DreamNet: A Deep Riemannian Manifold Network for SPD Matrix Learning (Supplementary Material)

Rui Wang[1,2][0000−0002−9984−1752], Xiao-Jun Wu[1,2⋆][0000−0002−0310−5778], Ziheng Chen[1,2][0000−0002−5366−7293], Tianyang Xu[1,2][0000−0002−9015−3128], and Josef Kittler[1,2,3][0000−0002−8110−9205]

[1] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214122, China
[2] Jiangsu Provincial Engineering Laboratory of Pattern Recognition and Computational Intelligence, Jiangnan University, Wuxi 214122, China
[3] Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford GU2 7XH, U.K.
cs_wr@jiangnan.edu.cn, zh_chen@stu.jiangnan.edu.cn, j.kittler@surrey.ac.uk
{xiaojun_wu_jnu,tianyang_xu}@163.com

Abstract. In this supplementary material, the Riemannian matrix backpropagation algorithm for training the designed DreamNet is firstly elaborated in Section 1. Then, Section 2.1 discusses the impact of different data fusion methods associated with shortcut connection on the classification performance of our model. Afterward, the discussions of the role of the trade-off parameter λ and the reconstruction error term are illustrated in Section 2.2. Next, Section 2.3 explores the classification performance of a 182-layer DreamNet on the large-scale UAV-Human dataset used. Finally, the comparison of the proposed Riemannian network with the original residual learning framework [4] is discussed in Section 3.

1 Riemannian Matrix Backpropagation

The studied Riemannian network in this article can be regarded as the data embedding model $(f, \boldsymbol{\mathcal{W}})$, where $f = f^{(K)} \circ f^{(K-1)} \circ \cdots \circ f^{(2)} \circ f^{(1)}$ signifies a series of successive function compositions and $\boldsymbol{\mathcal{W}} = \{\boldsymbol{W}_K, \boldsymbol{W}_{K-1}, ..., \boldsymbol{W}_2, \boldsymbol{W}_1\}$ denotes the parameter tuple, satisfying the properties of metric spaces. Here, K represents the number of layers of our DreamNet, $f^{(k)}$ and $\boldsymbol{W}_k$ refer to the operation function and weight parameter of the k-th layer (we use the symbol k to denote any layer of DreamNet for simplicity), respectively. The loss of the k-th layer can be expressed as $L^{(k)} = \mathcal{L}_{\mathrm{E}} \circ f^{(K)} \circ ... \circ f^{(k)}$, where $\mathcal{L}_{\mathrm{E}}$ is the cross-entropy loss of the last (E-th) RAE.

To optimize the proposed network, two key problems need to be tackled: 1) updating the Stiefel manifold-valued weights in the BiMap layers; 2) computing

⋆ Corresponding author

the gradients of the SPD matrices in the ReEig and LogEig layers. By applying SGD settings on the Stiefel manifolds [5, 3, 1], the first issue can be solved. To cope with the second problem, we exploit the matrix backpropagation methodology developed in [6] to compute the gradients of SPD matrices in the layers of ReEig and LogEig.

BiMap Layer: Since the differentiation of $\boldsymbol{X}_k$ in the BiMap layer is computed by:

$$d\boldsymbol{X}_k = d\boldsymbol{W}_k^T \boldsymbol{X}_{k-1} \boldsymbol{W}_k + \boldsymbol{W}_k^T d\boldsymbol{X}_{k-1} \boldsymbol{W}_k + \boldsymbol{W}_k^T \boldsymbol{X}_{k-1} d\boldsymbol{W}_k, \tag{1}$$

we can obtain the following chain rule on the basis of the invariance of the first-order differential:

$$\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} : d\boldsymbol{X}_k = \frac{\partial L^{(k)}}{\partial \boldsymbol{W}_k} : d\boldsymbol{W}_k + \frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} : d\boldsymbol{X}_{k-1}. \tag{2}$$

