SC4D: A Sparse 4D Convolutional Network for Skeleton-Based Action Recognition

Lei Shi[1,2], Yifan Zhang[1,2], Jian Cheng[1,2,3] and Hanqing Lu[1,2]

[1]NLPR & AIRIA, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
[3]CAS Center for Excellence in Brain Science and Intelligence Technology

Abstract. In this paper, a new perspective is presented for skeleton-based action recognition. Specifically, we regard the skeletal sequence as a spatial-temporal point cloud and voxelize it into a 4-dimensional grid. A novel sparse 4D convolutional network (SC4D) is proposed to directly process the generated 4D grid for high-level perceptions. Without manually designing the hand-crafted transformation rules, it makes better use of the advantages of the convolutional network, resulting in a more concise, general and robust framework for skeletal data. Besides, by processing the space and time simultaneously, it largely keeps the spatial-temporal consistency of the skeletal data, and thus brings better expressiveness. Moreover, with the help of the sparse tensor, it can be efficiently executed with less computations. To verify the superiority of SC4D, extensive experiments are conducted on two challenging datasets, namely, NTU-RGBD and SHREC, where SC4D achieves state-of-the-art performance on both of them.

Keywords: Skeleton, Action Recognition, 4D Convolution.

1 Introduction

Action recognition is a popular research topic because it can be applied to many practical fields such as human-computer interaction and video surveillance [2,22,5,19]. In recent years, skeleton-based action recognition has drawn considerable attentions due to its small amount of data, higher-level semantic information and strong robustness for complicated environment [32,12,21]. In detail, the skeletal data is generally a sequence of frames each contains the position information of the human body joints, which is expressed as the 2D/3D coordinates of the camera coordinate system. It totally removes the background information, and thus it focuses more on the human body itself. Recently, with the success of the deep leaning, the data-driven methods have become the mainstream for skeleton-based action recognition. In most of the existing neural-network-based approaches, the joint coordinate is viewed as the attribute of each element, and various hand-crafted strategies are designed to transform these joints into various specific forms such as pseudo-images or graphs to feed them into RNNs [8,26], CNNs [12,1] or GCNs [32,20] for feature extraction.

Generally speaking, the GCN-based method, which structures the skeletal data as a spatiotemporal graph and feeds it into the graph convolutional network, shows great effectiveness and better performance. However, recent works found that parameterizing the graph topology performs better than using the fixed human-body-based graph [21,9]. It means the prior-knowledge-based graph structure is not the key of the success of the GCN-based method for skeletal data. Its success may largely owes to the great generalizability and robustness of the convolutional network. Thus, the GCN-based method is essentially the same with the CNN-based method, which tries to design rules to transform the skeletal data into a CNN-suitable form. However, there are three limitations for these methods: (1) Manually designed rules are not guaranteed to be an optimal choice due to the human factor. Although some works propose to learn these rules in the training process, it still more or less limits the flexibility of the model. For example, some works try to learn the graph topology, but the graph is still shifted based on the human-body-based graphs and is limited by the hand-crafted graph generation mechanism [21,9]. Besides, the raw skeletons exist in the metric space (3D coordinate system) with a class-specific pattern, which are naturally suitable for grid-based convolutional networks. It is unnecessary to still force a learning of the transformation rules for convolutions in such a data-driven framework. (2) Skeletal data is more fixed compared with other graph-structured data such as social networks or physical molecules. Its every element possesses a specific physical/semantic meaning, i.e., different human joints such as feet or hands, which is constant for various data samples. However, the convolution is weight-shared. Thus, the fixed rules force the convolutional model to mine constant interaction patterns for different joints and its neighbors of different samples, which limits the model's flexibility and capacity. There are also some works trying to avoid constant contributions by permuting the arrangement of joints [1] or multiplying attention weights for every joints [32], but they are treating symptoms and not the root cause. (3) In previous works for skeletal data, the position information is employed as features while the hand-crafted rules are used to organize convolutional operations. It is somewhat conflicted with the translation-invariant character of the convolutional network. Since the input is position-variant, once the skeleton is shifted a little bit, the extracted features also become completely different. This reduces the robustness and the generalizability of the model.

Instead, a new perspective is proposed in this work for skeleton-based action recognition. We regard the skeletal sequence as a spatial-temporal point cloud and voxelize it into a 4D grid. Similar to the RGB values of image pixels, each voxel is attached with a feature vector that denotes whether there is a joint in this position and which joint it is. Different with previous works, it needs no manual designs of transformation rules and can better utilize the power of the data-driven mechanism. By performing convolution on 4D grid, the weight-sharing mechanism is no longer worked for the fixed relation structure, which avoids constant contributions for different joints and its neighbors of different samples. Besides, the position information is not included in the feature vector, and thus it is more robust for position translation.

Besides, different with most existing methods that process the skeletal sequence frame by frame, we directly construct a 4D convolutional network to process the generated 4D grid. It hierarchically extracts the spatial-temporal features from low-levels to high-levels, which largely keeps the spatial-temporal consistencies of the skeletal data. To construct 4D convolutional networks, a naive method can be directly expanding the 3D convolutional kernels to 4D. However, due to the additional temporal dimension of the kernel, it has a large amount of computational cost. Since the skeletal data is very sparse, we propose to employ the sparse tensor and the sparse convolution to reduce the computational cost inspired by [4]. It only tracks the non-empty voxels of the 4d grid, along with their position information and associated feature vectors. The convolutional operation is performed on these sparse voxels hierarchically, which generates the sparse output accordingly. It can generate similar results compared with the dense convolution, but it is more efficient and needs less computations.

