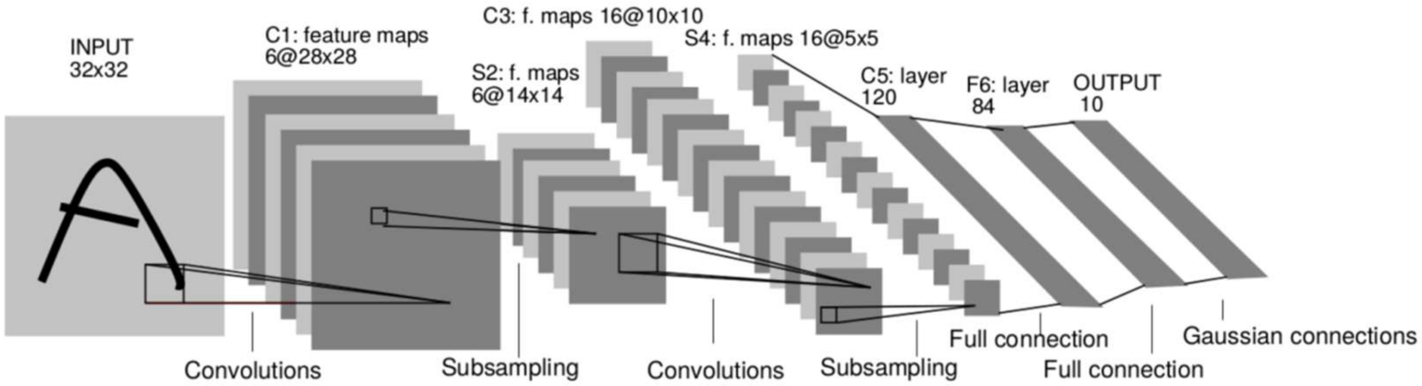
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Deep Manifold Learning for Human-focussed Video Classification

Zhiwu Huang

Computer Vision Lab @ ETH Zurich

Convolutional Networks for scalar inputs



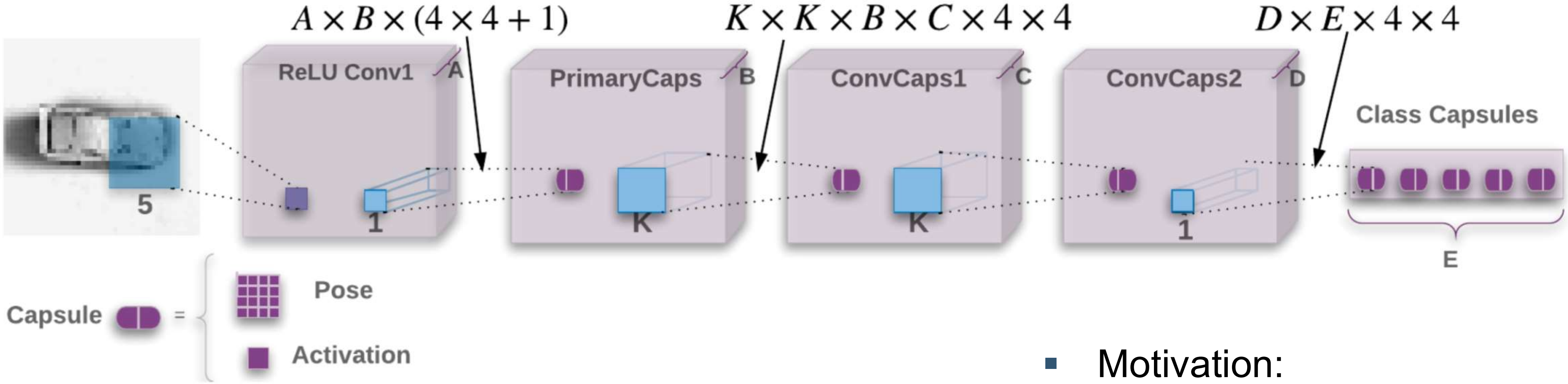
- **Translation invariance**
  - Convolutions
- **Multiscale structure**
  - Downsampling
- **Non-linearity**
  - Sigmoid, ReLU

LeCun et al., 1989

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Capsule Networks for matrix inputs

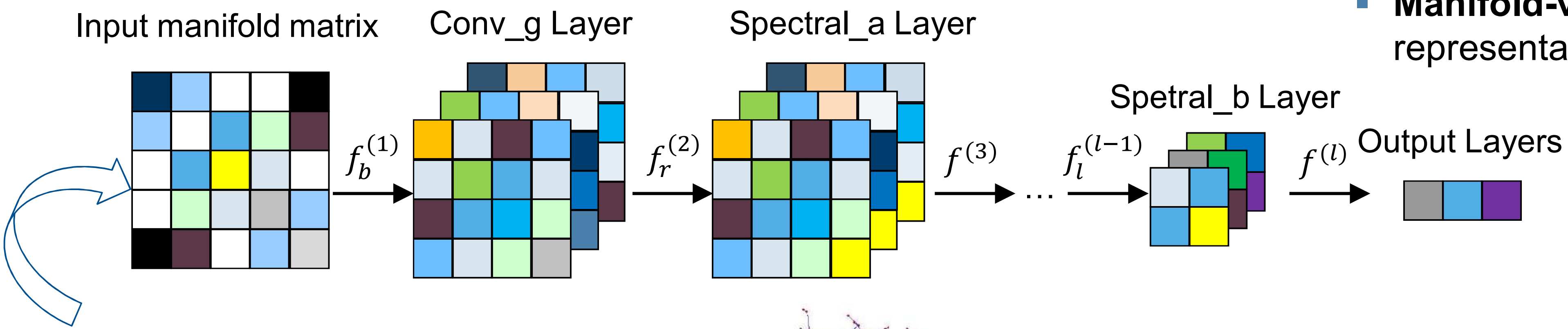


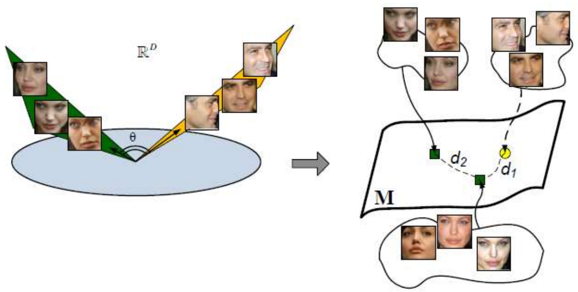
- Motivation:
  - **Pose-equivalent** representation

Hinton et al., NIPS'17, ICLR'18

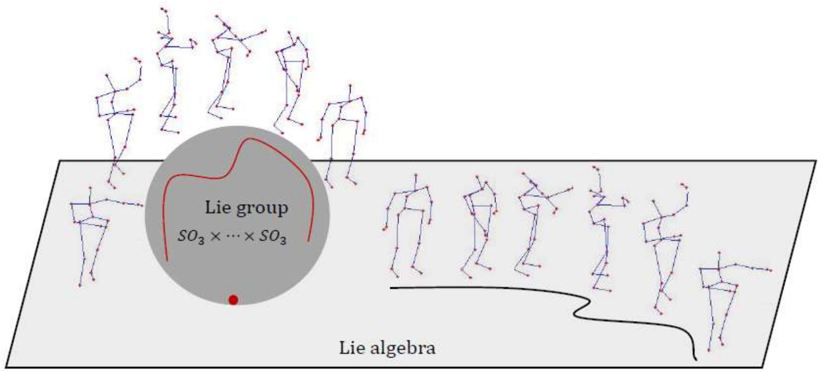
Manifold Networks for manifold-valued matrix inputs

- Motivation:
  - **Manifold-valued** representation





Face recognition on Grassmannian/SPD manifold-valued representation of facial videos



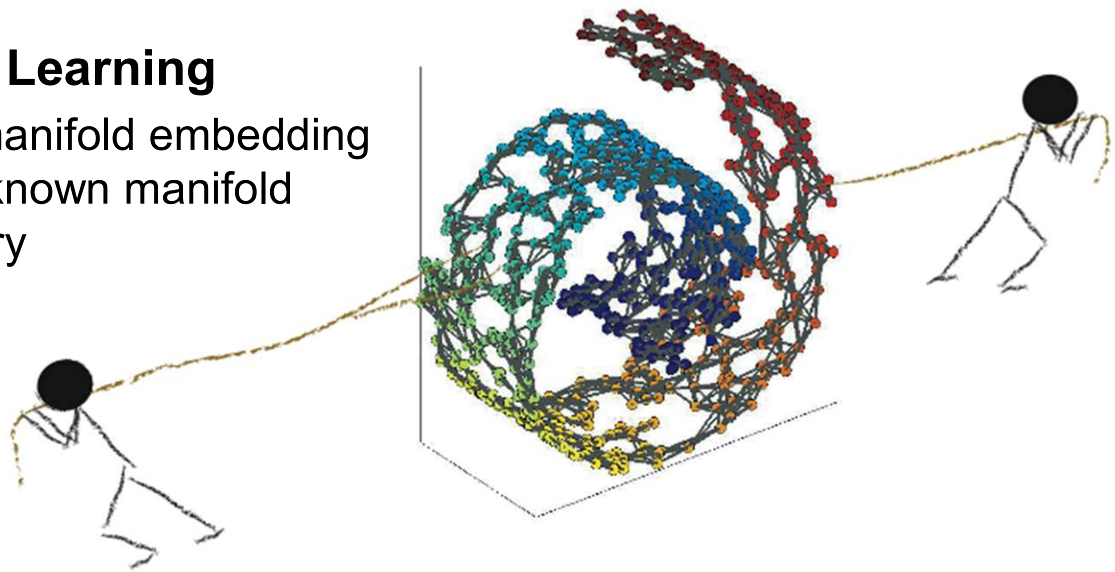
Activity recognition on group of special orthogonal matrices for 3D joint representation

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Manifold-valued Representation Learning vs. Manifold Learning

- **Manifold Learning**
  - Learn manifold embedding with unknown manifold geometry



Courtesy : Kilian Q. Weinberger

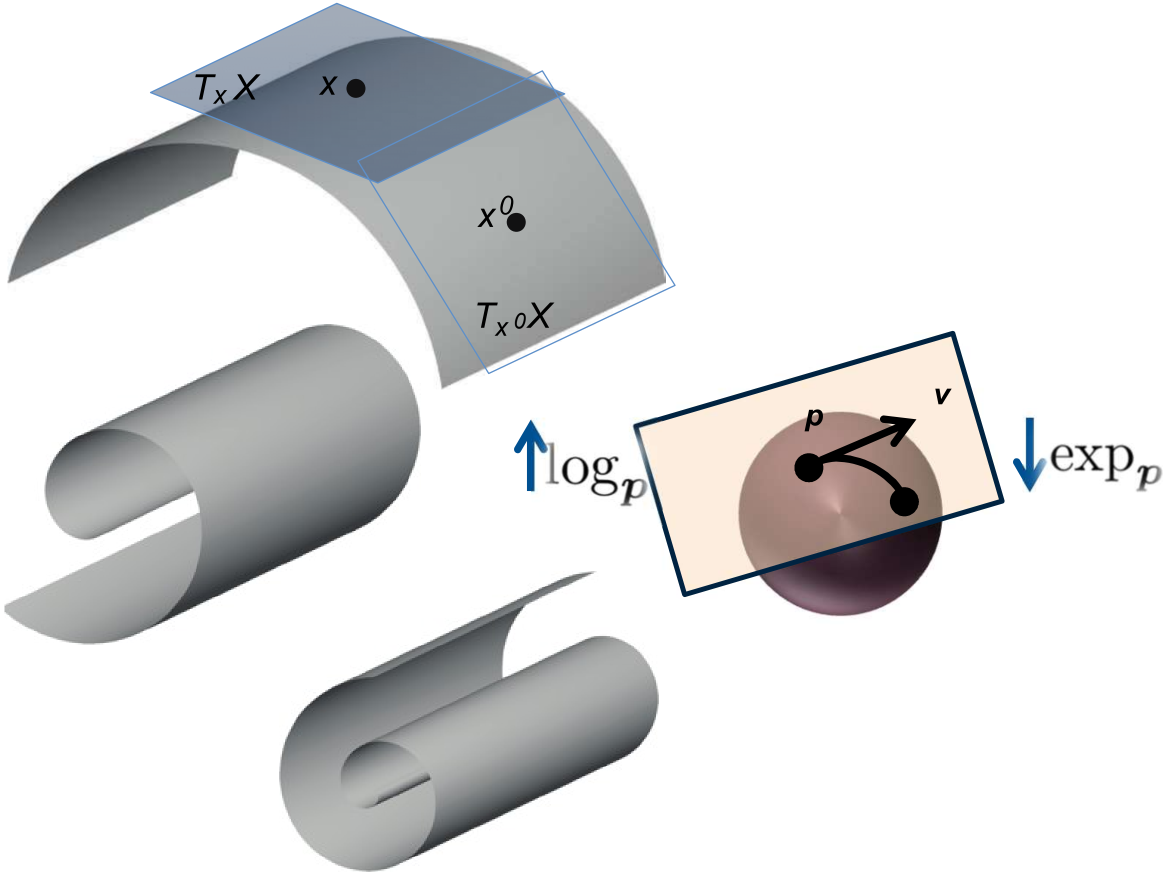
- **Manifold-valued Representation Learning**
  - Learn manifold-valued representation with known manifold geometry

Riemannian Geometry of Matrix Manifolds

- Riemannian manifold $\mathcal{X}$
  - Topological & differential space
  - Local Euclidean structure

- Tangent space $T_x\mathcal{X}$= local Euclidean structure of manifold $\mathcal{X}$ around x
  - Logarithm & Exponential maps

