Ske2Grid: Skeleton-to-Grid Representation Learning for Action Recognition

Dongqi Cai [1] **Yangyuxuan Kang** [1] **Anbang Yao** [1] **Yurong Chen** [1]

Abstract

This paper presents Ske2Grid, a new representation learning framework for improved skeleton-based action recognition. In Ske2Grid, we define a regular convolution operation upon a novel grid representation of human skeleton, which is a compact image-like grid patch constructed and learned through three novel designs. Specifically, we propose a graph-node index transform (GIT) to construct a regular grid patch through assigning the nodes in the skeleton graph one by one to the desired grid cells. To ensure that GIT is a bijection and enrich the expressiveness of the grid representation, an up-sampling transform (UPT) is learned to interpolate the skeleton graph nodes for filling the grid patch to the full. To resolve the problem when the one-step UPT is aggressive and further exploit the representation capability of the grid patch with increasing spatial size, a progressive learning strategy (PLS) is proposed which decouples the UPT into multiple steps and aligns them to multiple paired GITs through a compact cascaded design learned progressively. We construct networks upon prevailing graph convolution networks and conduct experiments on six mainstream skeleton-based action recognition datasets. Experiments show that our Ske2Grid significantly outperforms existing GCN-based solutions under different benchmark settings, without bells and whistles. Code and models are available at https://github.com/OSVAI/Ske2Grid.

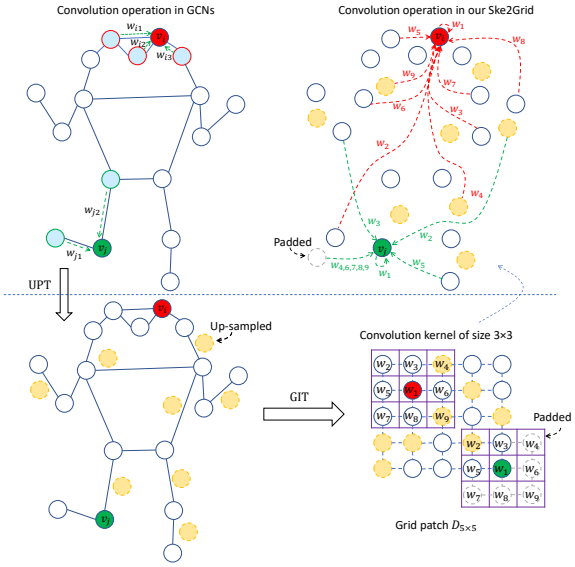

Figure 1. Comparison of convolution operations in GCNs and in our Ske2Grid. Graph convolution (top-left) typically convolves every node with their neighboring nodes using specific kernels. In Ske2Grid, we construct a regular grid patch for skeleton representation via up-sampling transform (UPT) and graph-node index transform (GIT). Convolution operation upon this grid patch convolves every grid cell using a shared regular kernel. It operates on a set of grid cells within a squared sub-patch which may be filled by a set of nodes distributed remotely on the graph, achieving a learnable receptive field on the skeleton for action feature modeling (top-right). In the figure, the up-sampled skeleton graph is visualized assuming the locations of the original graph nodes being unchanged for a better illustration.

1. Introduction

With the rapid development of 3D motion capturing systems and advanced real-time 2D/3D pose estimation algorithms, skeleton-based action recognition has attracted increasing attention from both industry and academia. The performance of a skeleton-based action recognition system depends on how well it can model the discriminative skeleton feature interactions among the active coordinated human joints when performing an action. Traditional methods (Ke et al., 2017; Hu et al., 2019; Qin et al., 2021) focus on designing hand-crafted features to model spatial structure and temporal dynamics of skeleton sequences, and use well-designed classifiers to recognize human actions. Early deep learning based methods (Du et al., 2015; Song et al., 2017; Li et al., 2018a) consider skeletons in videos as temporal vector sequences and use deep recurrent networks such as RNN and LSTM to model the skeleton dynamic-

[1]Intel Labs China. Correspondence to: Anbang Yao <anbang.yao@intel.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

s. In recent years, convolutional neural networks (CNNs) and graph convolution networks (GCNs) have become the mainstream learning models in skeleton-based action recognition research. CNN-based methods (Liu et al., 2017; Yan et al., 2019; Duan et al., 2022c) usually convert skeleton to image-like input and use 2D or 3D CNNs for end-to-end feature extraction. GCN-based methods (Yan et al., 2018; Li et al., 2019; Peng et al., 2020) treat skeleton as an irregular graph and learn to aggregate the skeleton features in terms of the pre-defined topology of graph. More recently, hypergraph-based models, such as Hao et al. (2021), also adopt a skeleton graph structure. Unlike conventional graph-based models that usually ignore non-physical dependencies among joints of the human body, hypergraph-based models address this gap by introducing local and global hyperedges to encode higher-order feature dependencies. Besides, newly emerging transformer-based models, such as Zhang et al. (2021), Plizzari et al. (2021) and Zhou et al. (2022), still retain a skeleton graph structure and treat each joint of the human body as a token, and then use spatial and temporal self-attention operators to capture feature dependencies.

Despite the prevalence of CNNs and GCNs based solutions, improved representation learning for skeleton-based action recognition is still a challenging problem. On the one side, CNN-based solutions benefit from the regular convolution for efficient feature modeling. However, they are restricted to image-like input, to which the conversion from skeleton typically ignores the critical topological information of human joints and would sacrifice the compactness of skeleton data to some extent. For example, Luvizon et al. (2018) directly encodes the temporal dimension, the joints and the coordinates as different axis of the image-like tensor. In the recent work of Duan et al. (2022c), skeleton is represented as the stacked heatmaps of joints obtained by 2D pose estimators. Besides, the image-like input may then be fixed during training, reducing the flexibility of skeleton in representing action patterns. On the other side, GCN-based solutions benefit from the utilization of topological relations among skeleton joints for feature modeling. However, they rely on the irregular topological structure and are forced to learn different shaped and separate convolution kernels node by node, lacking similar properties of feature aggregation in formulation. Furthermore, the receptive field of graph convolution usually covers a set of adjacent nodes with predefined distances to the target node, degrading the efficiency for modeling feature interactions especially among human joints that are actively coordinated yet topologically remote.