Moreover, to further improve the performance, we propose three data augmentation techniques. First, we suggest to interpolate points along bones, which can utilize the prior knowledge of the human body and make it more distinctive. The number of interpolations are determined based on the average length of the bones in the human body. Second, since the skeletal joint is very sparse, a dilation technique is used to enhance the spatial pattern to ease the recognition. The values of the new added points are reduced proportionally to distinguish them from the original joints. Third, as shown in previous methods [21], the bone information and the motion information are effective for skeleton-based action recognition. To also exploit them for our voxelization-based methods, we propose to transform the input from the coordinate space into the bone space and the motion space. By modeling the information of the three spaces with multiple streams and finally fusing the results, the performance is further improved.

The proposed method, namely, the sparse 4D convolutional network (SC4D), is a general framework for skeletal data. To verify the effectiveness of SC4D, extensive experiments are conducted on two popular datasets for different tasks, i.e., NTU-RGBD for action recognition and SHREC for gesture recognition. Although it is the first try to voxelize skeletons into a 4D grid, SC4D achieves state-of-the-art performance on both datasets, which illustrates that it is an effective method and a valuable-researched perspective.

Overall, our contributions lie in three aspects:

1. We propose a new perspective for skeleton-based action recognition, where the skeletal data is voxelized into a 4d grid and is directly processed with a sparse 4D convolutional network. The proposed framework is effective, concise, robust, and efficient.
2. Two data enhancement strategies are proposed to augment the generated skeletal grid to ease the recognition. We further transform the coordinate data into two other spaces, i.e., the bone space and the motion space, to utilize their complementarity to better recognize the human action.

3. The final model achieves state-of-the-art performance on two challenging datasets for different tasks. The code will be released to facilitate future works.

2 Related works

2.1 Skeleton-based action recognition

Skeleton-based action recognition has been studied for decades. Early-stage approaches concentrate on designing variable hand-crafted features [31]. With the rise of deep learning, the mainstream approaches in recent years lie in three aspect: (1) the RNN-based approaches where the skeleton sequence is fed into the RNN models and the joints are modeled in a predefined traversal order [35,10,25,24]. (2) the CNN-based approaches where the skeleton sequence is transformed into a pseudo-image and are fed into the CNNs for recognition [8,12,1]. (3) the GCN-based approaches where the skeleton sequence is encoded into a spatiotemporal graph according to the physical structure of the human body and is modeled with GCNs [32,28,21,20]. In contrast to previous works, we project the sparse joints into a 4D grid and employ a sparse 4D convolutional network to extract features and make predictions.

2.2 Perception tasks based on 3D point cloud

Here, since we regard the skeletal data as point clouds, we introduce some recent works for perception tasks based on the 3D point cloud. There have been many works investigating how to model the point cloud with deep neural networks. Some works exploit the metric space distance to aggregate features from local to global in a hierarchical manner [17,23]. Some works transform the original data into other forms such as surface [14], octree [18] or sparse lattice [27], which are modeled with low-dimension neural networks. Some works perform the volumetric representation for point cloud. For example, Tchapmi et al. [29] project the raw point cloud data into 3D volumes and employ 3D convolutional networks for 3D segmentation. It should be note that for many tasks, the point cloud is video-based, and thus have an additional temporal dimension. For these tasks, one strategy is to process the video frames sequentially and finally aggregate the temporal features. For example, You et al. [34] represent multiple peoples actions captured by depth cameras as a sequence of point-cloud-based volume and process the volumes frame by frame with a 3D convolutional network. Another strategy is to directly model the videos in a 4D manner. For example, Choy et al. [4] project the sequence of 3D scans into a 4D tesseract and introduce a 4D sparse spatiotemporal convolutional network to extract features for 3D video segmentation. This work follows the voxelization-based methods.

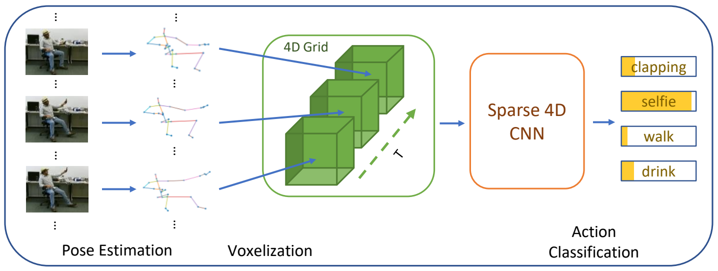

Fig. 1. Pipeline overview.

3 Methods

3.1 Pipeline Overview

Fig. 1 shows the pipeline of our method. Skeletal data can be obtained by motion-capture devices or pose estimation algorithms from videos. It is first normalized and voxelized into a 4D grid. Then the generated grid is directly processed with a sparse 4D convolutional network, which hierarchically extracts semantics from the 4D grid and outputs the high-level feature maps. Finally these feature maps are global-average-pooled and classified with the SoftMax classifier. The following sections will go over these steps.

3.2 Voxelization

The raw skeletal data is a sequence of frames, each of which records the Cartesian coordinates of the human joints in the current frame. It is formulated as a set of joint coordinates as $\{\mathbf{R}^{raw}_{t,m,n} : \mathbf{R}^{raw}_{t,m,n} \in \mathbb{R}^{C_{coor}}, t = 1, 2, \cdots, T, m = 1, 2, \cdots, M, n = 1, 2, \cdots, N\}$, where T, M and N denote the number of frames, persons and joints defined in the data acquisition system, respectively. In the rest of the paper, C_{coor} is default to 3, i.e., the joints are obtained in a 3D coordinate system.