By virtue of Eqn.(1) and the matrix inner product ":" properties [6], the left-hand side of Eqn.(2) can be split into the following three equations:

$$\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} : d\boldsymbol{W}_k^T \boldsymbol{X}_{k-1} \boldsymbol{W}_k = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \boldsymbol{X}_{k-1} \boldsymbol{W}_k : d\boldsymbol{W}_k, \tag{3}$$

$$\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} : \boldsymbol{W}_k^T \boldsymbol{X}_{k-1} d\boldsymbol{W}_k = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \boldsymbol{X}_{k-1} \boldsymbol{W}_k : d\boldsymbol{W}_k, \tag{4}$$

$$\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} : \boldsymbol{W}_k^T d\boldsymbol{X}_{k-1} \boldsymbol{W}_k = \boldsymbol{W}_k \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \boldsymbol{W}_k^T : d\boldsymbol{X}_{k-1}. \tag{5}$$

In this case, the left-hand side of Eqn.(2) can be rewritten as:

$$2\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \boldsymbol{X}_{k-1} \boldsymbol{W}_k : d\boldsymbol{W}_k + \boldsymbol{W}_k \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \boldsymbol{W}_k^T : d\boldsymbol{X}_{k-1}. \tag{6}$$

As Eqn.(6) equals to the right-hand of Eqn.(2), the following partial derivatives can be derived:

$$\frac{\partial L^{(k)}}{\partial \boldsymbol{W}_k} = 2\boldsymbol{X}_{k-1} \boldsymbol{W}_k \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}, \qquad \frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = \boldsymbol{W}_k \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \boldsymbol{W}_k^T. \tag{7}$$

To make readers self-contained, the updating criteria of $\boldsymbol{W}_k$ on the Stiefel manifold are given below [5]:

$$\tilde{\nabla} L^{(k)}_{\boldsymbol{W}_k} = \nabla L^{(k)}_{\boldsymbol{W}_k} - \nabla L^{(k)}_{\boldsymbol{W}_k} \boldsymbol{W}_k^T \boldsymbol{W}_k, \tag{8}$$

$$\boldsymbol{W}_k^{(t+1)} = \mathcal{R}(\boldsymbol{W}_k^t - \eta \tilde{\nabla} L^{(k)}_{\boldsymbol{W}_k}), \tag{9}$$

where $\tilde{\nabla} L^{(k)}_{\boldsymbol{W}_k}$ is the tangential component to the stiefel manifold, obtained by subtracting the normal component of the Euclidean gradient $\nabla L^{(k)}_{\boldsymbol{W}_k}$ computed by the first term of Eqn.(7), $\mathcal{R}$ represents the retraction operation, $\boldsymbol{W}_k^t$ is the

updated weight at the t-th iteration, and η is the learning rate. For detailed information about the optimization process, please kindly refer to [1].

ExpEig and LogEig Layers: Due to these two layers involve SVD operation, for convenience, a transition layer k^+ is introduced to receive $\boldsymbol{X}_{k-1}$ as input and output $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$, *i.e.*, $\boldsymbol{X}_{k^+} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$. Accordingly, the following chain rule can be obtained:

$$\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} : d\boldsymbol{X}_{k-1} = \frac{\partial L^{(k^+)}}{\partial \boldsymbol{U}} : d\boldsymbol{U} + \frac{\partial L^{(k^+)}}{\partial \boldsymbol{\Sigma}} : d\boldsymbol{\Sigma}. \tag{10}$$

Since $d\boldsymbol{X}_{k-1} = d\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T + \boldsymbol{U}d\boldsymbol{\Sigma}\boldsymbol{U}^T + \boldsymbol{U}\boldsymbol{\Sigma}d\boldsymbol{U}^T$, the differentiations of $d\boldsymbol{U}$ and $d\boldsymbol{\Sigma}$ are given as [5, 6]:

$$d\boldsymbol{U} = 2\boldsymbol{U}(\boldsymbol{\Psi}^T \circ (\boldsymbol{\Sigma}^T\boldsymbol{U}^T d\boldsymbol{X}_{k-1}\boldsymbol{U})_{sym}) \tag{11}$$

$$d\boldsymbol{\Sigma} = (\boldsymbol{U}^T d\boldsymbol{X}_{k-1}\boldsymbol{U})_{diag}, \tag{12}$$