Motivated by these two observations, we formulate a new skeleton representation called Ske2Grid for action recognition with the expectation to inherit the non-Euclidean spatial layout of the graph skeleton while enabling the use of regular convolution as in the Euclidean image space but with a compact shape. Specifically, we investigate the problem from two technical perspectives. (1) from representation perspective, how to convert the skeleton from an irregular graph to a regular image-like grid patch while maintaining its critical topological representation capability and compactness for action recognition? (2) from network perspective, how to guarantee that the learned grid patch has improved representation abilities over the skeleton graph for action representation? To the first question, we design a novel grid representation of skeleton constructed and learned through three novel designs. Specifically, we construct a regular grid patch through allocating the nodes in the skeleton graph one by one to the desired grid cells using a graph-node index transform (GIT), inspired by the recent work (Kang et al., 2023) for 3D human pose estimation. To ensure that GIT is a bijection and enrich the expressiveness of the grid representation, we propose an up-sampling transform (UPT) with regulation of the graph topology to interpolate the skeleton graph nodes for filling the grid patch to the full. To further exploit the representation capability of the grid patch with increasing spatial size under which the one-step UPT is aggressive and tends to lose its effectiveness, a progressive learning strategy (PLS) is proposed for decoupling the UPT into multiple steps and aligning them to multiple paired GITs through a compact cascaded design learned progressively. The resulting grid representation differs from the existing image-like skeleton inputs in the following ways: (1) the grid patch is filled by up-sampled graph nodes with its layout reflecting the topological relations of skeleton features for action modeling; (2) the grid patch representation is much more compact in size compared with images (e.g. $5\times5\times3$). Instead of being down-sampled as in CNNs, the spatial size of our grid patch is fixed during representation learning to maintain the compact skeleton semantics; (3) the GIT and UPT can be jointly learned with the Ske2Grid convolution network in an end-to-end manner, enabling the enriched topological connections on the skeleton being automatically learned conditioned on the target action recognition dataset. To the second question, we define a regular convolution operation upon our grid patch for efficiently modeling the skeleton feature interactions. As shown in Figure 1, with the above novel designs for Ske2Grid, we can readily use the regular convolution with the square-shaped convolution kernel to learn discriminative feature interactions by naturally sharing the kernel to all grid nodes. Benefiting from the cascaded structure of multiple UPT and GIT pairs facilitated by the PLS, the resulting grid patch incorporates extra nodes interpolated from the original graph nodes, which further enables the regular convolution to capture various ordered topological relations, significantly strengthening the learning ability of the regular convolution on the grid patch.

Our Ske2Grid could be readily applicable to popular GCN architectures for improved skeleton-based action recognition, without modifying their built-in temporal modules. We

construct networks through replacing spatial graph convolution in prevailing GCNs with regular 2D convolution upon our grid patch representation, and conduct experiments on six mainstream skeleton-based action recognition datasets. Experiments show that our method significantly outperforms many existing GCN-based solutions under different settings, demonstrating the effectiveness of our method in learning improved skeleton features for action recognition.

2. Related Works

Skeleton-based action recognition with CNNs. Inspired by the research progress in image-based tasks, it is natural to convert the graph-structured skeleton into regular image-like input and take the advantage of image recognition pipelines to handle skeleton-based action recognition task. This line of works obtains a sequence of 2D array via combining skeleton features spatially and temporally, and then directly transforms the 2D arrays to gray images. For example, Ke et al. (2017) transforms skeleton into three clips consisting of several frames represented by computed spatial temporal features, and uses a deep CNN to model long-term temporal dynamics of skeleton. Luvizon et al. (2018) directly encodes the temporal dimension, the joints and the coordinates to be different axis of a pseudo image, and uses a CNN-based multitask framework for pose estimation and action recognition. Duan et al. (2022c) takes the stacked heatmaps of skeleton joints obtained by 2D pose estimators as input, and utilizes 3D convolutional neural networks to recognize actions, which refreshes the state-of-the-art on many skeleton-based action recognition benchmarks. These solutions benefit from the regular 2D/3D convolution for efficient image feature modeling. However, the conversion from skeleton to image-like input usually ignores the critical topological information of human joints and would sacrifice the compactness of skeleton data to some extent. Besides, the image-like input is usually fixed during training. Unlike them, we construct and learn a compact grid representation for skeleton, and its layout reflects the topological relations of human joints for action modeling.

Skeleton-based action recognition with GCNs. Skeleton can be naturally represented by an irregular graph, in which nodes represent joint coordinates and edges naturally connect joints in human bodies. GCNs generalize CNNs to graphs of arbitrary structures which are mainstream learning models to handle graph-structured skeleton data. ST-GCN (Yan et al., 2018) extends GCNs to a spatial-temporal graph model and designs convolution kernels for skeleton modeling, which is the first work that achieves satisfactory performance on large-scale skeleton-based action recognition benchmarks. However, it uses a pre-defined topology according to human body structure, which is fixed during both training and testing phases. A lot of following methods improve ST-GCN through constructing skeleton graph with dynamic topologies (Duan et al., 2022a; Shi et al., 2019; 2020). Many recent methods improve the receptive field of graph convolution through incorporating extra contextual information. AS-GCN (Li et al., 2019) introduces an A-link module to capture action-specific latent dependencies and an S-link module to represent higher-order node dependencies of skeleton graph. CA-GCN (Zhang et al., 2020) introduces a context term for each vertex by integrating information from the entire skeleton graph to enlarge the receptive field of graph convolution. Shift-GCN (Cheng et al., 2020) proposes local shift graph operation and non-local shift graph operation to provide flexible receptive fields for spatial and temporal graphs. CTR-GCN (Chen et al., 2021) proposes to learn a shared topology and channel-specific correlations simultaneously, obtaining channel-wise topologies which improve feature aggregations in graph convolution. Despite the advancement of these solutions, they still model the irregular skeleton graph using standard graph convolution. As discussed in the previous section, we define a regular convolution operation upon our learnable grid patch for efficiently modeling the skeleton feature interactions.

3. Method

In this section, we first revisit the graph convolution from a general perspective, then define the convolution operation upon our grid patch along with three core designs in Ske2Grid, and finally introduce the construction of network.

