Adaptive Log-Euclidean Metrics for SPD Matrix Learning

Ziheng Chen, Yue Song*, Tianyang Xu, Zhiwu Huang, Xiao-Jun Wu, and Nicu Sebe

***Abstract*—Symmetric Positive Definite (SPD) matrices have received wide attention in machine learning due to their intrinsic capacity to encode underlying structural correlation in data. Many successful Riemannian metrics have been proposed to reflect the non-Euclidean geometry of SPD manifolds. However, most existing metric tensors are fixed, which might lead to sub-optimal performance for SPD matrix learning, especially for deep SPD neural networks. To remedy this limitation, we leverage the commonly encountered pullback techniques and propose Adaptive Log-Euclidean Metrics (ALEMs), which extend the widely used Log-Euclidean Metric (LEM). Compared with the previous Riemannian metrics, our metrics contain learnable parameters, which can better adapt to the complex dynamics of Riemannian neural networks with minor extra computations. We also present a complete theoretical analysis to support our ALEMs, including algebraic and Riemannian properties. The experimental and theoretical results demonstrate the merit of the proposed metrics in improving the performance of SPD neural networks. The efficacy of our metrics is further showcased on a set of recently developed Riemannian building blocks, including Riemannian batch normalization, Riemannian Residual blocks, and Riemannian classifiers.**

***Index Terms*—Riemannian geometry, SPD manifolds**

I. Introduction

THe Symmetric Positive Definite (SPD) matrices are ubiquitous in statistics, supporting a diversity of scientific areas, such as medical imaging [1]–[3] , signal processing [4]–[7], elasticity [8], [9], question answering [10], [11], graph and node classification [12], and computer vision [13]–[23]. Despite the ability to capture data variations, SPD matrices cannot simply interact as points in the Euclidean space, which becomes the main challenge in practice. To guarantee the manifoldness, several Riemannian metrics have been proposed, including Affine-Invariant Metric (AIM) [24], Log-Euclidean Metric (LEM) [25], and Log-Cholesky Metric (LCM) [26], to name a few. Equipped with these metrics, many Euclidean methods could be generalized into the domain of the Riemannian manifold [27]–[31]. It is essential to clarify that there are also some metric learning methods in SPD manifolds [28], [30]. However, the metrics these methods learned are distance functions induced by existing Riemannian metrics. In contrast, this paper focuses on Riemannian metrics, which are more fundamental than the metric learning methods mentioned above.

Recently, inspired by the vivid progress of deep learning [32]–[34], several deep networks were developed on the SPD manifold [1], [3], [5], [7], [11], [13], [19], [22], [23], [35]–[40]. Although different network structures are designed, the theoretical foundations of these methods are all built upon Riemannian metrics on the SPD manifold. Therefore, the design of the Riemannian metric is significantly important for the efficacy of the learning algorithms. However, most metric tensors in the existing popular Riemannian metrics on the SPD manifold are fixed, which could undermine the expressibility of the associated geometry. After analyzing several existing Riemannian metrics on SPD manifolds, we find that the pullback is a commonly used tool, which can be intuitively viewed as a bijection preserving Riemannian properties. For instance, [41] explained AIM as the pullback metric from a left-invariant metric on the Cholesky manifold. In [42], the authors generalized LEM by the pullback of the vanilla LEM. In [26], the authors proposed LCM by the pullback from the Cholesky manifold.

Inspired by the above observations, we leverage pullback techniques to introduce adaptive Riemannian metrics in this paper. In particular, we first show that several Riemannian metrics on SPD manifolds, including LEM, LCM, and their generalizations, can be explained as pullback metrics from the standard Euclidean space. We refer to these metrics as Pullback Euclidean Metrics (PEMs). Then, we propose a general framework for characterizing the properties of PEMs. Our framework can explain the widely used LEM [25] and LCM [26]. We focus on LEM on SPD manifolds and extend it into Adaptive Log-Euclidean Metrics (ALEMs). Besides, we present a complete theoretical discussion on the properties of ALEMs. Compared with the existing Riemannian metrics, our metrics are adjustable, adapting to the characteristics of the datasets. To the best of our knowledge, our work is the **first** to integrate learnable Riemannian metrics into Riemannian deep networks. The effectiveness of our metrics is demonstrated by experiments as well as the applications to recently developed Riemannian building blocks, including Riemannian batch normalization [22], Riemannian residual blocks [43], and Riemannian classifiers [39]. Drawing on

This work was partly supported by the MUR PNRR project FAIR (PE00000013) funded by the NextGenerationEU, the EU Horizon project ELIAS (No. 101120237), and a donation from Cisco. The authors also gratefully acknowledge the financial support from the China Scholarship Council (CSC).

Ziheng Chen, Yue Song, and Nicu Sebe are with the Department of Information Engineering and Computer Science, University of Trento, Trento, Italy. E-mail: ziheng_ch@163.com, songyue19960927@gmail.com, niculae.sebe@unitn.it. (Corresponding author: Yue Song)

Tianyang Xu and Xiao-Jun Wu are with the School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, China. E-mail: tianyang.xu; wu_xiaojun@jiangnan.edu.cn.

Zhiwu Huang is with the School of Electronics and Computer Science, University of Southampton, Southampton, U.K. E-mail: zhiwu.huang@soton.ac.uk.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material includes preliminaries, implementation details, and proofs.

This article has been accepted for publication in IEEE Transactions on Image Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIP.2024.3451930

this, our **contributions** are summarized as follows: **(a)** We reveal the connection of two popular Riemannian metrics (LEM and LCM) by the pullback technique and propose a general framework for PEMs; **(b)** Based on our framework, we propose specific ALEMs on SPD manifolds and conduct comprehensive analyses in terms of the algebraic, analytic, and geometric properties; **(c)** Extensive experiments on widely used SPD learning benchmarks demonstrate that our metrics exhibit consistent performance gain across datasets.

The rest of the paper is organized as follows: Sec. II reviews some essential backgrounds of differential geometry and the geometry of SPD manifolds. Sec. III-A rethinks the existing LEM and LCM from the perspective of pullback metrics. Sec. III-B provides a detailed discussion on PEMs. Secs. III-C and III-D extend the existing LEM into ALEMs based on the framework of PEMs. Sec. IV extensively analyzes the geometric properties of ALEM. Sec. V presents the application of our ALEM into SPD neural networks. Sec. VI discusses the gradient computations and parameter updates involved in our methods. Sec. VII validates our metric on three datasets. Sec. VIII further applies our ALEM to re-design other Riemannian blocks. Sec. IX discusses the limitations of this work, and Sec. X concludes this paper. For better representation, all proofs are left in the supplement.

II. Preliminaries

This section reviews some basic notations of differential geometry and the geometry of SPD manifolds. For a more detailed review, please refer to the supplementary.

We first briefly review the idea of pullback, which is a common trick in geometry to study metrics.

Definition II.1 (Pullback Metrics)**.** Suppose $\mathcal{M}, \mathcal{N}$ are smooth manifolds, g is a Riemannian metric on $\mathcal{N}$, and $f: \mathcal{M} \rightarrow \mathcal{N}$ is smooth. Then the pullback of the tensor field g by f is defined point-wisely,

$$(f^* g)_p(V_1, V_2) = g_{f(p)}(f_{*,p}(V_1), f_{*,p}(V_2)), \tag{1}$$

where $p \in \mathcal{M}$, $f_{*,p}(\cdot)$ is the differential map of f at p, and $V_i \in T_p\mathcal{M}$. If f^*g is positive definite, it is a Riemannian metric on $\mathcal{M}$, which is called the pullback metric defined by f.

The most common pullback metrics are the ones induced by diffeomorphism, *i.e.,* when f is a diffeomorphism.

