Skeleton-based Action Recognition via Spatial and Temporal Transformer Networks

Chiara Plizzari Marco Cannici Matteo Matteucci
Politecnico di Milano, Italy
chiara.plizzari@mail.polimi.it
{marco.cannici,matteo.matteucci}@polimi.it

Abstract

Skeleton-based Human Activity Recognition has achieved great interest in recent years as skeleton data has demonstrated being robust to illumination changes, body scales, dynamic camera views, and complex background. In particular, Spatial-Temporal Graph Convolutional Networks (ST-GCN) demonstrated to be effective in learning both spatial and temporal dependencies on non-Euclidean data such as skeleton graphs. Nevertheless, an effective encoding of the latent information underlying the 3D skeleton is still an open problem, especially when it comes to extracting effective information from joint motion patterns and their correlations. In this work, we propose a novel Spatial-Temporal Transformer network (ST-TR) which models dependencies between joints using the Transformer self-attention operator. In our ST-TR model, a Spatial Self-Attention module (SSA) is used to understand intra-frame interactions between different body parts, and a Temporal Self-Attention module (TSA) to model inter-frame correlations. The two are combined in a two-stream network, whose performance is evaluated on three large-scale datasets, NTU-RGB+D 60, NTU-RGB+D 120, and Kinetics Skeleton 400, outperforming the state-of-the-art on NTU-RGB+D w.r.t. models using the same input data, i.e., joint information.

1. Introduction

Human Action Recognition is achieving increasing interest in recent years for the progress achieved in deep learning and computer vision and for the interest of its applications in human-computer interaction, eldercare and healthcare assistance, and video surveillance. Recent advances in 3D depth cameras such as Microsoft Kinect [55, 38] and Intel RealSense [21] sensors, and advanced human pose estimation algorithms [5, 9] made it possible to estimate 3D skeleton coordinates quickly and accurately with cheap devices (refer to the survey by [13] for an in-depth analysis of recent devices). Nevertheless, several aspects of skeleton-based action recognition still remain open [53, 1, 37]. The most widespread method to perform skeleton-based action recognition has nowadays become Graph Neural Networks (GNNs), and in particular, Graph Convolutional Networks (GCNs) since, being an efficient representation of non-Euclidean data, they are able to effectively capture spatial (intra-frame) and temporal (inter-frame) information. Models making use of GCN were first introduced in skeleton-based action recognition by [51] and they are usually referred to as Spatial-Temporal Graph Convolutional Networks (ST-GCNs). These models process spatial information by operating on skeleton bone-connections along space, and temporal information by considering additional time-connections between each skeleton joint along time. Despite being proven to perform very well on skeleton data, ST-GCN models have some structural limitations, some of them already addressed by [41, 44, 6, 32].

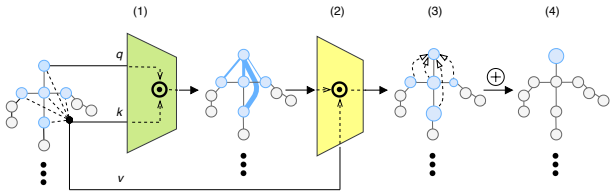


Figure 1. Self-attention on skeleton joints. (1) For each body joint, a query $\mathbf{q}$, a key $\mathbf{k}$ and a value vector $\mathbf{v}$ are calculated. (2) Then, the dot product ($\odot$) between the query of the joint and the key of all the other nodes is performed, representing the connection strength between each pair of nodes. (3) Finally, each node is scaled by its correlation w.r.t. the current node, (4) whose new features are obtained summing the weighted nodes together.

First of all, the topology of the graph representing the human body is fixed for all layers and all the actions; this may prevent the extraction of rich representations for the skeleton movements during time, especially if graph links are

directed and information can only flow along a predefined path. Secondly, both Spatial and Temporal Convolution are implemented starting from a standard 2D convolution. As such, they are limited to operate in a local neighborhood, somehow restricted by the convolution kernel size. And finally, as a consequence of the previous, correlations between body joints not linked in the human skeleton, e.g., the left and right hands, are underestimated even if relevant in actions such as "clapping". In this paper, we face all these limitations by employing a modified Transformer self-attention operator, as depicted in Figure 1. Despite being originally designed for Natural Language Processing (NLP) tasks, the sequentiality and hierarchical structure of human skeleton sequences, as well as the flexibility of Transformer self-attention [47] in modeling long-range dependencies, make this model a perfect solution to tackle ST-GCN weaknesses. Recently, [2] employed self-attention to overcome the locality of the convolution operator by capturing the global context of pixels in an image. In our work, we aim to apply the same mechanism to spatial-temporal skeleton-based architectures, and in particular to joints representing the human skeleton with the goal of modeling long-range interactions within human actions both in space, through a Spatial Self-Attention module (SSA), and time, through a Temporal Self-Attention module (TSA) module. [8] also proposed a Self-Attention Network (SAN) to extract long-term semantic information; however, since it focuses on temporally segmented clips, it solves the locality limitations of convolution only partially.

Main contributions of this paper are summarized as follows:

- We propose a novel two-stream Transformer-based model for skeleton activity recognition tasks, employing *self-attention* on both the spatial and the temporal dimensions
- We design a *Spatial Self-Attention* (SSA) module to dynamically build links between skeleton joints, representing the relationships between human body parts, conditionally on the action and independently from the natural human body structure. On the temporal dimension, we introduce a *Temporal Self-Attention* (TSA) module to study the dynamics of a joint along time. We made both layers publicly available for experiments replication and further use [1]
- Our model outperforms ST-GCN [51] and A-GCN [42] baselines on all datasets and outperforms previous state-of-the-art methods using the same input data on NTU.

[1]Code at https://github.com/Chiaraplizz/ST-TR

2. Related Works

2.1. Skeleton-based Action Recognition

Most of the early studies in skeleton-based action recognition relied on handcrafted features [16, 48, 17] exploiting relative 3D rotations and translations between joints. Deep learning revolutionized activity recognition by proposing methods capable of increased robustness [50] and able to achieve unprecedented performance. Methods that fall into this category rely on different aspects of skeleton data: (1) Recurrent neural network (RNN) based methods [23, 49, 29, 12] leverage on the sequentiality of joint coordinates, treating input skeleton data as time series. (2) Convolutional neural network (CNN) based methods [7, 44, 11, 30, 24] leverage spatial information, in a complementary way to RNN-based ones. Indeed, 3D skeleton sequences are mapped into a pseudo-image, representing temporal dynamics and skeleton joints respectively in rows and columns. (3) Graph neural network (GNN) based methods [51, 42, 41], make use of both spatial and temporal data by exploiting information contained in the natural topological graph structure of the human skeleton. These latter methods have demonstrated to be the most expressive among the three, and among these, the first model capturing the balance between spatial and temporal dependencies has been the Spatio-Temporal Graph Convolutional Network (ST-GCN) [51]. In ST-GCN the human skeleton is represented as a graph where joints are encoded as nodes and bones as arcs, while time is modeled with additional edges linking together the same joint along time. In this work, we used ST-GCN as the baseline model; its functioning is presented in details in Section 3.2.