3.1. Graph Convolution

Considering a skeleton graph $G = \{V, E, A\}$, which consists of a set of nodes V with $|V| = N$, a set of edges E with $|E| = M$ and the adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N\times N}$. If there is an edge between nodes v_i and v_j, the entry $\boldsymbol{A}(i,j) = 1$; otherwise, $\boldsymbol{A}(i,j) = 0$. We represent the corresponding feature of the graph as $\boldsymbol{X} \in \mathbb{R}^{N\times C}$, where C denotes the skeleton feature dimension, and $\boldsymbol{x}_i \in \mathbb{R}^{C\times 1}$ denotes the feature vector for node i. The output value of graph convolution operation for a single channel at the node i can be written as

$$f_{out}(v_i) = \sum_{v_j \in B(v_i)} \boldsymbol{w}_{i,j}\boldsymbol{x}_j, \tag{1}$$

where $B(v_i) = \{v_j | d(v_j, v_i) \leq D\}$ denotes the neighbor set of node i; $d(v_j, v_i)$ is the minimum length of any path from node j to node i, and normally D is set to 1; $\boldsymbol{w}_{i,j} \in \mathbb{R}^{1\times C}$ is a weight vector specific to node i for computing the inner product with the corresponding input feature $\boldsymbol{x}_j$, which is indexed within the neighbor set $B(v_i)$. This convolution operation models the feature interactions among a set of neighboring nodes defined by the graph topology. However,

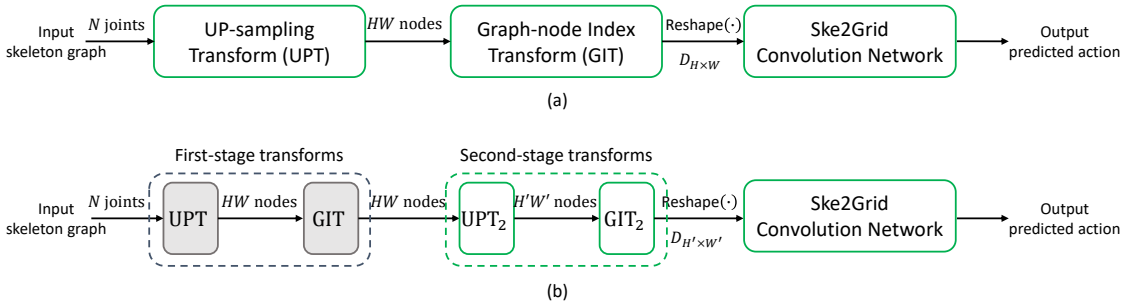

Figure 2. (a) The overall framework of Ske2Grid: the input skeleton graph with N joints is converted to a grid patch of size $H\times W$ using a pair of up-sampling transform (UPT) and graph-node index transform (GIT), which is then fed into the Ske2Grid convolution network for action recognition. (b) Ske2Grid with progressive learning strategy (PLS): the input skeleton is converted to a larger grid patch ($H' > H, W' > W$) using two-stage UPT plus GIT pairs. The well-trained Ske2Grid convolution network for the first-stage grid patch as in (a) is re-used to initialize the network for the second-stage grid patch as in (b), and the first-stage UPT plus GIT pair is fixed during training. PLS is used in a cascaded way to boost the performance of our Ske2Grid convolution network with increasing grid patch size.

the improved expressiveness of the skeleton action representation lies in its capability of modeling the discriminative feature interactions among a set of actively coordinated joints when performing an action, which may require the neighbor set activated in convolution operation to be flexible and expressive.

3.2. Convolution in Ske2Grid

It is intuitive to make the neighbor set in graph convolution be learnable to achieve improved expressiveness for action representation. However, the convolution kernel is specific to the target node and is difficult to model the correlations among a changing set of nodes. To solve this problem, we define a regular convolution operation in Ske2Grid.

Inspired by the recent progress in 3D human pose estimation (Kang et al., 2023), we construct a regular grid patch $D_{H\times W}$ with H and W being the spatial height and width, respectively. It consists of a set of grid cells $d_{i,j}$ filled by specific nodes from the skeleton graph. The feature of the grid patch is denoted as $\boldsymbol{Y} \in \mathbb{R}^{H\times W\times C}$, and $\boldsymbol{y}_{i,j} \in \mathbb{R}^{C\times 1}$ is the feature vector at the spatial location (i, j). Denote the size of a regular convolution kernel as $K \times K$. Similar to Equation (1), the output value of our convolution operation upon a grid cell at the spatial location (i, j) for a single channel is defined as

$$f_{out}(d_{i,j}) = \sum_{d_{m,n}\in B_D(d_{i,j})} \boldsymbol{w}_k \boldsymbol{y}_{m,n}, \tag{2}$$

where $B_D(d_{i,j})$ is the neighbor set of the grid cell $d_{i,j}$, which is the $K\times K$ sub-patch centered on $d_{i,j}$ containing K^2 grid cells; the set $\{\boldsymbol{w}_k \in \mathbb{R}^{1\times C} | k = 1, ..., K^2\}$ constitutes a squared-shaped convolution kernel for computing the inner product with features in the neighbor set of the grid cell $d_{i,j}$ as shown in Figure 1. In sharp contrast to graph convolution, the square-shaped convolution kernel in our Ske2Grid is shared everywhere on the grid patch, which is key to model a learnable set of grid cells within a regular neighbor set. Now, the question is how to learn a decent layout of the grid patch, making the neighboring grid cells be expressive for action representation.

3.3. Graph-node Index Transform

As mentioned above, a grid patch is constructed through assigning the nodes in the graph one by one to the grid cells. The problem is how to fill each grid cell with an appropriate graph node for improved feature interaction modeling. We propose to learn a mapping from the indexes of nodes in G to the spatial indexes of grid cells in $D_{H\times W}$, which is called graph-node index transform (GIT).

The GIT from N graph nodes to the grid patch $D_{H\times W}$ is defined by a binary matrix $\boldsymbol{\Phi} \in \{0, 1\}^{HW\times N}$, in which each row $\boldsymbol{\phi}_i \in \{0, 1\}^N$ is a one-hot vector indicating the index of the one selected node in the graph. That is, a grid cell d_i is filled with a specific graph node v_j only if $\boldsymbol{\phi}_{i,j} = 1$. With this definition, the feature of the grid patch $D_{H\times W}$ can be obtained by

$$\boldsymbol{Y} = \text{reshape}(\boldsymbol{\Phi} \cdot \boldsymbol{X}), \tag{3}$$

where "$\cdot$" is a row-by-column multiplication, with each row of the product matrix being the selected graph node feature. The $\text{reshape}(\cdot)$ operation rearranges the output of $\boldsymbol{\Phi}\cdot\boldsymbol{X}$ into an $H\times W$ grid patch representation. Thus, the layout of the grid patch can be learned along with $\boldsymbol{\Phi}$.

