different GCN backbones. Experimental results on NTU-60, NTU-120, and Chalearn2013 demonstrate that our approach consistently improves strong GCN baselines on NTU datasets and achieves competitive performance across all benchmarks. Furthermore, leveraging the properties of path development, our method exhibits significantly enhanced robustness under challenging conditions such as missing or irregular frames.

We summarize our main contributions as follows:

- We design a novel and principled graph development (**G-Dev**) layer for analyzing temporal graph data by extending the path development originally applied to vector-valued time series. As a generic network on graph data with time-varying node attributes, G-Dev exhibits efficacy and robustness to irregular sampling and missing data.
- We introduce the novel **GCN-DevLSTM** network for skeleton-based action recognition that effectively combines the G-Dev layer with LSTM to capture complex temporal dynamics.
- We further incorporate the line graph as an additional stream to capture complementary higher-order structural information, which provides further performance gains in strong multi-stream settings.

- Extensive experiments demonstrate that our method consistently improves strong GCN baselines on NTU datasets and achieves competitive performance across benchmarks, while providing enhanced robustness under challenging conditions.

2 Related Work

2.1 Skeleton-based Action Recognition

The main challenge in SAR is extracting the effective representation that captures both spatial and temporal dependency of skeleton sequences. Earlier methods typically use convolutional neural networks (CNNs) (Liu et al., 2017b) and RNN (Liu et al., 2017a) to address this challenge without considering structural topology. STGCN (Yan et al., 2018) is the first method to capture spatial and temporal relationships using GCN, taking the skeleton topology of the graph into consideration while this topology is not learnable, with only the spatial connection between adjacent joints being connected. As a result, some actions that rely on non-adjacent joints may lead to poor results. Liu et al. (2020) tackle this issue by introducing a learnable adjacency matrix alongside the original manually defined one, enabling connections between non-neighboring joints. Yet, this approach maintains the same graph for all action inputs, which might not be ideal, as different action classes may benefit from different graph topologies. To address this, Li et al. (2019b); Shi et al. (2019; 2020) propose adaptive GCN with the learnable and data-driven topology that shares among all channels. Cheng et al. (2020a); Chen et al. (2021) further propose the non-shared topology for each channel. To capture more comprehensive spatial dependencies, Kilic et al. (2025); Li et al. (2025) further enhance the skeleton graph by dynamically integrating additional hierarchical structural graphs. Additionally, BlockGCN (Zhou et al., 2024) reveals that purely learnable topologies may suffer from topology degradation, emphasizing the importance of explicitly preserving structural priors. Other recent studies improved model performance by enriching graph representations with higher-order features, such as angle-based information (Schlegel et al., 2024). While these approaches significantly improve spatial modeling, their temporal modeling is typically handled by simple temporal convolution (TCN) modules, which are limited in capturing long-range temporal dependencies. This suggests that temporal modeling remains under-explored in GCN-based frameworks.

Recently, transformer-based methods have been introduced to capture long-range spatial-temporal dependencies via self-attention. For example, LG-STFormer (Liu et al., 2025) and SkateFormer (Do & Kim, 2024) leverage attention mechanisms to model global joint interactions across both spatial and temporal dimensions, and hybrid GCN-transformer architectures (Chen et al., 2025) further combine local graph inductive bias with global attention modeling. Hierarchical transformer-based models such as SkelFormer (Yan et al., 2026) also explore multi-level spatio-temporal representations. However, these methods primarily focus on global dependency modeling through attention mechanisms, rather than improving temporal modeling within graph convolutional frameworks. Therefore, improving temporal modeling within the GCN paradigm remains an important and complementary research direction. In particular, designing more effective temporal modules that can be seamlessly integrated into GCN-based architectures is still underexplored.

2.2 Path signature & Path Development

The path signature is a core object in rough path theory that represents a time series through its iterated integrals, capturing both temporal order and interactions across feature dimensions (Lyons, 1998). Its first level records the path increment, while its second level encodes signed area terms; higher levels capture progressively richer geometric structure and higher-order sequential interactions along the path. Taken over all orders, the signature characterizes a path up to negligible time reparametrization, making it a faithful feature representation for sequential data (Levin et al., 2013). This property has enabled a wide range of applications in sequential data learning tasks, including handwritten recognition (Xie et al., 2017), writer identification (Yang et al., 2015), financial data analysis (Lyons et al., 2014), time-series data generation (Ni et al., 2021) and SAR (Liao et al., 2021; Cheng et al., 2023; Yang et al., 2022; Ahmad et al., 2019; Li et al., 2019a). Moreover, the signature is invariant to time reparameterization, meaning that traversing the same path at different speeds yields the same representation. This property is particularly desirable in action recognition, where the speed of performing an action should not affect its classification. In the specific

context of SAR, Ahmad et al. (2019) focus on employing path signature solely for extracting spatial features while Yang et al. (2022); Li et al. (2019a) utilize path signature to obtain both temporal and spatial features. However, relying solely on path signature to capture spatial relationships may be suboptimal, as it is not data adaptive and only considers local spatial correlations. Liao et al. (2021) leverage GCN to extract spatial features and employ the log signature to capture temporal dynamics. However, such approaches often suffer from the curse of dimensionality, as the feature dimension of the truncated signature grows rapidly with the path dimension, leading to overfitting in deep architectures.

More recently, the path development layer (Lou et al., 2024a) is proposed as a general sequential layer on time series data as a trainable alternative to the signature feature. Instead of computing a fixed set of features, path development maps the input path into a Lie group via a differential equation driven by the path. This results in a compact and learnable representation that retains key advantages of the signature, including sensitivity to temporal ordering and invariance to time reparameterization, while significantly reducing feature dimensionality. The development layer has demonstrated effectiveness in supervised learning on sequential data (Lou et al., 2024a), online signature verification (Shi et al., 2025) and synthetic time generation (Lou et al., 2024b). Path development originates from the (Cartan) development of a path (Driver, 1995), and plays an important role in understanding the properties of the path signature (Hambly & Lyons, 2010). In particular, Chevyrev & Lyons (2016) prove that under an appropriate Lie group, path development is a universal and characteristic feature of the path signature, which provides theoretical support for the development layer. Building upon their work (Lou et al., 2024a), we extend the application of path development from vector-valued time series to that of the temporal graph.