where $\circ$ represents the Hadamard product and $\boldsymbol{C}_{sym} = (\boldsymbol{C}+\boldsymbol{C}^T)/2$. Substituting Eqn.(11) and Eqn.(12) into Eqn.(10), the partial derivative of $L^{(k)}$ with respect to $\boldsymbol{X}_{k-1}$ for the layers of ReEig and LogEig layers can be computed by:

$$\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = 2\boldsymbol{U}(\boldsymbol{\Psi} \circ (\boldsymbol{U}^T\boldsymbol{Q}_1)_{sym})\boldsymbol{U}^T + \boldsymbol{U}\boldsymbol{Q}_2\boldsymbol{U}^T, \tag{13}$$

where $\boldsymbol{Q}_1 = \frac{\partial L^{(k^+)}}{\partial \boldsymbol{U}}$, $\boldsymbol{Q}_2 = (\frac{\partial L^{(k^+)}}{\partial \boldsymbol{\Sigma}})_{diag}$, and $\boldsymbol{\Psi}$ is expressed as:

$$\boldsymbol{\Psi}_{ij} = \begin{cases} \frac{1}{\sigma_i^2 - \sigma_j^2}, & i \neq j, \\ 0, & i = j, \end{cases} \tag{14}$$

where σ_i denotes the i-th ($i = 1 \rightarrow d_{k-1}$) eigenvalue of $\boldsymbol{\Sigma}$.

For the ReEig layer, it receives $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ as its input, and outputs $\boldsymbol{X}_k = \boldsymbol{U}\max(\epsilon\boldsymbol{I}, \boldsymbol{\Sigma})\boldsymbol{U}^T$. Since the variation of $\boldsymbol{X}_k$ is: $d\boldsymbol{X}_k = 2(d\boldsymbol{U}\max(\epsilon\boldsymbol{I}, \boldsymbol{\Sigma})\boldsymbol{U}^T)_{sym} + (\boldsymbol{U}\boldsymbol{G}d\boldsymbol{\Sigma}\boldsymbol{U}^T)_{sym}$, the partial derivatives of $L^{(k^+)}$ w.r.t $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ can be derived by imitating the chain rule of Eqn.(10):

$$\frac{\partial L^{(k^+)}}{\partial \boldsymbol{U}} = 2\left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\right)_{sym}\boldsymbol{U}\max(\epsilon\boldsymbol{I}, \boldsymbol{\Sigma}), \tag{15}$$

$$\frac{\partial L^{(k^+)}}{\partial \boldsymbol{\Sigma}} = \boldsymbol{G}\boldsymbol{U}^T\left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\right)_{sym}\boldsymbol{U}, \tag{16}$$

where $\boldsymbol{G}$ is the gradient of $\max(\epsilon\boldsymbol{I}, \boldsymbol{\Sigma})$: $\boldsymbol{G}_{ii} = 1$ if $\boldsymbol{\Sigma}_{ii} > \epsilon$, and 0 otherwise.

The variation of $\boldsymbol{X}_k$ in the LogEig layer becomes: $d\boldsymbol{X}_k = 2(d\boldsymbol{U}\log(\boldsymbol{\Sigma})\boldsymbol{U}^T)_{sym} + (\boldsymbol{U}\boldsymbol{\Sigma}^{-1}d\boldsymbol{\Sigma}\boldsymbol{U}^T)_{sym}$, and the following two partial derivatives can be obtained:

$$\frac{\partial L^{(k^+)}}{\partial \boldsymbol{U}} = 2\left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\right)_{sym}\boldsymbol{U}\log(\boldsymbol{\Sigma}), \tag{17}$$

$$\frac{\partial L^{(k^+)}}{\partial \boldsymbol{\Sigma}} = \boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T\left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\right)_{sym}\boldsymbol{U}. \tag{18}$$