However, directly learning a binary matrix will cut off the backward gradient flow, which makes the training non-differentiable. To address this problem, we use straight-through estimator (STE) (Courbariaux et al., 2015) for parameter update. Specifically, we introduce a real-value

matrix $\mathbf{\Psi} \in \mathbb{R}^{HW \times N}$ to assist the learning of $\mathbf{\Phi}$. During training, we obtain $\mathbf{\Phi}$ through binarizing $\mathbf{\Psi}$ row by row according to

$$\boldsymbol{\phi}_{i,j} = \begin{cases} 1 & \text{if } \boldsymbol{\psi}_{i,j} = \max(\boldsymbol{\psi}_{i,\cdot}), \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Specifically, only one element in each row of $\mathbf{\Phi}$ is set to one, at its position where the maximum value in the corresponding row of $\mathbf{\Psi}$ appears. In the backward, continuous gradient is used to update the real-value matrix $\mathbf{\Psi}$, instead of $\mathbf{\Phi}$. In principle, by introducing $\mathbf{\Psi}$ as a continuous approximation of $\mathbf{\Phi}$, it enables the searching for a decent layout of grid patch for improved action expressiveness.

3.4. Up-sampling Transform

To avoid information loss during the conversion from skeleton to grid representation, the number of grid cells should be set to no less than the number of skeleton joints. Then all the skeleton joints can be assigned to the grid patch through making the GIT a surjection. However, the number of grid cells in the target grid patch is typically larger than the number of graph nodes, which means there exist some extra grid cells need to be filled. When filling these grid cells with duplicated graph nodes, such an optimization problem is not trivial, potentially degrading the expressiveness of the grid representation. To further guarantee that GIT is a bijection, we introduce an up-sampling transform (UPT) to interpolate skeleton graph nodes to fill the grid patch to the full. The feature map of the up-sampled graph is obtained by

$$\boldsymbol{X}' = \mathbf{\Lambda} \cdot \boldsymbol{X}, \tag{5}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{HW \times N}$ denotes the up-sampling matrix and $\boldsymbol{X}' \in \mathbb{R}^{HW \times C}$ is the up-sampled feature. Considering that the original topology of skeleton graph reveals intuitive coordinated movements of joints when performing an action, we further propose to incorporate the prior adjacency matrix into Equation (5) to regulate the up-sampling process

$$\boldsymbol{X}' = (\mathbf{\Lambda} \cdot \boldsymbol{A}) \cdot \boldsymbol{X}. \tag{6}$$

The utilization of the adjacency matrix $\boldsymbol{A}$ encourages the UPT to interpolate new nodes using adjacent nodes along existing edges, which facilitates our skeleton-to-grid representation with the association of topological priors for improved action representation.

After the UPT, our grid representation can be obtained using Equation (3) with the up-sampled feature and one-to-one index mapping. In principle, the main idea behind the UPT is learning to interpolate the skeleton graph nodes for the improved expressiveness meanwhile ensuring that the following GIT for constructing our grid patch is a bijection. Generally, GIT can be made as a bijection in various ways. In our case, we simply add a non-repetitive constraint to the binarization process in Equation (4) through binarizing $\mathbf{\Psi}$ row by row with a simple greedy search.

3.5. Progressive Learning Strategy

With the GIT and UPT, the skeleton input can be converted into a grid patch of any size. In image-based tasks, a CNN model tends to show improved performance when increasing the resolution of the input. However, the performance improvement is marginal when directly increasing the size of grid patch in our method. We conjecture that there exists a mismatch between the one-step transform pair of UPT and GIT and the straightforward learning strategy, leading to slightly worse performance than the original skeleton graph when the one-step UPT is aggressive.

To solve this problem, we propose progressive learning strategy (PLS), a novel optimization scheme, which decouples the aggressive one-step UPT into multiple steps and aligns them to multiple paired GITs through a compact cascaded design learned in a progressive manner. In this way, the grid representation is gradually enriched through increasing the size of grid patch progressively.

Specifically, we first learn to convert the skeleton graph to a base grid patch $D_{H \times W}$. Then, the previous-stage transform pair of UPT and GIT for the base grid patch is reused to convert the skeleton to a larger grid patch $D_{H' \times W'}$ where $H' > H, W' > W$. This process can be formally defined as

$$\boldsymbol{Y} = \text{reshape}(\mathbf{\Phi}_2 \cdot (\mathbf{\Lambda}_2 \cdot (\mathbf{\Phi} \cdot ((\mathbf{\Lambda} \cdot \boldsymbol{A}) \cdot \boldsymbol{X})))), \tag{7}$$

where $\mathbf{\Phi} \cdot ((\mathbf{\Lambda} \cdot \boldsymbol{A}) \cdot \boldsymbol{X})$ constructs the base grid path $D_{H \times W}$ using the first-stage transform pair of UPT and GIT as defined by Equation (6) and Equation (3); $\mathbf{\Lambda}_2 \in \mathbb{R}^{H'W' \times HW}$ is the second-stage UPT for up-sampling the grid patch from the base size of $H \times W$ to the target size of $H' \times W'$, and $\mathbf{\Phi}_2 \in \mathbb{R}^{H'W' \times H'W'}$ is the second-stage GIT for allocating the up-sampled features to the target grid patch.

The transform pair of UPT and GIT learned in the first stage is fixed when learning the second-stage transform pair of UPT and GIT. Furthermore, the Ske2Grid convolution network maintains the same structure during different training stages, and thus the network trained for the first-stage grid patch is also used as a pre-trained model to initialize the network training in the second stage. These two aspects constitute the PLS, which can be naturally used in a cascaded way consisting of multiple transform pairs of UPT and GIT to boost the performance of the Ske2Grid convolution network. Thanks to the lightweight designs of UPT and

GIT, they introduce negligible extra computation cost to the Ske2Grid convolution network during inference. The cascaded structure of multiple UPT and GIT pairs facilitated by the PLS incorporates extra nodes interpolated from the original graph nodes and enables the regular convolution to capture various ordered topological relations, strengthening the representation learning ability of our grid patch.

3.6. Network Construction

With the above designs for grid representation learning, convolution operation in our Ske2Grid can model a learnable set of enriched grid cells for improved action representation. The grid patch is fed into a Ske2Grid convolutional network for action recognition, as shown in Figure 2. By jointly training the UPT, GIT and the Ske2Grid convolution network in an end-to-end manner facilitated with the PLS, the enriched topological connections on the skeleton can be automatically learned conditioned on a target action recognition dataset. To make our Ske2Grid be applicable to popular GCN architectures without modifying their built-in temporal modules, we construct the Ske2Grid convolution network upon prevailing GCNs through simply replacing the spatial graph convolution with the convolution operation upon our learnable grid patch, without changing the structure of the backbone network and their temporal modules.