Voxelizing these joint coordinates into a 4D grid is equivalent to calculating the new coordinates of these joints in the 4D-grid-based coordinate system. First, we add the temporal coordinate for every joints, i.e., $\mathbf{R}^{raw}_{t,m,n} \in \mathbb{R}^3 \rightarrow \mathbf{R}^{4d}_{t,m,n} \in \mathbb{R}^4$. The dimension order of the 4D coordinate system is default to time-z-y-x, which means $\mathbf{R}^{4d}_{t,m,n}(1)$, $\mathbf{R}^{4d}_{t,m,n}(2)$, $\mathbf{R}^{4d}_{t,m,n}(3)$ and $\mathbf{R}^{4d}_{t,m,n}(4)$ denote the t-coordinate, z-coordinate, y-coordinate and x-coordinate, respectively. The temporal coordinate is set as the index of frames, i.e., $\mathbf{R}^{4d}_{t,m,n}(1) = t$.

Then, we normalize the coordinate into the range of zero to the grid size. In detail, we use $\mathbf{S} \in \mathbb{R}^4$ to denote the grid shape, e.g., $\mathbf{S} = [32, 32, 32, 32]$. $\mathbf{S}(i)$ denotes the grid size of the coordinate dimension i, where $i = 1, 2, 3, 4$. $\mathbf{R}^{4d}_{t,m,n}$

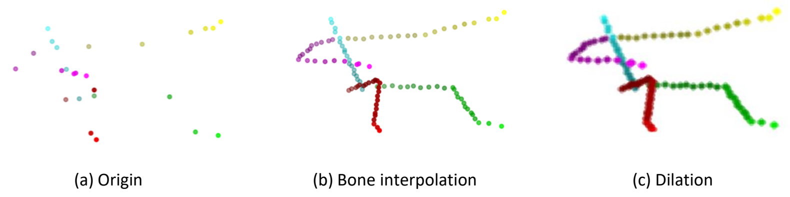

Fig. 2. Examples of two data enhancement techniques: bone interpolation and dilation.

is normalized to $\mathbf{R}^{nor}_{t,m,n}$ by:

$$\mathbf{R}^{nor}_{t,m,n}(i) = \frac{\mathbf{R}^{4d}_{t,m,n}(i) - \min\limits_{\forall t,m,n}\{\mathbf{R}^{4d}_{t,m,n}(i)\}}{\max\limits_{\forall t,m,n}\{\mathbf{R}^{4d}_{t,m,n}(i)\} - \min\limits_{\forall t,m,n}\{\mathbf{R}^{4d}_{t,m,n}(i)\}} \times S(i) \tag{1}$$

Finally, since the coordinate should be integer, the final coordinate of the joints in the 4d grid $\mathbf{R}^{jpt}_{t,m,n}$ is obtained by $\mathbf{R}^{jpt}_{t,m,n} = floor(\mathbf{R}^{nor}_{t,m,n} + 0.5)$, where floor means rounding down float coordinates to integers.

3.3 Feature Representation

After the voxelization process, we know the positions of all the joints in the 4D grid. Now we attach a feature vector for every joints to identify it. This is similar with the word embedding process of the NLP field, where words or phrases from the vocabulary are mapped to vectors of real numbers. In detail, $\mathbf{R}^{jpt}_{t,m,n}$ is attached with a feature vector $\mathbf{F}^{jpt}_{t,m,n} \in \mathbb{R}^{C_{fea}}$. In this work, we propose three strategies for embedding. The first strategy is setting the feature of the voxels that are occupied by joints as 1 and setting the feature of other voxels as 0. It means $C_{fea} = 1$. However, this strategy cannot tell which joint occupies the current voxel, and thus the information of the joint semantics is lost. The second strategy is $C_{fea} = M + N$. The first M elements construct a one-hot vector that indicates whether the current voxel is occupied by the joint of the m_{th} person. Similarly, the last N elements indicates whether it is the n_{th} joint. In formulation, $\mathbf{F}^{jpt}_{t,m,n}(i) = \mathcal{I}\{i == m || i == M + n\}$, where $i = 1, 2, \cdots, C_{fea}$. $||$ denotes “or”. If the expression is true, $\mathcal{I}\{expression\} = 1$, otherwise $\mathcal{I}\{expression\} = 0$. However, it can not distinguish the situation that the joints of multiple persons falling into a same voxel. The third strategy is $C_{fea} = MN$. If the current voxel is the n_{th} joint of the m_{th} person, $\mathbf{F}^{jpt}_{t,m,n}(i) = 1$, otherwise $\mathbf{F}^{jpt}_{t,m,n}(i) = 0$. In formulation, $\mathbf{F}^{jpt}_{t,m,n}(i) = \mathcal{I}\{i == (n + m \times N)\}$. The third strategy is competent for more situations, but it needs more data volumes. In the following paper, we use the third strategy.