Next, we review the basic geometry of SPD manifolds. We denote the set of $n \times n$ SPD matrices as $\mathcal{S}^n_{++}$, the set of $n \times n$ symmetric matrices as $\mathcal{S}^n$, and all the Cholesky matrices (lower triangular matrices with positive diagonal elements) as $\mathcal{L}^n_+$. As shown in the previous literature [25], [26], $\mathcal{S}^n_{++}$ and $\mathcal{L}^n_+$ form an SPD manifold and a Cholesky manifold, respectively. For an SPD matrix S, the matrix logarithm $\operatorname{mln}(\cdot): \mathcal{S}^n_{++} \rightarrow \mathcal{S}^n$ is defined as

$$\operatorname{mln}(S) = U \ln(\Sigma) U^\top, \tag{2}$$

where $S = U\Sigma U^\top$ is the eigendecomposition, and $\ln(\cdot)$ is the diagonal natural logarithm.

In [25], LEM on $\mathcal{S}^n_{++}$ is introduced by Lie group translation. The standard LEM is further generalized into two-parameter families of $\mathrm{O}(n)$-invariant metrics [42], namely (a,b)-LEM, by $\mathrm{O}(n)$-invariant inner product on $\mathcal{S}^n$

$$\langle X, X\rangle^{(a,b)} = a\|X\|_{\mathrm{F}} + b\,\mathrm{tr}(X)^2, \forall X \in \mathcal{S}^n, \tag{3}$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius inner product, and $(a,b) \in \mathbf{ST} = \{(a,b) \in \mathbb{R}^2 \mid \min(a, a+nb) > 0\}$. In [26], LCM is derived on $\mathcal{S}^n_{++}$ from the Cholesky manifold $\mathcal{L}^n_+$ by Cholesky decomposition. We denote (a,b)-LEM and LCM as $g^{(a,b)\text{-LE}}$ and g^{LC}, respectively. For an SPD matrix P and a tangent vector V in the tangent space $T_P\mathcal{S}^n_{++}$ at P, $g^{(a,b)\text{-LE}}$ is defined as

$$g_P^{(a,b)\text{-LE}}(V,V) = a\|\operatorname{mln}_{*,P}(V)\|^2_{\mathrm{F}} + b\,\mathrm{tr}(P^{-1}V)^2, \tag{4}$$

where $\operatorname{mln}_{*,P}$ is the differential map of matrix logarithm at $P \in \mathcal{S}^n_{++}$, V is a tangent vector in the tangent space $T_P\mathcal{S}^n_{++}$ at P, $(a,b) \in \mathbf{ST}$. Note that (a,b)-LEM incorporates the standard LEM when $(a,b) = (1,0)$.

For $L \in \mathcal{L}^n_+$ and $W \in T_L\mathcal{L}^n_+$, the metric on the Cholesky manifold [26] is defined as

$$g^{\mathrm{C}}_L(W,W) = \sum_{i>j} W_{ij}W_{ij} + \sum_{j=1}^{n} W_{jj}W_{jj}L_{jj}^{-2}, \tag{5}$$

The LCM is the pullback metric by the Cholesky decomposition $\mathscr{L}$ from g^{C} [26]:

$$g^{\mathrm{LC}} = \mathscr{L}^* g^{\mathrm{C}}. \tag{6}$$

III. Adaptive Log-Euclidean Metrics

As mentioned in Sec. I, pullbacks are ubiquitous for studying Riemannian metrics on SPD manifolds. In this section, we further show that both (a,b)-LEM and LCM are pullback metrics from the Euclidean space. Inspired by this observation, we present a general framework for characterizing PEMs. Then, we focus on generalizing LEM.

A. Rethinking (a,b)-LEM and LCM

Among the existing Riemannian metrics on the SPD manifold, LEM is popular in many applications, given its closed form for the Fréchet mean and clear vector space & Lie group structures. In addition, the nascent LCM, gaining increasing attention, also shares similar properties with LEM. LEM is derived from the Lie group translation [25], while LCM is derived by the pullback from $\mathcal{L}^n_+$ [26]. Besides, (a,b)-LEM is obtained by the pullback of LEM. However, theoretically, the mathematical logic beneath their derivation can be the same. We denote $\mathcal{L}^n$ as the Euclidean space of $n \times n$ lower triangular matrices. We define $\phi_{cln}: \mathcal{S}^n_{++} \rightarrow \mathcal{L}^n$ as

$$\phi_{cln}(P) = \lfloor L \rfloor + \ln(\mathbb{D}(L)), \tag{7}$$

where L is the Cholesky factor of the SPD matrix P, $\lfloor L \rfloor$ is the strictly lower part of L, and $\mathbb{D}(L)$ is a diagonal matrix with diagonal elements of L. Then, we have the following theorem.

Theorem III.1. *(a,b)-LEM is the pullback metric from the Euclidean space of $\mathcal{S}^n$ with an $\mathrm{O}(n)$-invariant inner product $\langle,\rangle^{(a,b)}$ by matrix logarithm. Specifically, the standard LEM is* *the pullback metric from the Euclidean space of $\mathcal{S}^n$ with the standard Frobenius inner product by matrix logarithm. LCM is the pullback metric from $\mathcal{L}^n$ with the Frobenius inner product by ϕ_{cln}.*

As n-dimensional Euclidean spaces are naturally isometric, it can be directly obtained that both (a,b)-LEM and LCM are pulled back from the standard Euclidean space $\mathcal{S}^n$.

Corollary III.2. *(a,b)-LEM and LCM are pullback metrics from $\mathcal{S}^n$ with standard Frobenius inner product.*

B. PEMs on SPD Manifolds

In Sec. III-A, we have shown how LEM is derived from matrix logarithm. Besides, as shown in [25], operations in Lie group and linear space on $\mathcal{S}^n_{++}$ are also induced from matrix logarithm. Now, let us explain the underlying mechanism in detail. A matrix logarithm is a diffeomorphism (a smooth bijection with a smooth inverse). The property of bijection offers the possibility of transferring algebraic structures from $\mathcal{S}^n$ into $\mathcal{S}^n_{++}$. The smoothness of matrix logarithm and its inverse suggest that smooth structures can be transferred into $\mathcal{S}^n_{++}$, like the Lie group and Riemannian metric. More generally, given an arbitrary diffeomorphism $\phi : \mathcal{S}^n_{++} \to \mathcal{S}^n$, it suffices to pull various properties from the Euclidean space back to the SPD manifold $\mathcal{S}^n_{++}$ by ϕ as well. Besides, the computation of the induced operators in $\mathcal{S}^n_{++}$ by ϕ is usually simple.

Lemma III.3. *Let $S_1, S_2, S \in \mathcal{S}^n_{++}, V_i \in T_S\mathcal{S}^n_{++}, k \in \mathbb{R}$ and g^{E} be the Frobenius inner product in $\mathcal{S}^n$. $\phi : \mathcal{S}^n_{++} \to \mathcal{S}^n$ is a diffeomorphism, and $\phi_{*,S}$ is the differential at S. We define the following operations,*

$$\text{Elements Addition: } S_1 \odot_\phi S_2 = \phi^{-1}(\phi(S_1) + \phi(S_2)), \tag{8}$$

$$\text{Scalar Product: } k \circledast_\phi S_2 = \phi^{-1}(k\phi(S_2)), \tag{9}$$

$$\text{Inner Product: } \langle S_1, S_2 \rangle_\phi = \langle \phi(S_1), \phi(S_2) \rangle, \tag{10}$$

$$\text{Riemannian Metric: } g^\phi = \phi^* g^{\mathrm{E}}, \tag{11}$$

Then, we have the following conclusions:

1) *$\{\mathcal{S}^n_{++}, \odot_\phi, \circledast_\phi, \langle \cdot,\cdot \rangle_\phi\}$ is a Hilbert space over $\mathbb{R}$.*
2) *$\{\mathcal{S}^n_{++}, \odot_\phi\}$ is an Abelian Lie group. $\{\mathcal{S}^n_{++}, g^\phi\}$ is a Riemannian manifold. The associated Riemannian operators are as follows*

$$d^\phi(S_1, S_2) = \|\phi(S_1) - \phi(S_2)\|_{\mathrm{F}}, \tag{12}$$

$$\mathrm{Exp}_{S_1} V = \phi^{-1}(\phi(S_1) + \phi_{*,S_1} V), \tag{13}$$

$$\mathrm{Log}_{S_1} S_2 = \phi^{-1}_{*,\phi(S_1)}(\phi(S_2) - \phi(S_1)), \tag{14}$$

$$\Gamma_{S_1 \to S_2}(V) = \phi^{-1}_{*,\phi(S_2)} \circ \phi_{*,S_1}(V), \tag{15}$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, $V \in T_{S_1}\mathcal{S}^n_{++}$ is a tangent vector, Exp_{S_1}, Log_{S_1} and $\Gamma_{S_1 \to S_2}$ are Riemannian exponential map at S_1, logarithmic map at S_1 and parallel transportation along the geodesics connecting S_1 and S_2 respectively, and ϕ^{-1}_* is the differential maps of ϕ^{-1}. Then g^ϕ is a bi-invariant metric, named Pullback Euclidean Metric (PEM) by ϕ.
3) *ϕ is an isomorphism: (a) a linear isomorphism preserving the inner product; (b) a Lie group isomorphism; (3) a Riemannian isometry.*

In fact, (a,b)-LEM and LCM are special cases of Lem. III.3, and so do linear space & Lie group in [25] and Lie group in [26]. In addition, neither [25] nor [26] reveals the Hilbert space structures in $\mathcal{S}^n_{++}$.

C. Adaptive Log-Euclidean Metrics

The key of Lem. III.3 lies in the diffeomorphism ϕ. If we have a proper ϕ, Riemannian metrics on SPD manifolds can be induced. In the following, we will present our mappings and then discuss the induced metrics.

As an eigenvalues function, the matrix logarithm in Eq. (2) is reduced into a scalar logarithm, which is a diffeomorphism between $\mathbb{R}_+$ and $\mathbb{R}$. Following this hint, the eigenvalues-based diffeomorphism between $\mathcal{S}^n_{++}$ and $\mathcal{S}^n$ is reduced to scalar diffeomorphism between $\mathbb{R}_+$ and $\mathbb{R}$. A very natural idea is to substitute the natural logarithm with scalar logarithms with arbitrary proper bases. In particular, we can define a general diagonal logarithm $\log(\cdot)$ as

$$\log_\alpha(X) = \mathrm{diag}(\log_{a_1}^{x_{11}}, \log_{a_2}^{x_{22}}, \cdots, \log_{a_n}^{x_{nn}}), \tag{16}$$

where $\alpha = (a_1, a_2, \cdots, a_n) \in \mathbb{R}^n_+ \setminus \{(1,1,\cdots,1)\}$ is the base vector, $\mathrm{diag}(\cdot)$ is the diagonalization operator, and X is an $n \times n$ diagonal matrix. By abuse of notation, we denote $\log_\alpha(\cdot)$ as $\log(\cdot)$ for a general diagonal logarithm, and $\log_a^{(\cdot)}$ as $\log^{(\cdot)}$ for a general scalar logarithm. Specially, $a_1 = \cdots = a_n = e \Rightarrow \log(\cdot) = \ln(\cdot)$. Together with eigendecomposition, a general matrix logarithm is:

$$\mathrm{mlog}(S) = U \log_\alpha(\Sigma) U^\top, \tag{17}$$

where $S = U\Sigma U^\top$ is the eigendecomposition. As a special case, when $\alpha = (e, e, \cdots, e)$, $\mathrm{mlog} = \mathrm{mln}$. Similar to the scalar logarithm, we have the following proposition.

Proposition III.4 (Diffeomorphism). *mlog is a diffeomorphism, a smooth bijection with a smooth inverse $\mathrm{mlog}^{-1}(\cdot) : \mathcal{S}^n \to \mathcal{S}^n_{++}$ defined as*

$$\mathrm{mlog}^{-1}(X) = \phi_{\mathrm{ma}}(X) = U\boldsymbol{\alpha}(\Sigma)U^\top, \tag{18}$$

where $\boldsymbol{\alpha}(\Sigma) = \mathrm{diag}(a_1^{\Sigma_{11}}, a_2^{\Sigma_{22}}, \cdots, a_n^{\Sigma_{nn}})$ is a diagonal exponentiation.

Remark III.5. Note that $\mathrm{mlog}(\cdot)$ should be more precisely understood as an arbitrary one from the following family

$$\{\mathrm{mlog}^\alpha | \alpha = (a_1, \cdots, a_n) \in \mathbb{R}^n_+ \setminus \{(1,\cdots,1)\}\}. \tag{19}$$

By abuse of notation, we will simply use $\mathrm{mlog}(\cdot)$. Besides, there could be some ambiguity in Eq. (17) under different arrangements of eigenvalues and eigenvectors. In fact, there is a correspondence between scalar $\log_{a_i}$ and eigenvalues & eigenvectors. Please refer to Supp. B-A for more details.

Since mlog is a diffeomorphism from $\mathcal{S}^n_{++}$ onto $\mathcal{S}^n$, all the results in Lem. III.3 hold true.

Theorem III.6. *Following the notations in Lem. III.3, we define $\odot_{mlog}, \circledast_{mlog}, \langle\cdot,\cdot\rangle_{mlog}$, and g^{mlog} as Eq. (8)-Eq. (11). Then, we have the following conclusions:*

1) *$\{\mathcal{S}^n_{++}, \odot_{mlog}, \circledast_{mlog}, \langle\cdot,\cdot\rangle_{mlog}\}$ is a Hilbert space over $\mathbb{R}$.*

2) *$\{\mathcal{S}^n_{++}, \odot_{mlog}\}$ is an Abelian Lie group. g^{mlog} is a Riemannian metric over $\mathcal{S}^n_{++}$. We call this metric Adaptive Log-Euclidean Metric (ALEM) and denote g^{mlog} as g^{ALE}. The associated Riemannian operators are as follows*

$$d^{\mathrm{ALE}}(S_1, S_2) = \| \operatorname{mlog}(S_1) - \operatorname{mlog}(S_2)\|_{\mathrm{F}}, \tag{20}$$

$$\operatorname{Exp}_{S_1} V = \phi_{\mathrm{ma}}(\operatorname{mlog}(S_1) + \operatorname{mlog}_{*,S_1} V), \tag{21}$$

$$\operatorname{Log}_{S_1} S_2 = \phi_{ma*,X_1}(\operatorname{mlog}(S_2) - \operatorname{mlog}(S_1)), \tag{22}$$

$$\Gamma_{S_1 \rightarrow S_2}(V) = \phi_{ma*,X_2} \circ \operatorname{mlog}_{*,S_1}(V), \tag{23}$$

where $X_i = \operatorname{mlog}(S_i) \in \mathcal{S}^n$ for $i = 1, 2$.

3) mlog *is an isomorphism: (a) a linear isomorphism preserving the inner product; (b) a Lie group isomorphism; (3) a Riemannian isometry.*

Remark III.7. Obviously, ALEM would vary with different mlog. We thus use the plural to describe our metrics. Besides, our metrics could be learnable. This is why we call them adaptive metrics.

Similar with (a, b)-LEM, we also can define (a, b)-ALEM as the pullback metric of $\mathrm{O}(n)$-invariant inner product:

$$g^{(a,b)\text{-ALE}} = \operatorname{mlog}^* g^{(a,b)\text{-E}}, \tag{24}$$

where we denote the $\mathrm{O}(n)$-invariant inner product $\langle , \rangle^{(a,b)}$ as $g^{(a,b)\text{-E}}$. $g^{(a,b)\text{-ALE}}$ also share the properties presented in Thm. III.6. Nevertheless, this paper focuses on $(a, b) = (1, 0)$.

D. Differentials of General Logarithms

Eq. (21)-Eq. (23) require the differential maps of mlog and ϕ_{ma}. This subsection introduces the concrete formulae of the associated differential maps.