2.2. Graph Neural Networks

Geometric deep learning [3] refers to all emerging techniques attempting to generalize deep learning models to non-Euclidean domains such as graphs. The notion of Graph Neural Network (GNN) was initially outlined by [14] and further elaborated by [39]. The intuitive idea underlying GNNs is that nodes in a graph represent objects or concepts while edges represent their relationships. Due to the success of Convolutional Neural Networks, the concept of convolution has later been generalized from grid to graph data. GNNs iteratively process the graph, each time representing nodes as the result of applying a transformation to nodes' and their neighbors' features. The first formulation of CNNs on graphs is due to [4], who generalized convolution to signals using a *spectral* construction. This approach had computational drawbacks that have been subsequently addressed by [15] and [10]. The latter has been further simplified and extended by [22]. A complementary approach is the *spatial* one, where graph convolution is defined as information aggregation [33, 35, 46]. In this work we make

use of the spectral construction proposed by [22], whose formulation is provided in Section 3.2.

2.3. Transformer

The Transformer is the leading neural model for Natural Language Processing (NLP), proposed by [47] as an alternative to recurrent networks. It has been designed to face two key problems: (i) the processing of very long sequences, which are often intractable both for LSTMs and RNNs, and (ii) the limitations in parallelizing sentence processing, which is usually performed sequentially, word by word, in standard RNNs architectures. The Transformer follows a usual encoder-decoder structure, but it relies solely on *multi-head self-attention* [47]. Recently, self-attention mechanisms have been also applied to visual tasks by [2], with the purpose of augmenting standard convolution. Our work is currently one of the few in literature applying self-attention on graphs [25].

3. Background

In this section, Spatial-Temporal Graph Convolutional Networks (ST-GCN) by [51] and the original Transformer self-attention mechanism by [47] are summarized, being the basic blocks of the model we propose in this paper.

3.1. Skeleton Sequences Representation

Given a sequence of skeletons, we define V as the number of joints representing each skeleton and T as the total number of skeletons composing the sequence, also named frames in the following. In order to represent the sequence, a spatial temporal graph is built, i.e., $G = (N, E)$, where $N = \{v_{ti} | t = 1, ..., T, i = 1, ..., V\}$ represents the set of all the nodes v_{ti} of the graph, i.e., the body joints of the skeleton along all the time sequence, and E represents the set of all the connections between nodes. E consists of two subsets; the first subset $E_S = \{(v_{ti}, v_{tj}) \mid i, j = 1, \dots, V, t = 1, \dots, T\}$ is composed by the intra-skeleton connections at each time interval t, for any pair of joints (i, j) connected by a bone in the human skeleton. The subset E_S of intra-skeleton connections is commonly further divided into K disjoint partitions, based on some criterion [51] (e.g., distance from the center of gravity), and encoded using a set of adjacency matrices $\tilde{\mathbf{A}}_k \in \{0, 1\}^{V \times V}$. The second subset $E_T = \{(v_{ti}, v_{(t+1)i}) \mid i = 1, \dots, V, t = 1, \dots, T\}$ consists of all the inter-frame connections between joints along consecutive time frames. The result is a graph extending on both the spatial and the temporal dimension.

3.2. Spatial Temporal Graph Convolutional Networks

Spatial Temporal Graph Convolutional Networks (ST-GCN) have been introduced by [51]. A ST-GCN is structured as a hierarchy of stacked spatial-temporal blocks, which are internally composed of a spatial convolution (GCN) followed by a temporal convolution (TCN).

The spatial sub-module uses the Graph Convolution formulation proposed by [22], which can be summarized as it follows:

$$\mathbf{f}_{out} = \sum_{k}^{K_s} (\mathbf{f}_{in} \mathbf{A}_k) \mathbf{W}_k, \tag{1}$$

$$\mathbf{A}_k = \mathbf{D}_k^{-\frac{1}{2}} (\tilde{\mathbf{A}}_k + \mathbf{I}) \mathbf{D}_k^{-\frac{1}{2}}, D_{\text{ii}} = \sum_{k}^{K_s} (\tilde{\mathbf{A}}_k^{ij} + \mathbf{I}_{\text{ij}}), \tag{2}$$

where K_s is the kernel size on the spatial dimension, $\tilde{\mathbf{A}}_k$ is the adjacency matrix of the undirected graph representing intra-body connections, $\mathbf{I}$ is the identity matrix and $\mathbf{W}_k$ is a trainable weight matrix. The temporal convolution sub-module (TCN) is implemented as a $1 \times K_t$ 2D convolution operating on (V, T) dimensions of the (C_{in}, V, T) input volume, where K_t is the number of frames considered within the kernel receptive field.

As shown in Equation 1, the graph structure is predefined, being the adjacency matrix fixed. In order to make it *adapative*, [42] introduced the *Adaptive Graph Convolutional Network* (A-GCN), where the GCN formulation in Equation 1 is replaced by the following:

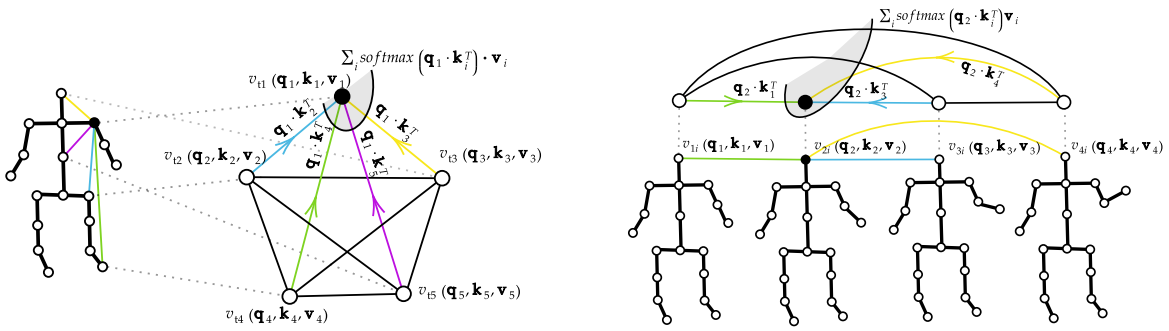
$$\mathbf{f}_{out} = \sum_{k}^{K_s} \mathbf{f}_{in} (\mathbf{A}_k + \mathbf{B}_k + \mathbf{C}_k) \mathbf{W}_k, \tag{3}$$

where $\mathbf{A}_k$ is the same as the one in Equation 1, $\mathbf{B}_k$ is learned during training, and $\mathbf{C}_k$ determines whether two vertices are connected or not through a similarity function.

