Spatio-Temporal Difference Descriptor for Skeleton-Based Action Recognition

Chongyang Ding,[1] **Kai Liu,**[1] **Jari Korhonen,**[2] **Evgeny Belyaev**[3]
[1] School of Computer Science and Technology, Xidian University
[2] School of Computer Science and Software Engineering, Shenzhen University
[3] International Laboratory Computer Technologies, ITMO University
dingcy@stu.xidian.edu.cn, kailiu@mail.xidian.edu.cn, jari.t.korhonen@ieee.org, eabelyaev@itmo.ru

Abstract

In skeletal representation, intra-frame differences between body joints, as well as inter-frame dynamics between body skeletons contain discriminative information for action recognition. Conventional methods for modeling human skeleton sequences generally depend on motion trajectory and body joint dependency information, thus lacking the ability to identify the inherent differences of human skeletons. In this paper, we propose a spatio-temporal difference descriptor based on a directional convolution architecture that enables us to learn the spatio-temporal differences and contextual dependencies between different body joints simultaneously. The overall model is built on a deep symmetric positive definite (SPD) metric learning architecture designed to learn discriminative manifold features with the well-designed non-linear mapping operation. Experiments on several action datasets show that our proposed method achieves up to 3% accuracy improvement over state-of-the-art methods.

Introduction

Human action recognition has become a challenging and active topic in video understanding in recent years. With the growing popularity of depth sensors, video formats based on three-dimensional depth maps are widely used in addition to the two-dimensional RGB videos. Previous biological study (Johansson 1973) suggests that human behavior can be represented as the motion of a human skeleton, which is structured as an articulated system of body parts, making it robust to the variations of viewpoint and location. In addition, recent pose estimation algorithm (Cao et al. 2017) can extract and gain human skeleton data from both the RGB videos and depth maps, making it possible to design a general skeleton-based framework for human action recognition. Therefore, skeleton-based action recognition has drawn more and more attention in recent years.

Recently, graph convolutional networks (GCNs) have been proposed and applied in many tasks (Niepert, Ahmed, and Kutzkov 2016; Atwood and Towsley 2016; Hamilton, Ying, and Leskovec 2017). GCNs generalize traditional convolution operation to graphs of different structure, and it has shown excellent performance for action recognition (Li et al. 2018; Yan, Xiong, and Lin 2018; Si et al. 2019) on account of the graph structure of skeleton data. GCNs structure the human skeleton as a graph whose vertices and edges represent body joints and natural connections between these body joints, to capture the joint dependencies of spatial and temporal domains. They use the convolution operation to obtain different weights for different neighbor subsets, indicating the influence of different subsets on the current body joint. However, these methods merely explore the spatio-temporal dependencies of body joints to model the skeleton sequences and cannot reflect the intra-frame differences and inter-frame dynamics information, thus lacking the ability to identify the inherent spatio-temporal differences of human skeletons.

Motivated by this observation, we introduce a second-order spatio-temporal descriptor based on our proposed directional convolution operation to address the lacking of spatio-temporal differences. Different from GCNs, the spatio-temporal descriptor is defined as a covariance-like descriptor extracted from the intrinsic attributes of the skeleton data, consisting of spatial difference matrix and temporal dynamic matrix. The directional convolution is first proposed in this paper to learn the joint dependencies of human skeletons. For a specific body joint, we allocate different directional weights to the neighboring joints located in different directions to aggregate to a new joint. Thus, a new human skeleton will be generated when the convolution is conducted on all body joints, containing the dependency information between adjacent joints in the original skeleton. The spatial difference matrix is introduced to learn the unique static pattern of a skeleton, allowing us to distinguish the current skeleton from the other skeletons to model the intra-frame differences. In addition, the temporal dynamic matrix is proposed to extract the temporal motion dynamics between adjacent frames for better temporal modeling. It is computed from the joint displacement information between current frame and the next frame to model the inter-frame dynamics (Wang et al. 2016; Zhao, Xiong, and Lin 2018). In order to measure the joint dependency information and spatio-temporal differences simultaneously, the spatial difference and temporal dynamic matrices are designed to be the aggregation of original convolutional features and second-order statistical features extracted from the convolutional features, which are actually SPD matrices residing on the Riemannian symmetric space. Then, a novel non-linear

mapping operation is proposed for deep SPD learning, mapping the input SPD matrices to be lower dimensional and more discriminative.

In summary, the main contributions of this paper are on the following three aspects:

1. We propose a directional convolution operation to learn the dependency relationships between different body joints. By assigning different weights to the joints located in different directions, the directional convolution will generate a new skeleton that contains the dependencies between adjacent joints.
2. We propose a spatio-temporal difference descriptor that consists of spatial difference matrix and temporal dynamic matrix to extract the intra-frame differences and inter-frame dynamics information.
3. We introduce a new deep SPD metric learning module to employ the non-linear mapping operation on the SPD manifold, which can map the SPD matrices to a low dimensional discriminative SPD space. The experimental results on several standard datasets demonstrate the effectiveness of the proposed network architecture.