Reconstruction Error Term: Since the reconstruction error term is defined as (Eqn.(7) of the main manuscript): $\mathcal{L}_2(\boldsymbol{\mathcal{Z}}_i, \hat{\mathbf{H}}_E) = ||\boldsymbol{\mathcal{Z}}_i - \hat{\mathbf{H}}_{\mathrm{E}}||_{\mathrm{F}}^2$, the partial derivatives of $\mathcal{L}_2$ w.r.t $\boldsymbol{\mathcal{Z}}_i$ and $\hat{\mathbf{H}}_{\mathrm{E}}$ can be formulated as:

$$\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\mathcal{Z}}_i} = 2(\boldsymbol{\mathcal{Z}}_i - \hat{\mathbf{H}}_{\mathrm{E}}), \qquad \frac{\partial \mathcal{L}_2}{\partial \hat{\mathbf{H}}_{\mathrm{E}}} = -2(\boldsymbol{\mathcal{Z}}_i - \hat{\mathbf{H}}_{\mathrm{E}}). \tag{19}$$

With these ingredients, the *Riemannian matrix Backpropagation* algorithm can be utilized to train the proposed network. The main implementation details of our DreamNet are summarized into **Algorithm 1**.

2 Ablation Studies

2.1 Ablation Study of the Data Fusion Methods Associated with Shortcut Connection

As mentioned in the main manuscript, we exploit the element-wise addition (EWA) strategy to implement the shortcut connections mainly based on the following two considerations: 1) it introduces neither parameters nor computational complexity; 2) it can make the resulting data points still lie on the SPD manifold. However, the Abelian group operation (AGO) (Definition 3.1 of [2]) seems to be a more appropriate method than EWA for SPD feature fusion. The main reason is that it is faithful to the Riemannian geometry of SPD manifolds and demonstrates strong theoretical and practical benefits in Riemannian data analysis [2]. To demonstrate that using EWA is valid, we compare DreamNet-27-EWA with DreamNet-27-AGO in terms of classification performance and computation time, choosing the FPHA dataset as an example. From Fig. 1(b), we can find that although the classification accuracy of DreamNet-27-AGO is 0.39% higher than that of DreamNet-27-EWA on the FPHA dataset, its superiority in computation time is not as good as DreamNet-27-EWA. In addition, Fig. 1(a) shows that the convergence speed of DreamNet-27-EWA is a bit faster than that of DreamNet-27-AGO. These observations suggest that it is feasible and effective to view EWA as a compromise.

To realize the forward and backward pass of the SPD matrices in AGO, we introduce three auxiliary layers for the proposed DreamNet. In particular, the input SPD matrices are first mapped to the logarithmic domain using the LogEig operation. As the mapped space conforms to the Euclidean geometry, the EWA is utilized to perform feature fusion. Finally, the ExpEig layer is applied to embed the fused data back onto the SPD manifold via the matrix exponential mapping, f_e. For simplicity, we formulate the ExpEig operation as: $\boldsymbol{X}_k = f_e(\boldsymbol{X}_{k-1}) = \boldsymbol{U}\exp(\boldsymbol{\Sigma})\boldsymbol{U}^T$. Wherein, $\boldsymbol{X}_{k-1} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$ denotes the eigenvalue decomposition, and $\exp(\boldsymbol{\Sigma})$ is a diagonal matrix composed of the eigenvalue exponentials. Due to the variation of $\boldsymbol{X}_k$ is: $d\boldsymbol{X}_k = 2(d\boldsymbol{U}\exp(\boldsymbol{\Sigma})\boldsymbol{U}^T)_{sym} + (\boldsymbol{U}\exp(\boldsymbol{\Sigma})d\boldsymbol{\Sigma}\boldsymbol{U}^T)_{sym}$, the following partial derivatives can be derived by referring to the computing

Algorithm 1 Deep Riemannian Manifold Network for SPD Matrix Learning

Input: Training set $\mathfrak{S}$, training epochs $\mathcal{T}$, learning rate η, batch size B, number of B of a given dataset $\mathfrak{b}$, threshold ϵ, number of layers K, and trade-off parameter λ.