3.3. Transformer Self-Attention

The original Transformer model of [47] employs *self-attention*, i.e., a *non-local operator* originally designed to operate on words in NLP tasks with the goal of enriching the embedding of each word based on the surrounding context. In the Transformer, new word embeddings are computed by comparing pairs of words an then mixing their embeddings together based on how much a word is relevant w.r.t. the others. By gathering clues from the surrounding context, self-attention enables to extract a better meaning from each word, dynamically building relations within and between phrases.

In particular, for each word embedding $\mathbf{w}_{\text{i}} \in W = \{\mathbf{w}_1, ..., \mathbf{w}_n\}$, a query $\mathbf{q} \in \mathbb{R}^{d_q}$, a key $\mathbf{k} \in \mathbb{R}^{d_k}$ and a value vector $\mathbf{v} \in \mathbb{R}^{d_v}$ are computed through trainable linear transformations starting from each the word embeddings, independently. Then, a score for each word embedding is obtained by taking the dot product $\alpha_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j^T$ $\forall i, j = 1, ..., n$, where n is the total number of nodes being considered. This score represents how much the word



(a) Spatial Self-Attention

(b) Temporal Self-Attention

Figure 2. Spatial Self-Attention (SSA) and Temporal Self-Attention (TSA). Self-attention operates on each pair of nodes, by computing a weight for each of them which represents the strength of their correlation. Those weights are then used to score the contribution of each body joint v_{ti}, proportionally to how relevant the node is w.r.t. to all the others. Please notice that on SSA (a), the procedure is illustrated only of a group of five nodes for simplicity, while in practice it operates on all the nodes.

j is relevant for word i. To compute the final embedding for word i, a weighted sum is computed by first multiplying the value vector of each other word $\mathbf{v}_j$ by the corresponding score α_{ij}, scaled through the softmax function, and than summing these vectors together.

This process, also called *scaled dot-product attention*, can be written in matrix form as it follows:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_{\mathrm{k}}}}\right)\mathbf{V}, \qquad (4)$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are matrices containing the predicted query, key and value vectors, respectively, packed together and d_{k} is the channel dimension of the key vectors. The division by $\sqrt{d_{\mathrm{k}}}$ is performed in order to increase gradients stability during training. In order to obtain better performance, a mechanism called *multi-headed attention* is usually applied, which consists in applying attention, i.e., a head, multiple times with different learnable parameters and then finally combining the results.

4. Spatial Temporal Transformer Network

We propose the *Spatial Temporal Transformer (ST-TR)* network, an architecture which uses Transformer self-attention to operate on both space and time. We propose to achieve this goal using two modules, the *Spatial Self-Attention (SSA)* and the *Temporal Self-Attention (TSA)* modules, each one focusing on extracting correlations on one of the two dimensions.

4.1. Motivation

The idea behind the original Transformer self-attention is to allow the encoding of both short- and long-range correlations between words in the sentence. Our intuition is that the same approach can be applied to skeleton-based action recognition as well, as correlations between nodes are crucial both on the spatial and on the temporal dimension. We consider the joints comprising the skeleton as a bag-of-words and make use of the Transformer self-attention to extract node embeddings encoding the relation between surrounding joints, just like words in a phrase in NLP. Contrary to a standard graph convolution, where only the adjacent nodes are compared, we discard any predefined skeleton structure and instead let the Transformer self-attention automatically discover joint relations which are relevant for predicting the current action. The resulting operation acts similarly to a graph convolution, but in which the kernel values are *dynamically predicted* based on the discovered joint relations. The same idea is also applied at the sequence level, by analyzing how each joint changes during the action and building *long-range relations* that span different frames, similarly to how relations between phrases are built in NLP. The resulting operator is capable of obtaining a dynamical representation extending both on the spatial and the temporal dimension.

4.2. Spatial Self-Attention (SSA)

The Spatial Self-Attention module applies self-attention *inside each frame* to extract low-level features embedding the relations between body parts. This is achieved by computing correlations between each pair of joints in every single frame independently, as depicted in Figure 2a. Given the frame at time t, for each node v_{ti} of the skeleton, a *query* vector $\mathbf{q}_i^t \in \mathbb{R}^{dq}$, a *key* vector $\mathbf{k}_i^t \in \mathbb{R}^{dk}$ and a *value* vector $\mathbf{v}_i^t \in \mathbb{R}^{dv}$ are first computed by applying trainable linear transformations to the node features $\mathbf{n}_i^t \in \mathbb{R}^{C_{in}}$ with parameters $\mathbf{W}_q \in \mathbb{R}^{C_{in} \times dq}$, $\mathbf{W}_k \in \mathbb{R}^{C_{in} \times dk}$, $\mathbf{W}_v \in \mathbb{R}^{C_{in} \times dv}$, shared across all nodes. Then, for each pair of body nodes (v_{ti}, v_{tj}), a *query-key dot product* is applied to obtain a

weight $\alpha_{ij}^t = \mathbf{q}_i^t \cdot {\mathbf{k}_j^t}^T \in \mathbb{R}, \forall t \in T$ representing the strength of the correlations between the two nodes. The resulting score α_{ij}^t is used to weight each joint value $\mathbf{v}_j^t$, and a weighted sum is computed to obtain a new embedding $\mathbf{z}_i^t$ for node v_{ti}, as in the following:

$$\mathbf{z}_i^t = \sum_j softmax_j \left(\frac{\alpha_{ij}^t}{\sqrt{d_k}} \right) \mathbf{v}_j^t, \tag{5}$$

where $\mathbf{z}_i^t \in \mathbb{R}^{C_{out}}$ (with C_{out} the number of output channels) constitutes the new embedding of node v_{ti}.

Multi-head attention is applied by repeating this embedding extraction process N_h times, each time with a different set of learnable parameters. The set $(\mathbf{z}_{i_1}^t, ..., \mathbf{z}_{i_H}^t)$ of node embeddings thus obtained, all referring to the same node v_{ti}, is then combined with a learnable transformation, i.e., $concat(\mathbf{z}_{i_1}^t, ..., \mathbf{z}_{i_H}^t) \cdot \mathbf{W}_o$, and constitutes the output features of SSA.

As shown in Figure 2a, the relations between nodes (i.e., the α_{ij}^t scores) are dynamically *predicted* in SSA; the correlation structure in the skeleton is then not fixed for all the actions, but it changes adaptively for each sample. SSA operates similar to a graph convolution on a fully connected graph where, however, the kernel values (i.e., the α_{ij}^t scores) are predicted dynamically based on the skeleton pose.