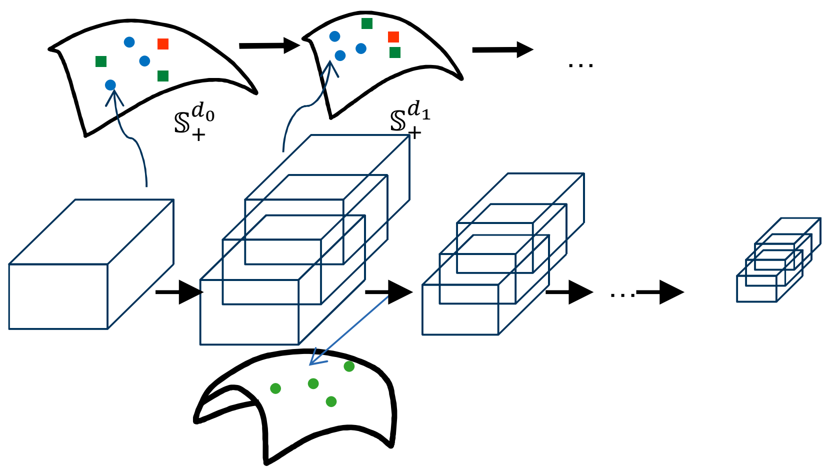
- Riemannian metric:
  - $\langle\cdot,\cdot\rangle_{T_x\mathcal{X}}: T_x\mathcal{X} \times T_x\mathcal{X} \longrightarrow \mathbb{R}$ depending smoothly on x



Courtesy: Michal Bronstein

Challenges of Deep Learning on Riemannian manifolds

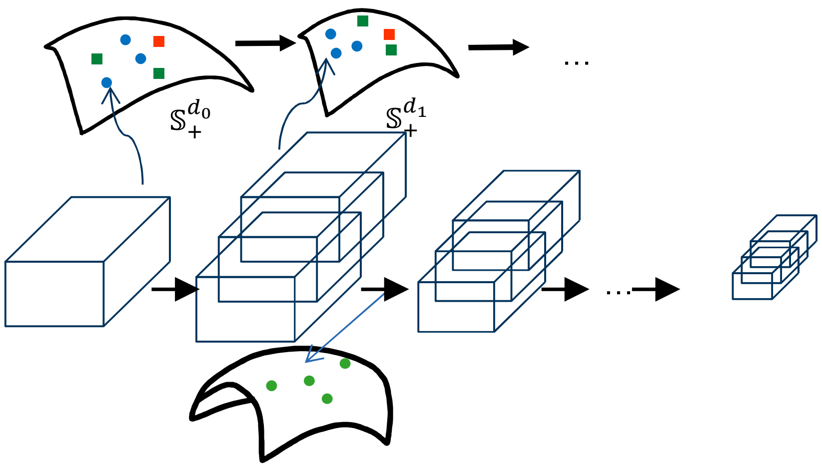
- Extend neural network techniques to manifold-structured matrix inputs
  - How to define **translation invariance**?
    - Manifold-to-manifold transform (convolution)
      - Geometry-aware transfrom
  - How to achieve **multiscale structure**?
    - Manifold sampling (pooling)
  - How to allow for **non-linearity**?
    - Spectral activation



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

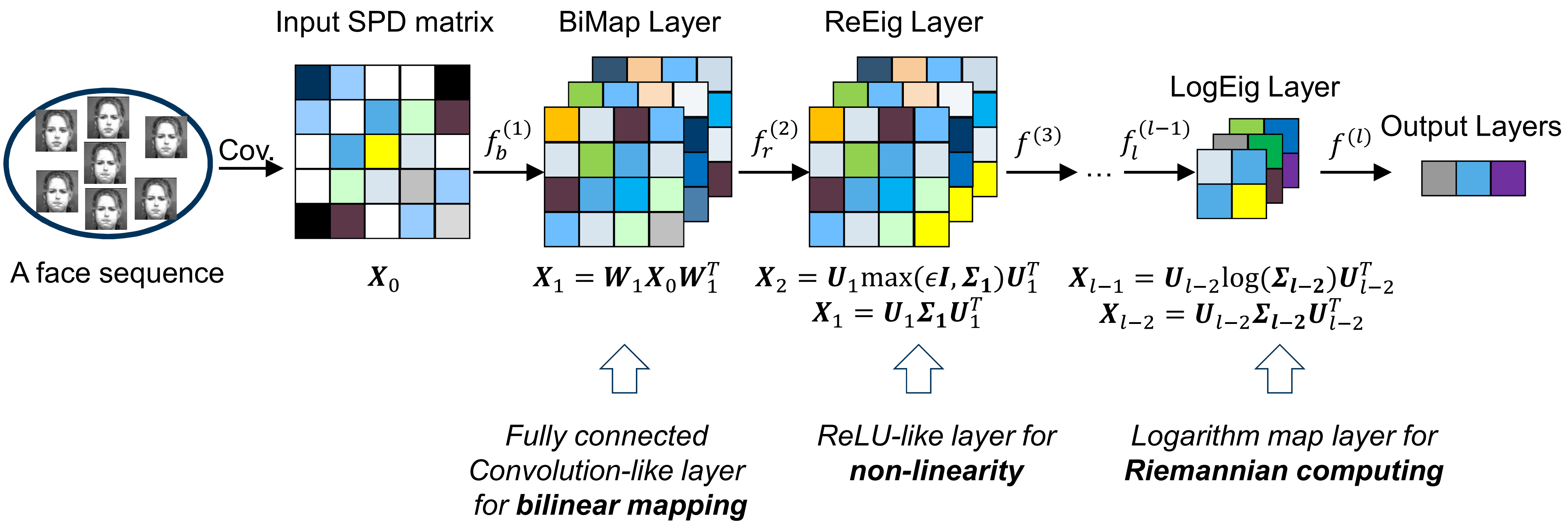
Proposed Manifold Networks



- **A Riemannian Network for SPD Matrix Learning**
  - Zhiwu Huang and Luc Van Gool, AAAI 2017
- **Building Deep Networks on Grassmann Manifolds**
  - Zhiwu Huang, Jiqing Wu, Luc Van Gool, AAAI 2018
- **Deep Learning on Lie Group for Skeleton-based Action Recognition**
  - Zhiwu Huang, Chengde Wan, Thomas Probst, Luc Van Gool, CVPR 2017

A Riemannian Network for SPD Matrix Learning

- Building a deep network on SPD manifolds
  - Accept SPD matrices as inputs, and keep the SPD structure across layers



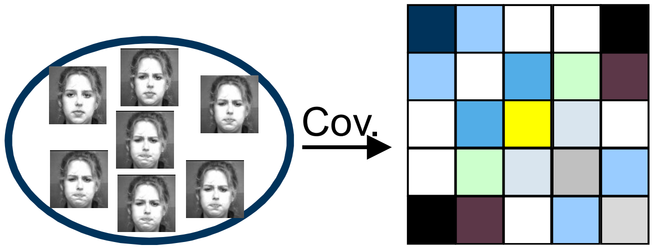
ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

A Riemannian Network for SPD Matrix Learning

- SPDNet Input:
  - SPD matrices, usually covariance matrices derived on the data

Input SPD matrix



A face sequence

X_0

◆N samples with d_0-dimension intensity feature

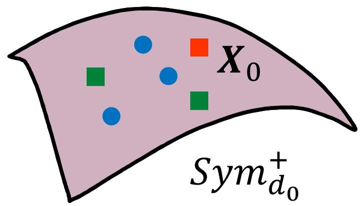
$$F = [f_1, f_2, \dots, f_N]_{d_0 \times N}$$

◆Second-order Pooling: $d_0 \times d_0$ symmetric semi*-positive definite matrix computing

$$X_0 = \frac{1}{N-1} \sum_{i=1}^{N} (f_i - \bar{f}_i)(f_i - \bar{f}_i)^T \in Sym_{d_0}^{+}$$

*: *use regularization to tackle singularity problem*

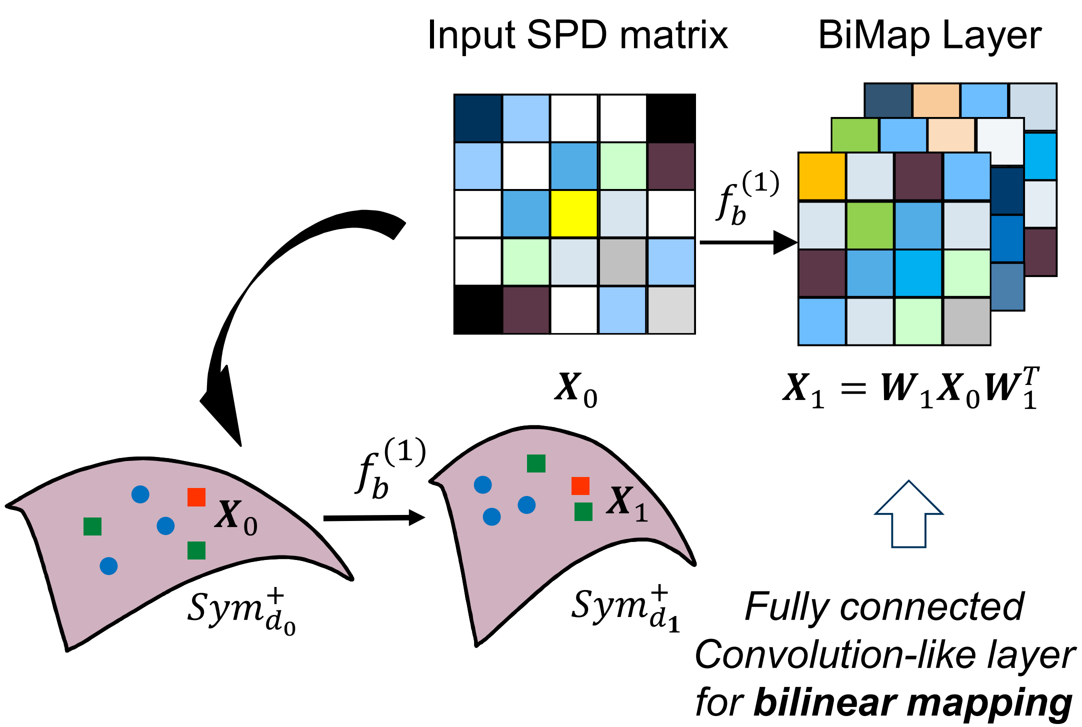
Wang et al., CVPR'12; Liu et al., ICMI'14; Huang et al., ICML'15



A Riemannian Network for SPD Matrix Learning

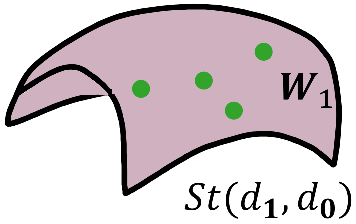
- BiMap Layer:
  - Generate compact & discriminative SPD matrices with a bilinear mapping scheme

Compact & Discriminative



◆ $\boldsymbol{X}_1 \in Sym^+_{d_1}$? → $\boldsymbol{W}_1 \in \mathbb{R}_*^{d_1 \times d_0}$ full rank

◆ The space of full rank matrices is non-compact $\xrightarrow{\boldsymbol{W}_1\boldsymbol{W}_1^T=\boldsymbol{I}}$ $\boldsymbol{W}_1 \in St(d_1, d_0)$



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

A Riemannian Network for SPD Matrix Learning

- ReEig layer
  - Perform a non-linear rectification on SPD matrices

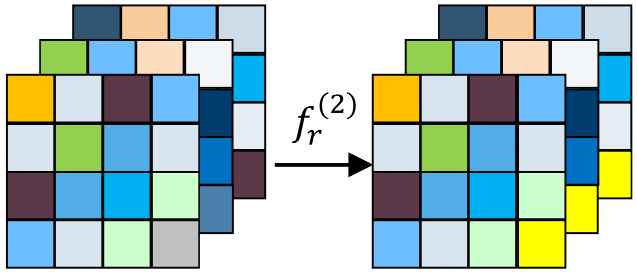
ReLU: $\max(0, \boldsymbol{x})$

◆ $\mathrm{A} = \max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma}_{\mathbf{1}}) \rightarrow$ diagnoal

◆ $\mathrm{A}(i,i) = \begin{cases} \boldsymbol{\Sigma}_{\mathbf{1}}(i,i) & \boldsymbol{\Sigma}_{\mathbf{1}}(i,i) > \epsilon \\ \epsilon & \boldsymbol{\Sigma}_{\mathbf{1}}(i,i) \leq \epsilon \end{cases}$

Prevent the input SPD matrices being close to non-positive ones

BiMap Layer　　ReEig Layer



$\boldsymbol{X}_1 = \boldsymbol{W}_1 \boldsymbol{X}_0 \boldsymbol{W}_1^T$　　$\boldsymbol{X}_2 = \boldsymbol{U}_1 \max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma}_{\mathbf{1}}) \boldsymbol{U}_1^T$

$\boldsymbol{X}_1 = \boldsymbol{U}_1 \boldsymbol{\Sigma}_{\mathbf{1}} \boldsymbol{U}_1^T$

Introduce non-linearity

SVD

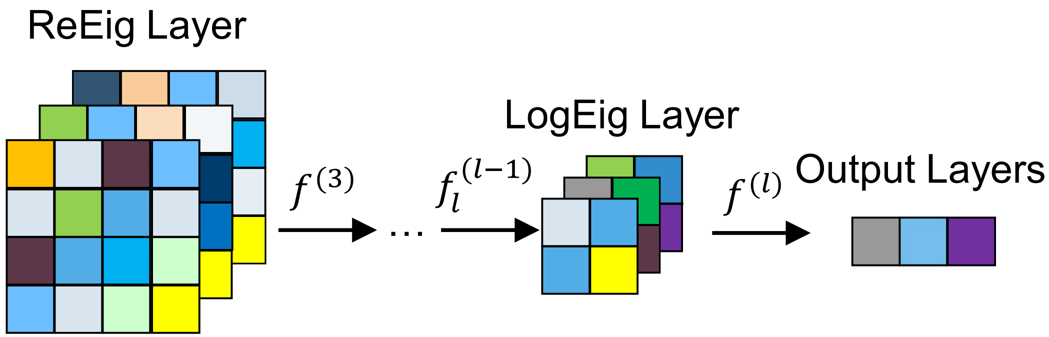
ReLU-like layer for ***non-linearity***

A Riemannian Network for SPD Matrix Learning

- LogEig layers
  - Perform Riemannian computing on SPD matrices for regular output layers with objective function