4. Experiments

4.1. Datasets

Six mainstream datasets are considered in the experiments.

NTU-60 (Shahroudy et al., 2016) is the first large-scale multi-modality skeleton-based action recognition dataset. It contains 56,880 skeleton action sequences which are performed by 40 volunteers and categorized into 60 classes. There are two popular validation protocols for this dataset: cross-subject (XSub) and cross-view (XView).

NTU-120 (Liu et al., 2019) is the extended version of NTU-60. It contains extra 57,367 skeleton sequences from 60 extra action classes. Two popular validation protocols for this dataset are: cross-subject(XSub) and cross-setup (XSet).

FineGym99 (Shao et al., 2020) is a newly released fine-grained action recognition dataset with 29,000 videos of 99 gymnastic action classes. **HMDB51** (Kuehne et al., 2011) and **UCF101** (Soomro et al., 2012) are two early popular action recognition datasets collected from the web. HMDB51 contains 6,700 videos from 51 classes and UCF101 contains 13,000 videos from 101 classes. **Diving48** (Li et al., 2018b) contains over 18,000 video clips of competitive diving actions, spanning 48 fine-grained dive classes.

Skeletons for all datasets. Regarding the experiments using 2D estimated skeleton, we use the 2D poses on these datasets provided by PYSKL (Duan et al., 2022b) for fair comparison, which are detected by HRNet (Sun et al., 2019) pre-trained on COCO (Lin et al., 2014). While there're ground-truth (GT) human bounding boxes on FineGym99, the human detector for all the other datasets is Faster-RCNN (Ren et al., 2015) with ResNet50 (He et al., 2016) as backbone. For the experiments on NTU-60 and NTU-120 using 3D GT skeleton, we also use the preprocessed 3D poses on these datasets provided by PYSKL for fair comparison.

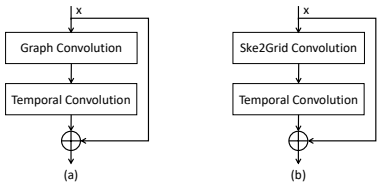

Figure 3. The difference between the ST-GCN block (a) and our ST-GCN* block (b).

4.2. Implementation Details

Regarding skeleton input, we use 2D joint coordinates plus the estimated scores as joint features for 2D skeleton input, and use 3D joint coordinates for GT 3D skeleton input, and thus $C = 3$ in both cases. For the initialization of the skeleton to grid representation transforms, we use random initialization for $\mathbf{\Psi}$ in the GIT. To better enjoy the regulation from the adjacency matrix in the UPT, $\mathbf{\Lambda}$ is initialized as an identity matrix of size $N \times N$ cascaded by a random matrix.

Regarding network construction, we choose the prevailing ST-GCN (Yan et al., 2018) as backbone network and modify the basic block through replacing the spatial graph convolution with regular 2D convolution upon our grid patch, as shown in Figure 3. We denote this modified network as ST-GCN*. The kernel size is 3×3, unless otherwise stated. The transform pair of UPT and GIT for $D_{H\times W}$ is added before the action recognition network, formulating our basic Ske2Grid$_{H\times W}$ framework.

For fair comparisons, we use the popular comprehensive skeleton-based action recognition toolbox PYSKL (Duan et al., 2022b) to implement all the experiments. There are mainly two training settings. For the experiments conducted to explore the effects of our core designs, we use the vanilla training strategy of PYSKL as in ST-GCN (Yan et al., 2018) for fair and clean comparisons, in which each model is trained for 80 epochs with the learning rate decayed by 10 at 10^{th} and 50^{th} epochs respectively. For the main experiments to explore the capability of Ske2Grid as shown in Table 2,Table 3, Table 7 and Table 8, we use the latest common experimental setups in PYSKL, in which each model is trained for 80 epochs with the cosine schedule of learning rate. In both settings, the initial learning rate is set to 0.1, the batch size is 128, the momentum is set to 0.9, the weight

Table 1. Effects of the three key designs for learning the grid representation in our Ske2Grid using NTU-60 XSub benchmark.

METHOD	GIT	UPT	PLS	TOP-1 ACC.(%)
ST-GCN	–	–	–	85.25
SKE2GRID$_{5\times5}$	√	×	×	84.70
	√	√	×	**86.20**
SKE2GRID$_{6\times6}$	√	×	×	84.04
	√	√	×	85.29
	√	×	√	85.61
	√	√	√	**87.87**
SKE2GRID$_{7\times7}$	√	×	×	85.93
	√	√	×	86.35
	√	×	√	85.96
	√	√	√	**88.26**
SKE2GRID$_{8\times8}$	√	×	×	84.81
	√	√	×	85.81
	√	×	√	86.11
	√	√	√	**88.55**
SKE2GRID$_{9\times9}$	√	×	×	84.10
	√	√	×	84.95
	√	×	√	86.60
	√	√	√	**88.37**

decay is 5×10^{-4}, and the Nesterov momentum is used for the optimizer. We report the validation top-1 accuracy in all the experiments except for Table 8. When comparing with state-of-the-art methods, we also follow the common practice as in Duan et al. (2022c) to report the testing top-1 accuracy for fair comparison, which is slightly better than the validation performance due to data augmentation.

4.3. Effects of Core Designs in Ske2Grid

We first conduct experiments to analyze the effects of our three core designs for learning our grid representation, including the UPT, GIT and PLS. We compare the performance of ST-GCN* using different combinations of these three designs for learning the grid patch, as shown in Table 1. The "PLS" setting indicates training the Ske2Grid$_{H\times W}$ progressively from Ske2Grid$_{(H-1)\times(W-1)}$ in a cascaded manner.

Ske2Grid of "GIT-only" fills the grid cells with all the distinct graph nodes and the remaining grid cells without the repetitive constraint. As shown, there is no obvious correlation between the performance and grid patch size. Together with "PLS", the performance improves progressively with the increase of the grid patch size, but the performance gain is not significant. This is probably due to the uncertainty brought by the redundant grid cells.