4.3. Temporal Self-Attention (TSA)

With the Temporal Self-Attention (TSA) module, the dynamics of each joint is studied separately *along all the frames*, i.e., each single joint is considered as independent and correlations between frames are computed by comparing the change in the embeddings of the same body joint along the temporal dimension (see Figure 2b). The formulation is symmetrical to the one reported in Equation (5) for SSA:

$$\alpha_{tu}^v = \mathbf{q}_t^v \cdot \mathbf{k}_u^v \quad \forall v \in V, \quad \mathbf{z}_t^v = \sum_j softmax_u \left(\frac{\alpha_{tu}^v}{\sqrt{d_k}} \right) \mathbf{v}_u^v, \tag{6}$$

where v_{ti}, v_{ui} indicate the same joint v in two different instants t, u, $\alpha_{tu}^i \in \mathbb{R}$ is the correlation score, $\mathbf{q}_t^i \in \mathbb{R}^{dq}$ is the query associated to v_{ti}, $\mathbf{k}_u^i \in \mathbb{R}^{dk}$ and $\mathbf{v}_u^i \in \mathbb{R}^{dv}$ are the key and value associated to joint v_{ui} (all computed using trainable linear transformations as in SSA), and $\mathbf{z}_t^i \in \mathbb{R}^{C_{out}}$ is the resulting node embedding. Note that the notation used in this section is opposite w.r.t. the one used in Section 4.2; subscripts indicate time while superscripts indicate the joint. Multi-head attention is applied in TSA as in SSA. An example of TSA is depicted in Figure 2b.

The TSA module, by extracting inter-frame relations between nodes in time, can learn how to correlate frames apart from each other (e.g., nodes in the first frame with those in the last one), capturing discriminant features that are not otherwise possible to capture with a standard ST-GCN convolution, being this limited by the kernel size.

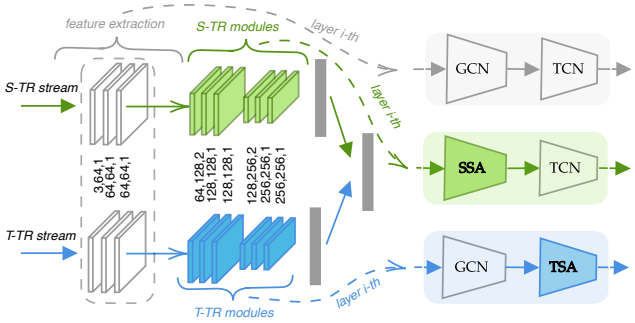


Figure 3. Illustration of two 2s-ST-TR architecture. On each stream, the first three layers extract low level features. On the S-TR stream, at each layer SSA is used to extract spatial information, followed by a 2D convolution on time dimension (TCN), while on the T-TR stream, at each layer, TSA is used to extract temporal information, while spatial features are extracted by a standard graph convolution (GCN) [51].

4.4. Two-Stream Spatial Temporal Transformer Network

To combine the SSA and TSA modules, a two-stream architecture named 2s-ST-TR is used, as similarly proposed by [42] and [41]. In our formulation, the two streams differentiate on the way the proposed self-attention mechanisms are applied: SSA operates on the spatial stream (named S-TR), while TSA on the temporal one (named T-TR). On both streams, node features are first extracted by a three-layers residual network, where each layer processes the input on the spatial dimension through graph convolution (GCN), and on the temporal dimension through a standard 2D convolution (TCN), as done by [51][2]. SSA and TSA are then applied on the S-TR and on the T-TR stream in the subsequent layers in substitution to the GCN and TCN feature extraction modules respectively (Figure 3). The sub-networks outputs are eventually fused together by summing up their softmax output scores to obtain the final prediction, as proposed by [42] and [41].

Spatial Transformer Stream (S-TR) In the spatial stream, self-attention is applied at the skeleton level through a SSA module, which focuses on spatial relations between joints. The output of the SSA module is passed to a 2D convolutional module with kernel K_t on the temporal dimension (TCN), as done by [51], in order to extract temporally relevant features, as shown in Figure 3 and expressed in the

[2]In principle other features, e.g., visual features, could be added here but we want in this paper to focus on pure skeleton base action recognition and we leave this option for future investigations.

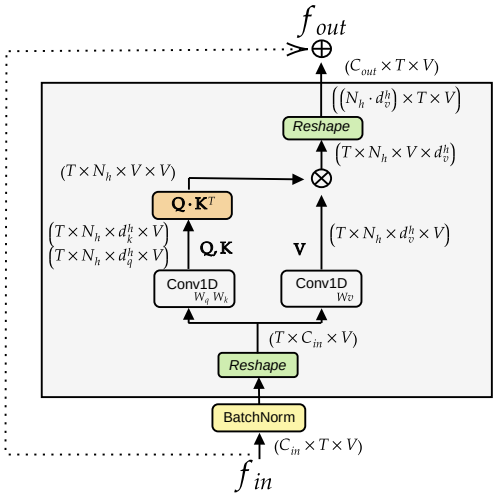


Figure 4. Illustration of a SSA module (the implementation of TSA is the same, with the only difference that the dimension V corresponds to T and viceversa). The input f_{in} is reshaped by moving T in the batch dimension, such that self-attention operates on each time frame separately. SSA is implemented as a matrix multiplication, where $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are the query, key and value matrix respectively, and $\otimes$ denotes the matrix multiplication.

following:

$$\textbf{S-TR}(x) = Conv_{2D(1\times K_t)}(\textbf{SSA}(x)). \tag{7}$$

Following the original Transformer structure, the input is pre-normalized passing through a Batch Normalization layer [18, 34], and skip connections are used to sum the input to the output of the SSA module (see Figure 4).

Temporal Transformer Stream (T-TR) The temporal stream, instead, focuses on discovering inter-frame temporal relations. Similarly to the S-TR stream, inside each T-TR layer, a standard graph convolution sub-module [51] is followed by the proposed Temporal Self-Attention module:

$$\textbf{T-TR}(x) = \textbf{TSA}(GCN(x)). \tag{8}$$

TSA operates on graphs linking the same joint along all the time dimension (e.g., all left feet, or all right hands).

4.5. Implementation of SSA and TSA

The matrix implementation of SSA (and of TSA) is based on the implementation of Transformer on pixels by [2]. As shown in Figure 4, given an input tensor of shape (C_{in}, T, V), where C_{in} is the number of input features, T is the number of frames and V is the number of nodes, a matrix $\mathbf{X}_V \in \mathbb{R}^{T\times C_{in}\times V}$ is obtained by rearranging the input. Here the T dimension is moved inside the batch dimension, effectively implementing parameter sharing along the temporal dimension and applying the transformation separately on each frame:

$$head_h(\mathbf{X}_V) = Softmax\left(\frac{(\mathbf{X}_V\mathbf{W}_\mathrm{q})(\mathbf{X}_V\mathbf{W}_\mathrm{k})^\mathrm{T}}{\sqrt{d_k^h}}\right)(\mathbf{X}_V\mathbf{W}_v)$$
$$SelfAttention_V = Concat(head_1, ..., head_{N_h})\mathbf{W}^\mathrm{o}, \tag{9}$$