◆Log-Euclidean Riemannian metric:

$$d_l^2(\boldsymbol{X}_1, \boldsymbol{X}_2) = || \log(\boldsymbol{X}_1) - \log(\boldsymbol{X}_2) ||_{\mathcal{F}}^2$$



$$\boldsymbol{X}_{l-1} = \log(\boldsymbol{X}_{l-2}) = \boldsymbol{U}_{l-2}\log(\boldsymbol{\Sigma}_{l-2})\boldsymbol{U}_{l-2}^T$$

$$\boldsymbol{X}_{l-2} = \boldsymbol{U}_{l-2}\boldsymbol{\Sigma}_{l-2}\boldsymbol{U}_{l-2}^T$$

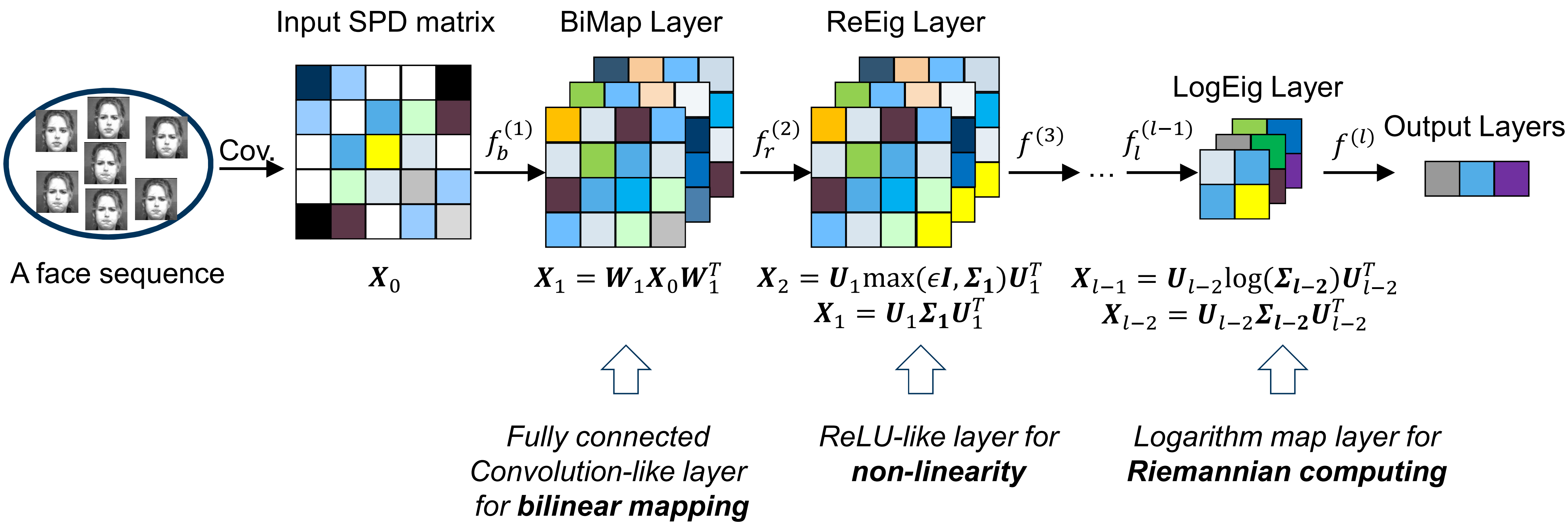
Riemannian Computing

SVD

Logarithm map layer for
Riemannian computing

A Riemannian Network for SPD Matrix Learning

- Building a deep network on SPD manifolds
  - perform deep learning on input SPD matrices, and output compact & discriminative SPD matrices



A Riemannian Network for SPD Matrix Learning

- Training the SPD network?
  - Conventional backpropagation

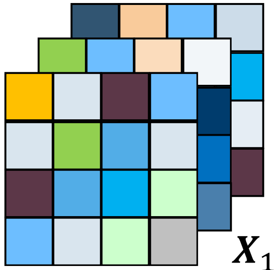
$$\frac{\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)}{\partial \boldsymbol{W}_k} = \frac{\partial L^{(k+1)}(\boldsymbol{X}_k, y)}{\partial \boldsymbol{X}_k} \frac{\partial f^{(k)}(\boldsymbol{X}_{k-1})}{\partial \boldsymbol{W}_k},$$

$$\frac{\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)}{\partial \boldsymbol{X}_{k-1}} = \frac{\partial L^{(k+1)}(\boldsymbol{X}_k, y)}{\partial \boldsymbol{X}_k} \frac{\partial f^{(k)}(\boldsymbol{X}_{k-1})}{\partial \boldsymbol{X}_{k-1}},$$

  - **Two key issues**
    - **Issue#1**: Update a new valid orthogonal weights?
    - **Issue#2**: Update the structured data within the matrix factorizations like SVD?

A Riemannian Network for SPD Matrix Learning

BiMap Layer

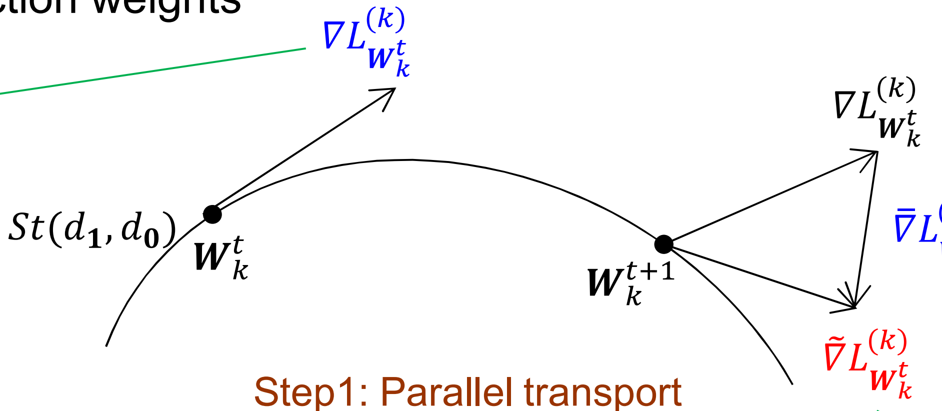


$X_1 = W_1 X_0 W_1^T$

- Training the SPD network?
  - **Issue#1**: exploit a stochastic gradient descent (SGD) setting on Stiefel manifolds to update the orthogonal connection weights

Euclidean gradient:

$$\nabla L_{W_k^t}^{(k)} = \frac{\partial L^{(k+1)}}{\partial X_k} \frac{\partial f^{(k)}}{\partial W_k^t} = 2\frac{\partial L^{(k+1)}}{\partial X_k} W_k^t X_{k-1}$$



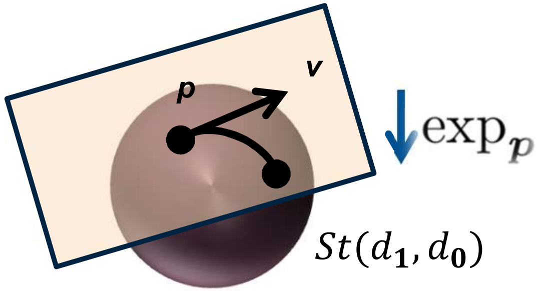
Normal component

$$\bar{\nabla} L_{W_k^t}^{(k)} = \nabla L_{W_k}^{(k)} (W_k^t)^T W_k^t$$

Step1: Parallel transport
(Absil et al., 2008)

Riemannian gradient

$$\tilde{\nabla} L_{W_k^t}^{(k)} = \nabla L_{W_k^t}^{(k)} - \nabla L_{W_k^t}^{(k)} (W_k^t)^T W_k^t$$



Step2: Retraction operation
(Absil et al., 2008)

$$W_k^{t+1} = \Gamma(W_k^t - \lambda \tilde{\nabla} L_{W_k}^{(k)})$$

A Riemannian Network for SPD Matrix Learning

- Training the SPD network
  - **Issue#2**: Matrix backprop for computing the gradients of the involved data within SVD operations

◆Matrix Backprop *[Ionescu et al., ICCV'15]*

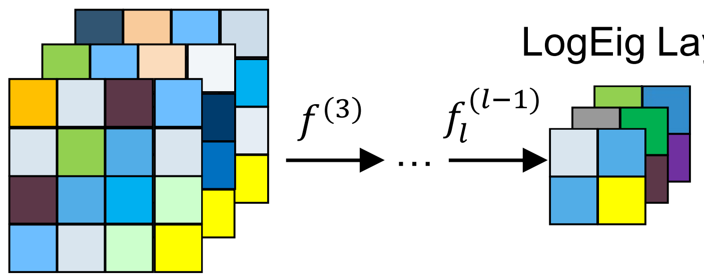
$\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = \mathcal{F}^*\left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\right)$, where $\mathrm{d}\boldsymbol{X}_k = \mathcal{F}(\mathrm{d}\boldsymbol{X}_{k-1}), a:$
$\mathcal{F}(b) = \mathcal{F}^*(a): b, A : B = Tr(A^T B)$

$$\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} : \mathrm{d}\boldsymbol{X}_{k-1} = \frac{\partial L^{(k')}}{\partial \boldsymbol{U}} : \mathrm{d}\boldsymbol{U} + \frac{\partial L^{(k')}}{\partial \boldsymbol{\Sigma}} : \mathrm{d}\boldsymbol{\Sigma} = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_{k+1}} : \mathrm{d}\boldsymbol{X}_k$$

$$\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = ???$$

ReEig Layer

LogEig Layer



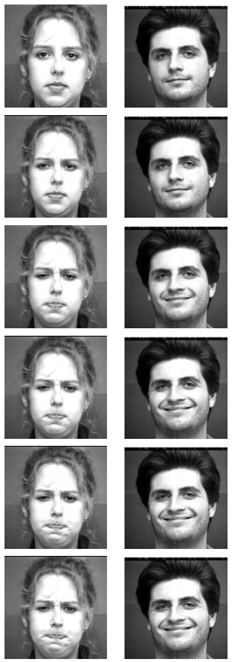
$\boldsymbol{X}_2 = \boldsymbol{U}_1 \max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma}_1) \boldsymbol{U}_1^T$ $\quad \boldsymbol{X}_{l-1} = \boldsymbol{U}_{l-2} \log(\boldsymbol{\Sigma}_{l-2}) \boldsymbol{U}_{l-2}^T$

$\boldsymbol{X}_1 = \boldsymbol{U}_1 \boldsymbol{\Sigma}_1 \boldsymbol{U}_1^T$ $\quad \boldsymbol{X}_{l-2} = \boldsymbol{U}_{l-2} \boldsymbol{\Sigma}_{l-2} \boldsymbol{U}_{l-2}^T$

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{X}} = ?$$

A Riemannian Network for SPD Matrix Learning

- Evaluation datasets

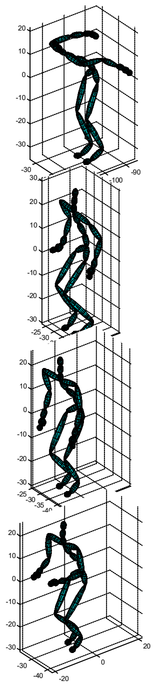


◆AFEW: *Emotion Recognition*

- ❖ 1,345 videos
- ❖ 7 emotions

◆ Data augment

- ❖ 1,747 train subvideos

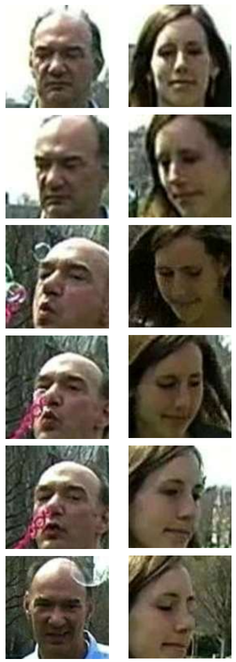


◆HDM05: *Activity recognition*

- ❖ 2,337 sequences
- ❖ 130 actions

◆ Data augment

- ❖ 18,000 train subsequences



◆PaSC: *Face verification*

- ❖ 2802 videos
- ❖ 265 subjects

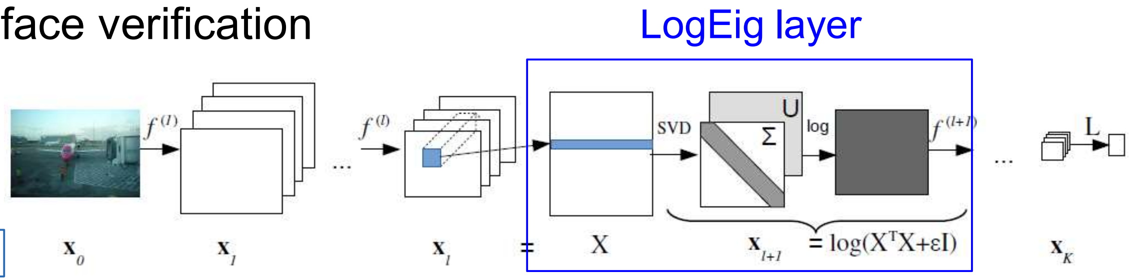
◆ Data augment

- ❖ 12,529 train subvideos

A Riemannian Network for SPD Matrix Learning

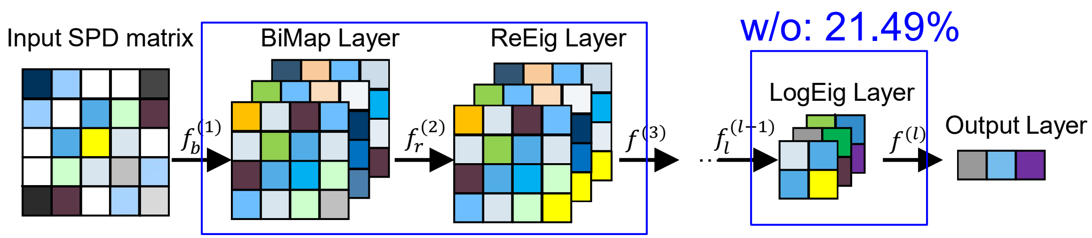
- Evaluation results
  - Emotion recognition, action recognition and face verification