Related Work

Skeleton-based action recognition has become a popular technique due to its effective and convenient representation of human actions. Recent data-driven works that can automatically learn the skeleton representation from input data have drawn substantial attention. Some convolutional neural network (CNN) based models (Liu, Liu, and Chen 2017; Kim and Reiter 2017) employ convolution operation designed manually (including the step and size) to learn the specific feature information and achieve remarkable performance. The graph-based methods (Yan, Xiong, and Lin 2018; Thakkar and Narayanan 2018) that generalize traditional convolutional operation from images to graphs have been proven to be effective with the graph structure representation. In addition, deep long short-term memory (LSTM) models (Shahroudy et al. 2016; Liu et al. 2016; Song et al. 2017) and bidirectional-RNNs (Du, Wang, and Wang 2015) have been proposed to model the contextual dependency in the temporal domain with the temporal recurrent architecture.

On the other hand, the manifold-based approaches explore the view-invariant representation of skeletal data (Anirudh et al. 2017; Huang and Van Gool 2017) for this task. Some representations (Vemulapalli, Arrate, and Chellappa 2014; Vemulapalli and Chellapa 2016) use geometric transformations to embed the spatial information of skeletons into a Lie group space and use the corresponding Lie algebra map to conduct the feature learning. The transported square-root velocity fields (TSRVF) framework (Anirudh et al. 2017) employs elastic functional coding to embed the motion trajectories on manifold space to a low dimensional and effective space for action representation. (Ben Tanfous, Drira, and Ben Amor 2018) represents human actions using an intrinsic formulation of sparse coding and dictionary learning of skeletal shapes in the Kendall's shape space, and effective coding approach is then used on the tangent space

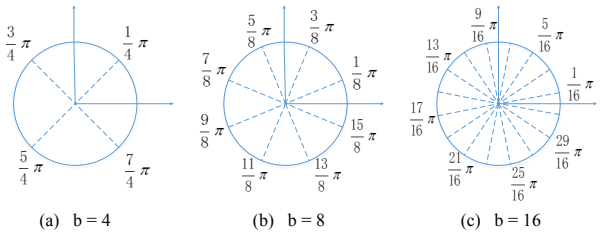

Figure 1: Illustration of the Cartesian plane that is divided into b partitions.

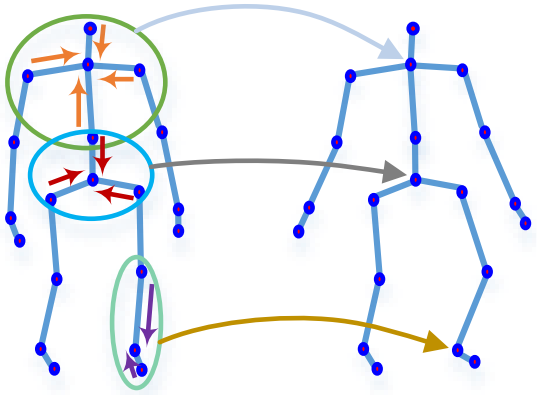

Figure 2: Conceptual illustration of the directional convolution on the human skeleton. The neighbor joints of a body joint all have different influences on the current joint, and our directional convolution operation aggregates this information to generate a new body joint.

where the dictionary of shapes is mapped to make the function of sparse coding lies in Euclidean space.

Proposed Method

The overall network architecture of our proposed model is illustrated in Figure 3. It consists of four parts: Directional convolution operation, spatio-temporal difference descriptor, deep SPD metric learning and classification layers. The directional convolution operation is first introduced to generate new skeletons containing the joint dependency information. Next, the spatio-temporal difference descriptor is aggregated from the second-order statistical features and directional convolution features. Deep SPD metric learning module is then designed to extract more compact and discriminative features. Finally, standard classification methods are conducted on the feature space to identify the action classes.

Directional Convolution on Human Skeletons

The convolution operation applied to the skeleton data is aimed to learn the joint dependencies that combine body joint features with its correlated variation information. The skeleton data is naturally constructed as a specific graph structure, whose vertices and edges are body joints and human body parts, respectively. Thus, the conventional convo-