With the "GIT & UPT" setting, the performance of Ske2Grid becomes stably improved over the baseline. However, when the grid patch grows to $D_{9\times9}$, the performance drops below the baseline. In this case, the number of grid cells is about 5 times the number of joints in the skeleton input. It becomes challenging to interpolate the skeleton graph for satisfactory expressiveness using only one-step UPT.

Combining all the three designs, our Ske2Grid consistently outperforms the baseline with significant performance margin. And the performance improves progressively with the increase of the grid patch size, yielding 3.3% top-1 margin when using $D_{8\times8}$, well demonstrating the effectiveness of our Ske2Grid in learning improved skeleton features for action recognition. In the following main experiments, we train our Ske2Grid models using all the three designs. Considering the training overhead, we use the progressive learning strategy cascaded up to 3 times (4 stages), e.g. from $D_{5\times5}$ to $D_{8\times8}$ by a step of 1 for 2D estimated skeleton input.

Table 2. Main results of our Ske2Grid on six datasets using 2D estimated skeletons. We report the performance of our Ske2Grid$_{8\times8}$.

METHOD	DATASET		TOP-1 ACC.(%)	ΔTOP-1(%)
ST-GCN	NTU-60	XSUB	88.23	**+3.70**
OURS			**91.93**	
ST-GCN		XVIEW	96.63	**+1.12**
OURS			**97.75**	
ST-GCN	NTU-120	XSUB	83.56	**+1.24**
OURS			**84.80**	
ST-GCN		XSET	83.84	**+3.69**
OURS			**87.53**	
ST-GCN	FINEGYM99		91.17	**+0.65**
OURS			**91.82**	
ST-GCN	UCF101		69.23	**+3.83**
OURS			**73.06**	
ST-GCN	HMDB51		47.25	**+1.12**
OURS			**48.37**	
ST-GCN	DIVING48		38.32	**+5.79**
OURS			**44.11**	

Table 3. Performance comparison of our Ske2Grid using 3D GT skeleton inputs. We report the performance of Ske2Grid$_{9\times9}$.

METHOD	DATASET		TOP-1 ACC.(%)	ΔTOP-1(%)
ST-GCN	NTU-60	XSUB	87.62	**+0.63**
OURS			**88.25**	
ST-GCN		XVIEW	95.08	**+0.64**
OURS			**95.72**	
ST-GCN	NTU-120	XSUB	81.13	**+1.60**
OURS			**82.73**	
ST-GCN		XSET	83.63	**+1.44**
OURS			**85.07**	

4.4. Main Results

Table 2 shows the main results of our Ske2Grid on six action recognition datasets using $D_{8\times8}$. As can be seen, our method consistently outperforms the baseline on all datasets, showing its good generalization ability for improved skeleton-based action recognition. The performance margin on FineGym99 is relatively small among these datasets. This maybe because the 2D estimated skeleton on this dataset is based on GT human bounding boxes which is more accurate, compared to the other datasets. The performance improvement from our grid representation becomes marginal. This conjecture is further verified by the large top-1 accuracy margin on Diving48. Since it contains challenging fine-grained dive classes and relatively inaccurate 2D estimated skeleton benefits the most from the enriched grid representation to recognize the fine-grained actions.

Table 4. Impacts of the convolution operation in our Ske2Grid using NTU-60 XSub benchmark. We use ST-GCN to directly model our grid patch obtained by the UPT and GIT. ST-GCN* is with the convolution defined in our Ske2Grid. "$\diamond$" denotes that the transform is well-learned and fixed during training.

NETWORK	UPT	GIT	D SIZE	TOP-1 ACC.(%)
ST-GCN	–	–	–	85.25
ST-GCN	$\surd$	$\surd$	5×5	80.39
	$\diamond$	$\surd$		82.14
	$\surd$	$\diamond$		81.94
	$\diamond$	$\diamond$		84.34
ST-GCN*	$\surd$	$\surd$		**86.20**
ST-GCN	$\surd$	$\surd$	6×6	80.63
	$\diamond$	$\surd$		79.71
	$\surd$	$\diamond$		81.26
	$\diamond$	$\diamond$		84.53
ST-GCN*	$\surd$	$\surd$		**87.87**

4.5. Ablative Studies

We further provide a number of ablative studies to have a deep analysis of our Ske2Grid.

Ske2Grid using 3D skeleton input. In addition to 2D estimated skeleton input, we use 3D GT skeleton consisting of 25 keypoints on NTU-60 and NTU-120 as input to conduct experiments as shown in Table 3. Our Ske2Grid is trained from $D_{6\times6}$ to $D_{9\times9}$ by a step of 1. As shown, our method outperforms the baseline under different validation protocols on both datasets, demonstrating the strong generalization capability of our Ske2Grid for improved motion-captured skeleton-based action recognition.

Impacts of the convolution operation. To analyze the impacts of the convolution operation in our Ske2Grid, we use graph convolution to directly model our grid representation through treating the grid patch as a regular graph with vertical and horizontal connections between adjacent grid cells. The comparison is shown in Table 4.

Firstly, ST-GCN is used to model a grid patch with both the UPT and GIT being learnable. Not surprisingly, the performance drops significantly. As discussed before, graph convolution convolves each grid cell with their adjacent grid cells using its specific kernels. As the layout of our grid patch is learned with the network, it is difficult to adapt the graph convolution for modeling the changing topological relationship of grid cells during training.

Further, we reuse the UPT and GIT well learned by our Ske2Grid to train ST-GCN for modeling the grid patch. As shown, the performance is relatively good when fixing both the UPT and GIT, illustrating the capability of graph convolution in modeling fixed graph. However the fixed regular layout of grid patch still limits the expressiveness of skeleton for action representation. ST-GCN* with learnable grid patch achieves the best performance under different size settings, demonstrating the effectiveness of our convolution operation in modeling the feature interactions among a learnable set of grid cells for improved action recognition.

Table 5. Performance (%) comparison of different base grid patches and progressive steps for the PLS setting in our Ske2Grid using NTU-60 XSub benchmark. D_K is short for $D_{K\times K}$.

START	PLS_1	ACC.	PLS_2	ACC.	PLS_3	ACC.
D_5	D_6	87.87	D_7	88.26	D_8	**88.55**
	D_7	87.98	D_9	88.15	D_{11}	88.47
	D_8	87.92	D_{11}	87.82	–	–
D_6	D_7	86.27	D_8	86.41	D_9	87.57
	D_8	86.23	D_{10}	87.69	D_{12}	88.49
	D_9	86.44	D_{12}	87.59	–	–
D_7	D_8	87.15	D_9	87.30	D_{10}	87.72
	D_9	86.85	D_{11}	87.18	D_{13}	87.20
	D_{10}	87.17	D_{13}	87.59	–	–

Table 6. Performance comparison when using other grid patch sizes in our Ske2Grid using NTU-60 XSub benchmark.

