A neural network based on SPD manifold learning for skeleton-based hand gesture recognition

Xuan Son Nguyen, Luc Brun, Olivier Lézoray, Sébastien Bougleux
Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

Presented at the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)
2019, June 16-21, Long Beach, CA (http://cvpr2019.thecvf.com/)

Abstract

This paper proposes a new neural network based on SPD manifold learning for skeleton-based hand gesture recognition. Given the stream of hand's joint positions, our approach combines two aggregation processes on respectively spatial and temporal domains. The pipeline of our network architecture consists in three main stages. The first stage is based on a convolutional layer to increase the discriminative power of learned features. The second stage relies on different architectures for spatial and temporal Gaussian aggregation of joint features. The third stage learns a final SPD matrix from skeletal data. A new type of layer is proposed for the third stage, based on a variant of stochastic gradient descent on Stiefel manifolds. The proposed network is validated on two challenging datasets and shows state-of-the-art accuracies on both datasets.

1. Introduction

Hand gesture recognition is an important research topic with applications in many fields, e.g., assisted living, human-robot interaction or sign language interpretation. A large family of hand gesture recognition methods is based on low-level features extracted from images, e.g., spatio-temporal interest points. However, with the introduction of affordable depth sensing cameras, e.g., Intel Realsense or Microsoft Kinect, and the availability of highly accurate joint tracking algorithms, skeletal data can be obtained effectively with good precision. Skeletal data provides a rich and high level description of the hand. This has led to an extensive development of approaches for skeleton-based hand gesture recognition in recent years.

Early works on the recognition of hand gestures or human actions from skeletal data are based on a modeling of the skeleton's movement as time series [49, 52]. The recognition step is thus based on the comparison of sequences of features describing skeleton's movements using, e.g., Dynamic Time Warping [49] or Fourier Temporal Pyramid [52].

Such approaches ignore the high correlations existing between the movement of two adjacent hand joints (e.g., two joints of a same finger) within a hand gesture. Taking into account this information is a crucial step for hand gesture recognition and requires the definition and the appropriate processing of hand joints' neighborhoods.

Recently, graph convolutional networks for action recognition [27, 56] have shown excellent performance by taking into account physical connections of body joints defined by the underlying structure of body skeleton. While the use of physical connections of skeleton joints are important for capturing discriminating cues in hand gesture and action recognition, the identification of other connections induced by the performed gestures and actions are also useful and can greatly improve recognition accuracy [45].

Motivated by this observation, we model in this work the hand skeleton as a 2D grid where connections, different from the classical physical connections of hand joints, are added to better capture patterns defined by hand joints' movements. Figure 1(a) shows the hand joint positions estimated by an Intel Realsense camera. Since the hand skeleton has an irregular geometric structure that differs from grid-shaped structures, the 2D grid is constructed from the hand skeleton by removing some hand joints and adding connections between neighboring joints. Figure 1(b) shows a 2D grid corresponding to the hand skeleton in Fig. 1(a). This 2D grid integrates adjacency relationships between hand joints that often have correlated movements. Moreover, this modeling allows us to use a classical convolutional layer instead of a graph convolutional operator on an arbitrary geometric graph [56].

Our approach relies on SPD matrices to aggregate features resulting from the convolutional layer. The SPD matrices considered in this work combine mean and covariance information which have been shown effective in various vision tasks [15, 21, 22]. Since SPD matrices are known to lie on a Riemannian manifold, specific layers for deep neural networks of SPD matrices should be designed [19, 60]. Beside good performances on action recognition tasks, these networks do not put a focus on spatial and temporal rela-

tionships of skeleton joints. This motivates us to design a neural network model for learning a SPD matrix-based gesture representation from skeletal data with a special attention on those relationships. In our work, the encoding of spatial and temporal relationships of hand joints is performed using different network architectures. This allows to capture relevant statistics for individual hand joints as well as groups of hand joints whose movement is highly correlated with that of other joints in the group. The experimental evaluation shows that our method significantly improves the state-of-the-art methods on two standard datasets.

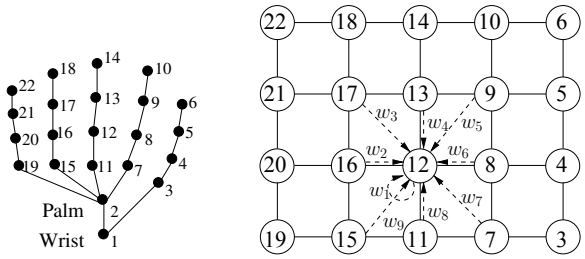
2. Related Works

This section presents representative works for skeleton-based hand gesture recognition (Sec. 2.1) and deep neural networks for SPD manifold learning (Sec. 2.2).

2.1. Skeleton-Based Gesture Recognition

Most of approaches can be categorized as hand-crafted feature-based approaches or deep learning approaches. Hand-crafted feature-based approaches describe relationships of hand and body joints in different forms to represent gestures and actions. The simplest proposed relationships are relative positions between pairs of joints [35, 46, 57]. More complex relationships were also exploited, e.g., skeletal quad [8] or 3D geometric relationships of body parts in a Lie group [49]. Temporal relationships have also been taken into account and proven effective [50]. While all joints are involved in the performed gestures and actions, only a subset of key joints is important for the recognition task. These are called informative joints and they can be automatically identified using information theory [37]. This allows to avoid considering non-informative joints that often bring noise and degrade performance.

Motivated by the success of deep neural networks in various vision tasks [13, 17, 26], deep learning approaches for action and gesture recognition have been extensively studied in recent years. To capture spatial and temporal relationships of hand and body joints, they rely mainly on Convolutional Neural Network (CNN) [4, 25, 32, 33, 36, 53], Recurrent Neural Network (RNN) [6, 51] and Long Short-Term Memory (LSTM) [31, 36, 44]. While hand-crafted feature-based approaches have used informative joints to improve recognition accuracy, deep learning approaches were based on attention mechanism to selectively focus on relevant parts of skeletal data [30, 55]. Recently, deep learning on manifolds and graphs has increasingly attracted attention. Approaches following this line of research have also been successfully applied to skeleton-based action recognition [19, 20, 23, 27, 56]. By extending classical operations like convolutions to manifolds and graphs while respecting the underlying geometric structure of data, they have demonstrated superior performance over other approaches.



(a) (b)

Figure 1: (a) Hand joints estimated by an Intel RealSense camera (b) graph of hand skeleton and the weights associated with the neighbors of node 12 in the convolutional layer.

2.2. Deep Learning of SPD Matrices

In recent years the deep learning community has shifted its focus towards developing approaches that deal with data in a non-Euclidean domain, e.g., Lie groups [20], SPD manifolds [19] or Grassmann manifolds [23]. Among them, those that deal with SPD manifolds have received particular attention. This comes from the popular applications of SPD matrices in many vision problems [1, 14, 16, 58].

Deep neural networks for SPD matrix learning aim at projecting a high-dimensional SPD matrix into a more discriminative low-dimensional one. Differently from classical CNNs, their layers are designed so that they preserve the geometric structure of input SPD matrices, i.e., their output are also SPD matrices. In [5], a 2D fully connected layer was proposed for the projection, while in [19] it was achieved by a Bimap layer. Inspired by ReLU layers in CNNs, different types of layers that perform non-linear transformations of SPD matrices were also introduced [5, 7, 19]. To classify the final SPD matrix, a layer is generally required to map it to an Euclidean space. Most of approaches rely on the two widely used operations in many machine learning models, i.e., singular value decomposition (SVD) and eigen value decomposition (EIG) for constructing this type of layers [19, 29, 54]. As gradients involved in SVD and EIG cannot be computed by traditional backpropagation, they exploit the chain rule established by Ionescu et al. [24] for backpropagation of matrix functions in deep learning.

3. The Proposed Approach

In this section, we present our network model referred to as Spatial-Temporal and Temporal-Spatial Hand Gesture Recognition Network (ST-TS-HGR-NET). An overview of our network is given in Section 3.1. The different components of our network are explained in Sections 3.2, 3.3, 3.4, and 3.5. In Section 3.6, we show how our network is trained for gesture recognition. Finally, Section 3.7 points out the

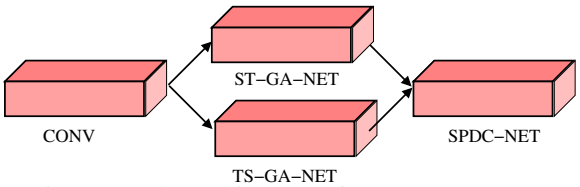


Figure 2: The architecture of ST-TS-HGR-NET.

relations of our approach with previous approaches.