Proposition III.8 (Differentials)**.** *For a tangent vector $V \in T_S\mathcal{S}^n_{++}$, the differential $\operatorname{mlog}_{*,S} : T_S\mathcal{S}^n_{++} \rightarrow T_{\operatorname{mlog}(S)}\mathcal{S}^n$ of* mlog *at $S \in \mathcal{S}^n_{++}$ is given by*

$$\operatorname{mlog}_{*,S}(V) = Q + Q^\top + W, \tag{25}$$

where $Q = D_U \log(\Sigma) U^\top$,

$$D_U = \begin{pmatrix} (\sigma_1 I - S)^+ V u_1 & \cdots & (\sigma_n I - S)^+ V u_n \end{pmatrix},$$

$$W = U \operatorname{diag}(\frac{u_1^\top V u_1}{\sigma_1 \ln a_1}, \cdots, \frac{u_n^\top V u_n}{\sigma_n \ln a_n}) U^\top,$$

$()^+$ is the Moore–Penrose inverse, $u_1, \cdots, u_n$ are orthonormal eigenvectors of S, and the associated eigenvalues are $\sigma_1, \cdots, \sigma_n$.

Symmetrically, for a tangent vector $\widetilde{V} \in T_X\mathcal{S}^n$, the differential $\phi_{ma*,X} : T_X\mathcal{S}^n \rightarrow T_{\phi_{\mathrm{ma}}(X)}\mathcal{S}^n_{++}$ of ϕ_{ma} at $X \in \mathcal{S}^n$ is given by

$$\phi_{ma*,X}(\widetilde{V}) = \widetilde{Q} + \widetilde{Q}^\top + \widetilde{W}, \tag{26}$$

where $S = \widetilde{U}\widetilde{\Sigma}\widetilde{U}^\top$ is the eigendecomposition, $D_{\widetilde{U}}$ is defined similarly, $\widetilde{Q} = D_{\widetilde{U}} \boldsymbol{\alpha}(\widetilde{\Sigma})\widetilde{U}^\top$, and

$$\widetilde{W} = \widetilde{U} \operatorname{diag}(\ln^{a_1} a_1^{\widetilde{\sigma}_1} \widetilde{u}_1^\top \widetilde{V} \widetilde{u}_1, \cdots, \ln^{a_n} a_n^{\widetilde{\sigma}_n} \widetilde{u}_n^\top \widetilde{V} \widetilde{u}_n) \widetilde{U}^\top.$$

In [25], the differential of the matrix exponential is written as an infinite series. The differential of our ϕ_{ma} can also be rewritten in this way.

Proposition III.9 (Differential as Infinite Series)**.** *Following the notation in Prop. III.8, the differential of ϕ_{ma} can also be formulated as*

$$\begin{aligned} &\phi_{ma*,X}(\widetilde{V}) \\ &= \sum_{k=1}^{\infty} \frac{1}{k!} (\sum_{l=0}^{k-1} (\widetilde{P}X)^{k-l-1} (D_{\widetilde{P}} X + \widetilde{P}\widetilde{V})(\widetilde{P}X)^l), \end{aligned} \tag{27}$$

where $\widetilde{P} = \widetilde{U} B \widetilde{U}^\top$, $B = \operatorname{diag}(\ln^{a_1}, \cdots, \ln^{a_n})$, $D_{\widetilde{P}} = D_{\widetilde{U}} B \widetilde{U}^\top + \widetilde{U} B D_{\widetilde{U}}^\top$.

When $\phi_{\mathrm{ma}}(\cdot)$ is reduced into matrix exponential, Eq. (27) becomes Eq. 8 in [25], and our ALEM becomes exactly LEM.

IV. Properties of ALEM

Since our ALEMs are natural generalizations of LEM. Therefore, intuitively, ALEMs would share every property of LEM. This section introduces some useful properties of our ALEMs for machine learning, including Fréchet mean and invariance properties.

Fréchet means are important tools for SPD matrices learning [1], [5], [30], [44]. Like LEM, our ALEM also enjoys closed forms of Fréchet means. We present a more general result, the weighted Fréchet mean.

Proposition IV.1 (Weighted Fréchet Means)**.** *For m points $S_1, \cdots S_m$ in SPD manifolds with associated weights $w_1, \cdots, w_m \in \mathbb{R}_+$, the weighted Fréchet mean M over the metric space $\{\mathcal{S}^n_{++}, d^{\mathrm{ALE}}\}$ has a closed form*

$$M = \phi_{\mathrm{ma}}(\sum_{i=1}^m \frac{w_i}{\sum_{j=1}^m w_i} \operatorname{mlog}(S_i)). \tag{28}$$

Like LEM, although our ALEM does not conform with the affine-invariance, our ALEM enjoys some other kinds of invariance.

Proposition IV.2 (Bi-invariance)**.** *ALEM is a Lie group bi-invariant metric.*

Proposition IV.3 (Exponential Invariance)**.** *The Fréchet means under ALEM are exponential-invariant. In other words, for $S_1, \cdots S_m \in \mathcal{S}^n_{++}$ and $\beta \in \mathbb{R}$,*

$$(\mathrm{FM}(S_1, \cdots S_m))^\beta = \mathrm{FM}(S_1^\beta, \cdots S_m^\beta), \tag{29}$$

where $\mathrm{FM}(S_1, \cdots S_m))$ means the Fréchet mean of $S_1, \cdots S_m$.

Except for the exponential invariance, the Fréchet mean induced by our ALEM also satisfies various properties presented in [45].

Proposition IV.4. *For any SPD matrices A, B, C, A_0, B_0, C_0, denote $\mathrm{FM}(A, B, C)$ as the Fréchet mean of A, B, C under ALEM. Then the Fréchet mean satisfies the following properties.*

(U1) Permutation invariance. For any permutation $\pi(\{A, B, C\})$ of $\{A, B, C\}$,

(U2) $\mathrm{FM}(A, A, A) = A$

The following properties hold if A, B, C, A_O, B_0, C_0 commute.

(V1) Joint homogeneity. $\mathrm{FM}(aA, bB, cC) = (abc)^{1/3}\mathrm{FM}(A, B, C), \forall a, b, c > 0.$

(V2) Monotonicity. The map $(A,B,C) \mapsto \mathrm{FM}(A,B,C)$ *is monotone, i.e., , if* $A \geq A_0$*,* $B \geq B_0$*, and* $C \geq C_0$*, then* $\mathrm{FM}(A,B,C) \geq \mathrm{FM}(A_0,B_0,C_0)$ *in the positive semidefinite ordering.*

(V3) Self-duality. $\mathrm{FM}(A,B,C) = \mathrm{FM}(A^{-1},B^{-1},C^{-1})^{-1}$.

(V4) Determinant identity. $\det \mathrm{FM}(A,B,C) = (\det A \cdot \det B \cdot \det C)^{1/3}$.

In fact, Prop. IV.4 holds true for any finite number of SPD matrices. Besides, the geodesic distance induced by ALEMs has similarity invariance.

Proposition IV.5 (Similarity Invariance)**.** *The geodesic distance under ALEM is similarity invariant. In other words, let* $R \in SO(n)$ *be a rotation matrix,* $s \in \mathbb{R}_+$ *is a scale factor. Given any two SPD matrices* S_1 *and* S_2*, we have*

$$d^{\mathrm{ALE}}(S_1,S_2) = d^{\mathrm{ALE}}(s^2 R S_1 R^\top, s^2 R S_2 R^\top). \tag{30}$$

Let us explain a bit more about the above three kinds of invariance. Firstly, among metrics on Lie groups, bi-invariant metrics are the most convenient ones [46, Chapter V]. Secondly, exponential invariance offers a fast computation for Fréchet means under exponential scaling. At last, similarity-invariance is significant for describing the frequently encountered covariance matrices [25].

The above discussion focuses on theoretical side. Now, let us reconsider Eq. (17) in a numerical way.