3 G-Dev Layer: Development layer of a temporal graph

In this section, we start with a concise overview of the development of the path as the principled feature representation of d-dimensional time series. Subsequently, we proceed with introducing the path development layer proposed in (Lou et al., 2024a). For details of the path development, we refer readers to (Lou et al., 2024a). Building on this, we propose the G-Dev layer, a novel extension of the development layer applied to temporal graphs, wherein the features of nodes are represented by time series.

3.1 The development of a Path

Before the formal definition, we first provide an intuitive explanation of path development. A time series can be viewed as a path evolving over time, and the goal is to construct a representation that captures its temporal dynamics, such as the order of events and how the signal changes, while being robust to variations like speed or sampling rate. Path development achieves this by mapping the input path into a trajectory on a Lie group. Informally, each local segment of the path is associated with a Lie group element, and the path development is obtained by taking the matrix product of these Lie group elements in sequential order. Since this product is order-sensitive, the resulting representation preserves information about the ordering of path segments, thereby encoding the history of the sequence as a matrix-valued representation. The path development is faithful in the sense that it preserves discriminative information about the shape of the underlying path, making it expressive for distinguishing time series with different geometric and sequential structures. Moreover, its invariance under time reparametrization means that the representation is unaffected by changes in traversal speed, and is therefore robust to variations in sampling rate and temporal resolution.

Let $X \in V([0,T],\mathbb{R}^d)$, where $V([0,T],\mathbb{R}^d)$ represents the space of continuous paths of finite length in $\mathbb{R}^d$ defined on the time interval $[0,T]$. To fix the notations, let G denote a finite-dimensional Lie group and its associate Lie algebra $\mathfrak{g}$. Examples of the matrix Lie algebra associated with the special orthogonal group (denoted by $\mathfrak{so}$) and special Euclidean group ($\mathfrak{se}$) are given as follows:

$$
\begin{aligned}
&\mathfrak{gl}(m;\mathbb{F}) = \{m \times m \text{ matrices over } \mathbb{F}\} \quad (\mathbb{F} = \mathbb{R} \text{ or } \mathbb{C});\\
&\mathfrak{so}(m,\mathbb{R}) := \{A \in \mathfrak{gl}(m;\mathbb{R}) : A^T + A = 0\};\\
&\mathfrak{se}(m) = \left\{ \begin{bmatrix} \omega & v \\ 0 & 0 \end{bmatrix} \mid \omega \in \mathfrak{so}(m,\mathbb{R}),\ v \in \mathbb{R}^m \right\}.
\end{aligned}
$$

Definition 3.1 (Path Development) *Let $M : \mathbb{R}^d \to \mathfrak{g} \subset \mathfrak{gl}(m)$ be a linear map. The path development of $X \in V([0,T], \mathbb{R}^d)$ on G under M is the solution $Z_T \in G$ to the below equation*

$$dZ_t = Z_t \cdot M(dX_t) \quad \text{for all } t \in [0,T] \quad \text{with } Z_0 = Id_m, \tag{1}$$

where Id_m is the identity matrix and $\cdot$ denotes the matrix multiplication.

Example 3.1 (Linear path) *Let $X \in V([0,T], \mathbb{R}^d)$ be a* linear *path. For any linear map $M \in L(\mathbb{R}^d, \mathfrak{g})$, the development of the path under M admits the explicit formula and is simply*

$$D_M(X)_{0,T} = \exp\left(M(X_T - X_0)\right), \tag{2}$$

where exp *is the matrix exponential.*

Thanks to the multiplicative property of the path development (Lemma 2.4 in (Lou et al., 2024a)), the development of two paths $X \in V([0,s], \mathbb{R}^d)$ and $Y \in V([s,t], \mathbb{R}^d)$ under M is the same as the development of the concatenation of these two paths, in formula,

$$D_M(X * Y) = D_M(X) \cdot D_M(Y), \tag{3}$$

where $X * Y$ denotes the concatenation of X and Y and $\cdot$ is the matrix multiplication. Eq. equation 3 enables us to compute the development of any piecewise linear path explicitly.

The path development enjoys several desirable theoretical properties, making it a principled and efficient representation of sequential data. We summarize the following two properties, that are directly relevant to action recognition.

Invariance under Time-reparametrization

Similar to the path signature, path development is proven to be invariant under time reparametrization; See Lemma 2.3 Lou et al. (2024a)). This means that the development of a path is unchanged by the speed at which the path is traversed. This invariance arises because the development is defined through integration along the path, which accumulates increments in order but does not depend on how quickly the path is traversed. As a result, only the sequence of values (i.e., the path itself), rather than its parameterization in time, determines the final representation. It is well suited to action recognition, as the prediction of actions should not depend on the speed of the actions or the frame rate. Time-reparametrization invariance of path development may bring massive dimension reduction and enhance the robustness to variation in speed.

Uniqueness of the path development Intuitively, the path development effectively captures the order of events in time series, leveraging the non-commutative nature of matrix multiplication. Changing the order of events may lead to different development in general, highlighting its capacity to capture the temporal dynamics of sequential data. It is proven in (Chevyrev & Lyons, 2016) the path development is a characteristic and universal feature of an un-parameterized path in $V_1([0,T], \mathbb{R}^d)$ under the appropriate Lie group. In other words, transforming a path of finite length to its development map $M \mapsto \mathrm{Dev}_M(X)$ can uniquely determine the path X up to translation. Thus, the path development combined with the starting point effectively summarizes the path without any loss of information.

3.2 Path Development Layer

Proposed in (Lou et al., 2024a), the path development layer is a novel neural network inspired by the path development by exploiting the representation of the matrix Lie group. More specifically, this layer utilizes the parameterized linear map $M \in L(\mathbb{R}^d, \mathfrak{g})$ by its linear coefficients $\theta \in \mathfrak{g}^d$.

Definition 3.2 (Path development layer) *The path development layer, denoted by D_θ, is defined as a mapping $\mathbb{R}^{d \times (N+1)} \to G$, which transforms any input time series $x = (x_1, x_2, ..., x_N)$ to the development $z_N \in G$ under the trainable linear map $M_\theta \in L(\mathbb{R}^d, \mathfrak{g})$. For $n \in \{0, ..., N-1\}$, we set*

$$z_{n+1} = z_n \exp(M_\theta(x_{n+1} - x_n)), z_0 = Id_m, \tag{4}$$

where $\theta \in \mathfrak{g}^d$ is the model parameter.