3.1. Overview of The Proposed Network

Our network illustrated in Fig. 2 is made up of three components. The first component, referred to as CONV, is a convolutional layer applied on the 2D grid encoding the hand skeletal data (Fig. 1). Filter weights are shared over all frames of the sequence.

The second component is based on the Gaussian embedding method of [34] and is used to capture first- and second-order statistics. This component is composed of two different architectures for feature aggregation referred to as Spatial-Temporal Gaussian Aggregation Sub-Network (ST-GA-NET) and Temporal-Spatial Gaussian Aggregation Sub-Network (TS-GA-NET).

The third component, referred to as SPD Matrix Learning and Classification Sub-Network (SPDC-NET), learns a SPD matrix from a set of SPD matrices and maps the resulting SPD matrix, which lies on a Riemannian manifold, to an Euclidean space for classification.

In the following, we explain in detail each component of our network. The backpropagation procedures of our network's layers are given in Appendix A.

3.2. Convolutional Layer

The convolutional layer (Fig. 3) used in the first place of our network allows to combine joints with correlated variations (Section 1). Let N_J and N_F be respectively the number of hand joints and the length of the skeleton sequence. Let us denote by $\mathbf{p}_{0,i}^t \in \mathbb{R}^3, i = 1, \ldots, N_J, t = 1, \ldots, N_F$, the 3D coordinates of hand joint i at frame t. We define a 2D grid where each node represents a hand joint i at a frame t (Section 1). The grid has three channels corresponding to the x, y, and z coordinates of hand joints. Fig. 1(b) shows the 2D grid corresponding to the hand skeleton in Fig. 1(a), where each node has at most 9 neighbors including itself. Let d_{out}^c be the output dimension of the convolutional layer. Let us denote by $\mathbf{p}_i^t \in \mathbb{R}^{d_{out}^c}, i = 3, \ldots, N_J, t = 1, \ldots, N_F$, the output of the convolutional layer. The output feature vector at node i is computed as:

$$\mathbf{p}_i^t = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{l(j,i)} \mathbf{p}_{0,j}^t, \tag{1}$$

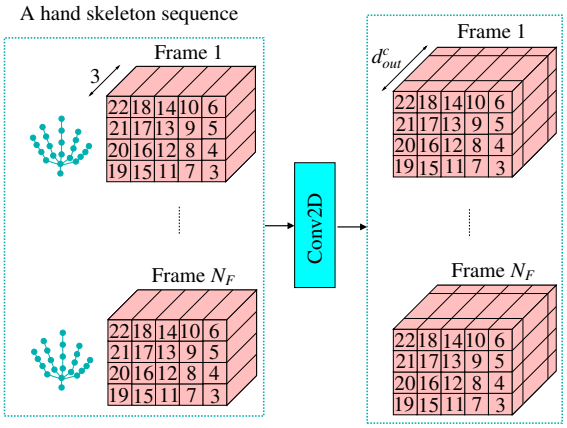


Figure 3: Illustration of sub-network CONV.

where $\mathcal{N}_i$ is the set of neighbors of node i, $\mathbf{W}_{l(j,i)}$ is the filter weight matrix, and $l(j,i)$ is defined as:

$j-i$	0	4	5	1	-3	-4	-5	-1	3
$l(j,i)$	1	2	3	4	5	6	7	8	9

(2)

3.3. Spatial-Temporal Gaussian Aggregation Sub-Network

To capture the temporal ordering of a skeleton sequence, a number of sub-sequences are constructed and then fed to different branches of ST-GA-NET (see Fig. 4). A branch of ST-GA-NET is designed to aggregate features for a sub-sequence of a specific finger. In this paper, we construct six sub-sequences for each skeleton sequence. The first sub-sequence is the original sequence. The next two sub-sequences are obtained by dividing the sequence into two sub-sequences of equal length. The last three sub-sequences are obtained by dividing the sequence into three sub-sequences of equal length. This results in 30 branches for ST-GA-NET (6 sub-sequences $\times$ 5 fingers).

To aggregate features in a branch associated with sub-sequence s and finger f, $s = 1, \ldots, 6, f = 1, \ldots, 5$, each frame of sub-sequence s is processed through 4 layers. Let J_f be the set of hand joints belonging to finger f, t_b^s, t_e^s be the beginning and ending frames of sub-sequence s, t be a given frame of sub-sequence s, $\{\mathbf{p}_{s,j}^i | j \in J_f, i = t_b^s, \ldots, t_e^s\}$ be the subset of output feature vectors of the convolutional layer that are fed to the branch. Let us finally consider a sliding window $\{t - t_0, \ldots, t + t_0\}$ centered on frame t. Following previous works [28, 43], we assume that $\mathbf{p}_{s,j}^i$, $j \in J_f, i = t - t_0, \ldots, t + t_0$, are independent and identically distributed samples from a Gaussian distribution (hereafter abbreviated as Gaussian for simplicity):

$$\mathcal{N}(\mathbf{p};\boldsymbol{\mu}^t_{s,f},\boldsymbol{\Sigma}^t_{s,f}) = \frac{1}{|2\pi\boldsymbol{\Sigma}^t_{s,f}|^{\frac{1}{2}}}\exp(-\frac{1}{2}(\mathbf{p}-\boldsymbol{\mu}^t_{s,f})^T(\boldsymbol{\Sigma}^t_{s,f})^{-1}(\mathbf{p}-\boldsymbol{\mu}^t_{s,f})), \tag{3}$$

where $|.|$ is the determinant, $\boldsymbol{\mu}^t_{s,f}$ is the mean vector and $\boldsymbol{\Sigma}^t_{s,f}$ is the covariance matrix. The parameters of the Gaussian can be estimated as:

$$\boldsymbol{\mu}^t_{s,f} = \frac{1}{(2t_0+1)|J_f|}\sum_{j\in J_f}\sum_{i=t-t_0}^{t+t_0}\mathbf{p}^i_{s,j}, \tag{4}$$

$$\boldsymbol{\Sigma}^t_{s,f} = \frac{1}{(2t_0+1)|J_f|}\sum_{j\in J_f}\sum_{i=t-t_0}^{t+t_0}(\mathbf{p}^i_{s,j}-\boldsymbol{\mu}^t_{s,f})(\mathbf{p}^i_{s,j}-\boldsymbol{\mu}^t_{s,f})^T. \tag{5}$$

Based on the method in [34] that embeds the space of Gaussians in the Riemannian symmetric space, the Gaussian can be identified as a SPD matrix given by:

$$\mathbf{Y}^t_{s,f} = \begin{bmatrix}\boldsymbol{\Sigma}^t_{s,f}+\boldsymbol{\mu}^t_{s,f}(\boldsymbol{\mu}^t_{s,f})^T & \boldsymbol{\mu}^t_{s,f}\\ (\boldsymbol{\mu}^t_{s,f})^T & 1\end{bmatrix}. \tag{6}$$

The GaussAgg layer is designed to perform the computation of Eq. 6, that is:

$$\mathbf{Y}^t_{s,f} = h_{ga}(\{\mathbf{p}^i_{s,j}\}^{i=t-t_0,\ldots,t+t_0}_{j\in J_f}), \tag{7}$$

where h_{ga} is the mapping of the GaussAgg layer, $\mathbf{Y}^t_{s,f}$ is the output of the GaussAgg layer.

The next layer ReEig [19] introduces non-linear transformations of SPD matrices via a mapping defined as:

$$\mathbf{Y}^t_{s,f} = h_r(\mathbf{X}^t_{s,f}) = \mathbf{U}\max(\epsilon\mathbf{I},\mathbf{V})\mathbf{U}^T, \tag{8}$$

where h_r is the mapping of the ReEig layer, $\mathbf{X}^t_{s,f}$ and $\mathbf{Y}^t_{s,f}$ are the input and output SPD matrices, $\mathbf{X}^t_{s,f} = \mathbf{U}\mathbf{V}\mathbf{U}^T$ is the eigen-decomposition of $\mathbf{X}^t_{s,f}$, ϵ is a rectification threshold, $\mathbf{I}$ is the identity matrix, $\max(\epsilon\mathbf{I},\mathbf{V})$ is a diagonal matrix whose diagonal elements are defined as:

$$(\max(\epsilon\mathbf{I},\mathbf{V}))(i,i) = \begin{cases}\mathbf{V}(i,i) & \text{if } \mathbf{V}(i,i)>\epsilon\\ \epsilon & \text{if } \mathbf{V}(i,i)\le\epsilon.\end{cases} \tag{9}$$

After the ReEig layer, the LogEig layer [19] is used to map SPD matrices to Euclidean spaces. Formally, the mapping of this layer is defined as:

$$\mathbf{Y}^t_{s,f} = h_l(\mathbf{X}^t_{s,f}) = \log(\mathbf{X}^t_{s,f}) = \mathbf{U}\log(\mathbf{V})\mathbf{U}^T, \tag{10}$$

where h_l is the mapping of the LogEig layer, $\mathbf{X}^t_{s,f}$ and $\mathbf{Y}^t_{s,f}$ are the input and output SPD matrices, as before.