3.4 Data Enhancement

The number of human joints are generally small. We propose two spatial feature enhancement techniques to augment the local patterns as shown in Fig. 2. The first strategy is interpolating points along the human bones, which are defined as the natural connections of the human body. Usually the human body can be viewed as a tree structure [33,11]. Thus, the number of bones is one less than the number of joints. For example, Fig. 4 shows the definition of the joints and bones for the two datasets. First, we count the average length of every bones in the dataset. The bone with the shortest length is defined as the basic bone, which means the number of interpolated points for this bone is one. Then, the number of interpolated points for other bones are determined by the multiples of their lengths to the length of the basic bone. The points are uniformly interpolated in the segment between the two end joints of the bone. For example, if the length of the basic bone is 2 and the length of the forearm is 10, we will uniformly interpolate 5 points in the segment between the wrist and the elbow. As for the features of the new interpolated points, they are calculated by the weighted sum of the features of two end joints, where the weight is inversely proportional to its distance from the corresponding end joint. In formulation, the number of bones is $B = N - 1$. The number of the interpolated joints for every bones, i.e., $\mathbf{J} \in \mathbb{R}^B$, is calculated by above strategies. $\mathbf{J}(b)$ denotes the number of the interpolated joints of the b_{th} bone, where $b = 1, 2, \cdots, B$ denotes the bone index. A map $\mathbf{I} \in \mathbb{R}^{B\times 2}$ is defined to record the indices of the two end joints of every bones. $\mathbf{I}(b,1)$ and $\mathbf{I}(b,2)$ are the indices of two end joints of the b_{th} bone, i.e., $\mathbf{I}(b,1), \mathbf{I}(b,2) \in \{1, 2, \cdots, N\}$. The coordinate and the feature of the interpolated point are represented as $\mathbf{R}^{inter}_{t,m,b,j} \in \mathbb{R}^4$ and $\mathbf{F}^{inter}_{t,m,b,j} \in \mathbb{R}^{C_{fea}}$, respectively, where $j = 1, 2, \cdots, \mathbf{J}(b)$ denotes the indices of the interpolated points of the b_{th} bone. Given $\mathbf{R}^{jpt}_{t,m,n}$ and $\mathbf{F}^{jpt}_{t,m,n}$, they are calculated as:

$$\begin{aligned}\mathbf{R}^{inter}_{t,m,b,j} &= \mathbf{R}^{jpt}_{t,m,\mathbf{I}(b,1)} + floor(\frac{\mathbf{R}^{jpt}_{t,m,\mathbf{I}(b,2)} - \mathbf{R}^{jpt}_{t,m,\mathbf{I}(b,1)}}{\mathbf{J}(b)+1} \times j + 0.5) \\ \mathbf{F}^{inter}_{t,m,b,j} &= \mathbf{F}^{jpt}_{t,m,\mathbf{I}(b,1)} \times (1 - \frac{j}{\mathbf{J}(b)}) + \mathbf{F}^{jpt}_{t,m,\mathbf{I}(b,2)} \times \frac{j}{\mathbf{J}(b)}\end{aligned} \quad (2)$$

The second strategy is the spatial dilation. We expand the coordinate of one point along all its spatial dimensions according to the dilation number. Using the 1D data as an example, assume there is one point in position 2, after dilating it with the dilation number 1, there will be three points in the position 1, 2 and 3. For simple, when performing dilation on one dimension, the coordinates of other dimensions are kept the same as before. The features of the new dilated points are the same with the original points, but it is divided by a scale, which is inversely proportional to its distance from the original point. In formulation, given the dilation value Σ, the number of the new added points of one person in one frame along the coordinate dimension c is $D_c = 2\Sigma N'$. $c \in \{2, 3, 4\}$ because that dilation is only performed along spatial dimensions. N' denotes the number of the original points, e.g., $N' = N + \sum_b^B \mathbf{J}(b)$ if the bone interpolation is performed. Then, the coordinate and the feature of the dilated point are represented as $\mathbf{R}^{dil}_{t,m,c,d} \in \mathbb{R}^4$ and $\mathbf{F}^{dil}_{t,m,c,d} \in \mathbb{R}^{C_{fea}}$, where $d = 1, 2, \cdots, D_c$ denotes the index of the new dilated points along the coordinate dimension c. Given $\mathbf{R}^{jpt}_{t,m,n}$ and

$\mathbf{F}^{jpt}_{t,m,n}$, they are calculated as:

$$\begin{aligned}
\sigma &= ceil(\frac{(d-1)\%(2N')+1}{2}) \\
\phi &= ceil(\frac{d}{2\sigma}) \\
\mathbf{R}^{dil}_{t,m,c,d}(i) &= \mathbf{R}^{jpt}_{t,m,\phi}(i) + (-1)^d \times \sigma \times \mathbb{I}\{i == c\} \\
\mathbf{F}^{dil}_{t,m,c,d} &= \mathbf{F}^{jpt}_{t,m,\phi}/\sigma
\end{aligned} \tag{3}$$

where $i = 1, 2, 3, 4$ denotes the index of the coordinate dimension of the dilated points. $\sigma \in \{1, 2, \ldots, \varSigma\}$ denotes the distance from the new dilated points to the original point. $\phi \in \{1, 2, \ldots, N\}$ denotes the index of the corresponding original point in the current dilation step. % denotes calculating remainder. Ceil means rounding up the float coordinates to integers.

3.5 Dense Convolution versus Sparse Convolution

After the voxelization and the data enhancement, we remove the duplicate points that fall into the same voxel. In detail, we simply keep the first one during the traversal of points. Now the raw skeletal joint coordinate has been transformed into a coordinate vector of the 4d grid $\mathbf{R}^{in}_n \in \mathbb{R}^4$ and a corresponding feature vector $\mathbf{F}^{in}_n \in \mathbb{R}^{C_{fea}}$, where $n = 1, 2, \cdots, N_{in}$, N_{in} denotes the total number of points after performing the data enhancement techniques and removing duplications. Then, the problem is how to process them with neural networks. The conventional method is updating the traditional 3D CNNs to 4D CNNs by expanding the convolutional kernel dimension to 4, which we called dense convolution. Specifically, we create a 5D tensor $\mathbf{F}^{den} \in \mathbb{R}^{\mathbf{S}(1)\times\mathbf{S}(2)\times\mathbf{S}(3)\times\mathbf{S}(4)\times C_{fea}}$ to represent a 4D grid and fill its values based on $\mathbf{R}^{in}_n$ and $\mathbf{F}^{in}_n$. $\mathbf{S} \in \mathbb{R}^4$ denotes the grid shape. $\mathbf{F}^{den}$ can be directly fed into a 4D CNN.