GRID PATCH	TOP-1 ACC.(%)
$D_{5\times5}$	**86.20**
$D_{4\times6}$	85.61
$D_{4\times7}$	84.64
$D_{5\times6}$	85.91
$D_{5\times7}$	84.25

Impacts of different PLS settings. We exhaustively explore the starting size of grid patch and the progressive step in the PLS, as shown in Table 5. The performance increases progressively under all different settings. As shown, training our Ske2Grid from a small grid patch with the number of grid cells being close to the number of joints in the skeleton input progressively with a step of 1 performs better.

Ske2Grid using other grid patch sizes. Table 6 shows the performance of our Ske2Grid when using different grid patch sizes without the PLS setting. We can see that the performance of using squared grid patches is slightly better than that of using rectangular grid patches. Therefore, we use squared grid patches as default setting in the experiments.

Ske2Grid applying to other GCNs. We construct a network upon CTR-GCN (Chen et al., 2021) through replacing the graph convolution in the basic block of spatial modeling with our convolution operation, and the performance is shown in Table 7. Although CTR-GCN is a recent advanced GCN-based solution, our Ske2Grid still achieves acceptable performance gains under the same benchmark settings.

Visualization of the learnt layouts in Ske2Grid. The grid patch in Ske2Grid is filled by the up-sampled graph nodes, and its layout reflects the topological relations of joints on the skeleton graph. We visualize the learnt connections among graph nodes on the skeleton using the layouts of progressively learnt grid patches in $\text{Ske2Grid}_{8\times8}$, as shown in Figure 4, illustrating the capability of our method for modeling learnable skeleton feature interactions.

Table 8. Performance (%) comparison of our Ske2Grid with state-of-the-art methods on NTU-60 and NTU-120 using estimated 2D skeletons. We report the performance of Ske2Grid$_{8\times8}$. The "J", "B", "JM" and "BM" represent the joint, bone, joint motion and bone motion data modalities, respectively. The "$\diamond$" denotes using the same human skeletons as that provided by Duan et al. (2022b). The "$*$" represents the average model fusion of our Ske2Grid from $D_{5\times5}$ to $D_{8\times8}$. Best results are bolded.

METHOD	N60-XSUB	N60-XVIEW	N120-XSUB	N120-XSET
2S-AGCN (SHI ET AL., 2019) (J+B)	88.5	95.1	–	–
MS-AAGCN (SHI ET AL., 2020) (J+B+JM+BM)	90.0	96.2	–	–
AS-GCN (LI ET AL., 2019) (J)	86.8	94.2	78.3	79.8
CA-GCN (ZHANG ET AL., 2020) (J)	83.5	91.4	–	–
SHIFT-GCN (CHENG ET AL., 2020)(J+B+JM+BM)	90.7	96.5	85.9	87.6
MS-G3D (LIU ET AL., 2020)(J+B)	91.5	96.2	86.9	88.4
CTR-GCN (CHEN ET AL., 2021)(J+B+JM+BM)	92.4	96.8	**88.9**	90.6
HYPERGNN (HAO ET AL., 2021)(J+B+JM+BM)	89.5	95.7	–	–
SYBIO-GNN (LI ET AL., 2021)(J+B)	90.1	96.4	–	–
STST (ZHANG ET AL., 2021)(J+B+JM+BM)	91.9	96.8	–	–
ST-TR (PLIZZARI ET AL., 2021)(J)	88.7	95.6	81.9	84.1
HYPERFORMER (ZHOU ET AL., 2022)(J)	90.7	95.1	86.5	88.1
POSECONV3D (DUAN ET AL., 2022C)(J) $\diamond$	93.7	96.6	86.0	89.6
MS-G3D (J+B) $\diamond$	92.2	96.6	87.2	89.0
ST-GCN (J) $\diamond$	88.9	96.8	84.0	84.1
SKE2GRID (J) $\diamond$	92.3	97.9	85.3	87.9
SKE2GRID$*$ (J) $\diamond$	93.0	98.2	85.9	88.7
SKE2GRID$*$ (J+B+JM+BM) $\diamond$	**93.8**	**98.6**	87.3	**90.8**

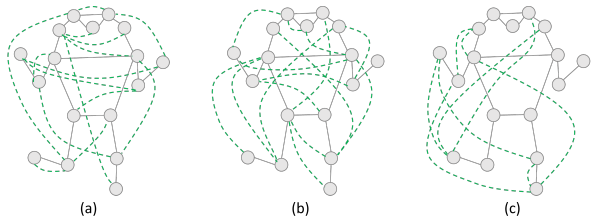

Figure 4. Visualization of the learnt topological relations on the skeleton graph reflected by (a) $D_{5\times5}$; (b) $D_{6\times6}$; and (c) $D_{7\times7}$.

Table 7. Performance (%) comparison when constructing a network upon CTR-GCN (Chen et al., 2021) by our Ske2Grid using 3D GT skeletons. We report the performance of Ske2Grid$_{9\times9}$.

METHOD	NTU-60		NTU-120	
	XSUB	XVIEW	XSUB	XSET
CTR-GCN	89.62	94.83	84.57	85.95
OURS	**90.08**	**95.09**	**84.79**	**86.04**

Comparison of Ske2Grid with state-of-the-art methods. Table 8 shows a performance comparison of our Ske2Grid with a lot of state-of-the-art methods on NTU-60 and NTU-120. We collect the best results reported in the original papers of all the methods shown above PoseConv3D (Yan et al., 2019). Prevailing GCN-based methods (the upper 7 solutions in Table 8) improve ST-GCN through constructing skeleton graph with dynamic topologies and incorporating extra contextual information to increase the receptive field of graph convolution, as we discussed in Section 2. Recently proposed hypergraph and transformer based methods (the middle 5 solutions in Table 8) either introduce local and global hyperedges to encode higher-order feature dependencies, or treat each joint of the human body as a token and use spatial and temporal self-attention operators to capture feature dependencies. As shown, our Ske2Grid with single "joint" modality outperforms or achieves comparable performance with these existing solutions. PoseConv3D (Yan et al., 2019) takes the stacked heatmaps (with the spatial size of 56×56) of skeleton joints obtained by 2D pose estimators as input and utilizes computationally intensive 3D CNN to recognize actions. Using much more compact grid representation (spatial size of 8×8) and lightweight 2D convolution network, our Ske2Grid achieves comparable performance with PoseConv3D, which is promising. Taking the advantage of our Ske2Grid with the progressive training, we ensemble the four models from $D_{5\times5}$ to $D_{8\times8}$ and obtain a significant performance boost, showing the complementarity among different grid patch scales. When further combining with the bone, joint motion and bone motion modalities commonly used in other methods, our Ske2Grid performs the best on three out of four benchmarks.