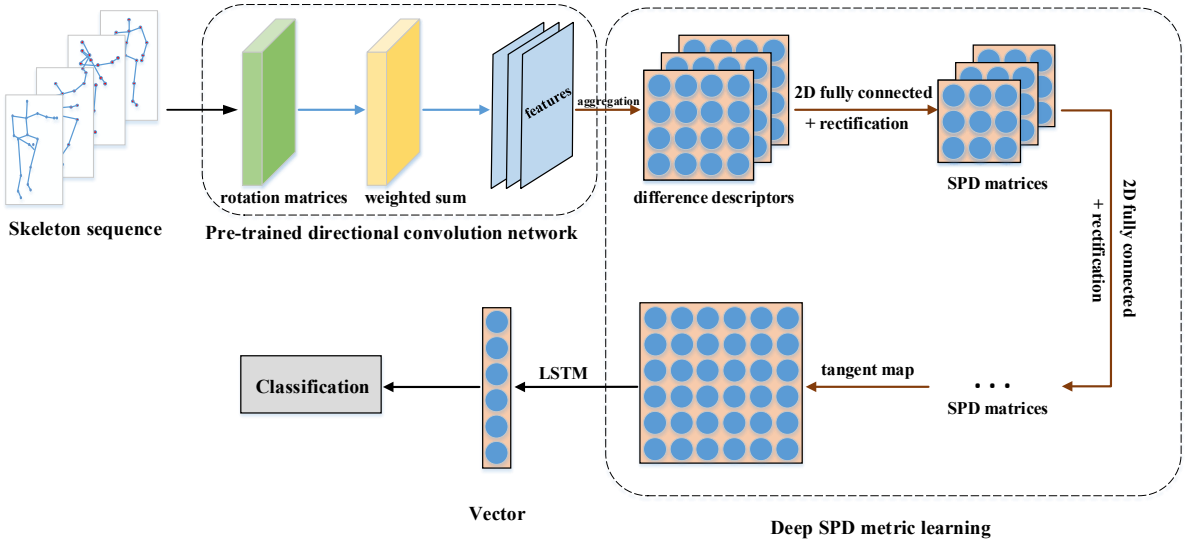

Figure 3: The overall architecture of our proposed model.

lution on grid structure data is no longer suitable for handling skeleton data. In this work, we construct a directional convolution kernel based on the human skeleton structure.

First, the human skeletons are rotated to make the ground plane projection of the vector from left hip to right hip parallel to the global x-axis. This rotation guarantees that all human skeletons are in the same perspective and can be projected from the front view on the same Cartesian plane as 2D maps. Then, the Cartesian plane is divided into b partitions as shown in Figure 1 according to the previous partition strategy (Xia, Chen, and Aggarwal 2012). Let N and L denote the number of human skeletal body joints and the length of a skeleton sequence, respectively. Let $p_i^t \in R^3$ $(i = 1, ...N, t = 1, ...L)$ represents the 3D position (x, y, z) of body joint i at frame t and $n(i, t)$ denotes the neighbor set of body joint i. The neighbor set of joint i is defined as a set of joints that are connected to joint i in the human skeleton.

Figure 2 illustrates the operational process of our directional convolution. Given the body joint i as the origin of the Cartesian plane, its neighbors are distributed in different parts. The neighbors located in different directions usually have different influence on the center point. Thus, different weights are assigned to these directions. Then, the directional convolution at joint i can be defined as:

$$f(p_i^t) = \frac{1}{|n(i,t)|} \sum_{j \in n(i,t)} \boldsymbol{W}_{k_j} p_j^t, \tag{1}$$

where $|n(i, t)|$ denotes the number of elements in neighbor set $n(i, t)$, k_j denotes the direction where the joint j is located in relative to the joint i, and $\boldsymbol{W}_{k_j}$ indicates the directional weight matrix.

Different from the conventional 2D convolution operating on pixels, the directional convolution in this work is aimed to aggregate 3D geometric characteristics of different joint locations p_j^t. The 3D joint positions contain two aspects of information: direction information and size information. The direction information can be obtained from the directional weight $\boldsymbol{W}$ by allocating different directional angles with corresponding weights. Thus, to preserve the size information, we use the rotation operation rather than numerical values as the directional weight parameters. Compared with the numerical value weights that may change the size information of inputs, the rotation operation can keep the size information invariant. Thus, the directional weights $\boldsymbol{W}_{k_j}$ are designed to be 3×3 rotation matrices in this work.

Spatio-Temporal Difference Descriptor

The spatio-temporal difference descriptor, consisting of spatial difference matrix and temporal dynamic matrix, is introduced to resolve the lack of spatio-temporal difference information. With regard to the spatial difference features of a skeleton, the spatial difference matrix is employed to measure the intra-frame body joint differences, which represents the static characteristics of the convolutional features at a specific frame. For body joint i at current frame t, the corresponding joint differences are first computed, then the products of these difference vectors and their transposes are added together to obtain the statistical matrix that indicates the specificity distinguishing each joint from the other joints:

$$\boldsymbol{C}_{S,i}^t = \frac{1}{N-1} \sum_{j=1}^{N} (f(p_i^t) - f(p_j^t))(f(p_i^t) - f(p_j^t))^T, \tag{2}$$

where $\boldsymbol{C}_{S,i}^t$ and $f(p_i^t)$ denote the statistical and convolutional features of joint i at time stamp t. Then, the spatial difference matrix of joint i at frame t can be formulated based on the aggregation rule (Nguyen et al. 2019) that embeds the feature space in the Riemannian symmetric space by:

$$\boldsymbol{S}^{t_i} = \begin{bmatrix} \boldsymbol{C}_{S,i}^t + f(p_i^t) f(p_i^t)^T & f(p_i^t) \\ f(p_i^t)^T & 1 \end{bmatrix}, \tag{3}$$

where $\boldsymbol{C}_{S,i}^t$ is a 3×3 square matrix and $f(p_i^t)$ is a 3×1 column vector, thus making the spatial difference matrix $\boldsymbol{S}^{t_i}$

a 4×4 square matrix.