Proposition IV.6. mlog *can be rewritten as*

$$\mathrm{mlog}(S) = U \log_\alpha(\Sigma) U^\top, \tag{31}$$
$$= U A \ln(\Sigma) U^\top, \tag{32}$$
$$= U \frac{\ln(\Sigma)}{B} U^\top, \tag{33}$$

where $\frac{X}{Y}$ *is the diagonal division,* $B = \mathrm{diag}(\ln^{a_1}, \cdots, \ln^{a_n})$*, and* $A = \frac{I}{B}$*.*

Based on the above proposition, more analyses could be carried out from a numerical point of view. First, $\mathrm{mlog}(\cdot)$ can balance the eigenvalues of an input SPD matrix S by exploiting different bases for different eigenvalues. In Riemannian algorithms, manifold-valued features usually contain vibrant information. We expect that by the above adaptation, manifold-valued data could be better fitted and the learning ability of algorithms could be further promoted.

Remark IV.7. Note that the discussion in Sec. III-C and Sec. IV can also be readily transferred into LCM, generating an adaptive version of LCM.

V. Applications to SPD Neural Networks

Since Riemannian metrics are the foundations of Riemannian learning algorithms, our ALEM has the potential to rewrite Riemannian algorithms, especially the algorithms based on LEM. Besides, the base vector in mlog could bring vibrant diversity to our ALEM. This adaptive mechanism could help the algorithm better fit with complicated manifold-valued data. Especially in Riemannian neural networks, as we will show, optimization of base vectors can be easily embedded into the standard backpropagation (BP) process. Therefore, we focus on the applications of our metrics to SPD neural networks.

In the existing SPD neural networks, on activation or classification layers, SPD features would interact with the logarithmic domain by matrix logarithm [3], [13], [17], [38], [47]. The underlying mechanism of this interaction is that the matrix logarithm is an isomorphism, identifying the SPD manifold under LEM with the Euclidean space $\mathcal{S}^n$. This projection can, therefore, maintain the LEM-based geometry of SPD features. However, in deep networks, the geometry might be more complex. Since ALEM can vibrantly adapt to network learning, compared with the plain LEM, our ALEM could more faithfully respect the geometry of SPD deep features. mlog thus possesses more advantages than the vanilla matrix logarithm mln. We, therefore, replace the vanilla matrix logarithm with our mlog, to respect the more advantageous geometry, *i.e.,* the ALEM-based geometry.

We focus on the most classic SPD network, SPDNet [13]. There are three basic layers in SPDNet, *i.e.,* BiMap, ReEig, and LogEig, which are defined as

$$\text{BiMap: } S^k = W^k S^{k-1} W^k, \tag{34}$$
$$\text{ReEig: } S^k = U^{k-1} \max(\Sigma^{k-1}, \epsilon I_n) U^{k-1\top}, \tag{35}$$
$$\text{LogEig: } S^k = \mathrm{mln}(S^{k-1}), \tag{36}$$

where W^k is semi-orthogonal and $S^{k-1} = U^{k-1}\Sigma^{k-1}U^{k-1\top}$ is the eigendecomposition. The BiMap (Bilinear Mapping) is a generalized version of conventional linear mapping. The ReEig (Eigenvalue Rectification) mimics the ReLu-like nonlinear activation functions by eigen-rectification. The LogEig layer projects SPD-valued data into the Euclidean space for further classification.

The matrix logarithm in the LogEig layer is substituted by our mlog. We call this layer the adaptive logarithm (ALog) layer. We set the base vector α as a learnable parameter. In this way, as mlog is an isomorphism, the network can implicitly respect the ALEM-based Riemannian geometry by learning the mlog explicitly. Besides, since our ALog layer is independent of specific network architectures, it can also be plugged into other SPD deep networks.

VI. Parameters Learning

We first present the gradient computation and then discuss in detail how to optimize the parameters in the ALog layer.

A. Gradients Computation

Two gradients need calculation in the proposed ALog layer: one w.r.t the parameters and another w.r.t the input of the ALog layer. Since structural matrix decomposition is involved in mlog, the following contents heavily rely on the structural matrix BP [48], the key idea of which is the invariance of first-order differential form. For the ALog layer, it is essentially a special case of eigenvalue functions. Based on the formula offered in [49] and matrix BP techniques presented in [48], we can obtain all the gradients, as presented in the following proposition.

TABLE I
PARAMETER LEARNING IN THE ALOG LAYER.

Name	Detail	Constraint	Method
RELU	Optimizing base vector α (Eq. (31))	Positive	shift-ReLu $\max(\epsilon, \alpha)$
MUL	Optimizing diagonal elements of A (Eq. (32))	Unconstrained	Standard BP
DIV	Optimizing diagonal elements of B (Eq. (33))	Unconstrained	Standard BP

Proposition VI.1. *Let us denote $X = \operatorname{mlog}(S)$, where $S \in \mathcal{S}^d_{++}$ is an input SPD matrix of the ALog layer. We have the following gradients:*

$$\nabla_S L = U[K \odot (U^T(\nabla_X L)U)]U^T, \tag{37}$$

$$\nabla_A L = [U^\top(\nabla_X L)U] \odot \log(\Sigma), \tag{38}$$

where $S = U\Sigma U^\top$ is the eigendecomposition of an SPD matrix and matrix K is defined as

$$K_{ij} = \begin{cases} \frac{f(\sigma_i)-f(\sigma_j)}{\sigma_i-\sigma_j} & \text{if } \sigma_i \neq \sigma_j \\ f'(\sigma_i) & \text{otherwise} \end{cases} \tag{39}$$

where $f(\sigma_i) = A_{ii}\log_e(\sigma_i)$ and $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \cdots, \sigma_d)$.

B. Parameters Updates

Let us explain how to optimize the proposed layer in a standard backpropagation (BP) framework. Denote the dimension of an input SPD matrix S as $d \times d$. Recalling Eq. (31)-Eq. (33), there are three ways to implement parameter learning. We could learn the base vector α in Eq. (31), diagonal matrix A in Eq. (32), or diagonal matrix B in Eq. (33), respectively.

For learning A in Eq. (32) or B in Eq. (33), since the parameters (diagonal elements) lie in a Euclidean space $\mathbb{R}^d$, the optimization can be easily integrated into the BP algorithm. We call learning A MUL and learning B DIV.

For the case of learning α in Eq. (31), since α lies in a non-Euclidean space, specific updating strategies should be considered. Without loss of generality, we focus on the case of a scalar parameter $a > 0 \& a \neq 1$. The condition of $a \neq 1$ can be further waived since we can set $a = 1+\epsilon$ if $a = 1$. Then, there is only one constraint about positivity. We use the shift-ReLU of an unconstrained parameter, *i.e.,* $\max(\epsilon, a)$ with $\epsilon \in \mathbb{R}_+$. This strategy is named RELU. Other tricks like square are also feasible, but we will focus on the RELU. In addition, positive scalar a can be directly optimized by Riemannian optimization [50]. We further prove that this strategy completely equals learning B directly. For more details, please refer to the Supp. B-B.

Therefore, there are three ways of updates, *i.e.,* RELU, DIV, and MUL, summarized in Tab. I.

VII. EXPERIMENTS

In this section, we validate the efficacy of our approaches on multiple datasets. We would like to clarify that our method does not necessarily aim to achieve the SOTA in a general sense for the following tasks but rather to promote the learning abilities of the family of SPD-based methods.

A. Datasets and Settings

As we discussed before, although the proposed ALog layers can be plugged into the existing SPD networks, we focus on the SPDNet framework [13]. We follow the PyTorch code provided by SPDNetBN[1] to reproduce SPDNet & SPDNetBN and implement our approaches.