5. Conclusions

This paper presents Ske2Grid, a new representation learning framework for improved skeleton-based action recognition. In Ske2Grid, we define a regular convolution operation upon a compact image-like grid patch constructed and learned through three novel designs namely UPT, GIT and PLS. The layout of the grid representation is learned with the Ske2Grid convolution network, and convolution operation upon the grid patch models a learnable set of grid cells, which improves the modeling of feature interactions among active coordinated human joints for effective action recognition. The efficacy of our Ske2Grid is well validated by thorough experiments on six public benchmarks.

References

Chen, Y., Zhang, Z., Yuan, C., Li, B., Deng, Y., and Hu, W. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 13359–13368, 2021.

Cheng, K., Zhang, Y., He, X., Chen, W., Cheng, J., and Lu, H. Skeleton-based action recognition with shift graph convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 183–192, 2020.

Courbariaux, M., Bengio, Y., and David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proceedings of the advances in neural information processing systems*, 2015.

Du, Y., Wang, W., and Wang, L. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1110–1118, 2015.

Duan, H., Wang, J., Chen, K., and Lin, D. Dg-stgcn: Dynamic spatial-temporal modeling for skeleton-based action recognition. *arXiv preprint arXiv:2210.05895*, 2022a.

Duan, H., Wang, J., Chen, K., and Lin, D. Pyskl: Towards good practices for skeleton action recognition. *arXiv preprint arXiv:2205.09443*, 2022b.

Duan, H., Zhao, Y., Chen, K., Lin, D., and Dai, B. Revisiting skeleton-based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2969–2978, 2022c.

Hao, X., Li, J., Guo, Y., Jiang, T., and Yu, M. Hypergraph neural network for skeleton-based action recognition. *IEEE Transactions on image processing*, 30:2263–2275, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hu, G., Cui, B., and Yu, S. Skeleton-based action recognition with synchronous local and non-local spatio-temporal learning and frequency attention. In *Proceedings of the IEEE international conference on multimedia and expo*, pp. 1216–1221, 2019.

Kang, Y., Liu, Y., Yao, A., Wang, S., and Wu, E. 3d human pose lifting with grid convolution. In *Proceedings of the AAAI conference on artificial intelligence*, 2023.

Ke, Q., Bennamoun, M., An, S., Sohel, F., and Boussaid, F. A new representation of skeleton sequences for 3d action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3288–3297, 2017.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. Hmdb: a large video database for human motion recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 2556–2563, 2011.

Li, M., Chen, S., Chen, X., Zhang, Y., Wang, Y., and Tian, Q. Actional-structural graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3595–3603, 2019.

Li, M., Chen, S., Chen, X., Zhang, Y., Wang, Y., and Tian, Q. Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction. *IEEE Transactions on pattern analysis and machine intelligence*, 44(6):3316–3333, 2021.

Li, S., Li, W., Cook, C., Zhu, C., and Gao, Y. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5457–5466, 2018a.

Li, Y., Li, Y., and Vasconcelos, N. Resound: Towards action recognition without representation bias. In *Proceedings of the European conference on computer vision*, pp. 513–528, 2018b.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Proceedings of the European conference on computer vision*, pp. 740–755, 2014.

Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L.-Y., and Kot, A. C. 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.

Liu, M., Liu, H., and Chen, C. Enhanced skeleton visualization for view invariant human action recognition. *Pattern recognition*, 68:346–362, 2017.

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

Luvizon, D. C., Picard, D., and Tabia, H. 2d/3d pose estimation and action recognition using multitask deep learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5137–5146, 2018.

Peng, W., Hong, X., Chen, H., and Zhao, G. Learning graph convolutional network for skeleton-based human action recognition by neural searching. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 2669–2676, 2020.

Plizzari, C., Cannici, M., and Matteucci, M. Skeleton-based action recognition via spatial and temporal transformer networks. *Computer vision and image understanding*, 208, 2021.

Qin, Z., Liu, Y., Ji, P., Kim, D., Wang, L., McKay, B., Anwar, S., and Gedeon, T. Fusing higher-order features in graph neural networks for skeleton-based action recognition. *arXiv preprint arXiv:2105.01563*, 2021.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the advances in neural information processing systems*, 2015.

Shahroudy, A., Liu, J., Ng, T.-T., and Wang, G. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1010–1019, 2016.

Shao, D., Zhao, Y., Dai, B., and Lin, D. Finegym: A hierarchical video dataset for fine-grained action understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2616–2625, 2020.

Shi, L., Zhang, Y., Cheng, J., and Lu, H. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 12026–12035, 2019.

Shi, L., Zhang, Y., Cheng, J., and Lu, H. Skeleton-based action recognition with multi-stream adaptive graph convolutional networks. *IEEE Transactions on image processing*, 29:9532–9545, 2020.

Song, S., Lan, C., Xing, J., Zeng, W., and Liu, J. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *Proceedings of the AAAI conference on artificial intelligence*, 2017.

Soomro, K., Zamir, A. R., and Shah, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Sun, K., Xiao, B., Liu, D., and Wang, J. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5693–5703, 2019.

Yan, A., Wang, Y., Li, Z., and Qiao, Y. Pa3d: Pose-action 3d machine for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7922–7931, 2019.

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

Zhang, X., Xu, C., and Tao, D. Context aware graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 14333–14342, 2020.

Zhang, Y., Wu, B., Li, W., Duan, L., and Gan, C. Stst: Spatial-temporal specialized transformer for skeleton-based action recognition. In *Proceedings of the ACM international conference on multimedia*, pp. 3229–3237, 2021.

Zhou, Y., Li, C., Cheng, Z.-Q., Geng, Y., Xie, X., and Keuper, M. Hypergraph transformer for skeleton-based action recognition. *arXiv preprint arXiv:2211.09590*, 2022.