where the product with $\mathbf{W}_q \in \mathbb{R}^{C_{in}\times N_h\times d_q^h}$, $\mathbf{W}_k \in \mathbb{R}^{C_{in}\times N_h\times d_k^h}$ and $\mathbf{W}_v \in \mathbb{R}^{C_{in}\times N_h\times d_v^h}$ gives rise respectively to $\mathbf{Q} \in \mathbb{R}^{T\times N_h\times d_q^h\times V}$, $\mathbf{K} \in \mathbb{R}^{T\times N_h\times d_k^h\times V}$ and $\mathbf{V} \in \mathbb{R}^{T\times N_h\times d_v^h\times V}$, being N_h the number of heads, and $\mathbf{W}^o$ a learnable linear transformation combining the heads outputs. The output of the Spatial Transformer is then rearranged back into $\mathbb{R}^{C_{out}\times T\times V}$. The TSA matrix implementation has the same expression as Equation (9), differing only in the way the input $\mathbf{X}$ is processed. Indeed, in order to be processed by each TSA module, the input is reshaped into a matrix $\mathbf{X}_T \in \mathbb{R}^{V\times C_{in}\times T}$, where the V dimension has been moved in the first position and aggregated to the batch dimension, not reported here explicitly, in order to operate separately on each joint along the time dimension. The formulation is analogous to Equation (9), differing only in the shape of matrices, which become $\mathbf{Q} \in \mathbb{R}^{V\times N_h\times d_q^h\times T}$, $\mathbf{K} \in \mathbb{R}^{V\times N_h\times d_k^h\times T}$ and $\mathbf{V} \in \mathbb{R}^{V\times N_h\times d_v^h\times T}$.

5. Model Evaluation

To understand the impact of both the Spatial and Temporal Transformer streams, we analyze their performance separately and in different configurations through extensive experiments on NTU-RGB+D 60 [40] (see Table 1-3). Then, for a comparison with the state-of-the-art, we test the resulting best configurations on the Kinetics dataset [19], which are used by most of previous works, and on the NTU-RGB+D 120 dataset [26], which represents to date one of the most complex skeleton-based action recognition benchmarks (see Table 4-5).

5.1. Datasets

NTU RGB+D 60 and NTU RGB+D 120 The NTU RGB+D 60 (NTU-60) dataset is a large-scale benchmark for 3D human action recognition collected using Microsoft Kinect v2 by [40]. It contains RGB videos, depth sequences, skeleton data, and infrared frames collected in $56,880$ RGB+D video samples. Skeleton information consists of 3D coordinates of 25 body joints and a total of 60 different action classes. The NTU-60 dataset follows two different criteria for evaluation. The first one, called *Cross-View Evaluation* (X-View), uses $37,920$ training and $18,960$ test samples, split according to the camera views from which the action is taken. The second one, called

Cross-Subject Evaluation (X-Sub), is composed instead of $40,320$ training and $26,560$ test samples. Data collection has been performed with 40 different subjects performing actions and divided into two groups, one for training and the other for testing. NTU RGB+D 120 [26] (NTU-120) is an extension of NTU-60, which adds $57,367$ new skeleton sequences representing 60 new actions, for a total of $113,945$ videos referring to 120 classes from 106 subjects under 32 camera setups. In order to perform the evaluation, the extended dataset follows two criteria: the first one is the *Cross-Subject Evaluation (X-Sub)*, the same used for NTU-60, while the second one is called *Cross-Setup Evaluation (X-Set)*, which substitutes Cross-View by splitting training and testing samples based on the parity of the camera setup IDs.

Kinetics The Kinetics skeleton dataset [51] is obtained by extracting skeleton annotations from videos composing the Kinetics 400 dataset [19], by using the OpenPose toolbox [5]. It consists of $240,436$ training and $19,796$ testing samples, representing a total of 400 action classes. Each skeleton is composed by 18 joints, each one provided with the 2D coordinates and a confidence score. For each frame, a maximum of 2 people are selected based on the highest confidence scores. To compare our methods with the literature, Top-1 and Top-5 accuracy are reported.

5.2. Model Complexity

Before studying the accuracy benefits of the proposed ST-TR, we perform an analysis on the complexity of the different self-attention modules we designed, and compare them to ST-GCN modules [51], based on standard convolution, and to 1s-AGCN [42] modules, based on adaptive graph convolution. First, we compare, singularly, a layer of standard convolution with our transformer mechanism, setting $C_{in} = C_{out}$ channels. The number of parameters of each configuration is shown in Figure 5a as a function of the channel dimension. The number of parameters introduced by self-attention is expressed mathematically in Equation 10:

$$SSA_{params} = TSA_{params} = C_{in}C_{out}(2k+v) + C_{out}^2 v^2, \tag{10}$$

where $k = \frac{d_k}{C_{out}} = 0.25$ and $v = \frac{d_v}{C_{out}} = 1$ in our case. This is the result of a $1 \times stride$ standard $2D$ convolution on C_{in} input channels and $2d_k + d_v$ output channels to calculate query, key, and value, and a 1×1 $2D$ convolution combining the output of each head with $C_{in} = C_{out} = d_v$ input and output channels. This results in the same number of parameters for both TSA and SSA, since the convolutions performed internally are the same (having fixed the same kernel dimensions) and both the query-key dot product and the logit-value product are parameter free.

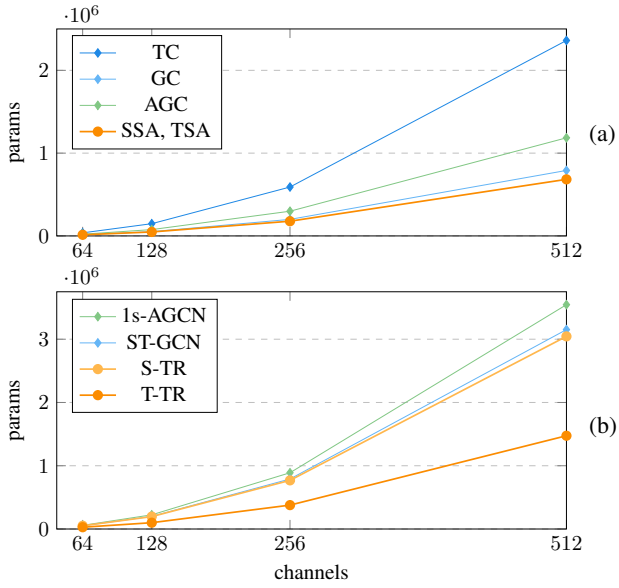


Figure 5. (a) Difference in terms of parameters between Graph Convolution (GC), Adaptive Convolution (AGC) Spatial Self-Attention (SSA) modules of $C_{in} = C_{out}$ channels, and between Temporal Convolution (TC) and Temporal Self-Attention (TSA) modules; (b) comparison in terms of parameters between ST-GCN, 1s-AGCN and our novel S-TR and T-TR. Best viewed in colors.