In this case, the difference matrices of all body joints represent the unique motion pattern of a skeleton, enabling us to distinguish the current skeleton from the other skeletons. Thus, the full spatial difference matrix of the frame t can be denoted as:

$$\boldsymbol{S}_t = [(\boldsymbol{S}^{t_1})^T, (\boldsymbol{S}^{t_2})^T, ..., (\boldsymbol{S}^{t_N})^T]^T. \quad (4)$$

On the other hand, the temporal dynamic matrix is proposed to derive the inter-frame dynamic changes for better temporal modeling. Compared with the spatial difference matrix, the temporal dynamic matrix is defined as the motion dynamic information of the convolutional features between frame t and its next frame t_n. Similar to the aforementioned calculation, the statistical matrix is first computed from the products of the joint displacement vectors and their corresponding transposes between adjacent frames:

$$\boldsymbol{C}_T^t = \frac{1}{N-1}\sum_{i=1}^{N}(f(p_i^{t_n}) - f(p_i^t))(f(p_i^{t_n}) - f(p_i^t))^T, \quad (5)$$

where $f(p_i^{t_n})$ denotes the convolutional feature of joint i at time stamp t_n and $\boldsymbol{C}_T^t$ represents the resulting 3×3 statistical matrix. To acquire the complete movement process of an action, we define $t_n = t + 1$ when $t < L$, and $t_n = 1$ when $t = L$. Thus, the temporal dynamic matrix of frame t can be formulated as:

$$\boldsymbol{T}_t = \begin{bmatrix} \boldsymbol{C}_T^t + \overline{f}(p_i^t)\overline{f}(p_i^t)^T & \overline{f}(p_i^t) \\ \overline{f}(p_i^t)^T & 1 \end{bmatrix}, \quad (6)$$

where $\overline{f}(p_i^t) = \frac{1}{N}\sum_{i=1}^{N} f(p_i^t)$ denotes the mean vector of the convolutional features and $\boldsymbol{T}_t$ is the temporal dynamic matrix of size 4×4.

Finally, the spatio-temporal difference descriptor of the frame t can be denoted as:

$$\boldsymbol{D}_t = [(\boldsymbol{S}_t)^T, (\boldsymbol{T}_t)^T]^T. \quad (7)$$

As studied in previous work (Lovrić, Min-Oo, and Ruh 2000), the spatial difference matrix $\boldsymbol{S}_i^t$ and temporal dynamic matrix $\boldsymbol{T}_t$ are all SPD matrices, leading to the spatio-temporal difference descriptor matrix residing on Riemannian symmetric space.

Deep SPD Metric Learning

2D fully connected layer. To extract the discriminative information contained in the SPD matrix and preserve the original matrix structure, we introduce a specific layer, called 2D fully connected layer, to generate a more compact and discriminative SPD matrix. The 2D fully connected layer is an extension of conventional fully connected layer, making it applicable to 2D matrices. To this end, we divide the neural connections into two-dimensional direction: row dimensional direction and column dimensional direction.

Matrix $\boldsymbol{X}$ of size $m \times n$ can be expressed in the form of row vector and column vector, respectively. The row vector form of the matrix can be represented as an n-dimensional row vector $[\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_n]$, where each element $\boldsymbol{X}_i, i \in \{1, 2, ..., n\}$, is an m-dimensional column vector $[X_i^1, X_i^2, ..., X_i^m]^T$. Similarly, the column vector form of the matrix can be represented as an m-dimensional column vector $[(\boldsymbol{X}^1)^T, (\boldsymbol{X}^2)^T, ..., (\boldsymbol{X}^m)^T]^T$, where each element $\boldsymbol{X}^j, j \in \{1, 2, ..., m\}$, is an n-dimensional row vector $[X_1^j, X_2^j, ..., X_n^j]$.

Given that $\boldsymbol{W}$ represents the $p \times m$ connection weight matrix, then all the m neurons in each column vector of the input matrix are fully connected to the p neurons in the corresponding column vector of the output matrix with the specific weight parameter $\boldsymbol{W} \in R^{p \times m}$ that can be formulated as:

$$\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X} = [\boldsymbol{W}\boldsymbol{X}_1, \boldsymbol{W}\boldsymbol{X}_2, \cdots, \boldsymbol{W}\boldsymbol{X}_n], \quad (8)$$

where $\boldsymbol{Y}$ represents the new $p \times n$ matrix generated by row dimensional connections.

Similarly, given that $\boldsymbol{V}$ represents the $n \times q$ connection weight matrix, the column dimensional connections conducted on the row vector of the matrix can be obtained as before. Therefore, the column dimensional connections that connect all the n neurons in each row vector of the input matrix with the q neurons of the output matrix using the corresponding weight parameter $\boldsymbol{V} \in R^{n \times q}$ can be formulated as:

$$\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{V} = \begin{bmatrix} \boldsymbol{Y}^1\boldsymbol{V} \\ \boldsymbol{Y}^2\boldsymbol{V} \\ \vdots \\ \boldsymbol{Y}^p\boldsymbol{V} \end{bmatrix}, \quad (9)$$

where $\boldsymbol{Z}$ denotes the output $p \times q$ matrix generated by column dimensional connections.

By generalizing the aforementioned situation to SPD matrices, we can find that $m = n$, $p = q$ and the column dimensional weight $\boldsymbol{V}$ can be represented as transpose of the row dimensional weight $\boldsymbol{W}$ due to the symmetry property of the SPD matrix. Thus, the 2D fully connected layer for the SPD matrix can be further formulated as:

$$\boldsymbol{Z} = \boldsymbol{W}\boldsymbol{X}\boldsymbol{V} = \boldsymbol{W}\boldsymbol{X}\boldsymbol{W}^T, \quad (10)$$

where $\boldsymbol{X}$ is the input $m \times m$ SPD matrix, $\boldsymbol{Z}$ is the output $p \times p$ matrix, $\boldsymbol{W} \in R^{p \times m}$ denotes the row dimensional weight. Here, we set $p < m$ for dimensionality reduction of the SPD matrix. Moreover, to ensure that the output matrix $\boldsymbol{Z}$ remains an SPD matrix, the row vectors of $\boldsymbol{W}$ should be linearly independent; in other words, $\boldsymbol{W}$ should be a row full rank matrix.