Eq. equation 4 is a direct consequence of Example 3.1 and the multiplicative property of the development (Eq. equation 3). Moreover, Eq. equation 4 shows the recurrence as an analogy to that of the RNNs, albeit in a much simpler manner, as depicted in Figure 1 (lower panel). Similar to RNNs, the path development can cope with time series input of variable length.

Dimension reduction The output of the path development has dimension m^2, where m is the matrix size. Note that its dimension does not depend on the path dimension d and time dimension T, resulting in the massive dimension reduction. This is in stark contrast to the signature feature of degree k, whose dimension $\left(\sum_{i=0}^{k} d^i = \frac{d^{k+1}-1}{d-1}\right)$ grows geometrically w.r.t d. Moreover, the path development is well suited to extract information from high-frequency time series or long time series without concern on the curse of dimensionality.

Robustness to irregular sampling As the path development can be interpreted as a solution to the differential equation (Eq. equation 1, it is a continuous time series model. Similar to other continuous time models such as signature and Neural CDEs (Kidger et al., 2020), the path development exhibits robustness in handling time series data with irregular sampling.

Computational Efficiency of analyzing online data Multiplicative property of the path development allows incremental computation of the development layer in streaming data, thereby reducing both computational load and memory requirements.

3.3 G-Dev Layer: Apply path development layer to a temporal graph

In this subsection, we propose the *G-Dev layer*, which extracts the local information of the temporal graph using the development layer in a rolling window fashion. As a sequence-based extension of the development layer, it generalizes the mapping from d-dimensional time series to temporal graph data. Concretely, the G-Dev layer operates over sliding windows along the time dimension, where the application of path development on each local window is referred to as the *G-Dev module*.

Throughout this paper, the temporal graph $\mathfrak{G}_X = (\mathcal{V}, \mathcal{E}, X)$ refers to the graph with each node associated with d-dimensional time series features X. Here $\mathcal{V} := \{v^1, \cdots, v^M\}$ denotes the vertex set and $\mathcal{E}$ can be represented by the adjacent matrix $A \in \mathbb{R}^{M\times M}$. We consider the general case where the time dimension of each node may vary across the nodes. More specifically, Let $X^i = (x_{t_1}, \cdots, x_{t_{L_i}}) \in \mathbb{R}^{d\times L_i}$ denote the features of the i^{th} vertex, where d and L_i are the path and time dimension, respectively. The feature sets $X = (X^{(1)}, \cdots, X^{(M)})$ is obtained by stacking X^i for all $i \in \{1, \cdots, M\}$. Clearly, the skeleton sequence is an example of temporal graph data.

G-Dev module. Let us first define the G-Dev module, which is a trainable layer with model parameters $\theta \in \mathfrak{g}^d \subset \mathbb{R}^{m^2\times d}$, where m is the matrix order of development. It maps a temporal graph $\mathfrak{G}_X = (\mathcal{V}, \mathcal{E}, X)$ to a static graph with matrix-valued features $\mathfrak{G}_Z = (\mathcal{V}, \mathcal{E}, Z)$ by applying the path development to the time-augmented transformation of each feature vector X_i of the vertex v_i, as shown in Figure 1. In formula, for each vertex index $i \in \{1, \cdots, M\}$, the output feature is $Z = (Z^{(i)})_{i=1}^M$, where $Z^{(i)} = D_\theta(X^{(i)}) \in G \subset \mathbb{R}^{m\times m}$.

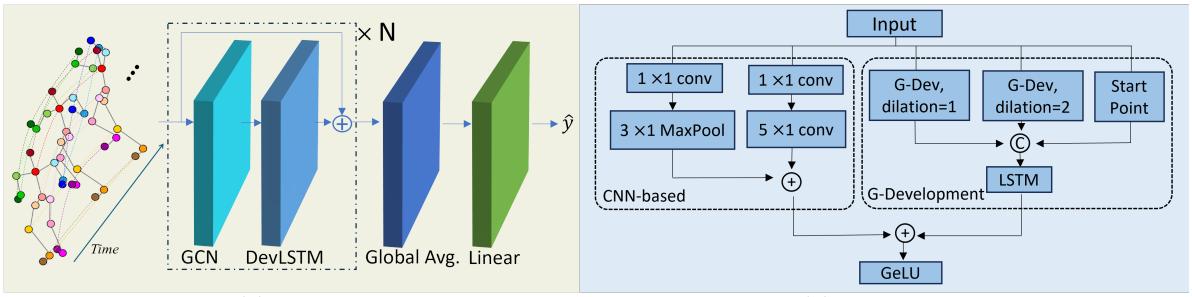
Built on top of the G-Dev module, we employ a sliding window strategy to construct the following G-Dev layer, which can summarize (potentially long) time series over coarse time partitions to reduce the time dimension effectively while retaining the fine-grained information by using the low-dimensional representation.

Without loss of generality, let us focus on the case where the graph features X share the same time stamps $(t_i)_{i=0}^{L-1}$ across vertices[2]. Fix the window (kernel) size K and stride S. Let $L' = \lceil \frac{L-k}{S} \rceil$. By applying padding to Y, for any $Y \in \mathbb{R}^{d\times N}$, we define a sequence of time series $\tilde{Y}_j$:

$$\tilde{Y}_j = (Y_{j\times S}, \cdots Y_{j\times S+K-1}) \in \mathbb{R}^{d\times K}, \forall j \in \{0, \cdots, L'-1\}. \tag{5}$$

Definition 3.3 (G-Dev layer) *A G-Dev layer with the matrix degree m, window size K and stride S is defined as a trainable map that takes a temporal graph $\mathcal{G}_X = (\mathcal{V}, \mathcal{E}, X)$ with $X \in \mathbb{R}^{M\times L'\times d}$ as an input and*

[2]Otherwise, we apply linear interpolation for data imputation, which would not affect the development feature thanks to invariance to time re-parameterization.