Initialization:

for $i \leftarrow 1$ **to** N ***do***:
  Computing $\boldsymbol{X}_i$ from $\boldsymbol{S}_i$ via $\boldsymbol{X}_i = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}(\boldsymbol{s}_j - \boldsymbol{\mu}_i)(\boldsymbol{s}_j - \boldsymbol{\mu}_i)^T$
  Using $\boldsymbol{X}_i \leftarrow \boldsymbol{X}_i + \alpha \boldsymbol{I}_d$ to ensure that $\boldsymbol{X}_i$ is SPD.
end
for $k \leftarrow 1$ **to** K ***do***:
  The SVD operation is applied to initialize $\boldsymbol{W}_k$ of the BiMap layer.
end

Training:

for $r \leftarrow 1$ **to** $\mathcal{T}$ ***do***:
  Randomly and equally divide the new training set $\boldsymbol{\mathcal{X}}$ into $\mathfrak{b}$ batches: $\boldsymbol{\mathcal{X}} = \text{random}[\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, ..., \boldsymbol{\mathcal{X}}_{\mathfrak{b}}]$.
  for $h \leftarrow 1$ **to** $\mathfrak{b}$ ***do***:
    for $k \leftarrow 1$ **to** K ***do***:

Forward Pass:
case 'BiMap layer'
$\boldsymbol{X}_k = f_b^{(k)}(\boldsymbol{W}_k, \boldsymbol{X}_{k-1}) = \boldsymbol{W}_k^T \boldsymbol{X}_{k-1} \boldsymbol{W}_k$.
case 'ReEig layer'
$\boldsymbol{X}_k = f_r^{(k)}(\boldsymbol{X}_{k-1}) = \boldsymbol{U}\max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma})\boldsymbol{U}^T$.
case 'LogEig layer'
$\boldsymbol{X}_k = f_l^{(k)}(\boldsymbol{X}_{k-1}) = \boldsymbol{U}\log(\boldsymbol{\Sigma})\boldsymbol{U}^T$.
case 'Cross-Entropy loss'
$\mathcal{L}_e = -\sum_{i=1}^{N}\sum_{t=1}^{c} \tau(l_i, t) \times \log(P(t|\boldsymbol{X}_i))$.
case 'Reconstruction error term'
$\mathcal{L}_2(\boldsymbol{\mathcal{Z}}_i, \hat{\mathbf{H}}_E) = ||\boldsymbol{\mathcal{Z}}_i - \hat{\mathbf{H}}_{\mathrm{E}}||_{\mathrm{F}}^2$.

Backward Pass:
case 'BiMap layer'
Updating $\boldsymbol{W}_k$ using Eqn.(8), Eqn.(9), and the first term of Eqn.(7).
case 'ReEig layer'
Computing $\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}}$ using Eqn.(13), Eqn.(15), and Eqn.(16).
case 'LogEig layer'
Computing $\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}}$ using Eqn.(13), Eqn.(17), and Eqn.(18).
case 'Cross-Entropy layer'
$\frac{\partial L^{(K)}}{\partial \boldsymbol{X}_{K-1}} = P(t|\boldsymbol{X}_i) - 1$.
case 'Reconstruction error term'
$\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\mathcal{Z}}_i} = 2(\boldsymbol{\mathcal{Z}}_i - \hat{\mathbf{H}}_{\mathrm{E}}), \quad \frac{\partial \mathcal{L}_2}{\partial \hat{\mathbf{H}}_{\mathrm{E}}} = -2(\boldsymbol{\mathcal{Z}}_i - \hat{\mathbf{H}}_{\mathrm{E}})$..

    end
  end
end

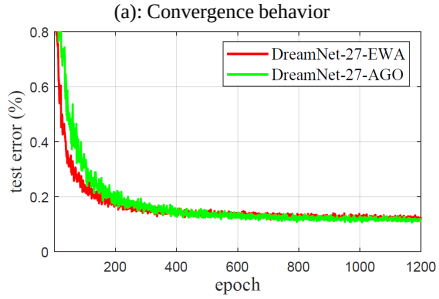
Output: $\boldsymbol{\mathcal{W}} = \{\boldsymbol{W}_K, \boldsymbol{W}_{K-1}, ..., \boldsymbol{W}_2, \boldsymbol{W}_1\}$.

process of the ReEig and LogEig layers:

$$\frac{\partial L^{(k^+)}}{\partial \boldsymbol{U}} = 2\left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\right)_{sym} \boldsymbol{U}\exp(\boldsymbol{\Sigma}), \tag{20}$$

$$\frac{\partial L^{(k^+)}}{\partial \boldsymbol{\Sigma}} = \exp(\boldsymbol{\Sigma})\boldsymbol{U}^T\left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\right)_{sym} \boldsymbol{U}. \tag{21}$$