However, because most of the elements in the generated 4D grid are 0, it is unnecessary to perform convolutions for every elements, which in practice causes the waste of computations and GPU memory. Instead, we consider to use the sparse tensor to reduce the computations. We follow the method introduced in [4]. To perform convolution or other operations sparsely, the key is to obtain a mapping $\mathcal{M}$ to identify which input affects which output according to the input coordinates $\mathbf{R}^{in}_n$ and the operation definitions. It is defined as pairs of lists of input indices $\mathbf{I}^{in} \in \mathbb{R}^{N_{in}}$, output indices $\mathbf{I}^{out} \in \mathbb{R}^{N_{out}}$ and weight indices $\mathbf{I}^{wei} \in \mathbb{R}^{N_{wei}}$ (optional), i.e., $\mathcal{M} = \{(\mathbf{I}^{in}(i), \mathbf{I}^{out}(j), \mathbf{I}^{wei}(k))\}$ where $i \in \{1, 2, \cdots, N_{in}\}$, $j \in \{1, 2, \cdots, N_{out}\}$ and $k \in \{1, 2, \cdots, N_{wei}\}$. Then the output is calculated according to the inputs ($\mathbf{R}^{in}_n$ and $\mathbf{F}^{in}_n$), the weight ($\mathbf{W}_{(opt.)}$), the mapping ($\mathcal{M}$) and the definition of the operation (f), i.e., $\mathbf{R}^{out}_n, \mathbf{F}^{out}_n \longleftarrow f(\mathbf{R}^{in}_n, \mathbf{F}^{in}_n, \mathbf{W}_{(opt.)}, \mathcal{M})$. For convolutional operation, the input features are multiplied with the corresponding weights, and then added to the corresponding output features based on $\mathcal{M}$. For pooling-based operations such as max-pooling and global-average-pooling, weights are not needed. The input features are gathered and directly reduced based on $\mathcal{M}$ to get the output features. For non-spatial functions such as Batch Normalization and ReLu, we can directly use the 1D dense operation on the input features.

3.6 SC4D

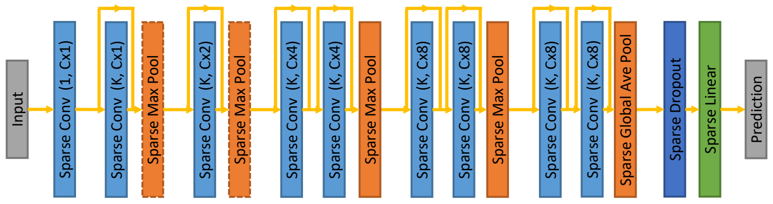

Fig. 3. Architecture of the SC4D. Each convolution is appended with a Batch Normalization layer and a ReLU layer. K denote the kernel size. C denote the basic number of filters.

After defining these operations, we can now build the network only with generalized sparse operations. With extensive experiments, the architecture of the sparse 4D convolutional network (SC4D) for skeleton-based action recognition is built as shown in Fig 3. It is inspired by the C3D network [30]. All layers are built with sparse operations. There are totally 9 sparse convolutional layers. Each sparse convolution is appended with a sparse Batch Normalization layer and a sparse ReLU layer. The kernel size is 1 for the first layer and K for others. The number of filters are C, C, $2C$, $4C$, $4C$, $8C$, $8C$, $8C$, $8C$, respectively. Both K and C can be adjusted to balance the model size and the performance. The first sparse convolutional layer serves as an embedding layer to embed the one-hot features into the feature space. The residual connections are added for every convolutions except for the first one to ease the gradient passing following [6]. Sparse max-pooling layer is added after the 2_{nd}, 3_{rd}, 5_{th} and 7_{th} convolutional layers. The stride of the max-pooling is 2 for all dimensions. If the input grid size is too small, the first several pooling layers will be removed to save the resolution. A sparse global-average-pooling layer and a sparse fully-connected layer is added in the end to make predictions. Dropout is used before the fully-connected layer to avoid overfitting.

3.7 Multiple-Streams

Previous methods have shown that apart from the position information, the bone information and the motion information of the skeletal data are also helpful for action recognition [21]. Here, we transform the raw skeletal data from the coordinate space into the bone space and the motion space to utilize these two types of information. In detail, for bone information, given the raw joint coordinate $\mathbf{R}^{raw}_{t,m,n} \in \mathbb{R}^3$ of the original space, we first calculate the corresponding raw coordinate $\mathbf{R}^{braw}_{t,m,b} \in \mathbb{R}^3$ of the bone space by:

$$\mathbf{R}^{braw}_{t,m,b} = \mathbf{R}^{raw}_{t,m,\mathbf{I}(b,1)} - \mathbf{R}^{raw}_{t,m,\mathbf{I}(b,2)} \tag{4}$$

where $b = 1, 2, \cdots, B$. The map $\mathbf{I} \in \mathbb{R}^{B\times 2}$ records the indices of the two end joints of every bones. Then, similar with the procedure of voxelizing $\mathbf{R}^{raw}_{t,m,n}$ introduced in Sec. 3.2, $\mathbf{R}^{braw}_{t,m,b}$ is voxelized into a bone-space-based 4D grid, resulting in the sparse bone-space-based coordinate vector $\mathbf{R}^{bone}_{t,m,b}$ and the corresponding feature vector $\mathbf{F}^{bone}_{t,m,b}$. $\mathbf{F}^{bone}_{t,m,b}$ is same as $\mathbf{F}^{jpt}_{t,m,n}$.