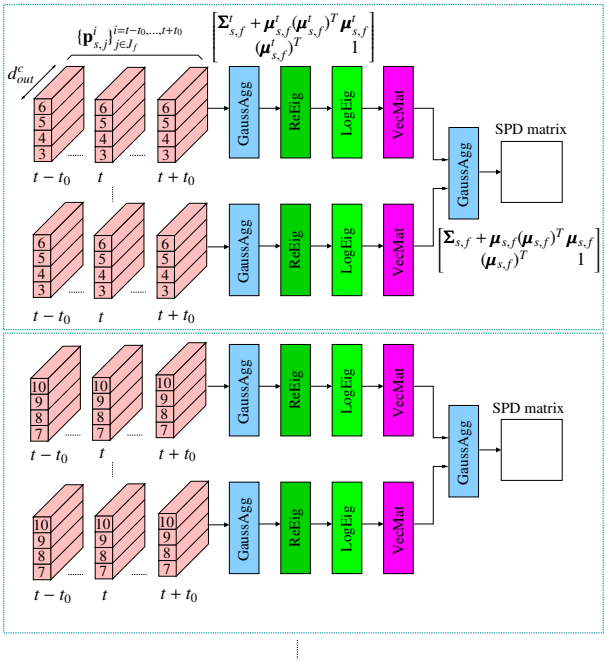


Figure 4: The architecture of ST-GA-NET (two different branches are shown).

The next layer, referred to as VecMat, vectorizes SPD matrices by the following mapping [48]:

$$\mathbf{y}^t_{s,f} = h_{vm}(\mathbf{X}^t_{s,f}) = [\mathbf{X}^t_{s,f}(1,1), \sqrt{2}\mathbf{X}^t_{s,f}(1,2), \ldots, \sqrt{2}\mathbf{X}^t_{s,f}(1,d^c_{out}+1), \mathbf{X}^t_{s,f}(2,2), \sqrt{2}\mathbf{X}^t_{s,f}(2,3), \ldots, \mathbf{X}^t_{s,f}(d^c_{out}+1,d^c_{out}+1)]^T, \tag{11}$$

where h_{vm} is the mapping of the VecMat layer, $\mathbf{X}^t_{s,f} \in \mathbb{R}^{d^c_{out}+1}$ is the input matrix, $\mathbf{y}^t_{s,f}$ is the output vector, $\mathbf{X}^t_{s,f}(i,i), i=1,\ldots,d^c_{out}+1$, are the diagonal entries of $\mathbf{X}^t_{s,f}$ and $\mathbf{X}^t_{s,f}(i,j), i<j, i,j=1,\ldots,d^c_{out}+1$, are the off-diagonal entries.

We again assume that $\mathbf{y}^t_{s,f}, t=t^s_b,\ldots,t^s_e$, are independent and identically distributed samples from a Gaussian $\mathcal{N}(\mathbf{y};\boldsymbol{\mu}_{s,f},\boldsymbol{\Sigma}_{s,f})$ whose parameters can be estimated as:

$$\boldsymbol{\mu}_{s,f} = \frac{1}{t^s_e-t^s_b+1}\sum_{t=t^s_b}^{t^s_e}\mathbf{y}^t_{s,f}, \tag{12}$$

$$\boldsymbol{\Sigma}_{s,f} = \frac{1}{t^s_e-t^s_b+1}\sum_{t=t^s_b}^{t^s_e}(\mathbf{y}^t_{s,f}-\boldsymbol{\mu}_{s,f})(\mathbf{y}^t_{s,f}-\boldsymbol{\mu}_{s,f})^T. \tag{13}$$

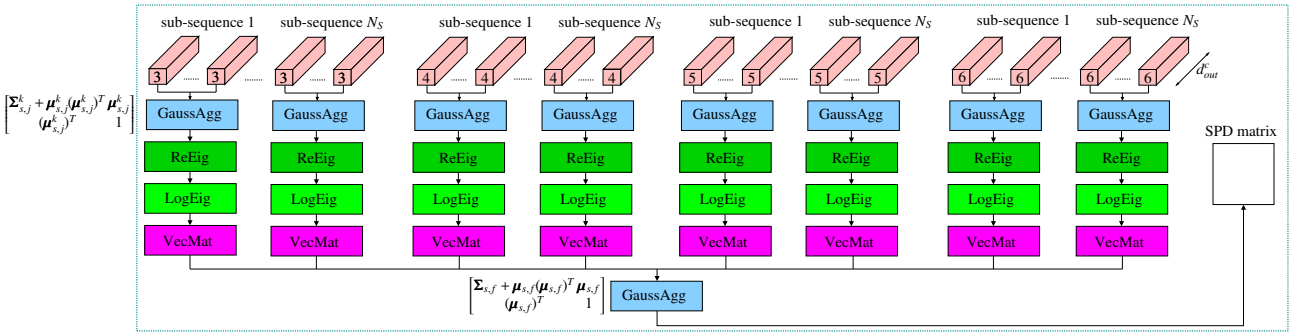


Figure 5: Illustration of a branch of TS-GA-NET.

The second GaussAgg layer then performs the mapping:

$$\begin{aligned}\mathbf{Y}_{s,f} &= h_{ga}(\{\mathbf{y}_{s,f}^{t}\}^{t=t_b^s,\ldots,t_e^s}) \\ &= \begin{bmatrix}\boldsymbol{\Sigma}_{s,f}+\boldsymbol{\mu}_{s,f}(\boldsymbol{\mu}_{s,f})^T & \boldsymbol{\mu}_{s,f}\\ (\boldsymbol{\mu}_{s,f})^T & 1\end{bmatrix}.\end{aligned} \tag{14}$$

The resulting SPD matrix $\mathbf{Y}_{s,f}$ describes variations of finger f along sub-sequence s.

3.4. Temporal-Spatial Gaussian Aggregation Sub-Network

Similarly to ST-GA-NET, TS-GA-NET is composed of 30 branches where each branch aggregates features for a sub-sequence of a specific finger. The sub-sequences are constructed in exactly the same way as ST-GA-NET. However, the feature aggregation procedure at the first and second GaussAgg layers are performed differently. More precisely, considering the branch associated with sub-sequence s and finger f. First, sub-sequence s is further divided into N_S sub-sequences of equal length. Let $t_{b,k}^s$ and $t_{e,k}^s$, $k = 1,\ldots,N_S$, be the beginning and ending frames of these sub-sequences. Then for a given hand joint $j \in J_f$ and sub-sequence k, the first GaussAgg layer computes a SPD matrix given as:

$$\mathbf{Y}_{s,j}^{k} = \begin{bmatrix}\boldsymbol{\Sigma}_{s,j}^{k}+\boldsymbol{\mu}_{s,j}^{k}(\boldsymbol{\mu}_{s,j}^{k})^T & \boldsymbol{\mu}_{s,j}^{k}\\ (\boldsymbol{\mu}_{s,j}^{k})^T & 1\end{bmatrix}, \tag{15}$$

where $\boldsymbol{\mu}_{s,j}^{k} = \frac{1}{t_{e,k}^s - t_{b,k}^s+1}\sum_{t=t_{b,k}^s}^{t_{e,k}^s}\mathbf{p}_{s,j}^{t}$ and $\boldsymbol{\Sigma}_{s,j}^{k} = \frac{1}{t_{e,k}^s - t_{b,k}^s+1}\sum_{t=t_{b,k}^s}^{t_{e,k}^s}(\mathbf{p}_{s,j}^{t}-\boldsymbol{\mu}_{s,j}^{k})(\mathbf{p}_{s,j}^{t}-\boldsymbol{\mu}_{s,j}^{k})^T$.