(a) Pipeline (b) DevLSTM Module

Figure 2: (a) The pipeline of our proposed approach consists of N blocks, with each block containing a GCN module and a DevLSTM module. (b) The detail of the DevLSTM module.

outputs a temporal graph $\mathcal{G}_Z = (\mathcal{V}, \mathcal{E}, Z)$ *with model parameter* $\theta \in \mathbb{R}^{d\times m\times m}$. *Here* $Z = (z_j)_{j=1}^{L'}$ *is a time series consisting of* $z_j \in \mathbb{R}^{M\times m\times m}$. *Each* $z_j = G\text{-}Dev_\theta(\tilde{X}_j)$, *where* $\tilde{X}_j$ *is obtained by applying sliding window by Eqn. 5.*

The G-Dev layer (Definition 3.3) can be easily extended to the moving window with the dilation parameter D, where we collect the path information at every D time steps during each moving window. To guarantee no information loss of the G-Dev layer, it is crucial to choose S and K to ensure a minimum one-point overlap between two consecutive moving windows.

4 GCN-DevLSTM Network for SAR

A skeleton sequence can be modeled by a temporal graph, where the vertex set corresponds to M possible joints, with each $v^i \in \mathbb{R}^2$ or $\mathbb{R}^3$ indicating 2 or $3D$ coordinates of the i^{th} joint, respectively. The edge set $\mathcal{E}$ denotes the set of all the joint pairs connected by a bone.

In Figure 2(a), we introduce the proposed GCN-DevLSTM Network, a novel framework for SAR tasks that integrates the GCN and DevLSTM modules into a single block. The GCN module is designed to detect spatial correlations among vertices, while the DevLSTM module handles the temporal dynamics across video frames. Further details on these modules are discussed in Sections 4.1 and 4.2, respectively. Our network is designed to capture discriminative spatial-temporal features with N such blocks, where $N = 9$ in this paper. Additionally, within each block, we build a residual path that establishes a shortcut connection with the input to mitigate gradient vanishing/ explosion issues, enhancing the training stability.

Following these GCN-DevLSTM blocks, we employ global average pooling to effectively reduce both temporal and spatial dimensions to 1. After that, a linear function is applied to adjust the feature dimension to match the number of action classes, thereby facilitating the classification process. Cross-entropy loss is employed at the end to compare the predicted action class with the ground truth.

4.1 GCN Module

In the GCN module, we basically follow the network of CTR-GC (Chen et al., 2021). However, we depart from the use of three initial adjacency matrices representing self, inward, and outward correlations. Instead, we propose a Hops-CTRGC to enhance the model by incorporating various hops of information to construct three adjacency matrices denoted as A_0, A_1, and A_2.

A_0 remains an identical (self) matrix. For A_1, the elements a_{ij} are set to 1 if vertex i and vertex j are physically connected in the skeleton graph; otherwise, a_{ij} in A_1 is set to 0. A_1 captures the direct connections in the graph. To enrich spatial information, we introduce $A_2 = A_1^2$, enabling a vertex to connect with its neighbor's neighbor. A_2 allows us to extend our understanding beyond direct connections to include relationships involving two hops in the graph. Since GCN module is not our main focus in this paper, readers can find more details about the network of CTR-GC in (Chen et al., 2021) or in our appendix **??**.

4.2 DevLSTM Module

The proposed DevLSTM module, illustrated in Figure 2(b) is mainly composed of the CNN-based branch and the G-development-based branch. The output of DevLSTM module is given by applying GeLU activation function to the summation of the output of both branches.

G-Development branch. Inspired by multi-scale TCN (Liu et al., 2020), we also design a multi-scale network to capture temporal dynamics. We utilize two G-dev layers with the dilation parameter $D = 1$ and $D = 2$ to enhance its ability to capture temporal patterns across different scales in the input sequence.

The path development combined with the starting point can determine the path uniquely. Hence, we append the starting point location to the output of the path development layer along the feature dimension over each sub-time interval. The resulting features are then fed into an LSTM network to aggregate local development summaries and capture long-term dependencies in time series data. We use the sequential output of LSTM, preserving the same time dimension as the input.

CNN-based branch. MaxPool and a convolutional layer with 5×1 kernel size have consistently demonstrated superior performance within the temporal module of other GCN-based studies. Introducing both MaxPool and the 5×1 convolutional operation to our DevLSTM temporal module has the potential to enhance model performance even further. We can reasonably design kernel size, padding, stride and dilation to ensure the output size of Maxpool, LSTM and 5×1 convolution remain the same.

5 Numerical Experiments

To validate the effectiveness of the proposed approach, we carry out numerical experiments on the following popular benchmarking datasets for skeleton-based action and gesture recognition. These datasets are chosen for their diversity in data size and task complexity, providing a comprehensive evaluation of the model's performance. (1) **Chalearn2013** (Escalera et al., 2013) is a public dataset for gesture recognition, consisting of 11,116 skeleton examples and 20 gesture categories, filmed by 27 people. Each frame has 20 joints. (2) **NTU RGB+D 60** (NTU-60) (Shahroudy et al., 2016) consists of 56,880 action samples and 60 action classes, filmed by 40 people using three Kinect cameras in different camera views. 3D skeletal data in the dataset have different sequence lengths, with each owning 25 joints. (3) **NTU RGB+D 120** (NTU-120) (Liu et al., 2019) extends upon NTU RGB+D 60, consisting of 114,480 samples and 120 action classes, filmed by 106 people in three camera views. We consider the Cross-subject (X-Sub) and Cross-view (X-View) evaluation benchmarks for both NTU-60 and NTU-120 as proposed by Shahroudy et al. (2016). We use only skeleton data from these datasets for our experiments.

Experiments are conducted using a single Nvidia A6000 graphics card, where a batch of 16 examples is fed to the network. Stochastic gradient descent optimiser (Bottou, 2012) is employed, with a Nesterov momentum = 0.9 and a weight decay = 0.0003. The training epoch is set to 65 and we apply the warm-up strategy (He et al., 2016) to the first 5 epochs. The initial learning rate is 0.02 and a step learning rate decay is applied with a factor of 0.1 at epoch 35 and 55, respectively. Data processing on NTU datasets mainly follows the procedure in (Chen et al., 2021), with the length of all input data sequences resizing to 64. We use $\mathfrak{se}$ Lie algebra in this paper, with a corresponding matrix size of $m = 10$. A detailed network architecture can be found in the supplementary. The replicate padding is used in our experiments.