Similarly, the raw coordinate $\mathbf{R}^{mraw}_{\tau,m,n} \in \mathbb{R}^3$ of the motion space is obtained by:

$$\mathbf{R}^{mraw}_{\tau,m,n} = \mathbf{R}^{raw}_{t+1,m,n} - \mathbf{R}^{raw}_{t,m,n} \tag{5}$$

where $\tau = 1, 2, \cdots, T-1$. It is also voxelized into a motion-space-based 4D grid, resulting in $\mathbf{R}^{motion}_{\tau,m,n}$ and $\mathbf{F}^{motion}_{\tau,m,n}$. Both the bone information and the motion information are modeled with two additional networks with the same architecture as SC4D. The SoftMax scores of three streams are averaged to get the final prediction.

4 Experiments

4.1 Datasets and Training Details

We specially select two datasets, namely, NTU-RGBD and SHREC, with different tasks to show the generalizability of our model.

NTU-RGBD consists of 56,000 action clips in 60 action classes. Each action is captured by 3 cameras. It provides 3D joint locations of 25 joints detected by Kinect-V2 depth sensors as shown in Fig. 4, left. Each video has no more than 2 subjects. Because the accuracy of the cross-view benchmark is nearly saturated, we conduct experiments on the cross-subject benchmark of the dataset, where the training and testing sets are split based on different subjects. Since the whole dataset is large and the training is time-consuming, we extract a subset of NTU-RGBD, namely, NTU-RGBD-SUB, for ablation studies. Specifically, because the samples are captured by 3 camera, the samples captured by the first camera are used to form the NTU-RGBD-SUB.

SHREC contains 2800 gesture sequences performed by 28 subjects in two ways: using one finger to perform gestures or using the whole hand to perform gestures. It provides the 3D coordinates of 22 hand joints captured by Intel-Real-Sense depth camera as shown in Fig. 4, right. This dataset has once been used for the competition of SHREC'17 in conjunction with the Eurographics 3DOR'2017 Workshop, and thus it reflects the highest level in this field.

All experiments are conducted on the PyTorch deep learning framework [15]. Stochastic gradient descent (SGD) with Nesterov momentum (0.9) is applied as

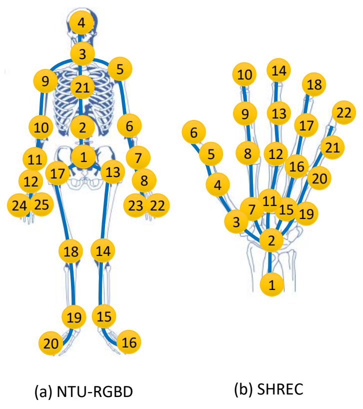

Fig. 4. Illustration of the joint index and the bones for (a) NTU-RGBD dataset and (b) SHREC dataset.

the optimization strategy. The batch size is set to 32. Cross-entropy is selected as the loss function to back-propagate gradients. The weight decay is set to 0.0005. Warm up is used with 5 epochs. Drop out rate is 0.2. To avoid overfitting, the voxelized skeletons are randomly rotated and scaled for spatial dimensions. The rotation range is [-10, 10]. The scale range is [0.8, 1.2].

Before the voxelization introduced in Sec. 3.2, the data is first preprocessed inspired by [21]. The center joint of each skeleton is set as the origin of the coordinate. It is the joint #1 for both NTU-RGBD and SHREC as shown in Fig. 4. Besides, for NTU-RGBD dataset, the skeletons are rotated to the same viewpoint to reduce the intra-class variations. In detail, we keep the line between the joint #9 and the joint #5 parallel with the x axis, and the line between the joint #1 and the joint #2 parallel with the z axis.

4.2 Ablation Study

Ablation studies are conducted on NTU-RGBD-SUB. Firstly, we investigate the strategies of the feature representation introduced in Sec. 3.3. We update the C3D network [30] to C4D by simply expanding the dimension of the kernel from 3 to 4. Others are kept the same as before. Due to the GPU memory limitations, we set the grid size to 16 (same for all dimensions) and the number of basic channels to 16, i.e., the output channels of 8 convolutional layers in C4D is 16, 32, 64, 64, 128, 128, 128, 128. Tab. 1 shows the results of the three strategies, where S3 performs better as expected.

Then, we replace the dense convolution with the sparse convolution to see the difference. To keep the comparison fair, instead of using the final SC4D showed in Fig. 3, we use a C4D-comparable model named "Sparse C4D" for comparison, whose architecture is the same with C4D except for the sparse operation. As

Table 1. Action recognition performance for different feature representation strategies. S1, S2 and S3 correspond to the strategies introduced in Sec. 3.3 sequentially.

Strategy	Acc. (%)
S1	80.6
S2	81.7
S3	82.3

shown in Tab 2, using dense C4D achieves only a little better than using sparse C4D. But the sparse C4D largely saves the computations. For dense C4D, it needs 4 TITANXP GPUs to train the model using the PyTorch framework. But for sparse C4D, it needs only one-tenth memory of one GPU.

Table 2. Comparison of the dense convolution and the sparse convolution.