Note that $\mathbf{Y}_{s,j}^{k}$ encodes the first- and second-order statistics of hand joint j computed within sub-sequence k. This temporal variation of individual joints is not captured by the first GaussAgg layer of ST-GA-NET. The resulting SPD matrices are processed through the ReEig, LogEig and VecMat layers. Let $\mathbf{y}_{s,j}^{k}, k = 1,\ldots,N_S, j \in J_f$, be the output vectors of the VecMat layer of the branch. The second GaussAgg layer of TS-GA-NET then performs the following mapping:

$$\begin{aligned}\mathbf{Y}_{s,f} &= h_{ga}(\{\mathbf{y}_{s,j}^{k}\}_{j\in J_f}^{k=1,\ldots,N_S}) \\ &= \begin{bmatrix}\boldsymbol{\Sigma}_{s,f}+\boldsymbol{\mu}_{s,f}(\boldsymbol{\mu}_{s,f})^T & \boldsymbol{\mu}_{s,f}\\ (\boldsymbol{\mu}_{s,f})^T & 1\end{bmatrix},\end{aligned} \tag{16}$$

where $\boldsymbol{\mu}_{s,f}$ and $\boldsymbol{\Sigma}_{s,f}$ can be estimated as:

$$\boldsymbol{\mu}_{s,f} = \frac{1}{N_S|J_f|}\sum_{j\in J_f}\sum_{k=1}^{N_S}\mathbf{y}_{s,j}^{k}, \tag{17}$$

$$\boldsymbol{\Sigma}_{s,f} = \frac{1}{N_S|J_f|}\sum_{j\in J_f}\sum_{k=1}^{N_S}(\mathbf{y}_{s,j}^{k}-\boldsymbol{\mu}_{s,f})(\mathbf{y}_{s,j}^{k}-\boldsymbol{\mu}_{s,f})^T. \tag{18}$$

3.5. SPD Matrix Learning and Classification Sub-Network

The outputs of sub-networks ST-GA-NET and TS-GA-NET are sets of SPD matrices. The objective of the classification sub-network (see Fig. 6) is to transform those sets to a new SPD matrix, then map it to an Euclidean space for classification. The mapping h_{spda} of the SPDAgg layer is defined as:

$$\begin{aligned}\mathbf{Y} &= h_{spda}((\mathbf{X}_1,\ldots,\mathbf{X}_N);\mathbf{W}_1,\ldots,\mathbf{W}_N) \\ &= \sum_{i=1}^{N}\mathbf{W}_i\mathbf{X}_i(\mathbf{W}_i)^T,\end{aligned} \tag{19}$$

where $\mathbf{X}_i \in \mathbb{R}^{d_{in}^s\times d_{in}^s}, i = 1,\ldots,N$, are the input SPD matrices, $\mathbf{W}_i \in \mathbb{R}^{d_{out}^s\times d_{in}^s}$ are the transformation matrices, $\mathbf{Y} \in \mathbb{R}^{d_{out}^s\times d_{out}^s}$ is the output matrix.

To guarantee that the output $\mathbf{Y}$ is SPD, we remark that

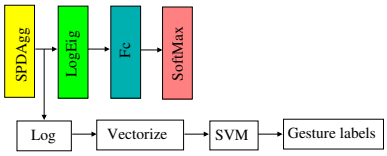


Figure 6: The architecture of SPDC-NET.

the right-hand side of Eq. (19) can be rewritten as:

$$\sum_{i=1}^{N} \mathbf{W}_i \mathbf{X}_i (\mathbf{W}_i)^T = \hat{\mathbf{W}} \, \mathrm{diag}(\mathbf{X}_1, \ldots, \mathbf{X}_N) (\hat{\mathbf{W}})^T, \quad (20)$$

where $\hat{\mathbf{W}} = [\mathbf{W}_1, \ldots, \mathbf{W}_N]$ and $\mathrm{diag}(\mathbf{X}_1, \ldots, \mathbf{X}_N)$ is constructed such that its diagonal contains the diagonal entries of $\mathbf{X}_1, \ldots, \mathbf{X}_N$:

$$\mathrm{diag}(\mathbf{X}_1, \ldots, \mathbf{X}_N) = \begin{bmatrix} \mathbf{X}_1 \ldots\ldots\ldots \\ \ldots \mathbf{X}_2 \ldots\ldots \\ \ldots\ldots\ldots \mathbf{X}_N \end{bmatrix}. \quad (21)$$

It can be easily seen that $\mathrm{diag}(\mathbf{X}_1, \ldots, \mathbf{X}_N)$ is a valid SPD matrix, as for any vector $\mathbf{x} \neq 0$, one has $\mathbf{x}^T \mathrm{diag}(\mathbf{X}_1, \ldots, \mathbf{X}_N)\mathbf{x} = \sum_{i=1}^{N} (\mathbf{x}_i)^T \mathbf{X}_i \mathbf{x}_i$, where $\mathbf{x} = [(\mathbf{x}_1)^T, \ldots, (\mathbf{x}_N)^T]^T$ and the vectors $\mathbf{x}_i, i = 1, \ldots, N$, have equal sizes. The right-hand side of the above equation is strictly positive since $(\mathbf{x}_i)^T \mathbf{X}_i \mathbf{x}_i \geq 0, \forall i = 1, \ldots, N$, and there must exist $i' \in \{1, \ldots, N\}$ such that $\mathbf{x}_{i'} \neq 0$ (as $\mathbf{x} \neq 0$), which implies that $(\mathbf{x}_{i'})^T \mathbf{X}_{i'} \mathbf{x}_{i'} > 0$.

Inspired by [19], we assume that the combined matrix $\hat{\mathbf{W}} = [\mathbf{W}_1, \ldots, \mathbf{W}_N]$ is a full row rank matrix. Then optimal solutions of the transformation matrices are achieved by additionally assuming that $\hat{\mathbf{W}}$ resides on a compact Stiefel manifold $St(d^s_{out}, N \times d^s_{in})$[1]. The transformation matrices $\mathbf{W}_i$ are updated by optimizing $\hat{\mathbf{W}}$ and projecting the optimal $\hat{\mathbf{W}}$ on its columns. Note that the constraint on the dimension d^s_{out} of the output $\mathbf{Y}$ is: $d^s_{out} \leq N d^s_{in}$.

To map the output SPD matrix of the SPDAgg layer to an Euclidean space, we use the LogEig layer, followed by a fully connected (FC) layer and a softmax layer.

3.6. Gesture Recognition

The SPDAgg layer outputs a matrix $\mathbf{B} \in \mathbb{R}^{d^s_{out}}$ for each gesture sequence (see Fig. 6). This matrix is then transformed to its matrix logarithm and finally vectorized. The final representation of the gesture sequence is $\mathbf{v} = [b_{1,1}, \sqrt{2}b_{1,2}, \sqrt{2}b_{1,3}, \ldots, \sqrt{2}b_{1,d^s_{out}}, b_{2,2}, \sqrt{2}b_{2,3}, \ldots, b_{d^s_{out},d^s_{out}}]^T$ where $b_{i,i}, i = 1, \ldots, d^s_{out}$, are the diagonal entries of $\log(\mathbf{B})$ and $b_{i,j}, i < j, i, j = 1, \ldots, d^s_{out}$, are the off-diagonal entries of $\log(\mathbf{B})$.

[1]A compact Stiefel manifold $St(d^s_{out}, N \times d^s_{in})$ is the set of d^s_{out}-dimensional orthonormal matrices of $\mathbb{R}^{N \times d^s_{in}}$.

3.7. Relation with Previous Works

Our approach is closely related to [19, 54]. We point out in the following paragraphs the relations between the proposed network and those introduced in [19, 54].

- Our network considers temporal modeling for hand gesture recognition, while temporal modeling is not considered in [54] as they focus on image classification tasks. Moreover, in our work, a Gaussian is identified as a SPD matrix, while [54] identifies a Gaussian as the square root of a SPD matrix.

- Our network takes directly 3D coordinates of hand joints as input, while in [19], covariance matrices must be computed beforehand as input of their network.

- Our network relies not only on the second-order information (covariance) as [19] but also on the first-order information (mean). The first-order information has been proven to be useful in capturing the extra distribution information of low-level features [42]. Moreover, we consider the first- and second-order information for different subsets of hand joints, while [19] uses the whole set of joints to compute statistics. Our network is thus based on a finer granularity than [19].

- Our network combines two different and complementary architectures to better capture relevant statistics for the recognition task, which makes our network distinct from those of [19, 54].

4. Experiments

We conducted experiments using the Dynamic Hand Gesture (DHG) dataset [46, 47] and the First-Person Hand Action (FPHA) dataset [12]. In all experiments, the dimension of a output feature vector of the convolutional layer was set to 9 ($d^c_{out} = 9$), the dimensions of the transformation matrices of the SPDAgg layer were set to 200×56 ($d^s_{in} = 56$, $d^s_{out} = 200$). All sequences of the two datasets were normalized to have 500 frames ($N_F = 500$)[2]. The batch size and the learning rate were set to 30 and 0.01, respectively. The rectification threshold ϵ for the ReEig layer was set to 0.0001 [19]. The network trained at epoch 15 was used to create the final gesture representation. The classifier was learned using the LIBLINEAR library [9] with L2-regularized L2-loss (dual) where C was set to 1, the tolerance of termination criterion was set to 0.1 and no bias term was added. For FPHA dataset, the non-optimized CPU implementation of our network on a 3.4GHz machine with 24GB RAM and Matlab R2015b takes about 22 minutes per epoch and 7 minutes per epoch for training and testing,

[2]We tested with $N_F = 300, 500, 800$ and the difference between obtained results were marginal.