5.1 Fusion results from different data streams

Most of the recent approaches that we are set to compare combine predicted scores from different input data streams for enhanced performance. The most commonly used data streams are joint (J), bone (B), joint motion (JM) and bone motion (BM) where details can be found in (Shi et al., 2019; 2020). Apart from the above-mentioned data streams, we introduce a novel stream: the line graph. In a line graph, vertices represent bones, and edges establish connections between these bones (Harary & Norman, 1960). More details about line graph can be found at Appendix **??**.

5.2 Comparison with state-of-the-art methods

Table 1: Results on Chalearn2013 dataset.

	Acc (%)
Two-streamLSTM (Wang & Wang, 2017)	91.70
ST-LSTM + Trust Gate (Liu et al., 2018)	92.00
Multi-path CNN (Liao et al., 2019)	93.13
3s_net_TTM (Li et al., 2019a)	92.08
GCN-Logsig-RNN (Liao et al., 2021)	92.86
ST-PSM+L-PSM (Cheng et al., 2023)	94.18
AGGP (Shi et al., 2024)	95.18
Our method	**95.61**

Table 1 shows the comparison results between our result and other state-of-the-art methods on Chalearn2013 dataset. We only employ the joint stream for fair comparison as other approaches only use the joint representation on Chalearn2013 dataset. It is evident that our model yields the best result on the small-scale gesture dataset, surpassing the second-best method by 0.42%, proving its effectiveness on the small dataset.

Table 2: Experimental results on NTU-60 and NTU-120 dataset. The best results for each benchmark are bolded.

						NTU-60		NTU-120	
						X-sub (%)	X-view (%)	X-sub (%)	X-view (%)
ST-GCN (Yan et al., 2018)						81.5	88.3	70.7	73.2
AGC-LSTM (Si et al., 2019)						89.2	95.0	-	-
2S-AGCN (Shi et al., 2019)						88.5	95.1	82.5	84.2
Shift-GCN (Cheng et al., 2020b)						90.7	96.5	85.9	87.6
Logsig-RNN (Liao et al., 2021)						-	-	75.8	78.0
CTR-GCN(J) (Chen et al., 2021)						89.8	94.8	84.9	86.7
CTR-GCN(J+B) (Chen et al., 2021)						92.0	96.3	88.7	90.1
CTR-GCN (Chen et al., 2021)						92.4	96.8	88.9	90.6
EfficientGCN (Song et al., 2022)						92.1	96.1	88.7	88.9
Ta-CNN (Xu et al., 2022)						90.4	94.8	85.4	86.8
FR-Head (Zhou et al., 2023)						92.8	96.8	89.5	90.9
SPIANet (Yin et al., 2024)						92.8	96.8	89.2	90.4
BlockGCN(J) (Zhou et al., 2024)						90.9	95.4	86.9	88.2
BlockGCN (Zhou et al., 2024)						**93.1**	**97.0**	**90.3**	**91.5**
FD-GCN (Huo et al., 2026)						92.8	96.7	89.4	90.7
	Joint	Bone	Line	Joint motion	Bone motion				
Our methods	✓					90.8	95.7	86.2	87.7
		✓				91.1	95.2	87.0	88.6
			✓			91.2	95.3	87.1	88.7
	✓	✓				92.4	96.4	89.0	90.4
	✓	✓		✓	✓	92.6	96.6	89.3	90.7
	✓	✓	✓	✓	✓	92.9	96.7	89.7	91.2

We compare our approach with other state-of-the-art methods on both NTU-60 and NTU-120 datasets, as shown in Table 2. Since our GCN module is following CTR-GCN (Chen et al., 2021) with the major difference lying in the temporal module, we show more results on CTR-GCN for a more comprehensive comparison. As shown in Table 2, all of our configurations, whether using only the joint stream or multi-stream fusion (e.g., joint and bone), consistently outperform CTR-GCN across all benchmarks. Our method's relatively limited performance gain from additional motion streams, compared to CTR-GCN, may stem from the path development layer focusing on position differences between consecutive time steps. As a result, motion streams provide overlapping information at early stages of the network, limiting their contribution. Compared to more recent methods such as BlockGCN (Zhou et al., 2024), which achieves stronger performance when using multi-stream inputs, our method attains comparable results under the joint-stream setting on NTU-60 and remains competitive on the more challenging NTU-120 benchmark.

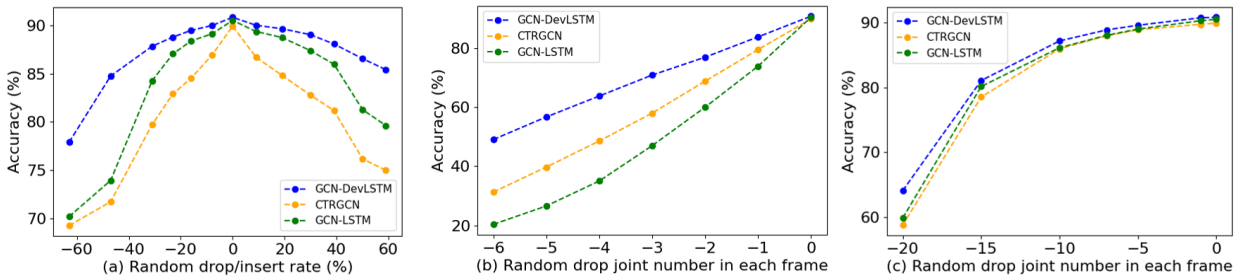


Figure 3: Robustness analysis on NTU60 X-sub benchmark. Figure 3a shows the results of randomly dropping or inserting a certain frames to original time series. Figure 3b and 3c shows the results of randomly dropping a number of joints in each frame with zero-filling and linear interpolation respectively.

Overall, our model, integrating multiple data streams, achieves competitive performance across benchmarks while maintaining a consistent network structure across datasets, demonstrating strong adaptability to different dataset sizes. Moreover, it is worth noting that methods such as BlockGCN primarily focus on improving spatial modeling, whereas our work targets temporal modeling, making the two approaches complementary.