Method	AFEW	HDM05	PaSC1	PaSC2
STM-ExpLet	31.73%	–	–	–
RSR-SPDML	30.12%	48.01%± 3.38	–	–
DeepO2P	28.54%	–	68.76%	60.14%
HERML-DeLF	–	–	58.0%	59.0%
VGGDeepFace	–	–	78.82%	68.24%
CDL	31.81%	41.74%± 1.92	78.29%	70.41%
LEML	25.13%	46.87%± 2.19	66.53%	58.34%
SPDML-AIM	26.72%	47.25%± 2.78	65.47%	59.03%
SPDML-Stein	24.55%	46.21%± 2.65	61.63%	56.67%
RSR	27.49%	41.12%± 2.53	–	–
SPDNet-0BiRe	26.32%	48.12%± 3.15	68.52%	63.92%
SPDNet-1BiRe	29.12%	55.26%± 2.37	71.75%	65.81%
SPDNet-2BiRe	31.54%	59.13%± 1.78	76.23%	69.64%
SPDNet-3BiRe	34.23%	61.45% ± 1.12	80.12%	72.83%



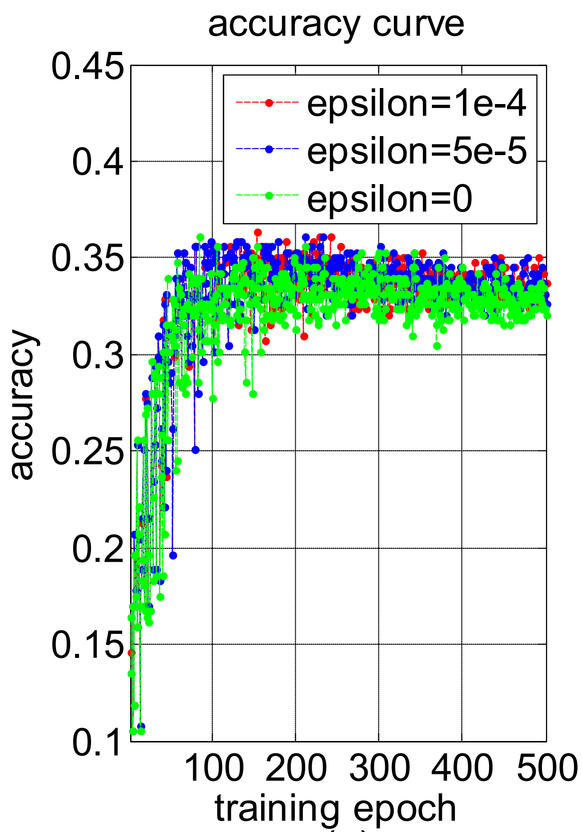
***DeepO2P**: Inserts one LogEig-like layer into a standard AlexNet architecture, Ionescu et.al., ICCV'15*

Break down when the SPD data are not Cov. matrix for images (e.g., HDM05 exp)

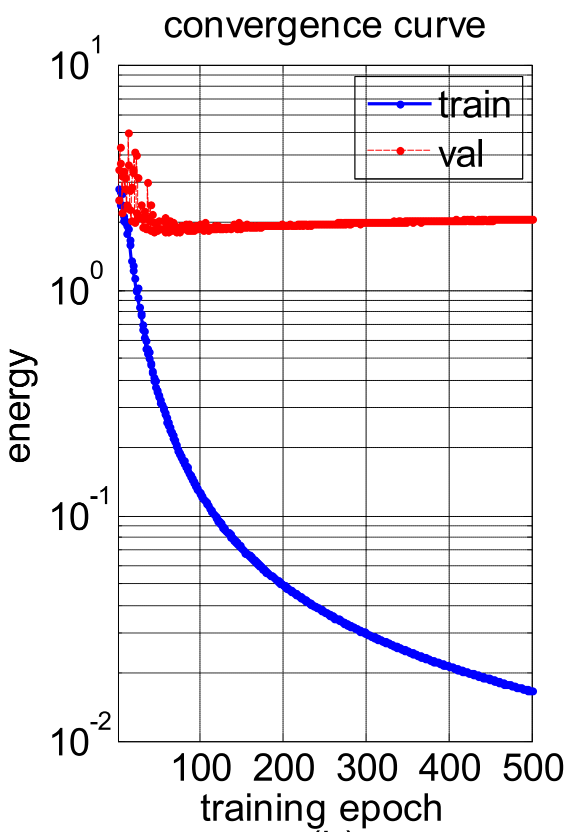


A Riemannian Network for SPD Matrix Learning

- Power of the designed rectification layer & Convergence behavior

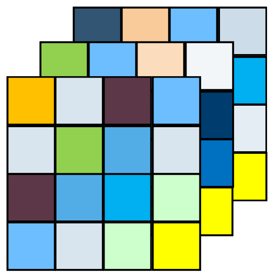


(a)



(b)

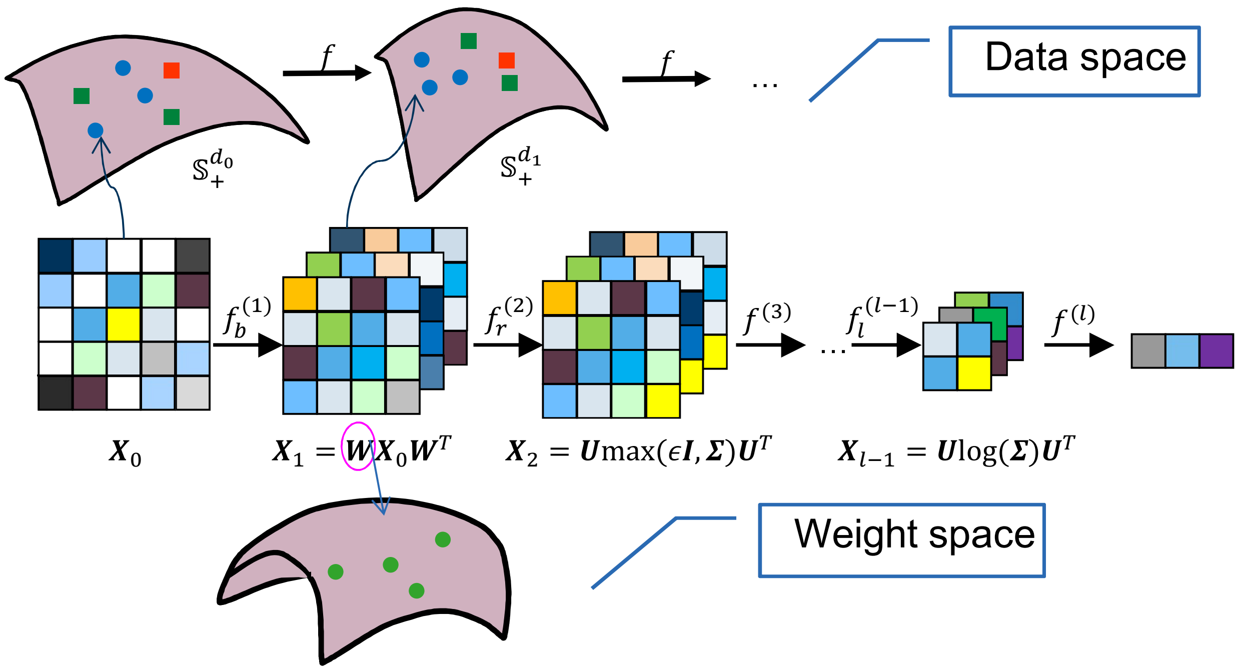
ReEig Layer



$$\boldsymbol{X}_2 = \boldsymbol{U}_1 \max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma}_1) \boldsymbol{U}_1^T$$
$$\boldsymbol{X}_1 = \boldsymbol{U}_1 \boldsymbol{\Sigma}_1 \boldsymbol{U}_1^T$$

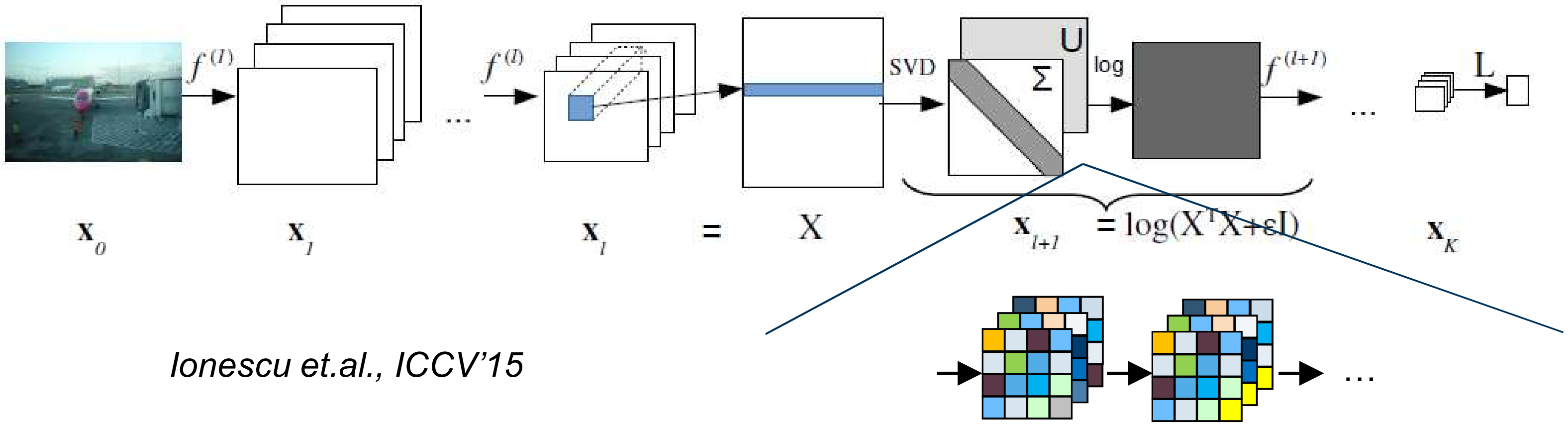
Summary

- A new methodology of deep learning on SPD manifolds
- A new SGD setting on Stiefel manifolds for backpropagation



Future Work

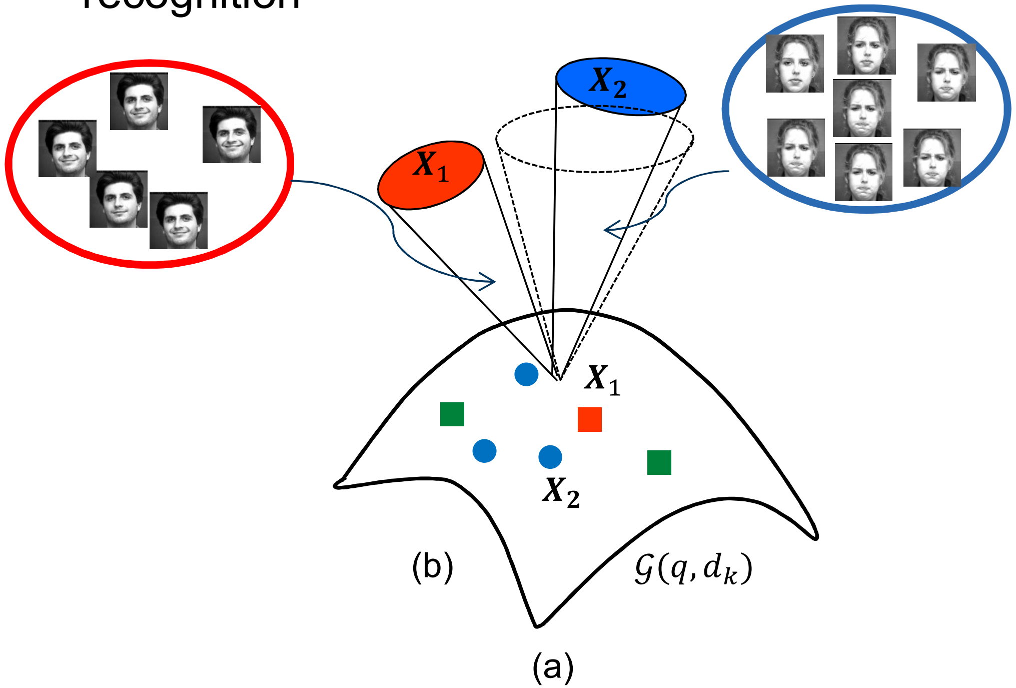
- Build SPDNets on the top of existing convolutional networks that start from images



Ionescu et.al., ICCV'15

Building Deep Networks on Grassmann Manifolds

- Deep learning on Grassmann manifolds $Gr(q, d_k)$
  - Applications (based on linear subspace modeling): face recognition, emotion recognition, action recognition



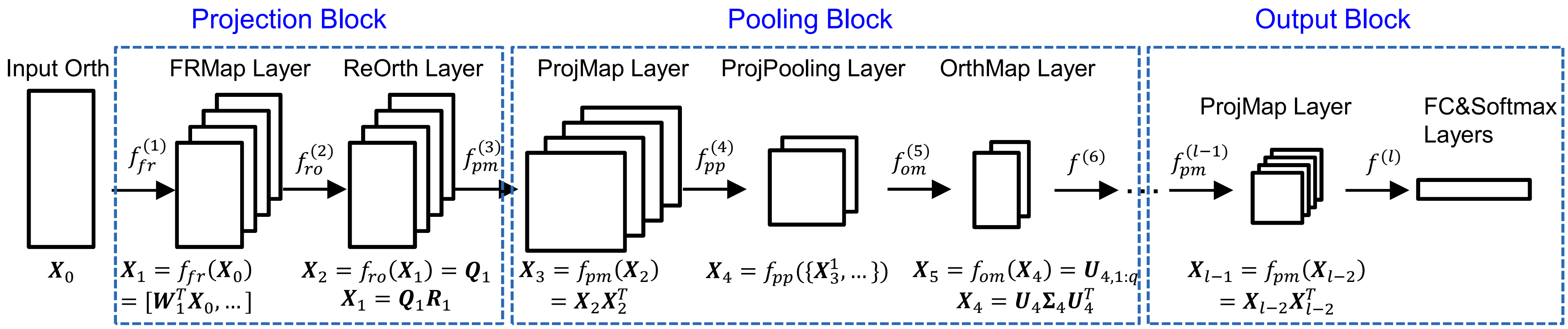
◆Linear subspace spanned by orthonormal basis matrix X_i derived on the data Y_i, i.e., the q-leadest eigenvectors