From Figure 5a it can be seen that Spatial Self-Attention introduces less parameters than Graph Convolution, especially when dealing with a large number of channels, where the maximum $\Delta_{GC--SSA}$, i.e., the decrease in terms of parameters, is 1.1×10^5. When dealing with Adaptive Graph Convolution (AGC), an additional number of parameters has to be considered, resulting in a difference with respect to SSA of $\Delta_{AGC--SSA} = 5 \times 10^5$. On the temporal dimension $\Delta_{TC--TSA}$ reaches a value of 16.8×10^5. Temporal convolution in [51] is implemented as a $2D$ convolution with filter $1 \times F$, where F is the number of frames considered along the time dimension, and it is usually set to 9, striding along $T = 300$ frames. Thus, substituting it with a self-attention mechanism results in a great complexity reduction, in addition to better performance, as reported in the next sections.

Finally, in Figure 5b we also compare the entire stream architectures, i.e., ST-GCN [51] and 1s-AGCN [42] with the proposed S-TR and T-TR streams in terms of parameters. As expected from the considerations above, the biggest improvement in parameters reduction is achieved by substituting temporal convolution with Temporal Self-Attention, i.e., in T-TR, with a $\Delta_{ST-GCN--T-TR} = 16.7 \times 10^5$. On the spatial dimension the difference in terms of parameters is not as pronounced as in temporal dimension, but it is still significant, with a $\Delta_{ST-GCN--S-TR} = 1.07 \times 10^5$ and

Method	GCN	TCN	Params $\times 10^5$	Top-1
ST-GCN	✓	✓	31.0	92.7
ST-GCN-fc	$\mathbf{A}_{fc}$	✓	26.5	93.7
1s-AGCN	$\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k$	✓	34.7	93.7
1s-AGCN w/o A	$\mathbf{B}_k, \mathbf{C}_k$	✓	33.1	93.4
S-TR	SSA	✓	30.7	**94.0**
T-TR	✓	TSA	17.6	93.6

Table 1. Comparison between the baseline and our self-attention modules in terms of both performance (accuracy (%)) and efficiency (number of parameters) on NTU-60 (X-View)

$\Delta_{1s-AGCN--S-TR} = 5.0 \times 10^5$.

5.3. Experimental Settings

Using PyTorch [36] framework, we trained our models for a total of 120 epochs with batch size 32 and SGD as optimizer on NTU-60 and NTU-120, while on Kinetics we trained our models for a total of 65 epochs, with batch size 128. The learning rate is set to 0.1 at the beginning and then reduced by a factor of 10 at the epochs $\{60, 90\}$ and $\{45, 55\}$ for NTU and Kinetics respectively. These schedulings have been selected as they have been shown to provide good results on ST-GCN networks used by [41]. Moreover, we preprocessed the data with the same procedure used by [42] and [41]. In order to avoid overfitting, we also used *DropAttention*, a particular dropout technique introduced by [52] for regularizing attention weights in Transformer netorks, that consists in randomly dropping columns of the attention logits matrix. In all of these experiments, the *number of heads* for multi-head attention is set to 8, and d_q, d_k, d_v embedding dimensions to $0.25 \times C_{out}$ in each layer, as done in [2]. We did not perform grid search on these parameters. As far as it concerns the model architecture, each stream is composed by 9 layers, of channel dimension $64, 64, 64, 128, 128, 128, 256, 256$ and 256. Batch normalization is applied to input coordinates, and a global average pooling layer is applied before the *softmax* classifier and each stream is trained using the standard cross-entropy loss.

5.4. Results

To verify in a fair way the effectiveness of our SSA and TSA modules, we compare the S-TR and T-TR streams individually against the ST-GCN [51] baseline (whose results are reported using our learning rate scheduling) and other models that modify its basic GCN module (see Table 1): (i) *ST-GCN (fc)*: we implemented a version of ST-GCN whose adjacency matrix is composed of all ones (referred as $\mathbf{A}_{fc}$), to simulate the fully-connected skeleton structure underlying our SSA module and verify the superiority of self-attention over graph convolution on the spatial dimension; (ii) *1s-AGCN*: Adaptive Graph Convolutional Network (AGCN) [42] (see Section 3.2), as it demonstrated in the literature to be more robust than standard ST-GCN, in order to remark the robustness of our SSA module over more recent methods; (iii) *1s-AGCN w/o A*: 1s-AGCN without the static adjacency matrix, to verify the effectiveness of our SSA over graph convolution in a similar setting where all the links between joints are exclusively learnt. All these methods use the same implementation of convolution on the temporal dimension (TCN). We make a comparison both in terms of model accuracy and number of parameters.

Method	Bones	X-Sub	X-View
S-TR		86.4	94.0
T-TR		86.0	93.6
ST-TR		88.7	**95.6**
S-TR	✓	87.9	94.9
T-TR	✓	87.3	94.1
T-TR-agcn	✓	86.1	94.3
ST-TR	✓	89.9	**96.1**
ST-TR-agcn	✓	89.3	**96.1**

(a)

Method	X-View
S-TR-all-layers	93.3
T-TR-all-layers	91.3
ST-TR-all-layers	95.0
S-TR-augmented	94.5
T-TR-augmented	90.2
ST-TR-augmented	94.9
ST-TR-1s	93.3

(b)

Table 2. a) Comparison of S-TR and T-TR streams, and the combination of the two (ST-TR) on NTU-60, w and w/o bones. b) Ablations of different model configurations

Regarding SSA, the performance of S-TR is superior to all methods mentioned above, demonstrating that self-attention can be used in place of graph convolution, increasing the network performance while also decreasing the number of parameters. In fact, as it can be seen from Table 1, S-TR introduces 0.3×10^5 parameters less then ST-GCN and 4×10^5 less than 1s-AGCN, with a performance increment w.r.t. all GCN configurations. Similarly, regarding TSA, what emerges from the comparison between T-TR and the ST-GCN baseline adopting standard convolution, is that by using self-attention on the temporal dimension the model is significantly lighter (13.4×10^5 less parameters), and achieves an increment of accuracy of 0.9%.

In Table 2 we first analyze the performance of the S-TR stream, T-TR stream and their combination by using input data consisting of joint information only. As it can be seen from Table 2a, on NTU-60 the S-TR stream achieves slightly better performance (+0.4%) than the T-TR stream, on both X-View and X-Sub. This can be motivated by the fact that SSA in S-TR operates on 25 joints only, while on temporal dimension the number of correlations is proportional to the huge number of frames. Again, as shown in Table 1, applying self-attention instead of convolution clearly benefits the model on both spatial and temporal dimensions. The combination of the two streams achieves 88.7% of ac-

curacy on X-Sub and 95.6% of accuracy on X-View, outperforming the baseline ST-GCN by up to 3% and surpassing other two-stream architectures (see Table 4).