5.3 Comparison with signature-based models

Since path development layer can be seen as a data-adaptive alternative to the signature, we further compare our method with other signature-based models in SAR task. Compared to the Logsig-RNN (Liao et al., 2021), one of the recent signature-based action recognition methods (using the joint stream only), our model (J) outperforms 10.4% and 9.7% respectively on both benchmarks of NTU-120 dataset, and 2.75% on Chalearn2013 dataset. Compared to the latest signature-based action recognition approach (Cheng et al., 2023), we still outperform 1.43% on Chalearn2013 dataset while unfortunately, they didn't report their results on NTU dataset, leaving no way to compare. These results support the advantages of using path development that we claimed before, compared with path signature.

5.4 Robustness analysis

We comprehensively compare the robustness of our full model (GCN-DevLSTM), CTRGCN (Chen et al., 2021) and our model without path development layer (GCN-LSTM) in terms of missing frames, missing joints and variable sequence length. To this end, we firstly create a new test dataset by randomly dropping or inserting a certain number of frames out of 64 frames to test the robustness of missing frames and variable length. Figure 3a illustrates that our full model with the path development layer consistently outperforms the other methods in terms of accuracy, regardless of the proportion of frames missing or inserting, proving its robustness to irregular sampling and variable sequence length. Notably, at a 47% frame-drop rate, the accuracy of our full model decreases by only 6.07%, while CTR-GCN drops by 18.18% and GCN-LSTM by 16.56%. A similar trend is observed when frames are inserted.

Additionally, to validate the robustness of missing joints due to occlusions in real life, we create another dataset by randomly dropping a certain number of joints in each frame. To fill in these missing joints, two techniques are adopted: zero-filling (in Figure 3b) and linear interpolation (in Figure 3c). Both figures show that our model with G-Dev layer, is consistently more robust against missing joints than others. More details are in the supplementary material. We credit the superior robustness of our full model to the theory foundation provided by the path development layer, mentioned in 3.2.

5.5 Comparison with MS-TCN

To demonstrate the flexible integration of our DevLSTM temporal module with various GCN modules in a plug-and-play fashion, we employ the adaptive graph convolution (AGC) graph in 2S-AGCN (Shi et al., 2019) and a fixed graph without learnable parameters, respectively as an additional GCN module backbone

in our network. In addition, we compare our DevLSTM with multi-scale temporal convolution network (MS-TCN) (Chen et al., 2021), commonly used as the temporal module in other recent works across the different GCN backbones. Table 3 illustrates that the proposed DevLSTM module consistently outperforms MS-TCN with different GCN backbones by a large margin, indicating that our designed temporal module is effective in enhancing temporal modeling across various GCN backbones. An overview of adaptive graph convolution and fixed graph can be found in the supplementary material.

Table 3: Comparison between DevLSTM and MS-TCN across various GCN modules on the NTU60 dataset.

GCN Module	Temporal module	X-sub (%)	X-view (%)
CTR-GC	MS-TCN (J)	90.3	94.9
	DevLSTM (J)	**90.8**	**95.7**
AGC	MS-TCN (J)	89.8	94.5
	DevLSTM (J)	**90.6**	**95.6**
Fixed graph	MS-TCN (J)	88.7	93.1
	DevLSTM (J)	**89.5**	**94.5**

5.6 Ablation studies

Table 4: Ablation studies of DevLSTM module on NTU-60 dataset. w/o means without.

Components	X-sub (%)	X-view (%)
with all components (J)	90.8	95.7
w/o residual path (J)	89.9	95.2
w/o 1× 1 conv & 5×1 conv (J)	90.7	95.3
w/o 1× 1 conv & 3× 1 maxpool (J)	90.6	95.4
w/o G-Dev (J)	90.5	95.2

We further investigate the importance of residual path design to connect the input and output of each block and each component in our temporal module. As evident from Table 4, residual path plays an important role to enhance the model performance since it can help alleviate gradient vanishing/explosion in the deep neural network. For temporal components in our DevLSTM module, the path development layer emerges as the primary contributor to the overall performance improvement since removing it causes the largest performance degradation, with other components also contributing to enhanced performance. Importantly, the improvement brought by the G-Dev layer is consistent across both X-sub (+0.3%) and X-view (+0.5%) settings, suggesting that it provides a stable and reliable enhancement rather than a dataset-specific gain.

As further demonstrated in the robustness analysis (Section 5.4), the advantage of the G-Dev layer becomes significantly more pronounced under challenging conditions such as missing frames, missing joints, and variable sequence lengths. This indicates that the primary contribution of the G-Dev layer lies in improving robustness and temporal consistency, rather than only optimizing standard benchmark accuracy.

Table 5 further shows that the G-Dev layer consistently improves performance across different numbers of blocks, with more pronounced gains in smaller models (e.g., a 3.3% improvement with a single block). As model capacity increases and performance approaches saturation, the marginal gains naturally diminish. Nevertheless, the improvement remains consistent across all configurations.

Table 5: Effect of the G-Dev layer across different numbers of blocks on the NTU-60 X-view dataset. Each block consists of one GCN module followed by one DevLSTM module. All values are reported in %.

Number of blocks	1	2	3	4	5	6	7	8	9
With G-Dev layer	89.1	92.7	93.7	94.2	95.0	95.1	95.4	95.5	95.7
w/o G-Dev layer	85.8	90.8	92.3	93.5	94.2	94.4	94.7	94.9	95.2
Performance drop	3.3	1.9	1.4	0.7	0.8	0.7	0.7	0.6	0.5

5.7 Model complexity analysis

Table 6 demonstrates the trade-off between accuracy and model complexity for compact GCN-DevLSTM networks with different numbers of blocks. Increasing the number of blocks from 1 to 9 improves the accuracy from 89.1% to 95.7%, while increasing the number of parameters from 0.13M to 4.61M. However, even with fewer blocks, our results are competitive with much less model parameters in comparison with other stoa methods. As illustrated in Table 6, adding more blocks to our networks can further improve accuracy, albeit with increased parameter number and computational complexity. It is worthy noting that our model with 5 blocks only has 68% of model parameters of the strong baseline CTRGCN, reducing 0.49 million parameters (from 1.46M to 0.98M) while slightly outperforming it. Besides, our model with a single block surpasses ST-GCN, despite the differences in preprocessing or training strategies between ST-GCN and our method. The slower inference speed compared to CTR-GCN and 2s-AGCN mainly stems from the sequential nature of LSTM and the current implementation of the G-Dev layer. Specifically, LSTM limits temporal parallelization due to recurrent hidden-state updates, while the G-Dev layer requires matrix exponential computations that currently lack optimized CUDA kernel support in common deep learning frameworks. Therefore, the additional runtime overhead is largely implementation-related rather than an intrinsic limitation of path development, and may decrease as GPU support for Lie group operations improves.