Rectification function. Following the 2D fully connected layer that transforms the input SPD matrix to a low dimensional discriminative SPD matrix, a non-linear operation should be employed to ensure the non-linearity of the mapping. In addition, the non-linear operation is required to preserve the SPD matrix structure. Considering the spectral decomposition of the SPD matrix $\boldsymbol{Z} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, we define a rectification function operating directly on the diagonal eigenvalue matrix $\boldsymbol{\Lambda}$ to generate a new diagonal matrix with its diagonal values as:

$$\boldsymbol{R}_{ii} = h(\boldsymbol{\Lambda}_{ii}) = \begin{cases} e^{(\boldsymbol{\Lambda}_{ii} - \epsilon)}, & if\ \Lambda_{ii} < \epsilon \\ \boldsymbol{\Lambda}_{ii}, & otherwise \end{cases}, \quad (11)$$

where ϵ is the rectification threshold that determines whether the eigenvalue $\boldsymbol{\Lambda}_{ii}$ is the principal component. According to

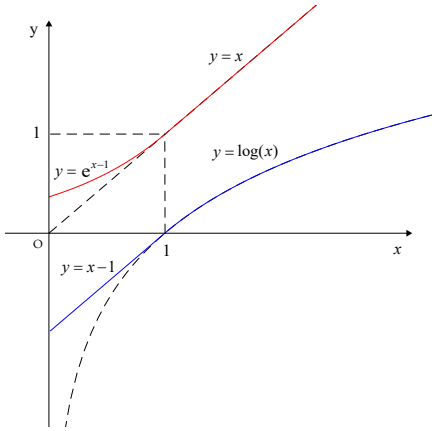

Figure 4: Images of the rectification function $h(\cdot)$ and $\log(\cdot)$ function operating on the eigenvalues with ϵ set to 1. The red curve represents the rectification function, and the blue curve denotes the resulting logarithm function.

the geometric characteristics of the feature space, ϵ is constrained as $\epsilon \geq 1$ to ensure the continuity of the feature space.

Therefore, the operation that rectifies the input SPD matrix can be defined as:

$$\boldsymbol{M} = g(\boldsymbol{Z}) = \boldsymbol{Q} diag(h(\boldsymbol{\Lambda}_{11}), ...h(\boldsymbol{\Lambda}_{ii}))\boldsymbol{Q}^T, \qquad (12)$$

where $\boldsymbol{M}$ is the rectified SPD matrix, $g(\cdot)$ denotes the rectification operation on SPD matrix, and $diag(\cdot)$ is the operation to construct a diagonal matrix.

As shown in Figure 4, when the eigenvalues are close to zero, the proposed rectification function adopt exponential operation to replace the linear map to generate new proper eigenvalues, which can effectively prevent the eigenvalues of the input SPD matrix from being close to zero. In this work, we set $\epsilon = 1$ for preliminary exploration. Figure 4 also illustrates that the logarithmic operation on small positive eigenvalues without our proposed rectification operation may produce large negative values that harm the performance of feature representation and learning. However, the proposed rectification operation can address the abovementioned problem. The logarithmic operation on the eigenvalues processed with rectification function can generate valid values and outputs even though the original inputs are close to zero, which also benefits the logarithm operation on the SPD matrix.

Other layers. Following the non-linear mapping operation, i.e., 2D fully connected layer and rectification function, tangent map metric learning layer is introduced to project the SPD manifold into its tangent space at identity matrix $\boldsymbol{I}$ (Tosato et al. 2012; Chavel 2006), which makes it convenient to calculate and avoids the complex and time-consuming Riemannian computations due to the non-Euclidean nature of the underlying space. In addition, LSTM and classical deep learning classification layers, i.e., fully connected layer and softmax layer, are conducted on the tangent vector space to train the deep SPD network.

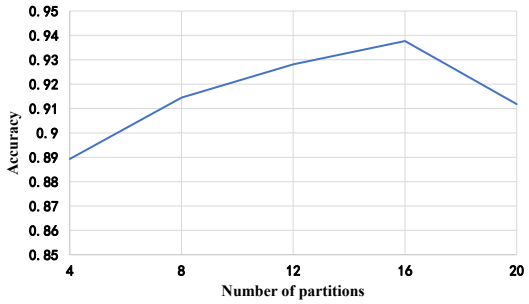

Figure 5: Recognition results of the directional convolution with different number of plane partitions on MSR-Action3D dataset.