As adding bones information demonstrated to lead to better results in previous works [41, 44], we also studied our Transformer modules on combined joint and bones information. For each node $\mathbf{v}_1 = (\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1)$ and $\mathbf{v}_2 = (\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_2)$, the bone connecting the two is calculated as $\mathbf{b}_{\mathbf{v}_1,\mathbf{v}_2} = (\mathbf{x}_2 - \mathbf{x}_1, \mathbf{y}_2 - \mathbf{y}_1, \mathbf{z}_2 - \mathbf{z}_1)$. Both joint and bone information are concatenated along the channel dimension, and then fed to the network. At each layer, the dimension of the input and output channels are doubled as done by [41] and [44]. Results are shown again in Table 2a, where all previous configurations improve when bones information is added as input. This highlights the flexibility of our method, which is capable of adapting to different input types and network configurations.

To further test its flexibility, we also perform additional experiments in which the GCN module is substituted by the AGCN adaptive module on the temporal stream. As it can be seen from Table 2a, these configurations (*T-TR-agcn*) achieve better results than the one using standard GCN (T-TR-agcn: 94.3%, T-TR: 94.1%) on X-View.

5.5. Effect of Applying Self-Attention since Feature Extraction

We designed our streams to operate starting from high-level features, rather than directly from coordinates, extracted using a sequence of residual GCN and TCN modules as reported in Section 4.4. This set of experiments validates our design choice. In these experiments SSA (TSA) substitutes GCN (TCN) on the S-TR (T-TR) stream, from the very first layer. The configurations reported in Table 2b (named *S-TR-all-layers*), perform worse than the corresponding ones in Table 2a, while still outperforming the baseline ST-GCN [41] by 2.3% (see Table 3). Notice that on T-TR, in order to deal with the great number of frames in the very first layers ($T = 300$), we divided frames into blocks within which SSA is applied, and then gradually reduce the number of blocks going deeper in the architecture ($d_{block} = 10$ where $C_{out} = 64$, $d_{block} = 10$ where $C_{out} = 128$, and a single block of $d_{block} = T^l$ on layers l with $C_{out} = 256$).

5.6. Effect of Augmenting Convolution with Self-Attention

Motivated by the results in [2], we studied the effect of applying the proposed Transformer mechanism as an augmentation procedure to the original ST-GCN modules. In this configuration, $0.75 \times C_{out}$ features result from GCN (TCN) and they are concatenated to the remaining $0.25 \times C_{out}$ features from SSA (TSA), a setup that has proven to be effective in [2]. To compensate the reduction of attention channels, *wide attention* is used, i.e., half of the attention channels are assigned to each head, then recombined together while merging heads. The results are reported in Table 2b (referred as *ST-TR-augmented*). Graph convolution is the one that benefits the most from SSA attention (S-TR-augmented, 94.5%), to be compared with S-TR's 94% in Table 2a. Nevertheless, the lower number of output features assigned to self-attention prevent temporal convolution improving on T-TR stream.

NTU-60			
Method	**Bones**	X-Sub	X-View
STA-LSTM [45]		73.4	81.2
VA-LSTM [54]		79.4	87.6
AGC-LSTM [43]		89.2	95.0
ST-GCN [51]		81.5	88.3
1s-AGCN [42]		86.0	93.7
1s Shift-GCN [6]		87.8	95.1
SAN [8]		87.2	92.7
ST-TR (Ours)		**88.7**	**95.6**
2s-AGCN [42]	✓	88.5	95.1
DGCNN [41]	✓	89.9	96.1
2s Shift-GCN [6]	✓	89.7	96.0
MS-G3D [32]	✓	**91.5**	**96.2**
ST-TR (Ours)	✓	89.9	96.1
ST-TR-agcn (Ours)	✓	89.3	96.1

Table 3. Comparison with state-of-the-art accuracy (%) on NTU60

5.7. Effect of combining SSA and TSA in a single stream

We tested the efficiency of the model when SSA and TSA are combined in a single stream architecture (see Table 2b, referred as *S-TR-1s*). In this configuration, feature extraction is still performed by the original GCN and TCN modules, while from the 4th layer on, each layer is composed by SSA followed by TSA, i.e., **ST-TR-1s**$(x) =$ **TSA**(**SSA**(x)).

We also tested this configuration on NTU-60, obtaining an accuracy of 93.3%, slightly lower than the 95.6% accuracy obtained by the two-stream configuration (see Table 1, ST-TR). However, it should be noted that the S-TR-1s configuration presents 17.4×10^5 parameters, drastically reducing the complexity of the baseline ST-GCN which consists in 31×10^5 parameters. Nevertheless, it outperforms the ST-GCN baseline by 0.6% using half of the parameters.

6. Comparison with State-Of-The-Art Results

In addition to NTU-60, we compare our methods on NTU-120 and Kinetics, for a fair comparison, w.r.t. other methods making use of joint or joint+bones information on

NTU-120		
Method	X-Sub	X-Set
ST-LSTM [27]	55.7	57.9
GCA-LSTM [28]	61.2	63.3
RotClips+MTCNN [20]	62.2	61.8
Pose Evol. Map [31]	64.6	66.9
1s Shift-GCN [6]	80.9	83.2
S-TR (Ours)	78.6	80.7
T-TR (Ours)	78.4	80.5
T-TR-agcn (Ours)	80.3	81.8
ST-TR (Ours)	81.9	84.1
ST-TR-agcn (Ours)	**82.7**	**84.7**

Table 4. Comparison with state-of-the-art accuracy (%) of S-TR, T-TR, and their combination (ST-TR) on NTU-120

Kinetics			
Method	**Bones**	Top-1	Top-5
ST-GCN [51]		30.7	52.8
SAN [8]		35.1	55.7
S-TR (Ours)		32.4	55.3
T-TR (Ours)		32.4	55.2
ST-TR (Ours)		34.5	57.6
2s-AGCN [42]	✓	36.1	58.7
DGCNN [41]	✓	36.9	59.6
MS-G3D [32]	✓	**38.0**	**60.9**
S-TR (Ours)	✓	35.4	57.9
T-TR (Ours)	✓	33.1	55.86
T-TR-agcn (Ours)	✓	33.7	55.1
ST-TR (Ours)	✓	37.0	59.7
ST-TR-agcn (Ours)	✓	37.4	59.8

Table 5. Comparison with state-of-the-art accuracy (%) of S-TR, T-TR, and their combination (ST-TR) on Kinetics

a one- or two-stream architecture, as we also did. On NTU-120 (Table 4), the model based on joint information only, achieves an accuracy of 82.7% on X-Sub and 84.7% on X-Set, outperforming all state-of-the-art methods that use the same information. On Kinetics and NTU-60, we test our model by using also bones information. On Kinetics (Table 5), our model using only joints outperforms the ST-GCN baseline by 3.8%. When using bones information, the best configuration that uses AGCN instead of GCN (ST-TR-agcn) as a backbone outperforms the baseline 2s-AGCN by 1.3%, and DGCNN [41] by 0.5%. On NTU-60 (Table 3), our configuration that uses joint information only, outperforms all the state-of-the-art models using the same type of information. In particular, it outperforms SAN [8], another method employing self-attention in skeleton-based action recognition, by up to 2.9%, and the baseline 1s-AGCN by up to 1.9%. Finally, when using bones information, both our configurations, the one based on the standard ST-GCN and the one making use of the AGCN backbone, outperform the 2s-AGCN baseline, with ST-TR performing the best and achieving an improvement up to 1.4%.