Num. of a hand joint's neighbors	FPHA	DHG (14 gestures)	DHG (28 gestures)
3	91.65	93.10	88.33
9	93.22	94.29	89.40

Table 1: Recognition accuracy (%) of our network for different settings of hand joint's neighborhood.

t_0	FPHA	DHG (14 gestures)	DHG (28 gestures)
1	93.22	94.29	89.40
2	93.04	94.17	89.04
3	93.04	94.29	89.40

Table 2: Recognition accuracy (%) of our network for different settings of t_0.

respectively. In the following, we provide details on the experimental settings and results obtained for each dataset.

4.1. Datasets and Experimental Settings

DHG dataset. The DHG dataset contains 14 gestures performed in two ways: using one finger and the whole hand. Each gesture is executed several times by different actors. Gestures are subdivided into fine and coarse categories. The dataset provides the 3D coordinates of 22 hand joints as illustrated in Fig. 1(a). It has been split into 1960 train sequences (70% of the dataset) and 840 test sequences (30% of the dataset) [47].

FPHA dataset. This dataset contains 1175 action videos belonging to 45 different action categories, in 3 different scenarios, and performed by 6 actors. Action sequences present high inter-subject and intra-subject variability of style, speed, scale, and viewpoint. The dataset provides the 3D coordinates of 21 hand joints as DHG dataset except for the palm joint. We used the 1:1 setting proposed in [12] with 600 action sequences for training and 575 for testing.

4.2. Ablation Study

In this section, we examine the influence of different components of our network on its accuracy. The default values of t_0 and N_S are set to 1 and 15, respectively.

Hand modeling. We evaluate the performance of our network when only physical connections of hand joints are used for the computations at the convolutional layer, i.e., connections between hand joints belonging to neighboring fingers are removed from the graph in Fig. 1 (b). Each joint is now connected to at most three joints including itself. Results shown in Tab. 1 confirm that the use of connections other than physical connections of hand joints bring performance improvement.

Time interval t_0. In this experiment, we vary t_0 and keep other components of our network unchanged. To ensure that the computation of covariance matrices is numerically stable, we set $t_0 > 0$. Tab. 2 shows the performance of our network with three different settings of t_0, i.e. $t_0 = 1, 2, 3$. Results suggest that using 3 consecutive frames for the input of the first GaussAgg layer of ST-GA-NET is sufficient to obtain good performance.

N_S	FPHA	DHG (14 gestures)	DHG (28 gestures)
15	93.33	94.29	89.40
20	92.87	94.05	88.93
25	92.70	94.29	89.04

Table 3: Recognition accuracy (%) of our network for different settings of N_S.

Network	FPHA	DHG (14 gestures)	DHG (28 gestures)
ST-HGR-NET	91.83	93.21	89.29
TS-HGR-NET	90.96	93.33	88.21
ST-TS-HGR-NET	**93.22**	**94.29**	**89.40**

Table 4: Recognition accuracy (%) of sub-networks ST-GA-NET and TS-GA-NET.

Number N_S of sub-sequences in a branch. This experiment is performed by varying N_S while keeping other components of our network unchanged. For the same reason related to the computation of covariance matrices, N_S must be in a certain interval. We tested with $N_S = 15, 20, 25$. Results given in Tab. 3 indicate that our network is not sensitive to different settings of N_S.

Contribution of ST-GA-NET and TS-GA-NET. We evaluate the performance of two networks, referred to as ST-HGR-NET and TS-HGR-NET by removing sub-networks TS-GA-NET and ST-GA-NET from our network, respectively. Results shown in Tab. 4 reveal that none of both ST-GA-NET and TS-GA-NET always provides the best performances on the datasets. This motivates the need for their combination using the component SPDC-NET and this contributes to the overall performance of our global network combining both TS-GA-NET and ST-GA-NET.

In the following, we report results obtained with default settings of t_0 and N_S, i.e. $t_0 = 1$ and $N_S = 15$.

4.3. Comparison with State-of-the-Art

DHG dataset. The comparison of our method and state-of-the-art methods on DHG dataset is given in Tab. 5. The accuracy of the method of [19] is obtained by using the implementation provided by the authors with their default parameter settings. Our method significantly outperforms the competing ones. The network of [19] also learns a SPD matrix-based representation from skeletal data which is similar in spirit to our network. However, they concatenate the 3D coordinates of joints at each frame to create the feature vector of that frame, and their network's input is the covariance matrix computed from feature vectors over the whole skeleton sequence. Thus, spatial and temporal relationships of joints are not effectively taken into account. By exploiting these relationships, our network improves the

Method	Year	Color	Depth	Pose	Accuracy (%)	
					14 gestures	28 gestures
Oreifej and Liu [40]	2013	✗	✓	✗	78.53	74.03
Devanne et al. [3]	2015	✗	✗	✓	79.61	62.00
Huang et al. [19]	2017	✗	✗	✓	75.24	69.64
Ohn-Bar and Trivedi [38]	2013	✗	✗	✓	83.85	76.53
Chen et al. [2]	2017	✗	✗	✓	84.68	80.32
De Smedt et al. [46]	2016	✗	✗	✓	88.24	81.90
Devineau et al. [4]	2018	✗	✗	✓	91.28	84.35
ST-TS-HGR-NET		✗	✗	✓	**94.29**	**89.40**

Table 5: Recognition accuracy comparison of our method and state-of-the-art methods on DHG dataset with 1960 sequences for training and 840 sequences for testing. The best result in each column is marked in bold.

Method	Year	Color	Depth	Pose	Accuracy (%)
De Smedt et al., [46]	2016	✗	✗	✓	83.1
CNN+LSTM [36]	2018	✗	✗	✓	85.6
Weng et al., [55]	2018	✗	✗	✓	85.8
ST-TS-HGR-NET		✗	✗	✓	**87.3**

Table 6: Recognition accuracy comparison of our method and state-of-the-art methods on DHG dataset using the leave-one-subject-out experimental protocol with 14 gestures. The best result in each column is marked in bold.

Method	Year	Color	Depth	Pose	Accuracy (%)
De Smedt et al., [46]	2016	✗	✗	✓	80.0
CNN+LSTM [36]	2018	✗	✗	✓	81.1
Weng et al., [55]	2018	✗	✗	✓	80.4
ST-TS-HGR-NET		✗	✗	✓	**83.4**

Table 7: Recognition accuracy comparison of our method and state-of-the-art methods on DHG dataset using the leave-one-subject-out experimental protocol with 28 gestures. The best result in each column is marked in bold.

recognition accuracy by 19.05% and 19.76% compared to the results of [19] for experiments with 14 and 28 gestures, respectively. For more comparison of our method and existing methods, we conducted experiments using the leave-one-subject-out experimental protocol. Results on Tabs. 6 (14 gestures) and 7 (28 gestures) demonstrate that our method achieves the best results compared to existing methods on this protocol. In particular, our method outperforms the most recent work [55] by 1.5 and 3 percent points for experiments with 14 and 28 gestures, respectively.

FPHA dataset. Tab. 8 shows the accuracies of our method and state-of-the-art methods on FPHA dataset. The accuracies of the methods of [19] and [23] are obtained by using the implementations provided by the authors with their default parameter settings. Despite the simplicity of our network compared to the competing deep neural networks, it is superior to them on this dataset. The best performing method among state-of-the-art methods is Gram Matrix, which gives 85.39% accuracy, 7.83 percent points inferior to our method. The remaining methods are outperformed by our method by more than 10 percent points. We observe that the method of [19] performs well on this dataset. However, since this method does not fully exploit spatial and temporal relationships of skeleton joints, it gives a significantly lower accuracy than our method. Results again confirm the effectiveness of the proposed network architecture for hand gesture recognition.

Method	Year	Color	Depth	Pose	Accuracy (%)
Two stream-color [10]	2016	✓	✗	✗	61.56
Two stream-flow [10]	2016	✓	✗	✗	69.91
Two stream-all [10]	2016	✓	✗	✗	75.30
HOG^2-depth [39]	2013	✗	✓	✗	59.83
HOG^2-depth+pose [39]	2013	✗	✓	✓	66.78
HON4D [40]	2013	✗	✓	✗	70.61
Novel View [41]	2016	✗	✓	✗	69.21
1-layer LSTM [62]	2016	✗	✗	✓	78.73
2-layer LSTM [62]	2016	✗	✗	✓	80.14
Moving Pose [59]	2013	✗	✗	✓	56.34
Lie Group [49]	2014	✗	✗	✓	82.69
HBRNN [6]	2015	✗	✗	✓	77.40
Gram Matrix [61]	2016	✗	✗	✓	85.39
TF [11]	2017	✗	✗	✓	80.69
JOULE-color [18]	2015	✓	✗	✗	66.78
JOULE-depth [18]	2015	✗	✓	✗	60.17
JOULE-pose [18]	2015	✗	✗	✓	74.60
JOULE-all [18]	2015	✓	✓	✓	78.78
Huang et al. [19]	2017	✗	✗	✓	84.35
Huang et al. [23]	2018	✗	✗	✓	77.57
ST-TS-HGR-NET		✗	✗	✓	**93.22**

Table 8: Recognition accuracy comparison of our method and state-of-the-art methods on FPHA dataset. The best result in each column is marked in bold.