❖ $Y_i Y_i^T \simeq X_i \Sigma_i X_i^T$

Hamm et al., ICML'08; Harandi et al., CVPR'11; Huang et al., CVPR'15

Building Deep Networks on Grassmann Manifolds

- Deep learning on Grassmann manifolds $Gr(q, d_k)$
  - Receive orthonormal matrices as inputs, and preserve the Grassmann structure across layers



Building Deep Networks on Grassmann Manifolds

- **Projection block**: Riemannian geometry-aware **dim reduction**

◆For **dim reduction** $X_k = \mathbf{W}_k X_{k-1}, \quad X_{k-1} \in Gr(q, d_k)$

❖ **Step#1**: Full rank mapping

FRMap

❑ W_k full rank, $X_k \in \mathbb{R}^{d_k \times d_{k-1}}$

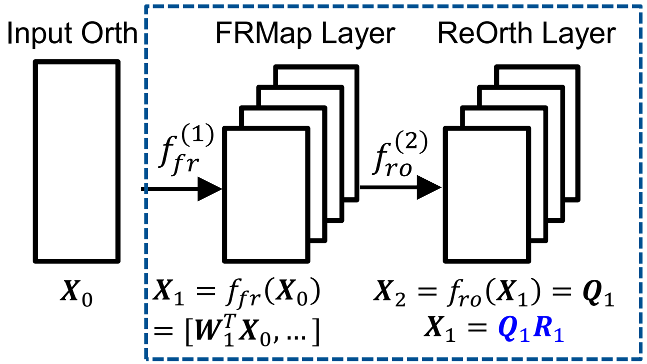
❑ Optimizing over the weight conjugate space, i.e., the manifolds of positive semidefinite (PSD) matrices

ReOrth

❖ **Step#2**: Re-orthonormalization

❑ QR decomposition for orthonormalization

❑ Non-linear matrix factorization Introduces a non-linearity



Building Deep Networks on Grassmann Manifolds

- **Pooling block:** reduce the complexity of deep model

◆ **Step#1(ProjMap)**

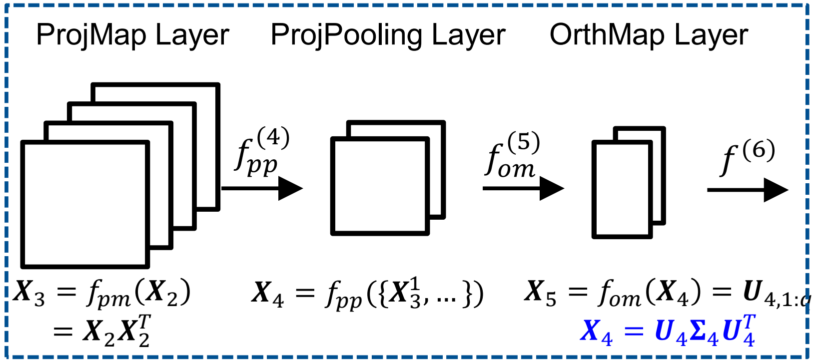
❖ Classical Projection map

❖ Grassmann → Euclidean

◆ **Step#3(OrthMap)**

❖ Retraction

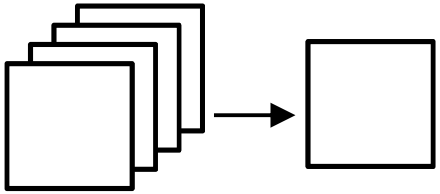
❖ Euclidean→Grassmann



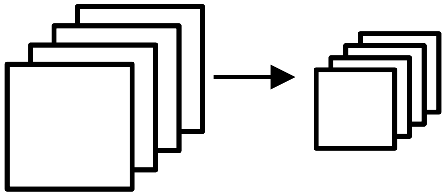
◆ **Step#2 (ProjPooling)**

❖ Two regular poolings

❑ Across projections



❑ Within projections



Building Deep Networks on Grassmann Manifolds

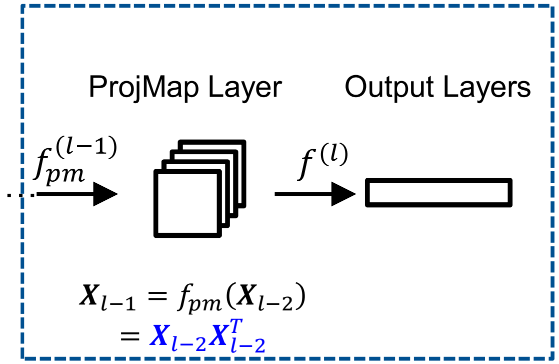
- **Output block**: enable regular output layers, e.g., fully connected and softmax layers

◆**Step#1(ProjMap)**

- ❖ Classical Projection map
- ❖ Grassmann → Euclidean

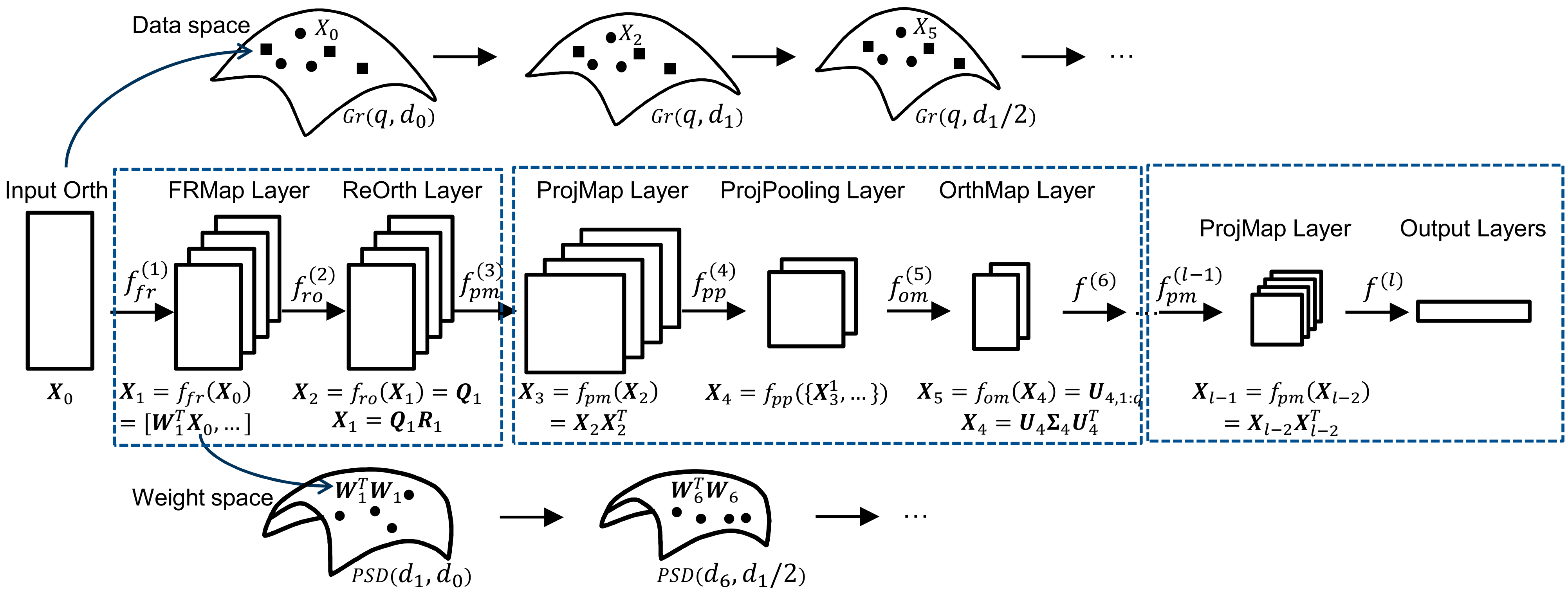
◆**Step#2(**Regular layers**)**

- ❖ Fully connected layer
- ❖ Softmax layer
- ❖ ….



Building Deep Networks on Grassmann Manifolds

- Deep learning on Grassmann manifolds $Gr(q, d_k)$
  - Deep learning on input Grassmannian data, and output compact & discriminative Grassmannian data

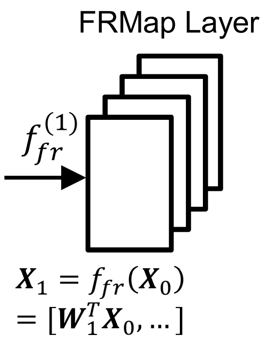


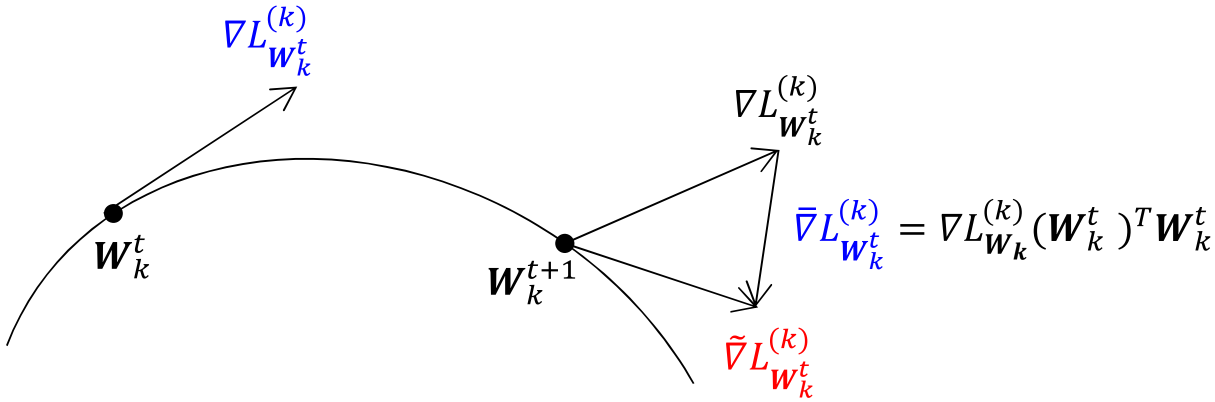
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

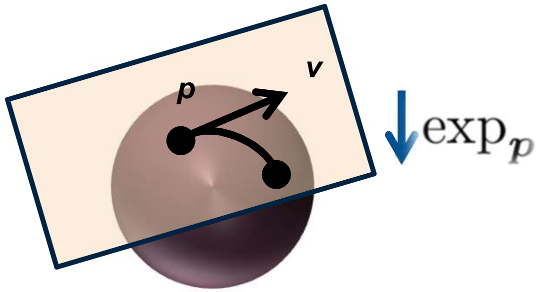
CVL Computer Vision Lab

Building Deep Networks on Grassmann Manifolds

- Training GrNet
  - Exploit a stochastic gradient descent (SGD) setting for updating the weights on the PSD manifold







ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

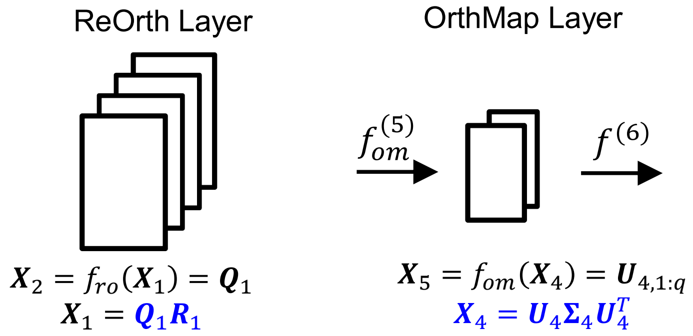
CVL Computer Vision Lab

Building Deep Networks on Grassmann Manifolds

- Training GrNet
  - Matrix backprop for computing the gradients of the involved Grassmannian matrices

◆Matrix Backprop *[Ionescu et al., ICCV’15]*

$\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = \mathcal{F}^*(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k})$, where $\mathrm{d}\boldsymbol{X}_k = \mathcal{F}(\mathrm{d}\boldsymbol{X}_{k-1}), a : \mathcal{F}(b) = \mathcal{F}^*(a) : b, A : B = Tr(A^T B)$