7. Conclusions

In this paper we propose a novel approach that introduces Transformer self-attention in skeleton activity recognition as an alternative to graph convolution. Through extensive experiments on NTU-60, NTU-120 and Kinetics, we demonstrated that our Spatial Self-Attention module (SSA) can replace graph convolution, enabling more flexible and dynamic representations. Similarly, Temporal Self-Attention module (TSA) overcomes the strict locality of standard convolution, enabling the extraction of long-range dependencies between joints in the action. Moreover, our final Spatial-Temporal Transformer network (ST-TR) achieves state-of-the-art performance on NTU-RGB+D w.r.t. methods using same input information and stream setup, and competitive results on Kinetics with no major hyperparameter tuning.

As combining exclusively self-attention modules revealed to be suboptimal w.r.t. using them separately on two different streams, a possible future work is to search for a fully self-attentional solution, leading to a unified Transformer architecture able to replace graph convolutional networks in a variety of tasks.

References

[1] Jake K Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):1–43, 2011. 1

[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*, pages 3286–3295, 2019. 2, 3, 6, 8, 9

[3] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 2

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014. 2

[5] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 1, 7

[6] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 183–192, 2020. 1, 9, 10

[7] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. *Proceedings of the IEEE international conference on computer vision*, pages 3218–3226, 2015. 2

[8] Sangwoo Cho, Muhammad Maqbool, Fei Liu, and Hassan Foroosh. Self-attention network for skeleton-based human action recognition. 2020. 2, 9, 10

[9] Xiao Chu, Wei Yang, Wanli Ouyang, Cheng Ma, Alan L Yuille, and Xiaogang Wang. Multi-context attention for human pose estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1831–1840, 2017. 1

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, pages 3844–3852, 2016. 2

[11] Zewei Ding, Pichao Wang, Philip O Ogunbona, and Wanqing Li. Investigation of different skeleton features for cnn-based 3d action recognition. *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 617–622, 2017. 2

[12] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1110–1118, 2015. 2

[13] Silvio Giancola, Matteo Valenti, and Remo Sala. *A survey on 3D cameras: Metrological comparison of time-of-flight, structured-light and active stereoscopy technologies*. Springer, 2018. 1

[14] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 2:729–734, 2005. 2

[15] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 2

[16] Jian-Fang Hu, Wei-Shi Zheng, Jianhuang Lai, and Jianguo Zhang. Jointly learning heterogeneous features for rgb-d activity recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5344–5352, 2015. 2

[17] Mohamed E Hussein, Marwan Torki, Mohammad A Gowayyed, and Motaz El-Saban. Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations. *Twenty-third international joint conference on artificial intelligence*, 2013. 2

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 6

[19] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 6, 7

[20] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. Learning clip representations for skeleton-based 3d action recognition. *IEEE Transactions on Image Processing*, 27(6):2842–2855, 2018. 10

[21] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. July 2017. 1

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations, ICLR*, 2017. 2, 3

[23] Guy Lev, Gil Sadeh, Benjamin Klein, and Lior Wolf. Rnn fisher vectors for action recognition and image annotation. *European Conference on Computer Vision*, pages 833–850, 2016. 2

[24] Bo Li, Yuchao Dai, Xuelian Cheng, Huahui Chen, Yi Lin, and Mingyi He. Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep cnn. *ICMEW*, 2017. 2

[25] Yuan Li, Xiaodan Liang, Zhiting Hu, Yinbo Chen, and Eric P. Xing. Graph transformer. *OpenReview*, 2019. 3

[26] Jun Liu, Amir Shahroudy, Mauricio Lisboa Perez, Gang Wang, Ling-Yu Duan, and Alex Kot Chichung. Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 6, 7

[27] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. *European conference on computer vision*, pages 816–833, 2016. 10

[28] Jun Liu, Gang Wang, Ling-Yu Duan, Kamila Abdiyeva, and Alex C Kot. Skeleton-based human action recognition with global context-aware attention lstm networks. *IEEE Transactions on Image Processing*, 27(4):1586–1599, 2017. 10

[29] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot. Global context-aware attention lstm networks for 3d action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1647–1656, 2017. 2

[30] Mengyuan Liu, Hong Liu, and Chen Chen. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition*, 68:346–362, 2017. 2

[31] Mengyuan Liu and Junsong Yuan. Recognizing human actions as the evolution of pose estimation maps. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1159–1168, 2018. 10

[32] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 143–152, 2020. 1, 9, 10

[33] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009. 2

[34] Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019. 6

[35] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. *International conference on machine learning*, pages 2014–2023, 2016. 2

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. pages 8026–8037, 2019. 8

[37] Bin Ren, Mengyuan Liu, Runwei Ding, and Hong Liu. A survey on 3d skeleton-based action recognition using learning method. *arXiv preprint arXiv:2002.05907*, 2020. 1

[38] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition with kinect sensor. *Proceedings of the 19th ACM international conference on Multimedia*, pages 759–760, 2011. 1

[39] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008. 2

[40] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016. 6

[41] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7912–7921, 2019. 1, 2, 5, 8, 9, 10

[42] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12026–12035, 2019. 2, 3, 5, 7, 8, 9, 10

[43] Chenyang Si, Wentao Chen, Wei Wang, Liang Wang, and Tieniu Tan. An attention enhanced graph convolutional lstm network for skeleton-based action recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1227–1236, 2019. 9

[44] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, pages 568–576, 2014. 1, 2, 9

[45] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, page 4263–4270, 2017. 9

[46] Felipe Petroski Such, Shagan Sah, Miguel Domínguez, Suhas Pillai, Chao Zhang, Andrew Michael, Nathan D. Cahill, and Raymond W. Ptucha. Robust spatial filtering with graph convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 2017. 2

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 3

[48] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 588–595, 2014. 2

[49] Hongsong Wang and Liang Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 499–508, 2017. 2

[50] Lei Wang, Du Q Huynh, and Piotr Koniusz. A comparative review of recent kinect-based action recognition algorithms. *IEEE Transactions on Image Processing*, 29:15–28, 2019. 2

[51] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *Thirty-second AAAI conference on artificial intelligence*, 2018. 1, 2, 3, 5, 6, 7, 8, 9, 10

[52] Lin Zehui, Pengfei Liu, Luyao Huang, Jie Fu, Junkun Chen, Xipeng Qiu, and Xuanjing Huang. Dropattention: A regularization method for fully-connected self-attention networks. *arXiv preprint arXiv:1907.11065*, 2019. 8

[53] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. A comprehensive survey of vision-based human action recognition methods. *Sensors*, 19(5):1005, 2019. 1

[54] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2117–2126, 2017. 9

[55] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012. 1