Following previous work [5], [13], we evaluate our methods on three datasets: the HDM05 [51] for skeleton-based actions recognition, the FPHA [52] for skeleton-based hand gestures recognition, and the AFEW [53] for emotions recognition. The HDM05 dataset comprises motion capture data (MoCap) covering 130 action classes. Each data point is a sequence of frames of 31 3D coordinates. Each sequence can be represented by a 93×93 temporal covariance matrix. For a fair comparison, we exploit the pre-processed 93×93 covariance features [2] released by [5], which trims the dataset down to 2086 points scattered throughout 117 classes by removing some under-represented classes. Following the settings in [5], we split the dataset into 50% for training and 50% for testing. FPHA includes 1,175 clips of 45 different action categories. Each frame is represented by 21 3D coordinates. Similarly, each sequence can be modeled by a 63×63 covariance matrix. For a fair comparison, we follow the experimental protocol in [52], where 600 sequences are used for training, and 575 sequences are used for testing. AFEW consists of 7 kinds of emotions, with 773 samples for training and 383 samples for validation. We use the released pre-trained FAN[3] [54] to extract deep features and establish a 512×512 temporal covariance matrix for each video.

We denote $\{d_0, d_1, \cdots, d_L\}$ as the dimensions of each transformation layer in the SPDNet backbone. Following the settings in [5], all networks are trained by the default Riemannian SGD [55] with a fixed learning rate γ and batch size of 30. To make ALog start from the vanilla matrix logarithm, the parameters in MUL, DIV, and RELU are initialized as 1,1 and e, respectively. By abuse of notation, SPDNet-ALog-MUL is abbreviated as ALog-MUL, denoting that we substitute the LogEig layer (matrix logarithm) in SPDNet with our proposed ALog optimized by MUL. All experiments use an Intel Core i9-7960X CPU with 32 GB RAM.

B. Experimental Results

On the three datasets, the training epochs are set to be 200, 500, and 100. We verify our ALog on the SPDNet with

[1]https://proceedings.neurips.cc/paper/2019/file/6e69ebbfad976d4637bb4b39de261bf7-Supplemental.zip
[2]https://www.dropbox.com/s/dfnlx2bnyh3kjwy/data.zip?dl=0
[3]https://github.com/Open-Debin/Emotion-FAN

This article has been accepted for publication in IEEE Transactions on Image Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIP.2024.3451930

TABLE II
RESULTS OF ALOG ON THE HDM05 DATASET.

Learning rate	$1e^{-2}$			$5e^{-2}$		
Architecture	{ 93, 30}	{ 93, 70, 30}	{ 93, 70, 50, 30}	{ 93, 30}	{ 93, 70, 30}	{ 93, 70, 50, 30}
SPDNet	62.92±0.81	62.87±0.60	63.03±0.67	63.89±0.73	64.00±0.65	63.72±0.61
SPDNetBN	63.03±0.75	58.27±1.7	52.02±2.34	63.75±0.69	48.78±5.15	37.84±6.10
ALog-MUL	63.52±0.75	63.86±0.58	**63.94±0.44**	64.4±0.68	64.60±0.69	64.36±0.49
ALog-DIV	**63.60±0.79**	63.93±0.52	63.81±0.7	**64.81±0.64**	**64.84±0.65**	**64.80±0.36**
ALog-RELU	63.02±0.79	**63.94±0.64**	63.14±0.65	63.97±0.75	64.10±0.63	63.78±0.46

various architectures. Besides, we further test the robustness of the proposed layer against different learning rates on the HDM05 and FPHA datasets. Generally speaking, among all three kinds of implementation, **ALog-MUL** shows the most robust performance gain and achieves consistent improvement over the vanilla matrix logarithm. Besides, we could also observe that ALog-MUL is comparable to or even better than SPDNetBN, which yet brings much more complexity than our approach. The main reason for the superiority of our ALog against the vanilla matrix logarithm is that our ALog can adaptively respect the vibrant geometry of SPD manifolds, depending on the characteristics of datasets, while only LEM can be respected by the matrix logarithm. The following are detailed observations and analyses.

Results on the HDM05 dataset. The 10-fold results are presented in Tab. II, where dataset split and weights initialization are randomized. Following [13], three architectures are implemented on this dataset, *i.e.,* { 93, 30}, { 93, 70, 30}, and { 93, 70, 50, 30}. Generally speaking, endowed with the ALog, SPDNet would achieve consistent improvement. Among all three kinds of implementation, RELU only brings limited improvement. The reason might be that RELU fails to respect the innate geometry of the positive constraint. There is another interesting observation worth mentioning. In [5], only the result of SPDNetBN under the architecture of $\{93, 30\}$ is reported on this dataset. Our experiments show that with the network going deeper, SPDNetBN tends to collapse, while our ALog layer performs robustly in all settings.

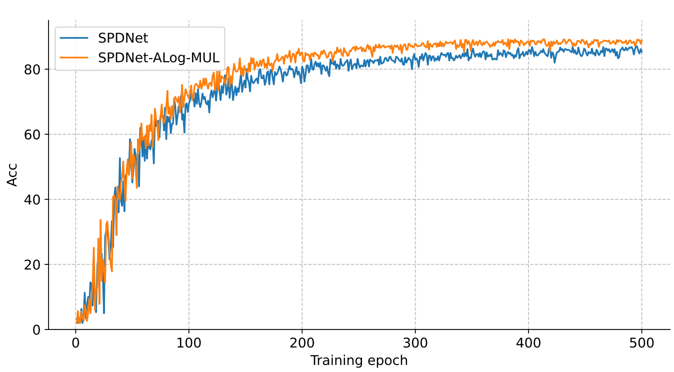

Fig. 1. Accuracy Curves on the FPHA Dataset.

Results on the FPHA dataset. We validate our approach on this dataset, with a learning rate of $1e^{-2}$, over 10-fold cross-validation on random initialization. Since our experiments indicate that the vanilla SPDNet is already saturated with 1 BiMap layer, we just report the results on the architecture

TABLE III
RESULTS OF ALOG ON THE FPHA DATASET.

SPDNet	SPDNetBN	ALog		
		MUL	DIV	RELU
85.73±0.80	86.83±0.74	87.8±0.71	**88.07±1.13**	86.65±0.68

of $\{63, 33\}$, which are presented in Tab. III. Although DIV performs best on this dataset, it presents the biggest variance. There is an underlying nonlinear scaling mechanism in the update of DIV, which might undermine its robustness. Without loss of generality, let us focus on a single scalar parameter b in Eq. (33). The ultimate factor multiplied by the plain logarithm is $1/b$. Therefore, the change of the multiplier after the update would be

$$1/(b - \Delta) - 1/b = \Delta/[(b - \Delta)b]. \tag{40}$$

Eq. (40) will scale the original Δ to some extent. This scaling mechanism might undermine the robustness of the ALog layer. However, ALog-MUL achieves robust improvement and even surpasses SPDNetBN. This again demonstrates the significance of our adaptive mechanism for Riemannian deep networks. Finally, in terms of convergence analysis, accuracy curves with and without ALog are also reported in Fig. 1.

TABLE IV
RESULTS OF ALOG ON THE AFEW DATASET.

Depth	1	2	3	4
SPDNet	48.53	46.89	48.24	47.22
SPDNetBN	46.89	46.65	47.62	48.35
ALog-MUL	**48.57**	**48.13**	**49.45**	**50.62**
ALog-DIV	48.42	48.02	48.13	49.89
ALog-RELU	48.06	47.25	48.86	48.1

Results on the AFEW dataset. On this dataset, the learning rate is $5e^{-2}$ and we validate our method under four network architectures, *i.e.,* {512, 100}, {512, 200, 100}, {512, 400, 200, 100}, and {512, 400, 300, 200, 100}. Note that, on this dataset, SPDNetBN tends to present relatively large fluctuations in performance, so we compute the median of the last ten epochs. On various architectures, consistent improvement can be observed when SPDNet is endowed with our ALog. In addition, MUL achieves the best among all three kinds of implementation. Another interesting observation is that SPDNetBN seems ineffective on these deep features, while our methods show consistent superior performance, particularly obvious for our ALog-MUL. This indicates that our adaptive layer maintains effectiveness when applied to covariance matrices from deep features.