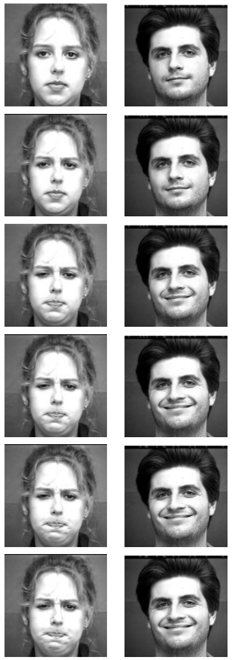


$$\Rightarrow \frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} : \mathrm{d}\boldsymbol{X}_{k-1} = \frac{\partial L^{(k')}}{\partial \boldsymbol{U}} : \mathrm{d}\boldsymbol{U} + \frac{\partial L^{(k')}}{\partial \boldsymbol{\Sigma}} : \mathrm{d}\boldsymbol{\Sigma} = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_{k+1}} : \mathrm{d}\boldsymbol{X}_k$$

$$\Rightarrow \frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = ???$$

Building Deep Networks on Grassmann Manifolds

- Evaluation datasets

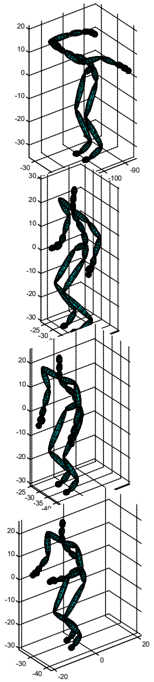


◆AFEW: *Emotion Recognition*

- ❖ 1,345 videos
- ❖ 7 emotions

◆ Data augment

- ❖ 1,747 train subvideos

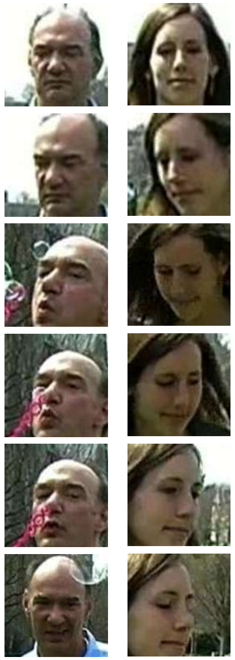


◆HDM05: *Activity recognition*

- ❖ 2,337 sequences
- ❖ 130 actions

◆ Data augment

- ❖ 18,000 train subsequences



◆PaSC: *Face verification*

- ❖ 2802 videos
- ❖ 265 subjects

◆ Data augment

- ❖ 12,529 train subvideos

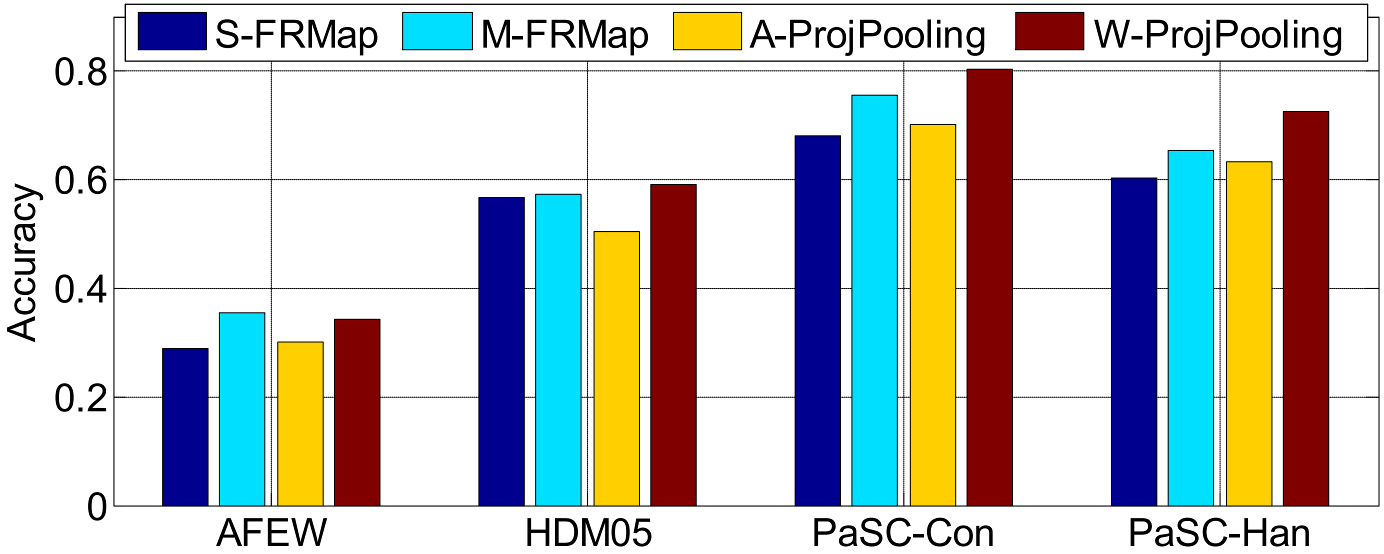
Building Deep Networks on Grassmann Manifolds

- Evaluation results
  - Emotion recognition, action recognition and face verification

Method	AFEW	HDM05	PaSC1	PaSC2
STM-ExpLet	31.73%	–	–	–
RSR-SPDML	30.12%	48.01%± 3.38	–	–
DCC	25.78%	41.34%± 1.05	75.83%	67.04%
GDA	29.11%	46.25%± 2.71	71.38%	67.49%
GGDA	29.45%	46.87%± 2.31	66.71%	68.41%
PML	28.98%	47.12%± 1.59	73.45%	68.32%
VGGDeepFace	–	–	78.82%	68.24%
DeepO2P	28.54%	–	68.76%	60.14%
SPDNet	34.23%	61.45%± 1.12	80.12%	72.83%
GrNet-0Block	25.34%	51.12%± 3.55	68.52%	63.92%
GrNet-1Block	32.08%	57.73%± 2.24	80.15%	72.51%
GrNet-2Blocks	34.23%	59.23%± 1.78	80.52%	72.76%

Building Deep Networks on Grassmann Manifolds

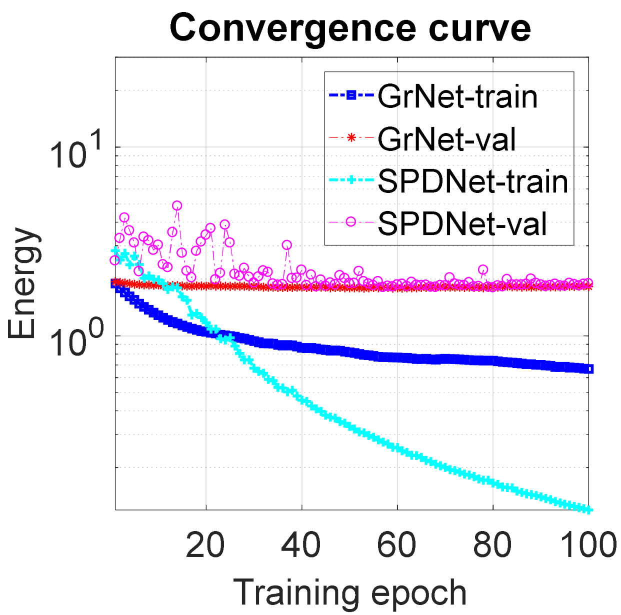
- Evaluation results
  - S-FRMap: Single mapping per FRMap layer
  - M-FRMap: Multiple mapping per FRMap layer
  - A-ProjPooling: Pooling between projection matrices + M-FRMap
  - W-ProjPooling: Pooling within projection matrices + M-FRMap

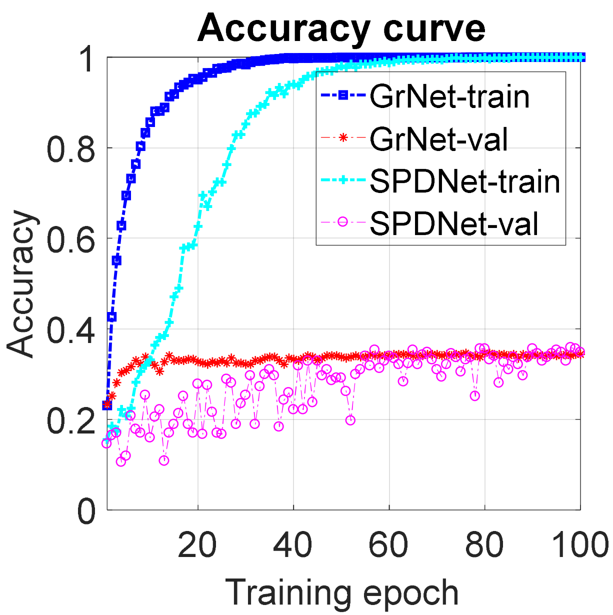


Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Building Deep Networks on Grassmann Manifolds

- Convergence behavior



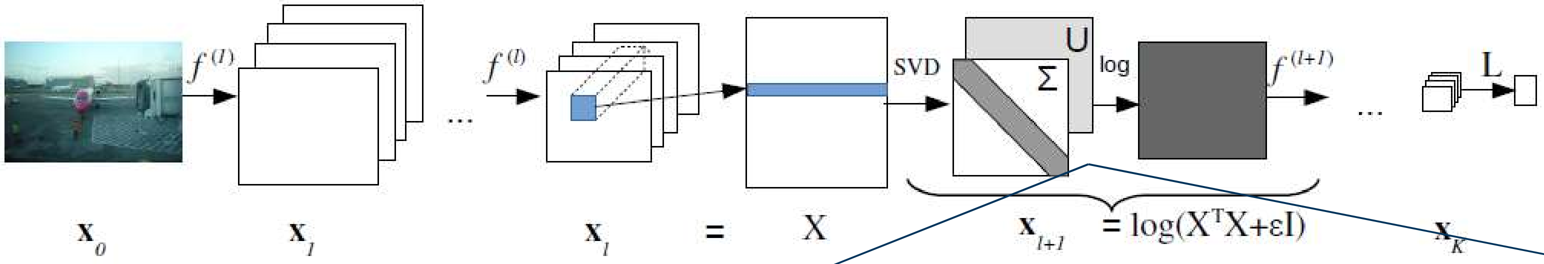


Building Deep Networks on Grassmann Manifolds

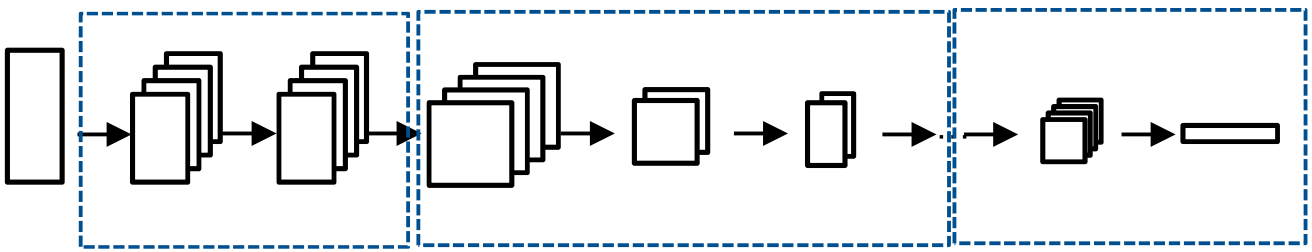
- Summary
  - A new direction of deeply learning Grassmannian data has been opened up
  - A procedure to train the new network
    - An update rule for the connection weights on PSD manifold
    - An update rule for the structured data within QR and EIG decompositions

Building Deep Networks on Grassmann Manifolds

- Future work
  - Deep learning from Euclidean to Grassmannin data?

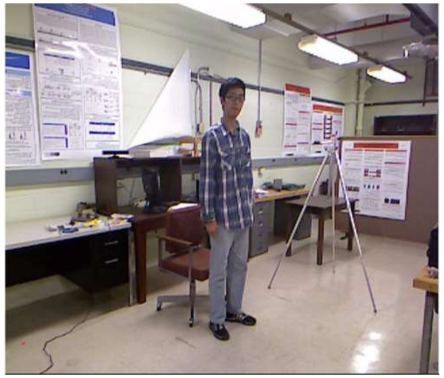


Ionescu et.al., ICCV'15

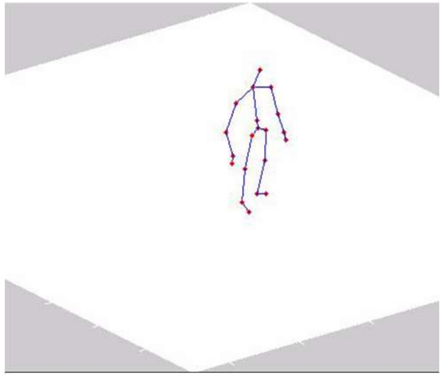


Deep Learning on Lie Group for Skeleton-based Action Recognition

Human performing an action → Cost effective depth sensors like Kinect + State-of-the-art depth-based skeleton estimation algorithm [Shotton 2011] → Real-time skeletal sequence



UTKinect-Action dataset [Xia2012]



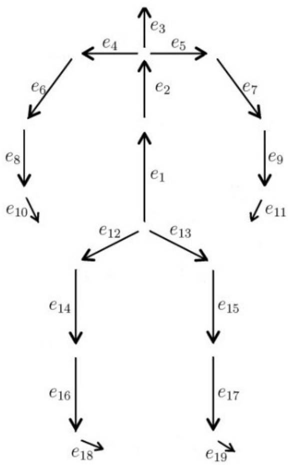
J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman and A. Blake, "Real-time Human Pose Recognition in Parts From a Single Depth Image", In CVPR, 2011.

L. Xia, C. C. Chen and J. K. Aggarwal, "View Invariant Human Action Recognition using Histograms of 3D Joints ", In CVPRW, 2012.