Experiments

Datasets and Training Settings

G3D-Gaming dataset (Bloom, Makris, and Argyriou 2012) contains 20 different gaming action classes performed by 10 different subjects. Each subject consists of 20 body joints, and the corresponding 3D joint locations (x, y, z) are captured in the camera coordinate system. Following the cross-subject test setting, samples performed by half of the subjects are used for training and the remaining half of the samples are used for testing. The recognition results reported on this dataset are evaluated by averaging over ten different configurations of training and testing sets. HDM05 dataset (Müller et al. 2007) contains 130 action classes executed by five different subjects. Each skeletal sequence is performed several times by different subjects. Each subject contains 31 body joints, and the corresponding 3D joint locations of these joints are provided in the dataset. Following the standard evaluation protocol, we randomly select half of the sequences for training and the rest for testing. The results reported on this dataset are evaluated averagely over ten experiments with different splits. Northwestern-UCLA dataset (Wang et al. 2014) is captured by three Kinect cameras with different views. It contains 1494 sequences covering 10 different action classes. Each sequence is performed by 10 different actors repeated from 1 to 6 times. Each subject consists of 20 body joints. We follow the standard evaluation protocol where samples from the first two cameras are used for training and samples captured by the last camera are used for testing.

In our experiments, the human skeletal sequence length is set to 100, 50 and 50 for G3D-Gaming, HDM05 and Northwestern-UCLA, respectively. The batch size is set to 30, the learning rate is fixed as 0.01, the directional weights are initialized as random rotation matrices, the row dimensional weights are initialized as random semi-orthogonal matrices and the rectification threshold ϵ is set to 1. The LSTM layer contains 100 neurons and it is trained using Adam optimization algorithm (Kingma and Ba 2014). All experiments were conducted on Matlab deep learning toolbox with a TITANXP GPU.

Method	Accuracy
LSTM	80.97%
StddNet	87.56%
Di-StddNet ($b = 8$)	88.97%
Di-StddNet ($b = 12$)	89.23%
Di-StddNet ($b = 16$)	90.06%

Table 1: Ablation study results of the spatio-temporal difference descriptor and directional convolution on the G3D-Gaming dataset. LSTM is the baseline network whose inputs are original body joint locations. StddNet represents the network architecture containing spatio-temporal difference descriptor, tangent map and LSTM. Di-StddNet equips StddNet with directional convolution. The different values of b indicate different directional settings employed in the corresponding model.

Method	Accuracy
Di-StddNet ($b = 16$) + 0block	90.06%
Di-StddNet ($b = 16$) + 1block	91.54%
Di-StddNet ($b = 16$) + 2blocks	92.03%
Di-StddNet ($b = 16$) + 3blocks	89.58%

Table 2: Comparisons of the recognition accuracy under different non-linear mapping block configurations on the G3D-Gaming dataset. 0, 1, 2 and 3 indicate the number of mapping blocks (2D fully connected layer + rectification operation) employed in the corresponding model. All these models listed for comparison are based on the Di-StddNet ($b = 16$) architecture.

Directional Convolution Network Pre-training

In this work, to capture the dependency relationships between adjacent body joints, we introduce a directional convolution subnetwork and pre-train it on the MSR-Action3D (Li, Zhang, and Liu 2010) dataset. The pre-trained network contains two main parts: directional convolution and LSTM. Then, the pre-trained directional convolution part is added into our main network and remains unchanged when we train our model. During pre-training, the directional convolution operation first generates spatial features according to the input joint positions. Then, LSTM is employed to explore the temporal relationships of feature sequences. The pre-trained network is processed under different directional configurations with the number of partitions b set to 4, 8, 12, 16 and 20, respectively. Figure 5 shows the recognition accuracies with different number of partitions. We select the top-3 best performing settings, i.e., $b = 8, 12, 16$, as the directional convolution part of our backbone network, and evaluate the recognition performances of using different subnetworks under the same test conditions.

Ablation Study

To demonstrate the effectiveness of our proposed components of the deep SPD learning model, we conduct extensive experiments on G3D-Gaming dataset.

	0block	1block	2blocks
$b = 8$	88.97%	89.46%	89.83%
$b = 12$	89.23%	90.16%	90.91%
$b = 16$	90.06%	91.54%	92.03%

Table 3: Recognition accuracies of Di-StddNets with different configurations on the G3D-Gaming dataset.

Method	Accuracy
RBM+HMM (Nie and Ji 2014)	86.40%
Lie Group (Vemulapalli, Arrate, and Chellappa 2014)	87.23%
Rolling Rotations (Vemulapalli and Chellapa 2016)	87.95%
LieNet (Huang et al. 2017)	89.10%
Di-StddNet ($b = 16$) + 2blocks	92.03%

Table 4: Comparisons of the recognition result on the G3D-Gaming dataset.

Spatio-temporal difference descriptor. First, we test the effect of introducing spatio-temporal difference descriptor into our network. The basic LSTM model is used as the baseline network architecture where body joint locations are employed as the network inputs without information extraction. Then, we add the spatio-temporal difference descriptor and the corresponding tangent map operation on the baseline LSTM as StddNet. Results shown in Table 1 demonstrate that the introduction of spatio-temporal difference descriptor improves the recognition performance.

Directional convolution. In this experiment, we equip the abovementioned StddNet with directional convolution, titled as Di-StddNet, to evaluate the performance of directional convolution. Table 1 shows the recognition performance of our proposed Di-StddNet with different directional settings of b, i.e., $b = 8, 12, 16$. Experimental results suggest that the models with directional convolution all outperform the StddNet, which demonstrates the effectiveness of the directional convolution. In addition, the best performance can be achieved when b is set to 16.