Model complexity. Our ALog manifests the same complexity, no matter how it is optimized. Without loss of generality, the discussion below focuses on ALog-MUL. The extra computation and memory costs caused by the ALog layer are minor. It only depends on the final dimension of the network. Let us take the deepest one on the AFEW dataset as an example. Our ALog only brings 100 unconstrained scalar parameters, while SPDNetBN needs an SPD matrix parameter for each Riemannian batch normalization (RBN) layer. The total number of the parameters in RBN layers is $400^2 + 300^2 + 200^2$, which is much bigger than ours. In addition, the SPDNetBN needs to store the running mean of SPD matrices in every RBN layer, while our ALog only needs to store a vector. In terms of computation, the extra cost of our ALog is secondary as well. The forward and backward computation of our ALog is generally the same as the plain matrix logarithm, while computation in the RBN layer is much more complex. All in all, our ALog can consistently improve the performance of the SPDNet and achieve comparable or better results against SPDNetBN with much cheaper computation and memory costs.

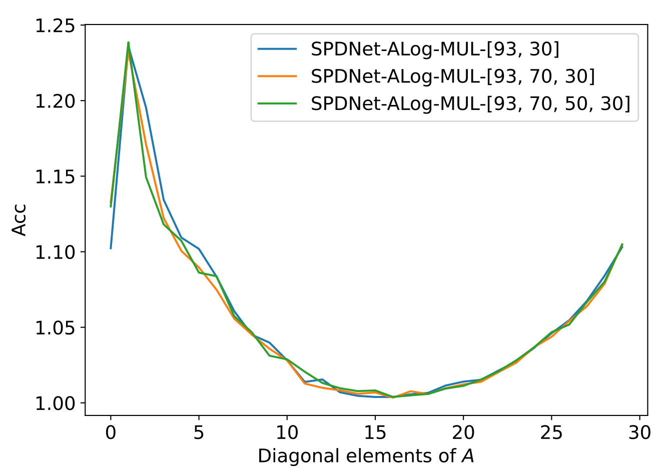

Fig. 2. Visualization of Parameters in the ALog Layer on the HDM05 Dataset.

Visualization. We visualize the final learned parameters of the ALog layer. Since ALog-MUL is the most robust strategy, we visualize the parameters of ALog-MUL. Specifically, we plot the final values of the diagonal elements of A in Eq. (32) and visualize the results in Figs. 2 and 3. We observe that the distribution of the parameters is consistent within the same dataset but varies between datasets. This indicates that our approach can capture vibrant patterns in different datasets, respecting their specific geometry.

Ablation studies. To further demonstrate the utility of the adaptive mechanisms in our approach, we further validate the ALog layer with fixed bases. As decimal and binary are the two most common systems, we use $\log_{10}$ and $\log_2$ as examples of shrinking and expanding $\log_e$. Specifically, we set $\log_\alpha = \log_{10}$ and $\log_\alpha = \log_2$ in Eq. (31), respectively. We refer to the network with binary/decimal base as SPDNet-Log2/SPDNet-Log10. Note that when $\log_\alpha = \log_e$, Eq. (31) is reduced to the vanilla matrix logarithm, and the network is our baseline, *i.e.*, SPDNet. We conduct 10-fold experiments on the HDM05 and FPHA datasets and set the learning rate to $5e^{-2}$ and $1e^{-2}$, respectively, while keeping the other settings consistent with previous experiments. The results are presented in Tab. V. We observe that the fixed logarithms show similar or slightly worse results than the vanilla $\log_e$, while our ALog shows consistent improvement. Besides, $\log_{10}$ does not converge in the FPHA dataset. In fact, $\log_{10}$ could shrink the gradient, slowing down convergence, especially under a small learning rate. In contrast, our ALog maintains consistent effectivity. In summary, our ALog can respect vibrant geometry induced by mlog and thus benefit SPD network learning.

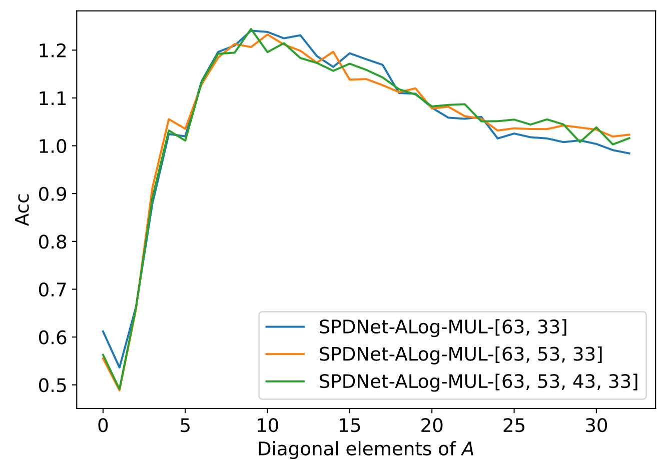

Fig. 3. Visualization of Parameters in the ALog Layer on the FPHA Dataset.

TABLE V
RESULTS OF FIXED BASES ON THE HDM05 AND FPHA DATASETS.

Dataset	HDM05			FPHA
Architecture	{93, 30}	{93, 70, 30}	{93, 70, 50, 30}	{63, 33}
SPDNet-Log2	63.93±0.81	63.54±0.50	63.98±0.63	86.65±0.67
SPDNet	63.89±0.73	64.00±0.65	63.72±0.61	85.73±0.80
SPDNet-Log10	63.45±0.33	63.8±0.71	63.64±0.64	78.42±0.77
SPDNet-ALog-MUL	**64.4±0.68**	**64.60±0.69**	**64.36±0.49**	**87.8±0.71**

VIII. APPLICATIONS TO OTHER RIEMANNIAN BLOCKS

Riemannian metrics are foundations for Riemannian neural networks. Therefore, our ALEM can re-design basic blocks in Riemannian neural networks. This section applies our ALEM to other Riemannian building blocks, including Riemannian batch normalization [22], Riemannian residual blocks [43], and Riemannian classifiers [39]. We also use the NTU60 [56] dataset as an example of the large-scale dataset. More implementation details are presented in Supp. C.

A. Riemannian Batch Normalization

In Euclidean neural networks, batch normalization [57] has been widely used since it can facilitate network training. Recently, Chen *et al.* [22] proposed a framework for Riemannian batch normalization (RBN) on Lie groups, referred to as

TABLE VI
COMPARISON OF RBN METHODS ON THE HDM05 DATASET.

Methods	Geometries	[93, 30]	[93, 70, 30]	[93, 70, 50, 30]
None	N/A	63.89±0.73	64.00±0.65	63.72±0.61
SPDNetBN	AIM	63.75±0.69	48.78±5.15	37.84±6.10
SPDBN	AIM	64.33±0.89	64.31±0.92	63.62±1.21
LieBN-LEM	LEM	63.67±0.85	65.77±0.89	65.34±0.83
LieBN-ALEM	ALEM	**65.24±0.71**	**70.11±0.96**	**68.86±0.72**

LieBN. LieBN can guarantee the normalization of sample statistics under the left- or right-invariant metric [22, Prop. 4.2]. As shown in Thm. III.6, $\{\mathcal{S}^n_{++}, \odot_{mlog}\}$ forms a Lie group. Besides, Prop. IV.2 demonstrates that ALEM is bi-invariant w.r.t. this group structure. Therefore, LieBN under ALEM can also normalize Riemannian sample statistics. We follow Alg. 1 and Thm 5.3 in [22] to implement the LieBN under ALEM, denoted as LieBN-ALEM. In addition, we compared LieBN-ALEM against other kinds of RBN methods, including AIM-based SPDNetBN [5] and SPDBN [58], and LieBN under LEM [22] (LieBN-LEM).

Following previous work [5], [22], [58], we adopt the SPDNet backbone. Tab. VI presents the 10-fold average results on the HDM05 dataset under different network architectures. Our LieBN-ALEM achieves the best performance compared with the other RBN methods. Especially, the AIM-based SPDNetBN brings worse performance under deeper architectures. In contrast, our LieBN-ALEM can consistently improve the performance across different architectures. Besides, compared with LieBN-LEM, our LieBN-ALEM shows better performance, demonstrating the effectiveness of our ALEM.