Table 6: Model complexity analysis. Compare the model parameters between our approach with the corresponding accuracy on NTU-60 X-view dataset using joint stream only.

	ST-GCN (Yan et al., 2018)	2s-AGCN (Shi et al., 2019)	CTRGCN (Chen et al., 2021)	**Ours**								
Blocks	-	-	-	9	8	7	6	5	4	3	2	1
Params (M)	1.22	1.55	1.46	4.61	3.29	1.78	1.38	0.98	0.53	0.40	0.26	0.13
Acc (%)	88.3	94.0	94.8	95.7	95.5	95.4	95.1	95.0	94.2	93.7	92.7	89.1
Inference (ms/sample)	-	1.7	1.7	11.2	10.2	8.9	8.3	7.1	5.4	4.1	2.8	1.4

5.8 Lie Group & Hidden Size Selection

Table 7: Evaluation of accuracy (%) across different hidden sizes in special Euclidean (SE) and special orthogonal (SO) Lie group on NTU60-Xview benchmark with the joint stream.

	8	9	10	11	12
SE	95.6	95.4	95.7	95.5	95.5
SO	95.5	95.5	95.4	95.6	95.4

Given that path development is key to our methodology, the choice of a suitable Lie Group, along with its corresponding hidden size, influences the ultimate performance of the model. Table 7 presents results for two commonly used Lie groups with varying hidden sizes. The results indicate that a larger hidden size does not necessarily ensure an enhancement in the model and the best result is obtained by utilizing the special Euclidean group with a hidden size of 10. These important hyperparameters should be tuned based on numerical results in practice.

6 Conclusion and future work

In summary, we propose a novel and generic GCN-DevLSTM network for analyzing temporal graph data via the path development layer. This approach not only offers flexibility when combined with various GCN modules but also significantly enhances performance by effectively capturing the temporal dependencies within data. The effectiveness of the proposed method is validated by the competitive results on three datasets in SAR. Moreover, it demonstrates strong robustness to missing data and variable frame rates.

The current G-Dev layer relies on other GCNs to extract spatial information, which limits its standalone utility. Future enhancements could include integrating the features of adjacent nodes directly into the input of the path development layer to better capture spatio-temporal dependencies.

Our proposed G-Dev layer provides a generic model for temporal graph data, opening up possibilities for its applications across various fields, including urban planning, social network analysis and others.

References

Tasweer Ahmad, Lianwen Jin, Jialuo Feng, and Guozhi Tang. Human action recognition in unconstrained trimmed videos using residual attention network and joints path signature. *IEEE Access*, 7:121212–121222, 2019.

Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade: Second Edition*, pp. 421–436. Springer, 2012.

Dong Chen, Mingdong Chen, Peisong Wu, Mengtao Wu, Tao Zhang, and Chuanqi Li. Two-stream spatio-temporal gcn-transformer networks for skeleton-based action recognition. *Scientific Reports*, 15(1):4982, 2025.

Yuxin Chen, Ziqi Zhang, Chunfeng Yuan, Bing Li, Ying Deng, and Weiming Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *IEEE International Conference on Computer Vision*, pp. 13359–13368, 2021.

Jiale Cheng, Dongzi Shi, Chenyang Li, Yu Li, Hao Ni, Lianwen Jin, and Xin Zhang. Skeleton-based gesture recognition with learnable paths and signature features. *IEEE Transactions on Multimedia*, 2023.

Ke Cheng, Yifan Zhang, Congqi Cao, Lei Shi, Jian Cheng, and Hanqing Lu. Decoupling gcn with dropgraph module for skeleton-based action recognition. In *European Conference on Computer Vision*, pp. 536–553, 2020a.

Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 183–192, 2020b.

Ilya Chevyrev and Terry J Lyons. Characteristic functions of measures on geometric rough paths. *Annals of probability: An official journal of the Institute of Mathematical Statistics*, 44(6):4049–4082, 2016.

Jeonghyeok Do and Munchurl Kim. Skateformer: skeletal-temporal transformer for human action recognition. In *European Conference on Computer Vision*, pp. 401–420. Springer, 2024.

BRUCE K Driver. A primer on riemannian geometry and stochastic analysis on path spaces. 1995.

Sergio Escalera, Jordi Gonzàlez, Xavier Baró, Miguel Reyes, Isabelle Guyon, Vassilis Athitsos, Hugo Escalante, Leonid Sigal, Antonis Argyros, Cristian Sminchisescu, et al. Chalearn multi-modal gesture recognition 2013: grand challenge and workshop summary. In *15th ACM on International conference on multimodal interaction*, pp. 365–368, 2013.

Ben Hambly and Terry Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics*, pp. 109–167, 2010.

Frank Harary and Robert Z Norman. Some properties of line digraphs. *Rendiconti del circolo matematico di palermo*, 9(2):161–168, 1960.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

Jinze Huo, Haibin Cai, and Qinggang Meng. Fast distance-enhanced graph convolutional network for skeleton-based action recognition. *Pattern Recognition*, 169:111827, 2026.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707, 2020.

Ugur Kilic, Ozge Oztimur Karadag, and Gulsah Tumuklu Ozyer. Agms-gcn: Attention-guided multi-scale graph convolutional networks for skeleton-based action recognition. *Knowledge-Based Systems*, 311:113045, 2025.

Rohit Lal, Saketh Bachu, Yash Garg, Arindam Dutta, Calvin-Khang Ta, Hannah Dela Cruz, Dripta S Raychaudhuri, M Salman Asif, and Amit Roy-Chowdhury. Stride: Single-video based temporally continuous occlusion-robust 3d pose estimation. In *Proceedings of the Winter Conference on Applications of Computer Vision*, pp. 794–803, 2025.