(a): Convergence behavior

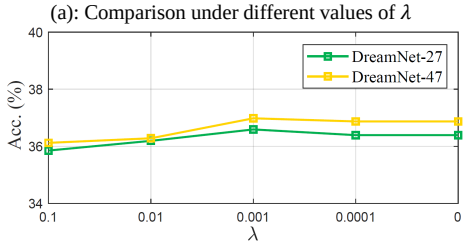


(b): Comparison under different data fusion methods

Methods	Acc. (%)	s/epoch
DreamNet-27-EWA	87.78	**2.60**
DreamNet-27-AGO	**88.17**	3.52

Fig. 1. Comparison of DreamNet-27-EWA and DreamNet-27-AGO on the FPHA dataset.

(a): Comparison under different values of λ



(b): Comparison under multiple reconstruction error terms

Metrics	DreamNet-27	DreamNet-47
Acc. (%)	36.84	37.17
s/epoch	33.64	50.46
#h-params	3	5

Fig. 2. Comparison of DreamNet-27/47 on the AFEW dataset.

With the above three auxiliary layers, according to Eqn.(20), Eqn.(21), and the backpropagation algorithm introduced in Section 1, the forward and backward pass of the SPD matrices in AGO can be achieved.

2.2 Ablation Study of the Role of the Trade-off Parameter λ and the Reconstruction Error Term

To measure the effectiveness of the reconstruction error term (RT) defined in Eqn.(7) of the main article, we select the AFEW dataset as an example to conduct the experiments. From Fig. 2(a), we have some interesting findings. Firstly, the performance of DreamNet-47 is better than that of DreamNet-27 in almost all cases, again demonstrating the benefits of increasing the network depth. Secondly, the classification accuracy of 27/47-layer DreamNets shows an increasing trend first and then decreasing. This is mainly attributed to the fact that the loss function of our DreamNet has two goals: 1) supervising the network to generate deep representations with richly structured semantic information; 2) enabling the network to reconstruct the input data better. Note that a large value of λ would make the network focus on deep reconstruction learning, which is not conducive to the training of effective classifiers. However, when λ takes values in the range of $\{0, 0.0001, 0.001\}$, the performance of 27/47-layer DreamNets is slightly improved. This supports our assertion that the RT helps to fine-tune the classification performance. In any case, our method is not very sensitive to this trade-off parameter.

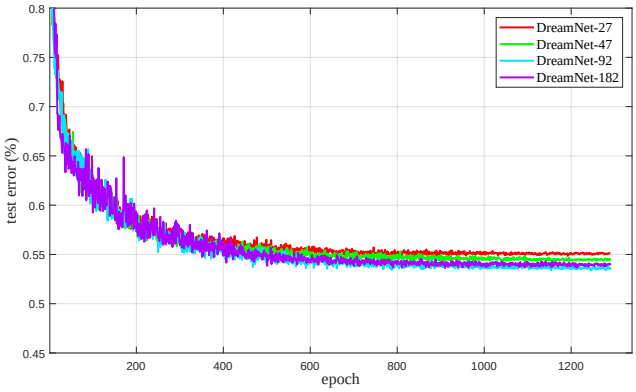


Fig. 3. The classification error of 27/47/92/182-layer DreamNets versus the number of training epochs on the UAV-Human dataset.

Based on the above results, we connected all the input layers of the remaining E-1 RAEs to the final layer of SRAE to include more RTs in DreamNet. Fig. 2(b) shows that this operation results in a relatively higher accuracy of 27/47-layer DreamNets, compared to Table 2 of the main paper. The main reason is that the multiple reconstruction learning helps high-level features capture the pivotal geometric structures conveyed by the raw data, thus facilitating effective classification. Meanwhile, the training time has not increased significantly. Nevertheless, as the number of hyperparameters (#h-params) of our network increases, we suggest using just one RT shown in Fig.2 of the main paper as a compromise. Considering the computational burden of DreamNet-92, we have not explored it here. All in all, these experimental evidences confirm the efficacy of the RT in guiding the proposed model to solve the degradation problem and to yield useful deep features.