Non-linear mapping block. Consisting of 2D fully connected layer and rectification operation, the mapping block is designed to generate more compact and discriminative SPD matrices. As mentioned in Section , the elements of the spatio-temporal descriptor are all 4×4 SPD matrices. Consequently, we set up three mapping blocks. The weight sizes of these three blocks are set to 3×4, 2×3 and 1×2, leading to the resulting SPD matrix sizes to be 3×3, 2×2 and 1×1. In this experiment, we set Di-StddNet ($b = 16$), the best performing model so far, as the baseline network to analyze the effect of non-linear mapping block. As shown in Table 2, we evaluate the recognition performance of Di-StddNet ($b = 16$) with different number of mapping blocks. The result shows that adding the mapping blocks appropriately can improve the recognition accuracy. And the highest recognition accuracy is achieved with 2 blocks added. After

	0block	1block	2blocks
$b = 8$	$74.60\%\pm2.12$	$76.31\%\pm2.73$	$77.83\%\pm3.07$
$b = 12$	$76.43\%\pm2.25$	$78.27\%\pm2.69$	$80.27\%\pm2.35$
$b = 16$	$77.58\%\pm2.98$	$79.14\%\pm2.51$	$82.32\%\pm2.27$

Table 5: Recognition accuracies of Di-StddNets with different configurations on the HDM05 dataset.

Method	Accuracy
Lie Group (Vemulapalli, Arrate, and Chellappa 2014)	$70.26\% \pm 2.89$
Rolling Rotations (Vemulapalli and Chellapa 2016)	$71.31\% \pm 3.21$
SPDNet (Huang and Van Gool 2017)	$61.45\% \pm 1.12$
LieNet (Huang et al. 2017)	$75.78\%\pm2.26$
PA-LSTM (Shahroudy et al. 2016)	$73.42\%\pm2.05$
ST-GCN (Yan, Xiong, and Lin 2018)	$82.13\% \pm 2.39$
Di-StddNet $(b = 16)$ + 2blocks	$82.32\%\pm2.27$

Table 6: Comparisons of the recognition result on the HDM05 dataset.

that, adding additional blocks will harm the performance. This is reasonable, because when 3 blocks are used, the final output SPD matrix will be reduced to be 1×1, resulting in the loss of too much useful information. We believe that the performance improvements obtained by Di-StddNet + 1block and Di-StddNet + 2blocks indicate the benefit of non-linear mapping blocks for SPD metric learning.

Comparison with the State-of-the-Art

G3D-Gaming dataset. In this dataset, we follow the standard cross-subject test protocol and use half of the subjects for training. As we discussed in Section , the original elements of the difference descriptor are all 4×4 SPD matrices, and the weight sizes of the non-linear mapping blocks are 3×4, 2×3 and 1×2. Thus, the resulting SPD matrices will be of size 1×1 after 3 blocks are added. In theory, this would result in the loss of too much useful information and decrease the recognition performance. The ablation studies of the non-linear mapping blocks justify our argumentation. Therefore, we equip the Di-StddNet with at most 2 blocks in our experiments. Table 3 shows the recognition performance of Di-StddNets with different configurations. As we can see, stacking some more mapping blocks of 2D fully connected and rectification function layers can improve the performance of Di-StddNet. And the best performance can be achieved when Di-StddNet $(b = 16)$ with 2 blocks added. Results shown in Table 4 suggest that our best performing Di-StddNet model outperforms all the previous methods.

HDM05 dataset. Results from Table 5 and 6 suggest that, compared with the manifold-based methods Lie group (Vemulapalli, Arrate, and Chellappa 2014), Rolling Rotations (Vemulapalli and Chellapa 2016), SPDNet (Huang and Van Gool 2017) and LieNet (Huang et al. 2017), the proposed Di-StddNet models gain significant performance improvements. Moreover, our Di-StddNet $(b = 16)$ with 2 blocks still outperforms the deep learning based LSTM and CNN models. The better performance of our proposed Di-StddNet with non-linear mapping blocks demonstrate the superiority of the proposed method.

	0block	1block	2blocks
$b = 8$	87.3%	89.5%	90.6%
$b = 12$	88.4%	91.7%	93.8%
$b = 16$	89.8%	90.9%	92.4%

Table 7: Recognition accuracies of Di-StddNets with different configurations on the Northwestern-UCLA dataset.

Method	Accuracy
Lie Group (Vemulapalli, Arrate, and Chellappa 2014)	74.2%
Actionlet ensemble (Wang et al. 2012)	76.0%
HBRNN (Du, Wang, and Wang 2015)	78.5%
Visualization CNN (Liu, Liu, and Chen 2017)	86.1%
Ensemble TS-LSTM (Lee et al. 2017)	89.2%
AGC-LSTM (Si et al. 2019)	93.3%
Di-StddNet $(b = 12)$ + 2blocks	93.8%

Table 8: Comparisons of the recognition result on the Northwestern-UCLA dataset.

Northwestern-UCLA dataset. As shown in Table 7, with the number of non-linear mapping blocks increasing, the Di-StddNet achieves better performance and our Di-StddNet $(b = 12)$ with 2 blocks obtains the highest recognition accuracy along different configurations. Table 8 gives the performance comparisons of our model with other state-of-the-art methods. Experimental results show that the proposed Di-StddNet $(b = 12)$ + 2blocks model outperforms the previous state-of-the-art methods.