5. Conclusion

We have presented a new neural network for hand gesture recognition that learns a discriminative SPD matrix encoding the first-order and second-order statistics. We have provided the experimental evaluation on two benchmark datasets showing that our method outperforms state-of-the-art methods.

Acknowledgments. This material is based upon work supported by the European Union and the Region Normandie under the project IGIL. We thank Guillermo Garcia-Hernando for providing access to FPHA dataset [12].

References

[1] P. Bilinski and F. Bremond. Video Covariance Matrix Logarithm for Human Action Recognition in Videos. In *IJCAI*, pages 2140–2147, 2015. 2

[2] X. Chen, H. Guo, G. Wang, and L. Zhang. Motion Feature Augmented Recurrent Neural Network for Skeleton-based Dynamic Hand Gesture Recognition. *CoRR*, abs/1708.03278, 2017. 8

[3] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, and A. D. Bimbo. 3-D Human Action Recognition by Shape Analysis of Motion Trajectories on Riemannian Manifold. *IEEE Transactions on Cybernetics*, 45(7):1340–1352, 2015. 8

[4] G. Devineau, F. Moutarde, W. Xi, and J. Yang. Deep Learning for Hand Gesture Recognition on Skeletal Data. In *IEEE International Conference on Automatic Face Gesture Recognition*, pages 106–113, May 2018. 2, 8

[5] Z. Dong, S. Jia, C. Zhang, M. Pei, and Y. Wu. Deep Manifold Learning of Symmetric Positive Definite Matrices with Application to Face Recognition. In *AAAI*, pages 4009–4015, 2017. 2

[6] Y. Du, W. Wang, and L. Wang. Hierarchical Recurrent Neural Network for Skeleton Based Action Recognition. In *CVPR*, pages 1110–1118, 2015. 2, 8

[7] M. Engin, L. Wang, L. Zhou, and X. Liu. DeepKSPD: Learning Kernel-matrix-based SPD Representation for Fine-grained Image Recognition. *CoRR*, abs/1711.04047, 2017. 2

[8] G. Evangelidis, G. Singh, and R. Horaud. Skeletal Quads: Human Action Recognition Using Joint Quadruples. In *ICPR*, pages 4513–4518, 2014. 2

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. 6

[10] C. Feichtenhofer, A. P., and A. Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. *CVPR*, pages 1933–1941, 2016. 8

[11] G. Garcia-Hernando and T.-K. Kim. Transition Forests: Learning Discriminative Temporal Transitions for Action Recognition. In *CVPR*, pages 407–415, 2017. 8

[12] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim. First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations. In *CVPR*, 2018. 6, 7, 8

[13] R. Girshick. Fast R-CNN. In *ICCV*, pages 1440–1448, 2015. 2

[14] K. Guo, P. Ishwar, and J. Konrad. Action Recognition From Video Using Feature Covariance Matrices. *IEEE Transactions on Image Processing*, 22(6):2479–2494, 2013. 2

[15] M. Harandi, M. Salzmann, and R. Hartley. Dimensionality Reduction on SPD Manifolds: The Emergence of Geometry-Aware Methods. *TPAMI*, 40:48–62, 2018. 1

[16] M. T. Harandi, C. Sanderson, A. Sanin, and B. C. Lovell. Spatio-temporal Covariance Descriptors for Action and Gesture Recognition. In *WACV*, pages 103–110, 2013. 2

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, June 2016. 2

[18] J. Hu, W. Zheng, J. Lai, and J. Zhang. Jointly Learning Heterogeneous Features for RGB-D Activity Recognition. In *CVPR*, pages 5344–5352, 2015. 8

[19] Z. Huang and L. V. Gool. A Riemannian Network for SPD Matrix Learning. In *AAAI*, pages 2036–2042, 2017. 1, 2, 4, 6, 7, 8, 11

[20] Z. Huang, C. Wan, T. Probst, and L. V. Gool. Deep Learning on Lie Groups for Skeleton-Based Action Recognition. In *CVPR*, pages 6099–6108, 2017. 2

[21] Z. Huang, R. Wang, X. Li, W. Liu, S. Shan, L. V. Gool, and X. Chen. Geometry-Aware Similarity Learning on SPD Manifolds for Visual Recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2513–2523, 2018. 1

[22] Z. Huang, R. Wang, S. Shan, X. Li, and X. Chen. Log-euclidean Metric Learning on Symmetric Positive Definite Manifold with Application to Image Set Classification. In *ICML*, pages 720–729, 2015. 1

[23] Z. Huang, J. Wu, and L. V. Gool. Building Deep Networks on Grassmann Manifolds. In *AAAI*, pages 3279–3286, 2018. 2, 8

[24] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix Backpropagation for Deep Networks with Structured Layers. In *ICCV*, pages 2965–2973, 2015. 2, 10, 11

[25] Q. Ke, M. Bennamoun, S. An, F. A. Sohel, and F. Boussaïd. A New Representation of Skeleton Sequences for 3D Action Recognition. In *CVPR*, pages 4570–4579, 2017. 2

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, pages 1097–1105, 2012. 2

[27] C. Li, Z. Cui, W. Zheng, C. Xu, and J. Yang. Spatio-Temporal Graph Convolution for Skeleton Based Action Recognition. In *AAAI*, pages 3482–3489, 2018. 1, 2

[28] P. Li, Q. Wang, H. Zeng, and L. Zhang. Local Log-Euclidean Multivariate Gaussian Descriptor and Its Application to Image Classification. *TPAMI*, 39(4):803–817, 2017. 3

[29] P. Li, J. Xie, Q. Wang, and W. Zuo. Is Second-order Information Helpful for Large-scale Visual Recognition? In *ICCV*, pages 2070–2078, 2017. 2

[30] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. In *ECCV*, pages 816–833, 2016. 2

[31] J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot. Global Context-Aware Attention LSTM Networks for 3D Action Recognition. In *CVPR*, pages 3671–3680, 2017. 2

[32] M. Liu, H. Liu, and C. Chen. Enhanced Skeleton Visualization for View Invariant Human Action Recognition. *Pattern Recognition*, 68:346–362, 2017. 2

[33] M. Liu and J. Yuan. Recognizing Human Actions as The Evolution of Pose Estimation Maps. In *CVPR*, 2018. 2

[34] M. Lovrić, M. Min-Oo, and E. A. Ruh. Multivariate Normal Distributions Parametrized As a Riemannian Symmetric Space. *Journal of Multivariate Analysis*, 74(1):36–48, 2000. 3, 4

[35] J. Luo, W. Wang, and H. Qi. Group Sparsity and Geometry Constrained Dictionary Learning for Action Recognition from Depth Maps. In *ICCV*, pages 1809–1816, Dec 2013. 2

[36] J. C. Nez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Vlez. Convolutional Neural Networks and Long Short-Term Memory for Skeleton-based Human Activity and

Hand Gesture Recognition. *Pattern Recognition*, 76(C):80–94, 2018. 2, 8

[37] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy. Sequence of The Most Informative Joints (SMIJ): A New Representation for Human Skeletal Action Recognition. *Journal of Visual Communication and Image Representation*, 25(1):24–38, 2014. 2

[38] E. Ohn-Bar and M. M. Trivedi. Joint Angles Similarities and HOG2 for Action Recognition. In *CVPRW*, pages 465–470, 2013. 8

[39] E. Ohn-Bar and M. M. Trivedi. Hand Gesture Recognition in Real Time for Automotive Interfaces: A Multimodal Vision-Based Approach and Evaluations. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2368–2377, 2014. 8

[40] O. Oreifej and Z. Liu. HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences. In *CVPR*, pages 716–723, June 2013. 8

[41] H. Rahmani and A. Mian. 3D Action Recognition from Novel Viewpoints. In *CVPR*, pages 1506–1515, June 2016. 8

[42] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *IJCV*, 105(3):222–245, 2013. 6

[43] G. Serra, C. Grana, M. Manfredi, and R. Cucchiara. GOLD: Gaussians of Local Descriptors for Image Representation. *CVIU*, 134:22–32, 2015. 3