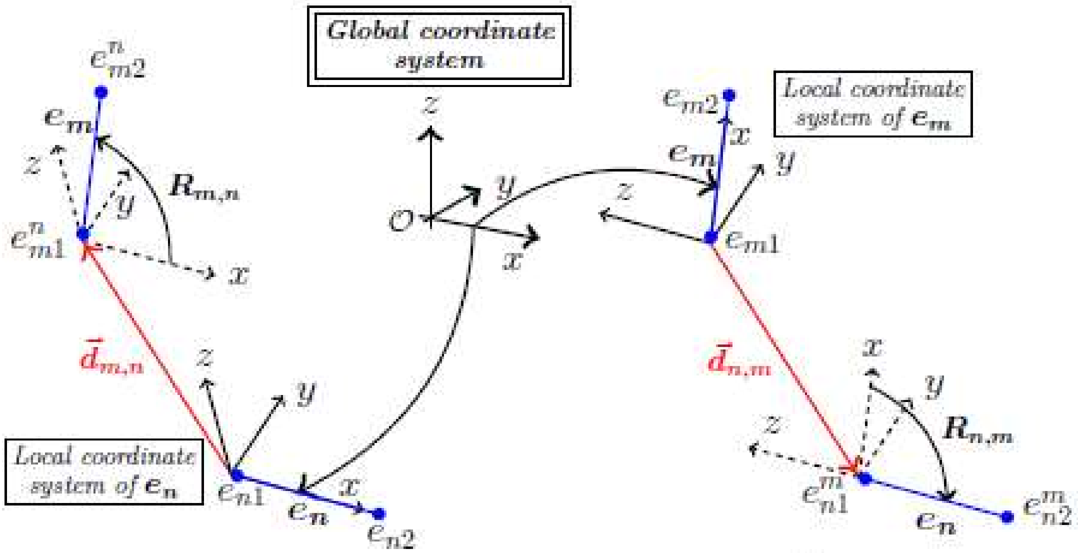
Courtesy: Raviteja Vemulapalli

Human Skeleton: Lie Group Representation

- Special Rotation Group (Lie Group) representation for one skeleton



Set of rigid rods
(body bones)



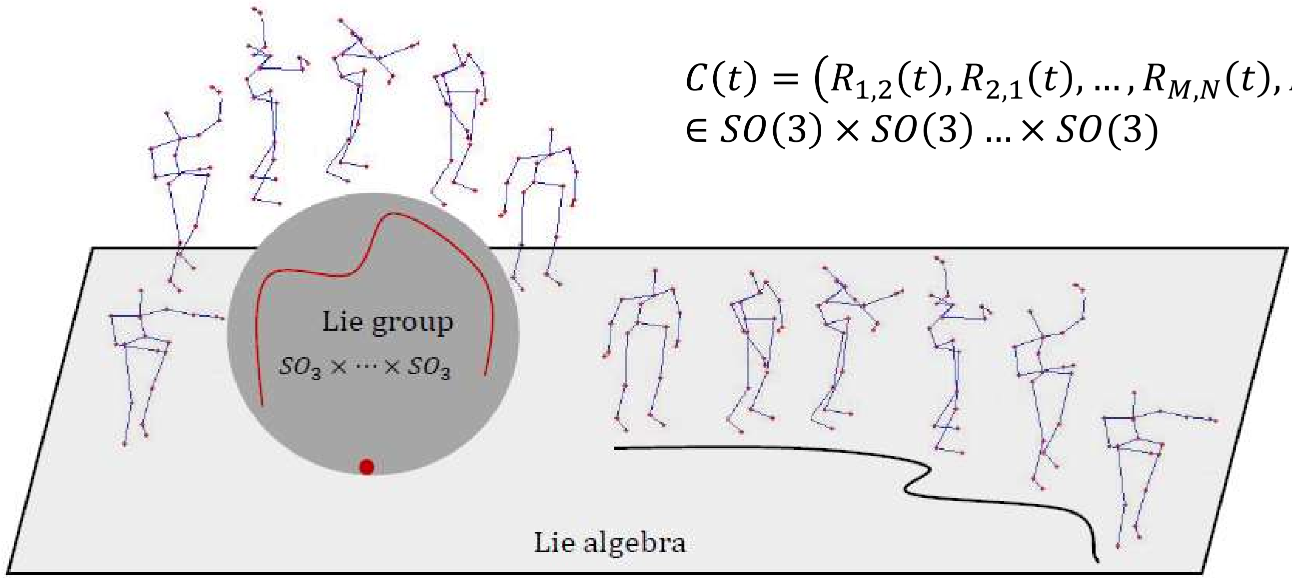
$$C(t) = \left(R_{1,2}(t), R_{2,1}(t), \dots, R_{M,N}(t), R_{N,M}(t)\right)$$
$$\in SO(3) \times SO(3) \dots \times SO(3)$$

Skeleton-based action recognition, *Vemulapalli (***Rama Chellappa group***) et al., CVPR'14, CVPR '16*

ZZ1

Deep Learning on Lie Group for Skeleton-based Action Recognition

- Lie Group curve representation for one moving skeleton



$$C(t) = \left(R_{1,2}(t), R_{2,1}(t), \ldots, R_{M,N}(t), R_{N,M}(t),\right) \in SO(3) \times SO(3) \ldots \times SO(3)$$

$$C = (C(1), C(2) \ldots, C(N))$$

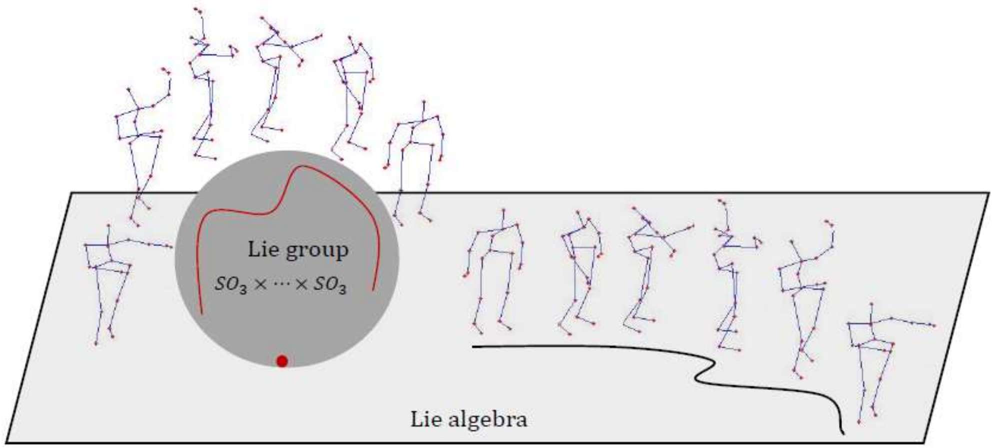
Skeleton-based action recognition, *Vemulapalli (***Rama Chellappa group***) et al., CVPR'14, CVPR '16*

Slide 39

ZZ1 Zhiwu Zhiwu; 24.10.2016

Deep Learning on Lie Group for Skeleton-based Action Recognition

- Motivation A
  - Speed variations (Temporal misalignment)
    - Compute a nominal curve and warp all the curves to this nominal using dynamic time warping (DTW) *[Muller, 2007]*



- **Cost additional time**
- **Two-step system**

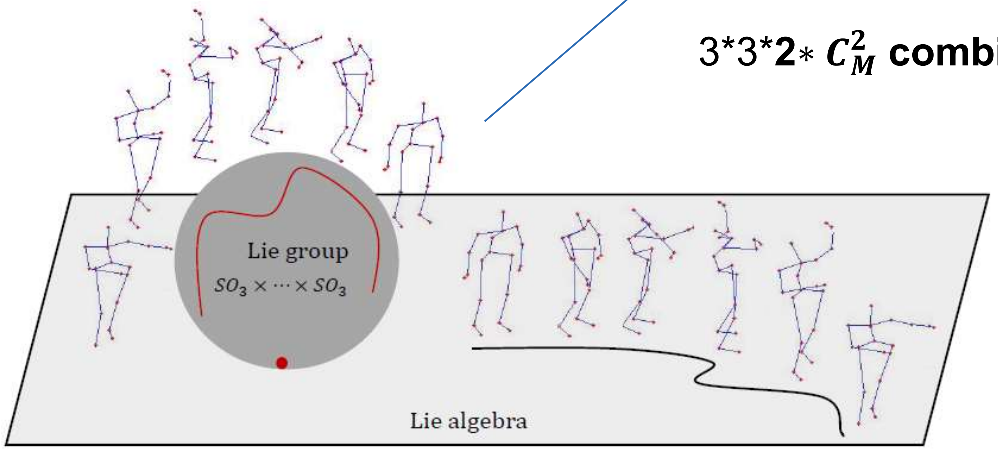
Deep Learning on Lie Group for Skeleton-based Action Recognition

- Motivation B
  - Lie group representations for action recognition tend to be extremely high-dimensional
    - Adopt SVM or PCA-like method to learn compact and discriminative features

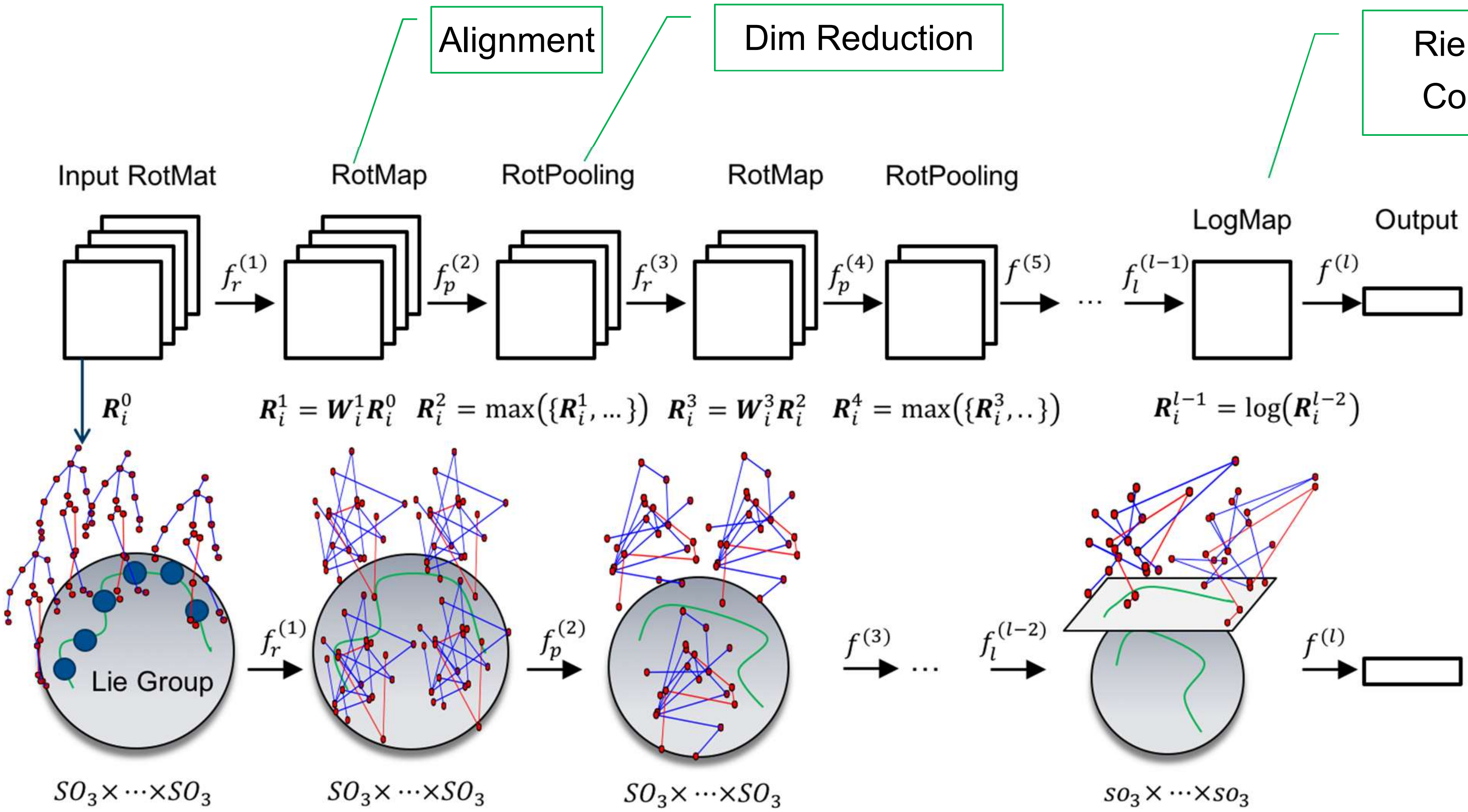
$$C(t) = \left(R_{1,2}(t), R_{2,1}(t), \ldots, R_{M,N}(t), R_{N,M}(t), \right) \in SO(3) \times SO(3) \ldots \times SO(3)$$

- High dimensionality
- Shallow learning

3*3***2*** C_M^2 **combinations/rotations* FrameNum**



Deep Learning on Lie Group for Skeleton-based Action Recognition

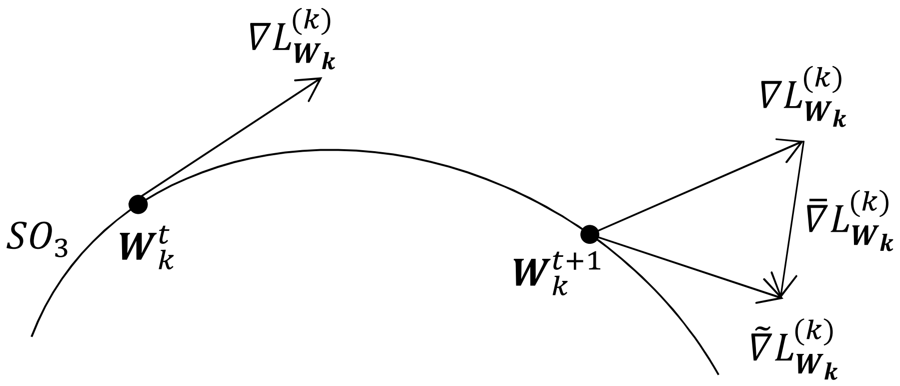


Deep Learning on Lie Group for Skeleton-based Action Recognition

- Training LieNet
  - used a stochastic gradient descent (SGD) setting on Lie groups to update the structured connection weights

$$\frac{\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)}{\partial \boldsymbol{W}_k} = \frac{\partial L^{(k+1)}(\boldsymbol{X}_k, y)}{\partial \boldsymbol{X}_k} \frac{\partial f^{(k)}(\boldsymbol{X}_{k-1})}{\partial \boldsymbol{W}_k},$$

$$\frac{\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)}{\partial \boldsymbol{X}_{k-1}} = \frac{\partial L^{(k+1)}(\boldsymbol{X}_k, y)}{\partial \boldsymbol{X}_k} \frac{\partial f^{(k)}(\boldsymbol{X}_{k-1})}{\partial \boldsymbol{X}_{k-1}},$$