Methods	Acc. (%)
Dense C4D	82.3
Sparse C4D	82.1

Now that we can save a large amount of GPU memories by using the sparse convolution, we use the network architecture showed in Fig. 3, i.e., SC4D, for experiments in the rest experiments. Compared with sparse C4D, residual connections are added for every convolutional layers. A feature embedding layer is added at the beginning. The basic number of filters is set to 32, i.e., C=32. The basic kernel size is 3, i.e., K=3. With these designations, SC4D performs better than sparse C4D. Then, we investigate the effect of the grid size as shown in Tab. 3. “Overlap” denotes the ratio of the number of points in sparse tensor to the number of original points. It reflects the degree of different joints falling into the same voxel, which causes the loss of information. It is 100% when there are no different joints falling into the same voxel. The result shows that properly increasing the grid size can improve the performance (Size=16 vs Size=32 vs Size=64). We found that using larger grid can reduce the overlaps of the joints. Thus, it can keep more useful information. However, it can not be increased too large (Size=64 vs Size=128). It is because along with the increasing of the grid size, the distance between the points also grows, which brings difficulties for relation modeling with the fix-size convolutional kernel.

The most important two factors that affect the model performance is the kernel size and the number of filters. Tab. 4 lists the performance of different configurations for the kernel size and the number of filters. The gird size is fixed to 32. It shows that increasing the kernel size improves the performance especially when the original kernel size is small (K=3 vs K=4 vs K=5). We believe it is because the large kernel size can help covering more points so that capture more information in one convolutional step. The improvement decreases or even becomes negative when the original kernel size is already large enough (K=5 vs K=6). It is because the larger kernel size also brings more parameters

Table 3. Action recognition performance for different grid sizes. "Overlap" denotes the ratio of the number of points in sparse tensor to the number of original points. It is 100% when there no different joints falling into the same voxel.

Grid size	Acc. (%)	Overlap (%)
Size=32	87.9	98.6
Size=16	86.6	93.1
Size=64	88.8	99.7
Size=128	88.5	99.9

that causes the difficulty for network training. The similar phenomenon is also observed for the number of filters (C=32 vs C=64 vs C=128).

Table 4. Action recognition performance for different kernel size (K) and the number of filters (C). K and C are corresponded with Fig. 3.

Configuration	Acc. (%)
K=3, C=32	87.9
K=4, C=32	89.0
K=5, C=32	90.2
K=6, C=32	89.8
K=3, C=64	89.1
K=3, C=128	89.4

Then, we test the effectiveness of two data enhancement strategies introduced in Sec. 3.3. The grid size is increased to 64 to make the grid more sparse. Tab. 1 shows that both of the two strategies bring improvement. However, since the number of points is also increased, the data becomes denser and it needs more computations and memories.

Table 5. Action recognition performance for different data enhancement strategies.

Strategy	Acc. (%)
no enhancement	88.8
+edge	88.9
+edge&dilate	89.7

Finally, we investigate the effectiveness of the three streams introduced in Sec. 3.7. They are processed by three networks with the same architecture, i.e., SC4D. The $SoftMax$ scores are fused to get the final prediction. It shows using the bone information or the motion information performs worse than the original data, but fusing three streams can largely improve the performance.

Table 6. Action recognition performance for different streams.

Modality	Acc. (%)
joint	88.8
bone	86.9
motion	81.5
fusion	90.7

4.3 Comparison with the SOTA

Although many strategies have been shown effective for improving the performance such as increasing the kernel size or the number of filters, due to the GPU memory limitations, we have to made some trade-offs. Finally, the kernel size of the SC4D is set to 3 for the temporal dimension and 5 for the spatial dimensions. The basic number of filters is 64, i.e., C=64. Grid size is 64 for NTU-RGBD dataset and is 32 for SHREC dataset because the "overlap" (introduced in Tab. 3) of the $32 \times 32 \times 32 \times 32$ grid already reaches 99.7% for SHREC. The bone interpolation strategy is used for the joint stream. We test our final model on two datasets: NTU-RGBD for action recognition and SHREC for gesture recognition. The results are showed in Tab. 7 and Tab. 8, where our SC4D achieves state-of-the-art performance on two datasets. More comparisons are showed in supplement materials. It verifies the effectiveness and generalizability of our method.

Table 7. Comparison with the state-of-art-methods on NTU-RGBD dataset.

Method	Year	Acc. (%)
AGC-LSTM [24]	2019	89.2
2s-AGCN [21]	2019	88.5
DGNN [20]	2019	89.9
Bayesian-GCN [36]	2020	81.8
NAS [16]	2020	89.4
SC4D (ours)	-	**90.5**

Table 8. Comparison with the state-of-art-methods on SHREC dataset.

Method	Year	14 gestures	28 gestures
ST-GCN [32]	2018	92.7	87.7
STA-Res-TCN [7]	2018	93.6	90.7
ST-TS-HGR-NET [13]	2019	94.3	89.4
DG-STA. [3]	2019	94.4	90.7
SC4D (ours)	-	**95.8**	**93.6**

5 Conclusion

In this work, we propose a new perspective for skeleton-based action recognition, where the skeletal data is viewed as a point cloud and is voxelized into a 4d grid for recognition. A novel sparse 4D convolutional network (SC4D) is proposed to directly model the 4D grid, which largely keeps the spatial-temporal consistencies of the skeletal data. The overall framework is concise, robust, and efficient due to the sparse operation. Besides, two data enhancement techniques are introduced to augment the spatial pattern and ease the recognition. The data is additionally projected into two other spaces to utilize the bone information and the motion information for better performance. Our method achieves state-of-the-art performance on two challenging datasets for different tasks, which confirms its effectiveness and generalizability.