[44] A. Shahroudy, J. Liu, T. T. Ng, and G. Wang. NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. In *CVPR*, pages 1010–1019, 2016. 2

[45] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Non-Local Graph Convolutional Networks for Skeleton-Based Action Recognition. *CoRR*, abs/1805.07694, 2018. 1

[46] Q. D. Smedt, H. Wannous, and J. Vandeborre. Skeleton-Based Dynamic Hand Gesture Recognition. In *CVPRW*, pages 1206–1214, June 2016. 2, 6, 8

[47] Q. D. Smedt, H. Wannous, J.-P. Vandeborre, J. Guerry, B. L. Saux, and D. Filliat. 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. In *Eurographics Workshop on 3D Object Retrieval*, pages 33–38, 2017. 6, 7

[48] O. Tuzel, F. Porikli, and P. Meer. Pedestrian Detection via Classification on Riemannian Manifolds. *TPAMI*, 30(10):1713–1727, 2008. 4

[49] R. Vemulapalli, F. Arrate, and R. Chellappa. Human Action Recognition by Representing 3D Skeletons as Points in a Lie Group. In *CVPR*, pages 588–595, 2014. 1, 2, 8

[50] C. Wang, Y. Wang, and A. L. Yuille. An Approach to Pose-Based Action Recognition. In *CVPR*, pages 915–922, 2013. 2

[51] H. Wang and L. Wang. Modeling Temporal Dynamics and Spatial Configurations of Actions Using Two-Stream Recurrent Neural Networks. *CVPR*, pages 3633–3642, 2017. 2

[52] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining Actionlet Ensemble for Action Recognition with Depth Cameras. In *CVPR*, pages 1290–1297, 2012. 1

[53] P. Wang, Z. Li, Y. Hou, and W. Li. Action Recognition Based on Joint Trajectory Maps Using Convolutional Neural Networks. In *ACM MM*, pages 102–106, 2016. 2

[54] Q. Wang, P. Li, and L. Zhang. G2DeNet: Global Gaussian Distribution Embedding Network and Its Application to Visual Recognition. In *CVPR*, pages 2730–2739, 2017. 2, 6, 12

[55] J. Weng, M. Liu, X. Jiang, and J. Yuan. Deformable Pose Traversal Convolution for 3D Action and Gesture Recognition. In *ECCV*, 2018. 2, 8

[56] S. Yan, Y. Xiong, and D. Lin. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*, pages 7444–7452, 2018. 1, 2

[57] X. Yang and Y. L. Tian. EigenJoints-based Action Recognition Using Naive-Bayes-Nearest-Neighbor. In *CVPRW*, pages 14–19, 2012. 2

[58] C. Yuan, W. Hu, X. Li, S. Maybank, and G. Luo. Human Action Recognition Under Log-euclidean Riemannian Metric. In *ACCV*, pages 343–353, 2010. 2

[59] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The Moving Pose: An Efficient 3D Kinematics Descriptor for Low-Latency Action Recognition and Detection. In *ICCV*, pages 2752–2759, 2013. 8

[60] T. Zhang, W. Zheng, Z. Cui, and C. Li. Deep Manifold-to-Manifold Transforming Network. *CoRR*, abs/1705.10732, 2017. 1

[61] X. Zhang, Y. Wang, M. Gou, M. Sznaier, and O. Camps. Efficient Temporal Sequence Comparison and Classification Using Gram Matrix Embeddings on a Riemannian Manifold. In *CVPR*, pages 4498–4507, 2016. 8

[62] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie. Co-occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks. In *AAAI*, pages 3697–3703, 2016. 8

Appendices

A. Backpropagation Procedures

This part provides details on the backpropagation procedures during the training process of our network. Our network can be encoded as a pair $(h, \mathbf{W})$ where $h = h^{(N_L)} \circ \ldots \circ h^{(1)}$ is a composition of N_L layers, $\mathbf{W} = (\mathbf{W}(N_L), \ldots, \mathbf{W}(1))$ represents the network parameters, $\mathbf{W}(k)$ are the parameters of layer k. Let $L^{(k)} = L \circ h^{(N_L)} \circ \ldots \circ h^{(k)}$ be the loss as a function of layer $k-1$. In the following, we omit the superscript k of $L^{(k)}$ for the sake of convenience.

A.1. SPDAgg layer

We present in this section a method based on the chain rule of [24] for the computations of partial derivatives. For more details on the established theory, we refer readers

to [24]. The variation of Y is given by:

$$d\mathbf{Y} = \sum_{i=1}^{N} \Big(d\mathbf{W}_i \mathbf{X}_i (\mathbf{W}_i)^T + \mathbf{W}_i d\mathbf{X}_i (\mathbf{W}_i)^T + \mathbf{W}_i \mathbf{X}_i d(\mathbf{W}_i)^T \Big). \tag{22}$$

The chain rule in this case is:

$$\frac{\partial L}{\partial \mathbf{Y}} : d\mathbf{Y} = \sum_{i=1}^{N} \Big(\frac{\partial L}{\partial \mathbf{W}_i} : d\mathbf{W}_i + \frac{\partial L}{\partial \mathbf{X}_i} : d\mathbf{X}_i \Big). \tag{23}$$

By replacing $d\mathbf{Y}$ in Eq. (23) with its expression in Eq. (22), the left-hand side of Eq. (23) becomes:

$$\sum_{i=1}^{N} \Big(\frac{\partial L}{\partial \mathbf{Y}} : d\mathbf{W}_i \mathbf{X}_i (\mathbf{W}_i)^T + \frac{\partial L}{\partial \mathbf{Y}} : \mathbf{W}_i \mathbf{X}_i d(\mathbf{W}_i)^T + \frac{\partial L}{\partial \mathbf{Y}} : \mathbf{W}_i d\mathbf{X}_i (\mathbf{W}_i)^T \Big). \tag{24}$$

Using the properties [24] of the matrix inner product ":" and by the fact that $\mathbf{Y}$ and $\mathbf{X}_i$ are symmetric, we have:

$$\frac{\partial L}{\partial \mathbf{Y}} : d\mathbf{W}_i \mathbf{X}_i (\mathbf{W}_i)^T = \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W}_i \mathbf{X}_i : d\mathbf{W}_i, \tag{25}$$

$$\frac{\partial L}{\partial \mathbf{Y}} : \mathbf{W}_i \mathbf{X}_i d(\mathbf{W}_i)^T = \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W}_i \mathbf{X}_i : d\mathbf{W}_i, \tag{26}$$

$$\frac{\partial L}{\partial \mathbf{Y}} : \mathbf{W}_i d\mathbf{X}_i (\mathbf{W}_i)^T = (\mathbf{W}_i)^T \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W}_i : d\mathbf{X}_i. \tag{27}$$

The expression (24) now becomes:

$$\sum_{i=1}^{N} \Big(2 \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W}_i \mathbf{X}_i : d\mathbf{W}_i + (\mathbf{W}_i)^T \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W}_i : d\mathbf{X}_i \Big). \tag{28}$$

Since the last expression is equal to the right-hand side of Eq. (23), we obtain the partial derivatives:

$$\frac{\partial L}{\partial \mathbf{W}_i} = 2 \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W}_i \mathbf{X}_i, \tag{29}$$

$$\frac{\partial L}{\partial \mathbf{X}_i} = (\mathbf{W}_i)^T \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W}_i. \tag{30}$$

To learn the weights of this layer, we use the method proposed in [19]. The weight $\hat{\mathbf{W}}$ is updated in two steps. First, the tangential component to the Stiefel manifold is obtained by subtracting the normal component of the Euclidean gradient:

$$\tilde{\nabla} L_{\hat{\mathbf{W}}^t} = \nabla L_{\hat{\mathbf{W}}^t} - \nabla L_{\hat{\mathbf{W}}^t} (\hat{\mathbf{W}}^t)^T \hat{\mathbf{W}}^t, \tag{31}$$

where $\hat{\mathbf{W}}^t$ is the updated weight at the t^{th} iteration and $\nabla L_{\hat{\mathbf{W}}^t} (\hat{\mathbf{W}}^t)^T \hat{\mathbf{W}}^t$ is the normal component of the Euclidean gradient $\nabla L_{\hat{\mathbf{W}}^t}$. Following Eq. (29), the Euclidean gradient $\nabla L_{\hat{\mathbf{W}}^t}$ is given by:

$$\nabla L_{\hat{\mathbf{W}}^t} = 2 \frac{\partial L}{\partial \mathbf{Y}} [\text{Proj}_1(\hat{\mathbf{W}}^t) \mathbf{X}_1, \ldots, \text{Proj}_N(\hat{\mathbf{W}}^t) \mathbf{X}_N], \tag{32}$$

where $\text{Proj}_i(\hat{\mathbf{W}}^t), i = 1, \ldots, N$, is the projection of $\hat{\mathbf{W}}^t$ on its columns corresponding to $\mathbf{W}_i$.