Deep Learning on Lie Group for Skeleton-based Action Recognition

- Evaluation datasets
  - G3D-gaming dataset *[Bloom et. al., CVPR'12 workshop]*
    - *663 motion sequences*
  - HDM05 dataset *[*Müller et al., Technical report'07*]*
    - *2,337 motion sequences*
  - **NTU RGB+D dataset *[Shahroudy et al., CVPR'16]***
    - ***> 56,000 motion sequences***

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Deep Learning on Lie Group for Skeleton-based Action Recognition

- Quantitative evaluation

Method	G3D-Gaming
RBM+HMM [31]	86.40%
SE [40]	87.23%
SO [41]	87.95%
LieNet-0Block	84.55%
LieNet-1Block	85.16%
LieNet-2Blocks	86.67%
LieNet-3Blocks	89.10%

Method	HDM05
SPDNet [18]	61.45%±1.12
SE [40]	70.26%±2.89
SO [41]	71.31%±3.21
LieNet-0Block	71.26%±2.12
LieNet-1Block	73.35%±1.14
LieNet-2Blocks	**75.78%±2.26**

Method	RGB+D-subject	RGB+D-view
HBRNN [13]	59.07%	63.97%
Deep RNN [36]	56.29%	64.09%
Deep LSTM [36]	60.69%	67.29%
PA-LSTM [36]	62.93%	70.27%
ST-LSTM [25]	**69.2%**	**77.7%**
SE [40]	50.08%	52.76%
SO [41]	52.13%	53.42%
LieNet-0Block	53.54%	54.78%
LieNet-1Block	56.35%	60.14%
LieNet-2Blocks	58.02%	62.52%
LieNet-3Blocks	61.37%	66.95%

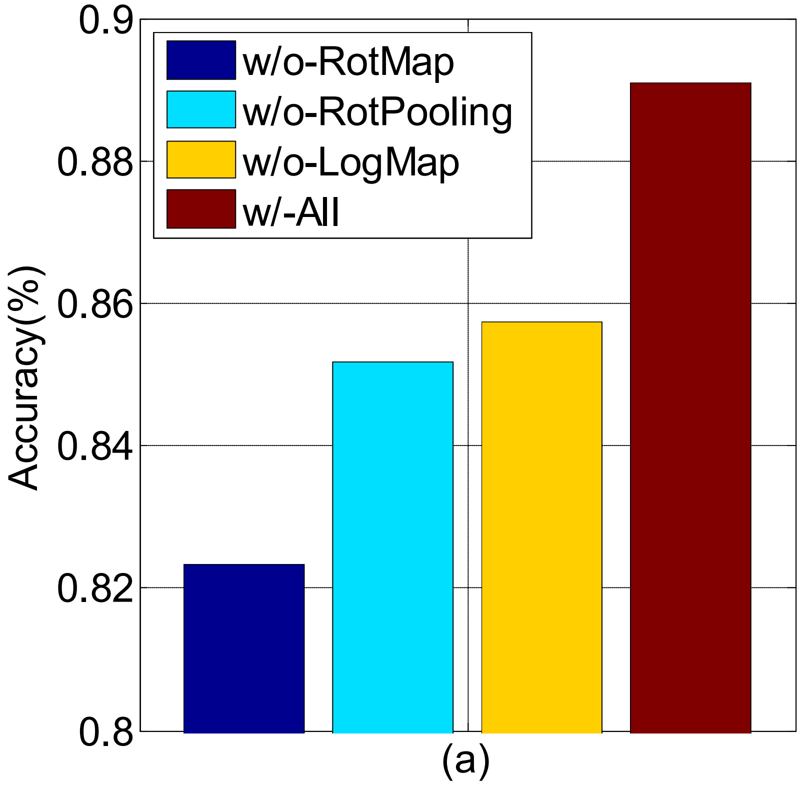
Temporal network

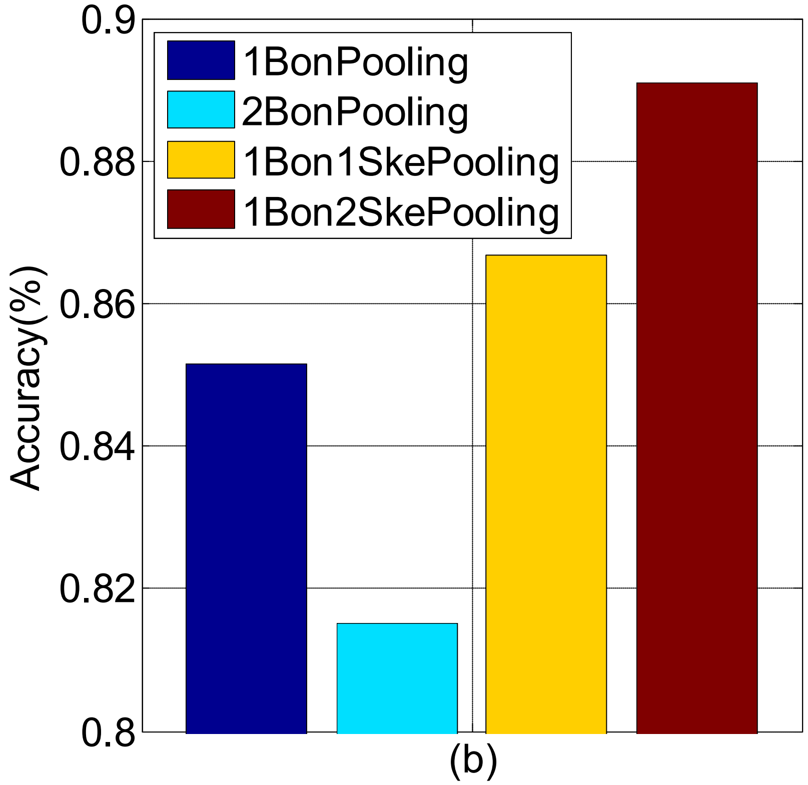
Spatio-temporal network

Spatial network

Deep Learning on Lie Group for Skeleton-based Action Recognition

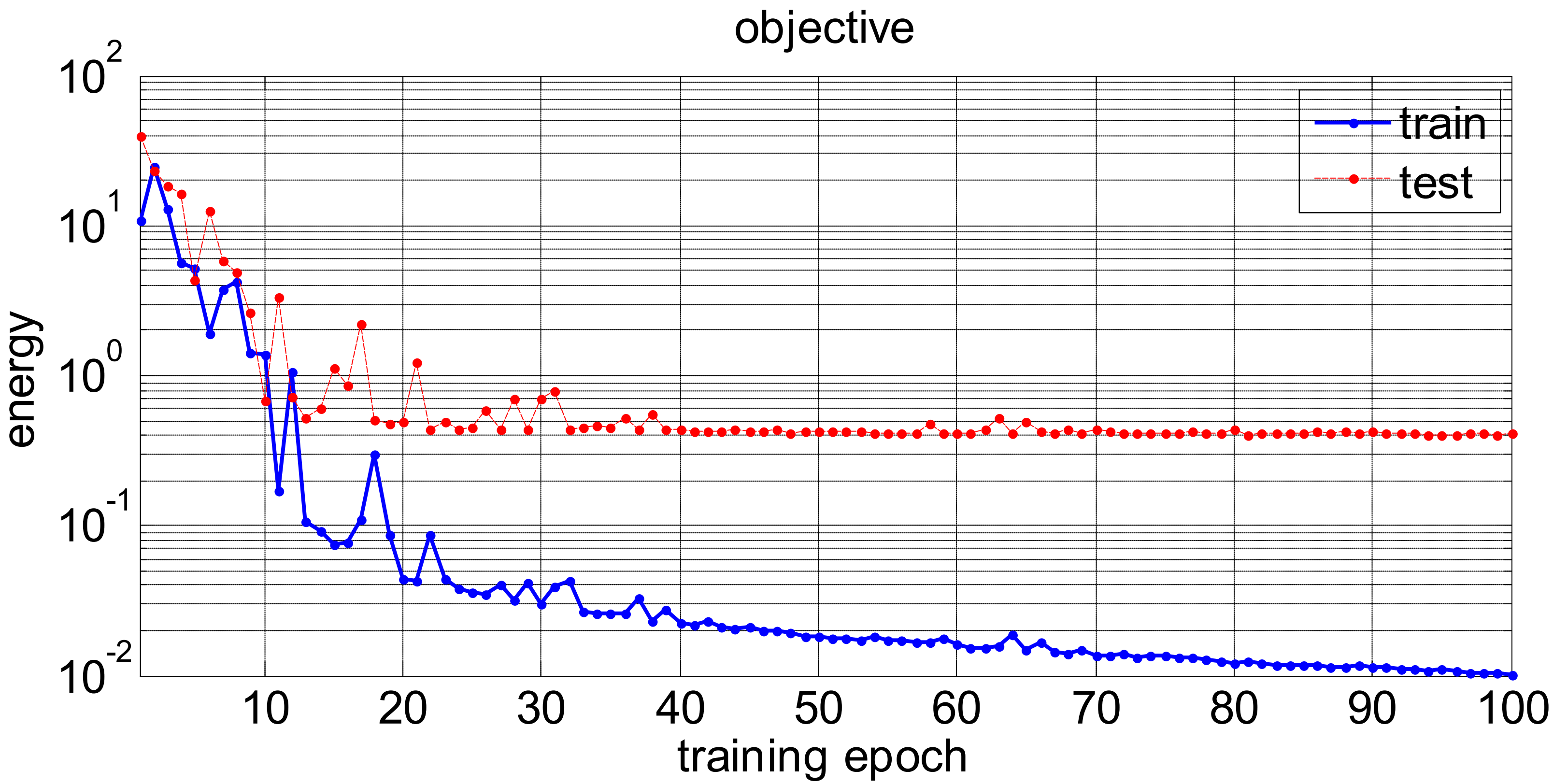
- Configuration analysis





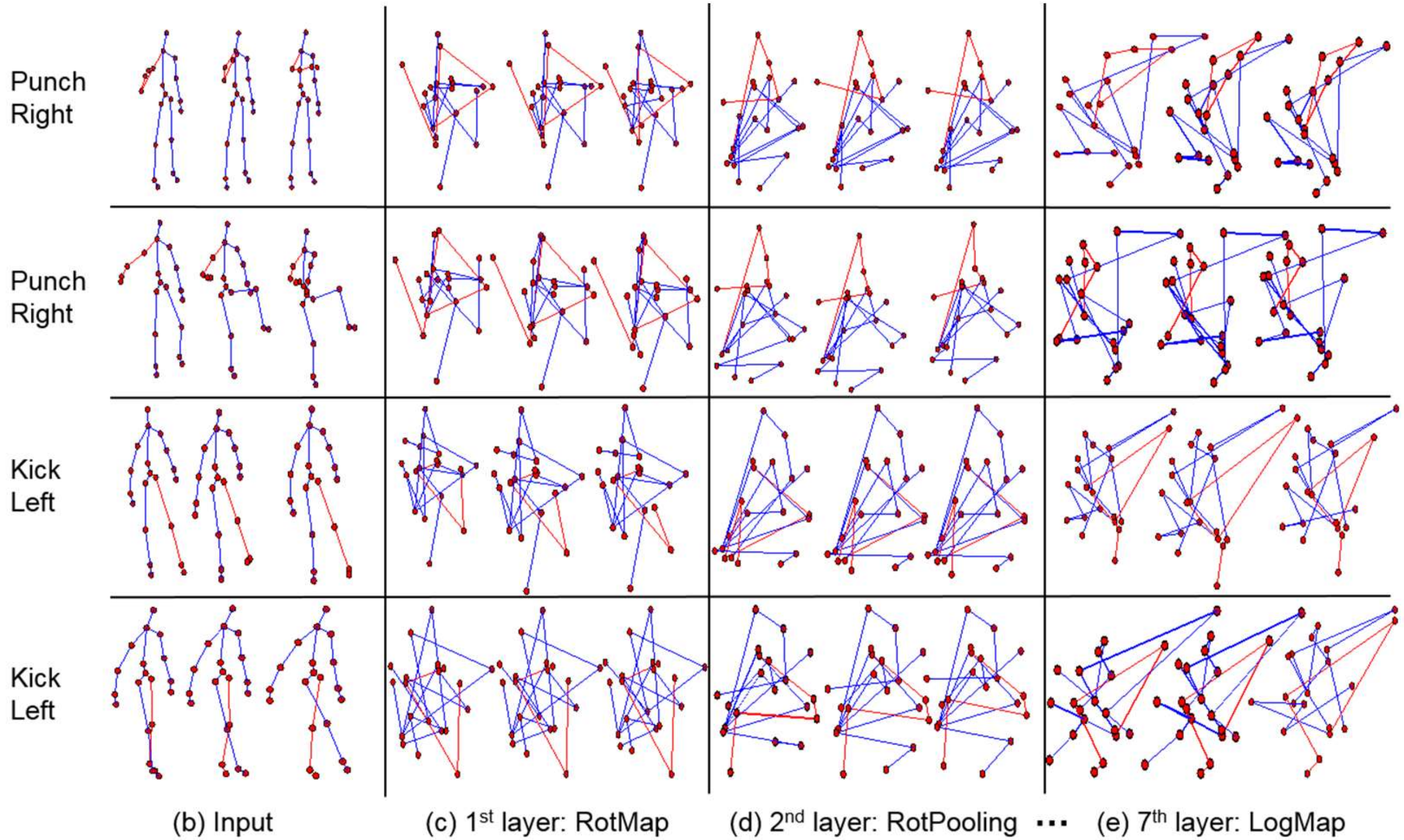
Deep Learning on Lie Group for Skeleton-based Action Recognition

- Convergence analysis on G3D-Gaming



Deep Learning on Lie Group for Skeleton-based Action Recognition

- Visualization of different LieNet layers for four action sequences



Deep Learning on Lie Group for Skeleton-based Action Recognition

- Conclusion
  - A novel neural network architecture is introduced to deeply learn more desirable Lie group representations for the problem of skeleton-based action recognition
  - A variant of stochastic gradient descent（SGD）optimization is exploited in the context of training Lie group network

Thank you for your time and attention!