References

1. Cao, C., Lan, C., Zhang, Y., Zeng, W., Lu, H., Zhang, Y.: Skeleton-Based Action Recognition with Gated Convolutional Neural Networks. IEEE Transactions on Circuits and Systems for Video Technology pp. 3247–3257 (2018)
2. Carreira, J., Zisserman, A.: Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6299–6308 (Jul 2017)
3. Chen, Y., Zhao, L., Peng, X., Yuan, J., Metaxas, D.N.: Construct Dynamic Graphs for Hand Gesture Recognition via Spatial-Temporal Attention. In: BMVC (2019)
4. Choy, C., Gwak, J., Savarese, S.: 4d Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. arXiv preprint arXiv:1904.08755 (2019)
5. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6202–6211 (2019)
6. He, K., Zhang, X., Ren, S., Sun, J.: Identity Mappings in Deep Residual Networks. In: The European Conference on Computer Vision (ECCV). pp. 630–645 (2016)
7. Hou, J., Wang, G., Chen, X., Xue, J.H., Zhu, R., Yang, H.: Spatial-temporal attention res-TCN for skeleton-based dynamic hand gesture recognition. In: The European Conference on Computer Vision (ECCV). pp. 0–0 (2018)
8. Li, C., Zhong, Q., Xie, D., Pu, S.: Skeleton-based Action Recognition with Convolutional Neural Networks. In: IEEE International Conference on Multimedia & Expo Workshops (ICMEW). pp. 597–600 (2017)
9. Li, M., Chen, S., Chen, X., Zhang, Y., Wang, Y., Tian, Q.: Actional-Structural Graph Convolutional Networks for Skeleton-Based Action Recognition. pp. 3595–3603 (2019)
10. Li, S., Li, W., Cook, C., Zhu, C., Gao, Y.: Independently recurrent neural network (indrnn): Building A longer and deeper RNN. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5457–5466 (2018)
11. Liu, J., Shahroudy, A., Xu, D., Wang, G., Liu, J., Shahroudy, A., Xu, D., Wang, G.: Spatio-Temporal LSTM with Trust Gates for 3d Human Action Recognition. In: The European Conference on Computer Vision (ECCV). vol. 9907, pp. 816–833 (2016)
12. Liu, M., Liu, H., Chen, C.: Enhanced skeleton visualization for view invariant human action recognition. Pattern Recognition **68**, 346–362 (2017)
13. Nguyen, X.S., Brun, L., Lzoray, O., Bougleux, S.: A neural network based on SPD manifold learning for skeleton-based hand gesture recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Apr 2019)
14. Pan, H., Liu, S., Liu, Y., Tong, X.: Convolutional neural networks on 3d surfaces using parallel frames. arXiv preprint arXiv:1808.04952 (2018)
15. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic Differentiation in PyTorch. In: Advancesin Neural Information Processing Systems Workshops (2017)
16. Peng, W., Hong, X., Chen, H., Zhao, G.: Learning Graph Convolutional Network for Skeleton-based Human Action Recognition by Neural Searching. In: AAAI (2020)
17. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)

18. Riegler, G., Osman Ulusoy, A., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3577–3586 (2017)
19. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Action Recognition via Pose-Based Graph Convolutional Networks with Intermediate Dense Supervision. arXiv:1911.12509 (Nov 2019)
20. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Skeleton-Based Action Recognition With Directed Graph Neural Networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7912–7921 (Jun 2019)
21. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12026–12035 (2019)
22. Shi, L., Zhang, Y., Jian, C., Hanqing, L.: Gesture Recognition using Spatiotemporal Deformable Convolutional Representation. In: IEEE International Conference on Image Processing (ICIP) (Oct 2019)
23. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–779 (2019)
24. Si, C., Chen, W., Wang, W., Wang, L., Tan, T.: An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1227–1236 (2019)
25. Si, C., Jing, Y., Wang, W., Wang, L., Tan, T.: Skeleton-Based Action Recognition with Spatial Reasoning and Temporal Stack Learning. In: The European Conference on Computer Vision (ECCV). pp. 103–118 (Sep 2018)
26. Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data. In: AAAI. pp. 4263–4270 (2017)
27. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2530–2539 (2018)
28. Tang, Y., Tian, Y., Lu, J., Li, P., Zhou, J.: Deep Progressive Reinforcement Learning for Skeleton-Based Action Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5323–5332 (2018)
29. Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S.: Segcloud: Semantic segmentation of 3d point clouds. In: 2017 international conference on 3D vision (3DV). pp. 537–547. IEEE (2017)
30. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning Spatiotemporal Features With 3d Convolutional Networks. In: The IEEE International Conference on Computer Vision (ICCV). pp. 4489–4497 (Dec 2015)
31. Vemulapalli, R., Arrate, F., Chellappa, R.: Human action recognition by representing 3d skeletons as points in a lie group. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 588–595 (2014)
32. Yan, S., Xiong, Y., Lin, D.: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In: AAAI (2018)
33. Yang, Y., Ramanan, D.: Articulated pose estimation with flexible mixtures-of-parts. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1385–1392. IEEE (2011)
34. You, Q., Jiang, H.: Action4d: Online Action Recognition in the Crowd and Clutter. pp. 11857–11866 (2019)

35. Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., Zheng, N.: View Adaptive Recurrent Neural Networks for High Performance Human Action Recognition From Skeleton Data. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2117–2126 (2017)
36. Zhao, R., Wang, K., Su, H., Ji, Q.: Bayesian Graph Convolution LSTM for Skeleton Based Action Recognition. In: AAAI (2020)