Then a retraction operation is used to map back the updated weight in the tangent space of the Stiefel manifold to that in the Stiefel manifold as:

$$\hat{\mathbf{W}}^{t+1} = \Gamma(\hat{\mathbf{W}}^t - \lambda \tilde{\nabla} L_{\hat{\mathbf{W}}^t}), \tag{33}$$

where Γ is the retraction operation, λ is the learning rate.

The updated weights of $\mathbf{W}_i, i = 1, \ldots, N$, at the $(t+1)^{th}$ iteration can be computed as:

$$(\mathbf{W}_i)^{t+1} = \text{Proj}_i(\hat{\mathbf{W}}^{t+1}). \tag{34}$$

A.2. LogEig and ReEig layers

To make this document self-contained for readers, we present here the computations of partial derivatives for the LogEig and ReEig layers. For more details, we refer readers to [19, 24]. For the LogEig layers, the first step receives matrix $\mathbf{X}_{s,f}^t$ as input and produces matrices $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{X}_{s,f}^t = \mathbf{U}\mathbf{V}\mathbf{U}^T$. The partial derivatives $\frac{\partial L}{\partial \mathbf{X}_{s,f}^t}$ can be computed from those of the outputs $\frac{\partial L}{\partial \mathbf{U}}$ and $\frac{\partial L}{\partial \mathbf{V}}$ as [24]:

$$\frac{\partial L}{\partial \mathbf{X}_{s,f}^t} = \mathbf{U} \Big\{ 2 \Big(\tilde{K}^T \circ \Big(\mathbf{U}^T \frac{\partial L}{\partial \mathbf{U}} \Big)_{sym} \Big) + \Big(\frac{\partial L}{\partial \mathbf{V}} \Big)_{diag} \Big\} \mathbf{U}^T, \tag{35}$$

where $\mathbf{D}_{sym} = \frac{1}{2}(\mathbf{D} + \mathbf{D}^T)$, $\mathbf{D}_{diag}$ is $\mathbf{D}$ with all off-diagonal elements being 0, and $\tilde{K}^T$ is defined as:

$$\tilde{K}_{ij} = \begin{cases} \frac{1}{\sigma_i - \sigma_j}, & i \neq j \\ 0, & i = j \end{cases} \tag{36}$$

The second step receives matrices $\mathbf{U}$ and $\mathbf{V}$ as input and produces matrix $\mathbf{Y}_{s,f}^t = \mathbf{U} \log(\mathbf{V}) \mathbf{U}^T$. The partial derivatives $\frac{\partial L}{\partial \mathbf{U}}$ and $\frac{\partial L}{\partial \mathbf{V}}$ can be computed from those of the output $\frac{\partial L}{\partial \mathbf{Y}_{s,f}^t}$ as [19]:

$$\frac{\partial L}{\partial \mathbf{U}} = 2 \Big(\frac{\partial L}{\partial \mathbf{Y}_{s,f}^t} \Big)_{sym} \mathbf{U} \log(\mathbf{V}), \tag{37}$$

$$\frac{\partial L}{\partial \mathbf{V}} = \mathbf{V}^{-1} \mathbf{U}^T \Big(\frac{\partial L}{\partial \mathbf{Y}_{s,f}^t} \Big)_{sym} \mathbf{U}. \tag{38}$$

The ReEig layers can be decomposed into two steps as the LogEig layers where the partial derivatives of the first

step are computed similarly to the LogEig layers. For the second step, the partial derivatives $\frac{\partial L}{\partial \mathbf{U}}$ and $\frac{\partial L}{\partial \mathbf{V}}$ can be computed from those of the output $\frac{\partial L}{\partial \mathbf{Y}^t_{s,f}}$ as:

$$\frac{\partial L}{\partial \mathbf{U}} = 2\Big(\frac{\partial L}{\partial \mathbf{Y}^t_{s,f}}\Big)_{sym} \mathbf{U} \max(\epsilon \mathbf{I}, \mathbf{V}), \tag{39}$$

$$\frac{\partial L}{\partial \mathbf{V}} = \mathbf{Q}\mathbf{U}^T\Big(\frac{\partial L}{\partial \mathbf{Y}^t_{s,f}}\Big)_{sym} \mathbf{U}, \tag{40}$$

where $\max(\epsilon \mathbf{I}, \mathbf{V})$ is defined in Eq. (9), and $\mathbf{Q}$ is the gradient of $\max(\epsilon \mathbf{I}, \mathbf{V})$ with diagonal elements being defined as:

$$\mathbf{Q}(i,i) = \begin{cases} 1 & \text{if } \mathbf{V}(i,i) > \epsilon \\ 0 & \text{if } \mathbf{V}(i,i) \leq \epsilon. \end{cases} \tag{41}$$

A.3. VecMat layer

For the VecMat layer, the expression for the partial derivatives $\frac{\partial L}{\partial \mathbf{X}^t_{s,f}}$ is straightforward and can be written as:

$$\frac{\partial L}{\partial \mathbf{X}^t_{s,f}} = \begin{bmatrix} \frac{\partial L}{\partial \mathbf{y}^t_{s,f}(1)} & \frac{\sqrt{2}\partial L}{\partial \mathbf{y}^t_{s,f}(2)} & \cdots & \frac{\sqrt{2}\partial L}{\partial \mathbf{y}^t_{s,f}(d^c_{out}+1)} \\ \frac{\sqrt{2}\partial L}{\partial \mathbf{y}^t_{s,f}(2)} & \frac{\partial L}{\partial \mathbf{y}^t_{s,f}(d^c_{out}+2)} & \cdots & \frac{\sqrt{2}\partial L}{\partial \mathbf{y}^t_{s,f}(2d^c_{out}+1)} \\ \frac{\sqrt{2}\partial L}{\partial \mathbf{y}^t_{s,f}(d^c_{out}+1)} & \frac{\sqrt{2}\partial L}{\partial \mathbf{y}^t_{s,f}(2d^c_{out}+1)} & \cdots & \frac{\partial L}{\partial \mathbf{y}^t_{s,f}(\frac{(d^c_{out}+1)(d^c_{out}+2)}{2})} \end{bmatrix}, \tag{42}$$

where $\mathbf{y}^t_{s,f} = [\mathbf{y}^t_{s,f}(1), \ldots, \mathbf{y}^t_{s,f}(\frac{(d^c_{out}+1)(d^c_{out}+2)}{2})]^T$.

A.4. GaussAgg layer

The general form for the mapping h_{ga} of the GaussAgg layers can be written as:

$$h_{ga}(\mathbf{X}) = \mathbf{Y} = \begin{bmatrix} \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix}, \tag{43}$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times d}$ is the input of the GaussAgg layer, $\mathbf{Y}$ is the output of the GaussAgg layer, $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i$ and $\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^N (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T$.

By the identity $\boldsymbol{\Sigma} = \frac{1}{N}\mathbf{X}^T\mathbf{X} - \boldsymbol{\mu}\boldsymbol{\mu}^T$, $\mathbf{Y}$ can be expressed as a function of $\mathbf{X}$ as [54]:

$$\mathbf{Y} = \frac{1}{N}\mathbf{B}\mathbf{X}^T\mathbf{X}\mathbf{B}^T + \frac{2}{N}\Big(\mathbf{B}\mathbf{X}^T\mathbf{1}\mathbf{b}^T\Big)_{sym} + \mathbf{C}, \tag{44}$$

where $\mathbf{B} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0}^T \end{bmatrix}$, $\mathbf{I}$ is the $d \times d$ identity matrix and $\mathbf{0}$ is the d-dimensional zero vector, $\mathbf{b} = [0, \ldots, 0, 1]^T$ is the $(d+1)$-dimensional vector with all elements being zero except the last one which is equal to one, $\mathbf{1}$ is the N-dimensional vector with all elements being one, $\mathbf{C} = \begin{bmatrix} \mathbf{O} & \mathbf{0} \\ \mathbf{0}^T & \mathbf{1} \end{bmatrix}$, $\mathbf{O}$ is the $d \times d$ zero matrix.

Based on Eq. (44), the expression for the partial derivatives $\frac{\partial L}{\partial \mathbf{X}}$ can be obtained as:

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{2}{N}\Big(\mathbf{X}\mathbf{B}^T + \mathbf{1}\mathbf{b}^T\Big)\Big(\frac{\partial L}{\partial \mathbf{Y}}\Big)_{sym}\mathbf{B}. \tag{45}$$