2.3 Investigating the Classification Performance of a 182-layer DreamNet

In this subsection, we explore an aggressively deep Riemannian network of over 180 layers. We set E = 20 (the number of stacked RAEs) that leads to a 182-layer DreamNet, which is trained with the Riemannian matrix backpropagation algorithm introduced above. According to Fig. 3, we can note that this 182-layer DreamNet exhibits no optimization difficulty, and its test accuracy (46.03%) is still higher than that of the competitors listed in Table 5 of the main paper.

In spite of this, the test result of DreamNet-182 is somewhat lower than that of DreamNet-92, although both have similar training error. We believe that one of the reasons for this phenomenon is overfitting. This 182-layer DreamNet may be a bit large (0.63M) for the UAV-Human dataset. For the basic reason of the inferiority of DreamNet-182 compared to DreamNet-92, we argue that it is due to the loss of some pivotal structural information embedded in the

input SPD matrices during multi-stage SRAE transformation. Since the reconstruction loss is introduced as a learning objective, the solution for the early stages of SRAE network will tend to diagonalise the respective SPD matrices (the visualized feature maps in Fig. 4(a), Fig. 4(b), and Fig. 4(c) of the main article support this notion experimentally). This follows from the well known fact that the optimal solution to the problem of minimising the signal (video clip, image set, or point cloud) approximation error using a reduced number of basis functions are the eigenvectors of the signal covariance matrix associated with the largest eigenvalues. The SPD matrix reconstruction problem is a proxy to the signal approximation problem. However, once an off-diagonal element of an SPD matrix becomes zero, it will never contribute to the generation of the lower-dimensional SPD matrices. This will considerably reduce the number of variables involved in the generation of these matrices. Although the used eigenvalue regularization (ReEig layer) allows the off-diagonal elements to contribute to the learning process, it will gradually exacerbate the amount of adjustment to the input SPD matrices with increasing the number E of the stacked RAEs. In this scenario, the more transformation stages of SRAE, the more distortion of the original data structure will be. This is also the fundamental reason why the classification score of DreamNet-92 on the FPHA dataset (shown in Table 3 of the main manuscript) is not as good as that of DreamNet-47.

3 Comparison with the Original Residual Learning Framework

According to the analysis in the main paper, we can speculate that the Riemannian residual functions learnt in this paper may not approach a zero mapping, which is different from the hypothesis verified in ResNet [4], *i.e.*, the residual functions might be generally close to zero. The basic reason is that the semi-orthogonality of $\boldsymbol{W}_k$ makes it impossible for the Riemannian solver to drive the weights of multiple layers towards zero. The incremental information conveyed by the visualized feature maps in Fig. 4 of the main paper confirms our speculation, experimentally. Other differences include: 1) both the inputs and outputs of our model are structured SPD matrices, rather than the image features of ResNet. In other words, the suggested architecture is strictly defined on the Riemannian manifolds, other than the Euclidean space; 2) the parameter optimization of our DreamNet is realized by exploiting the stochastic gradient descent (SGD) setting on the Stiefel manifolds with the Riemannian matrix backpropagation (Section 1) for preserving the Riemannian geometry of SPD data points, while the Euclidean SGD-based backpropagation is used in ResNet.

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization algorithms on matrix manifolds. Princeton University Press (2009)

2. Arsigny, V., Fillard, P., Pennec, X., Ayache, N.: Geometric means in a novel vector space structure on symmetric positive-definite matrices. SIAM J. Matrix Anal. Appl. pp. 328–347 (2007)
3. Edelman, A., Arias, T.A., Smith, S.T.: The geometry of algorithms with orthogonality constraints. SIAM J. Matrix Anal. Appl. pp. 303–353 (1998)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR pp. 770–778 (2016)
5. Huang, Z., Van Gool, L.: A riemannian network for spd matrix learning. In: AAAI pp. 2036–2042 (2017)
6. Ionescu, C., Vantzos, O., Sminchisescu, C.: Training deep networks with structured layers by matrix backpropagation. arXiv preprint arXiv:1509.07838 (2015)