Daniel Levin, Terry Lyons, and Hao Ni. Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv preprint arXiv:1309.0260*, 2013.

Chenyang Li, Xin Zhang, Lufan Liao, Lianwen Jin, and Weixin Yang. Skeleton-based gesture recognition using several fully connected layers with path signature features and temporal transformer module. In *AAAI Conference on Artificial Intelligence*, volume 33, pp. 8585–8593, 2019a.

Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3595–3603, 2019b.

Tianchen Li, Pei Geng, Xuequan Lu, Wanqing Li, and Lei Lyu. Skeleton-based action recognition through attention guided heterogeneous graph neural network. *Knowledge-Based Systems*, 309:112868, 2025.

Lufan Liao, Xin Zhang, and Chenyang Li. Multi-path convolutional neural network based on rectangular kernel with path signature features for gesture recognition. In *IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4. IEEE, 2019.

Shujian Liao, Terry Lyons, Weixin Yang, Kevin Schlegel, and Hao Ni. Logsig-rnn: a novel network for robust and efficient skeleton-based action recognition. *British Machine Vision Conference*, 2021.

Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot. Global context-aware attention lstm networks for 3d action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1647–1656, 2017a.

Jun Liu, Amir Shahroudy, Dong Xu, Alex C Kot, and Gang Wang. Skeleton-based action recognition using spatio-temporal lstm network with trust gates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):3007–3021, 2018.

Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C Kot. Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2684–2701, 2019.

Mengyuan Liu, Hong Liu, and Chen Chen. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition*, 68:346–362, 2017b.

Ruyi Liu, Yu Chen, Feiyu Gai, Yi Liu, Qiguang Miao, and Shuai Wu. Local and global spatial–temporal transformer for skeleton-based action recognition. *Neurocomputing*, 634:129820, 2025.

Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 143–152, 2020.

Hang Lou, Siran Li, and Hao Ni. Path development network with finite-dimensional lie group. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL `https://openreview.net/forum?id=YoWBLu74TL`.

Hang Lou, Siran Li, and Hao Ni. Pcf-gan: generating sequential data via the characteristic function of measures on the path space. *Advances in Neural Information Processing Systems*, 36, 2024b.

Terry Lyons, Hao Ni, and Harald Oberhauser. A feature set for streams and an application to high-frequency financial tick data. In *2014 International Conference on Big Data Science and Computing*, pp. 1–8, 2014.

Terry J Lyons. Differential equations driven by rough signals. *Revista Matemática Iberoamericana*, 14(2): 215–310, 1998.

Hao Ni, Lukasz Szpruch, Marc Sabate-Vidales, Baoren Xiao, Magnus Wiese, and Shujian Liao. Sig-wasserstein gans for time series generation. In *Second ACM International Conference on AI in Finance*, pp. 1–8, 2021.

Kevin Schlegel, Lei Jiang, and Hao Ni. Using joint angles based on the international biomechanical standards for human action recognition and related tasks. *arXiv preprint arXiv:2406.17443*, 2024.

Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1010–1019, 2016.

Dongzi Shi, Xin Zhang, Jiale Cheng, Tong Xiong, and Hao Ni. Adaptive global gesture paths and signature features for skeleton-based gesture recognition. In *International Conference on Pattern Recognition*, pp. 278–292. Springer, 2024.

Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12026–12035, 2019.

Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with multi-stream adaptive graph convolutional networks. *IEEE Transactions on Image Processing*, 29:9532–9545, 2020.

Yilin Shi, Lei Jiang, Hao Ni, and Lianwen Jin. Devinsight: Weaving path development into online signature verification. In *International Conference on Document Analysis and Recognition*, pp. 398–415. Springer, 2025.

Chenyang Si, Wentao Chen, Wei Wang, Liang Wang, and Tieniu Tan. An attention enhanced graph convolutional lstm network for skeleton-based action recognition. In *IEEE International Conference on Computer Vision*, pp. 1227–1236, 2019.

Yi-Fan Song, Zhang Zhang, Caifeng Shan, and Liang Wang. Constructing stronger and faster baselines for skeleton-based action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (2):1474–1488, 2022.

Chuanchuan Wang and Ahmad Sufril Azlan Mohamed. Deep learning for 3d skeleton-based action recognition: a comprehensive review of methods, datasets, and future directions. *International Journal of Machine Learning and Cybernetics*, 17(3):115, 2026.

Hongsong Wang and Liang Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 499–508, 2017.

Zecheng Xie, Zenghui Sun, Lianwen Jin, Hao Ni, and Terry Lyons. Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(8):1903–1917, 2017.

Kailin Xu, Fanfan Ye, Qiaoyong Zhong, and Di Xie. Topology-aware convolutional neural network for efficient skeleton-based action recognition. In *AAAI Conference on Artificial Intelligence*, volume 36, pp. 2866–2874, 2022.

Jiexing Yan, Xi Zhang, Caiyan Tan, and Dawen Li. Skelformer: An adaptive hierarchical transformer-based approach on skeleton graphs for human action recognition in video sequences. *PloS one*, 21(1):e0340390, 2026.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Weixin Yang, Lianwen Jin, and Manfei Liu. Chinese character-level writer identification using path signature feature, dropstroke and deep cnn. In *13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 546–550. IEEE, 2015.

Weixin Yang, Terry Lyons, Hao Ni, Cordelia Schmid, and Lianwen Jin. Developing the path signature methodology and its application to landmark-based human action recognition. In *Stochastic Analysis, Filtering, and Stochastic Optimization: A Commemorative Volume to Honor Mark HA Davis's Contributions*, pp. 431–464. Springer, 2022.

Xinpeng Yin, Jianqi Zhong, Deliang Lian, and Wenming Cao. Spatiotemporal progressive inward-outward aggregation network for skeleton-based action recognition. *Pattern Recognition*, 150:110262, 2024.

Huanyu Zhou, Qingjie Liu, and Yunhong Wang. Learning discriminative representations for skeleton based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10608–10617, 2023.

Yuxuan Zhou, Xudong Yan, Zhi-Qi Cheng, Yan Yan, Qi Dai, and Xian-Sheng Hua. Blockgcn: Redefine topology awareness for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2049–2058, 2024.