Conclusion

In this paper, we propose a spatio-temporal difference descriptor for non-linear metric learning on SPD manifold. A directional convolution network is first introduced to generate new human skeletons with the contextual joint dependencies learned in the pre-training process. Then, the proposed model constructs a set of spatio-temporal difference descriptors based on the new skeleton sequences. The difference descriptors can capture the intra-frame skeleton differences and inter-frame dynamic changes that are complementary to original static skeleton information for human action recognition. In addition, we introduce 2D fully connected layer and rectification operation on SPD matrices to further improve the performance of manifold metric learning. Experiments on several standard datasets demonstrate the effectiveness of the proposed model.

References

Anirudh, R.; Turaga, P.; Su, J.; and Srivastava, A. 2017. Elastic functional coding of Riemannian trajectories. *IEEE transactions on pattern analysis and machine intelligence* 39(5): 922–936.

Atwood, J.; and Towsley, D. 2016. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1993–2001.

Ben Tanfous, A.; Drira, H.; and Ben Amor, B. 2018. Coding Kendall's Shape Trajectories for 3D Action Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2840–2849.

Bloom, V.; Makris, D.; and Argyriou, V. 2012. G3D: A gaming action dataset and real time action recognition evaluation framework. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 7–12. IEEE.

Cao, Z.; Simon, T.; Wei, S.-E.; and Sheikh, Y. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7291–7299.

Chavel, I. 2006. *Riemannian geometry: a modern introduction*, volume 98. Cambridge university press.

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

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 1024–1034.

Huang, Z.; and Van Gool, L. 2017. A riemannian network for spd matrix learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Huang, Z.; Wan, C.; Probst, T.; and Van Gool, L. 2017. Deep learning on lie groups for skeleton-based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6099–6108.

Johansson, G. 1973. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics* 14(2): 201–211.

Kim, T. S.; and Reiter, A. 2017. Interpretable 3d human action analysis with temporal convolutional networks. In *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, 1623–1631. IEEE.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Lee, I.; Kim, D.; Kang, S.; and Lee, S. 2017. Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In *Proceedings of the IEEE international conference on computer vision*, 1012–1020.

Li, C.; Cui, Z.; Zheng, W.; Xu, C.; and Yang, J. 2018. Spatio-temporal graph convolution for skeleton based action recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 3482–3489.

Li, W.; Zhang, Z.; and Liu, Z. 2010. Action recognition based on a bag of 3d points. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, 9–14. IEEE.

Liu, J.; Shahroudy, A.; Xu, D.; and Wang, G. 2016. Spatio-temporal LSTM with trust gates for 3D human action recognition. In *European Conference on Computer Vision*, 816–833. Springer.

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

Lovrić, M.; Min-Oo, M.; and Ruh, E. A. 2000. Multivariate normal distributions parametrized as a Riemannian symmetric space. *Journal of Multivariate Analysis* 74(1): 36–48.

Müller, M.; Röder, T.; Clausen, M.; Eberhardt, B.; Krüger, B.; and Weber, A. 2007. Documentation Mocap Database HDM05. Technical Report CG-2007-2, Universität Bonn.

Nguyen, X. S.; Brun, L.; Lézoray, O.; and Bougleux, S. 2019. A Neural Network Based on SPD Manifold Learning for Skeleton-Based Hand Gesture Recognition. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 12028–12037.

Nie, S.; and Ji, Q. 2014. Capturing global and local dynamics for human action recognition. In *2014 22nd International Conference on Pattern Recognition*, 1946–1951. IEEE.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*, 2014–2023.

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

Si, C.; Chen, W.; Wang, W.; Wang, L.; and Tan, T. 2019. An attention enhanced graph convolutional lstm network for skeleton-based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1227–1236.

Song, S.; Lan, C.; Xing, J.; Zeng, W.; and Liu, J. 2017. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Thakkar, K.; and Narayanan, P. 2018. Part-based graph convolutional network for action recognition. *arXiv preprint arXiv:1809.04983* .

Tosato, D.; Spera, M.; Cristani, M.; and Murino, V. 2012. Characterizing humans on riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8): 1972–1984.

Vemulapalli, R.; Arrate, F.; and Chellappa, R. 2014. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 588–595.

Vemulapalli, R.; and Chellapa, R. 2016. Rolling rotations for recognizing human actions from 3d skeletal data. In

Proceedings of the IEEE conference on computer vision and pattern recognition, 4471–4479.

Wang, J.; Liu, Z.; Wu, Y.; and Yuan, J. 2012. Mining actionlet ensemble for action recognition with depth cameras. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 1290–1297. IEEE.

Wang, J.; Nie, X.; Xia, Y.; Wu, Y.; and Zhu, S.-C. 2014. Cross-view Action Modeling, Learning and Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, 20–36. Springer.

Xia, L.; Chen, C.-C.; and Aggarwal, J. K. 2012. View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 20–27. IEEE.

Yan, S.; Xiong, Y.; and Lin, D. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 7444–7452.

Zhao, Y.; Xiong, Y.; and Lin, D. 2018. Recognize actions by disentangling components of dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6566–6575.