B. Riemannian Residual Blocks

TABLE VII
EXPERIMENTS OF RRESNET UNDER DIFFERENT GEOMETRIES.

Methods	HDM05	NTU
SPDNet	63.89±0.73	45.90±1.11
RResNet-AIM	63.82±0.58	45.22 ± 1.23
RResNet-LEM	66.51±0.93	48.73±0.60
RResNet-ALEM	**69.03±1.06**	**57.09±0.59**

ResNets [34] have become ubiquitous in machine learning due to their beneficial learning properties. Recently, Katsman *et al.* [43] extended the Euclidean ResNet into Riemannian spaces, referred to as RResNet. On the SPD manifold, the Riemannian residual block under a given metric g is defined as

$$g(S) = \mathrm{Exp}_S(\ell(S)), \tag{41}$$

$$\ell(X) = Q \,\mathrm{diag}\left(f(\mathrm{spec}(X))\right) Q^T, \tag{42}$$

where Exp is the Riemannian exponentiation under g, $\ell : \mathcal{S}^n_{++} \to T\mathcal{S}^n_{++}$ constructs the vector field, $\mathrm{spec}(\cdot)$ is the spectral map that takes SPD matrices to a vector of their eigenvalues, $f : \mathbb{R}^n \to \mathbb{R}^n$ is parameterized as a neural network, and $Q \in \mathrm{O}(n)$. Since the Riemannian exponential in Eq. (41) is metric-dependent, the Riemannian residual blocks vary under different metrics. The Riemannian residual block under ALEM can be obtained by putting Eq. (21) into Eq. (41). We need further to show the gradient w.r.t. ϕ_{mexp}. As the inverse of Eq. (32), ϕ_{mexp} can be rewrote as

$$\begin{aligned} \phi_{\mathrm{mexp}}(X) &= U \boldsymbol{\alpha}(\Sigma) U\top \\ &= U \exp\left(\frac{\Sigma}{A}\right) U\top, \end{aligned} \tag{43}$$

where $X = U\Sigma U^\top \in \mathcal{S}^n$ is the eigendecomposition. Following Prop. VI.1, we can obtain the backpropagation of ϕ_{mexp}, which is presented in the following.

Proposition VIII.1. *Let us denote $X = \phi_{\mathrm{mexp}}(S)$ with $S \in \mathcal{S}^d_{++}$. We have the following gradients:*

$$\nabla_S L = U[K \odot (U^T(\nabla_X L)U)]U^T, \tag{44}$$

$$\nabla_A L = [U^\top(\nabla_X L)U] \odot \left(\boldsymbol{\alpha}(\Sigma)\frac{-\Sigma}{A^2}\right), \tag{45}$$

where $S = U\Sigma U^\top$ is the eigendecomposition of an SPD matrix and matrix K is defined as

$$K_{ij} = \begin{cases} \frac{f(\sigma_i)-f(\sigma_j)}{\sigma_i-\sigma_j} & \text{if } \sigma_i \neq \sigma_j \\ f'(\sigma_i) & \text{otherwise} \end{cases} \tag{46}$$

where $f(\sigma_i) = e^{\frac{\sigma_i}{A_{ii}}}$ and $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_d)$.

Following [43], we compare RResNet under different geometries on the HDM05 and NTU60 datasets. Tab. VII reports the 10-fold and 5-fold average results on these datasets. Compared with the vanilla SPDNet, RResNet-AIM brings little improvement, while LEM and ALEM show much better performance. Especially, the ALEM-based RResNet can bring a clear performance improvement, underscoring the effectiveness of our ALEM.

C. Riemannian Classifiers

TABLE VIII
COMPARISON OF GYRO MLRS ON THE NTU60 DATASETS.

Learning Rates	$1e^{-2}$	$5e^{-2}$
GyroMLR-AIM	54.28±0.47	41.41±0.71
GyroMLR-LCM	42.68±0.88	42.06±0.49
GyroMLR-LEM	53.22±0.47	39.62±1.30
GyroMLR-ALEM	**56.21±0.39**	**51.65±0.44**

Euclidean Multinomial Logistic Regression (MLR), which consists of FC and softmax, has become a standard classification block in Euclidean neural networks. Inspired by this, Nguyen and Yang [39] extended the Euclidean MLR into the SPD manifolds by gyro structures [11] for intrinsic classification, referred to as gyro MLR. Three gyro MLRs under LCM, AIM, and LEM was introduced in [39]. Following the logic in [39, Sec. 2.4.2], we can obtain the gyro MLR under ALEM.

Theorem VIII.2 (Gyro MLR)**.** *Given an SPD feature $S \in \mathcal{S}^n_{++}$ and C classes, the SPD gyro MLR under ALEM computes the multinomial probability of each class:*

$$\begin{aligned} &p(y = k \mid S) \\ &\propto \exp\left[\langle \mathrm{mlog}(S) - \mathrm{mlog}(P_k), \mathrm{mlog}_{*,P_k}(\tilde{A}_k)\rangle\right], \end{aligned} \tag{47}$$

where $k \in \{1, \ldots, C\}$, $P_k \in \mathcal{S}^n_{++}$, *and* $\tilde{A}_k \in T_{P_k}\mathcal{S}^n_{++}$.

Since A_k lies in $T_{P_k}\mathcal{S}^n_{++}$, and P_k varies during network training, A_k cannot be viewed as a Euclidean parameter. Following [59], we set $\tilde{A}_k = \Gamma_{I \to P_k}(A_k)$ with $A_k \in T_I\mathcal{S}^n_{++}$ (a fixed tangent space). Therefore, the RHS of Eq. (47) becomes

$$\exp\left[\langle \operatorname{mlog}(S) - \operatorname{mlog}(P_k), \operatorname{mlog}_{*,I}(A_k)\rangle\right], \qquad (48)$$

As $\operatorname{mlog}_{*,I}(A_k) \in T_0\mathcal{S}^n \cong \mathcal{S}^n$, we view $\operatorname{mlog}_{*,I}(A_k)$ as the parameter.

We use the SPDNet as the backbone. We compare Gyro MLR under our ALEM with the ones under LEM, LCM, and AIM on the NTU60 dataset. Tab. VIII presents the 5-fold average results under different learning rates. Our ALEM outperforms the other metrics within the gyro MLR framework. When the learning rate is $5e^{-2}$, our GyroMLR-ALEM shows more advantageous performance, especially compared with GyroMLR-LEM. These results demonstrate that the Riemannian networks can benefit from the adaptivity of our ALEM.

IX. Limitations

Our approach presents a general framework for PEMs and specifically focuses on extending LEM. Despite the fast and simple computations of PEMs, there are several other types of Riemannian metrics on SPD manifolds, such as AIM [24] and Bures-Wasserstein Metric (BWM) [60]. These metrics do not belong to PEMs but have shown successful performance on different applications. Therefore, the adaptive mechanisms of these types of Riemannian metrics should also be addressed in future work.

X. Conclusion

Riemannian metrics are foundations for Riemannian learning algorithms. In this paper, we propose a general framework for characterizing PEMs on SPD manifolds. According to this framework, we extend LEM into ALEMs for SPD matrix learning. We also present comprehensive and rigorous theories of our metrics. Extensive experiments indicate that SPD deep networks can benefit from our metrics. Eq. (7) indicates that LCM is pulled back by Cholesky decomposition and diagonal logarithm. Therefore, as a future avenue, the discussions in this paper can be readily transferred to LCM.

References

[1] R. Chakraborty, C.-H. Yang, X. Zhen, M. Banerjee, D. Archer, D. Vaillancourt, V. Singh, and B. Vemuri, "A statistical recurrent model on the manifold of symmetric positive definite matrices," *Advances in Neural Information Processing Systems*, vol. 31, 2018. [Online]. Available: https://papers.nips.cc/paper_files/paper/2018/hash/7070f9088e456682f0f84f815ebda761-Abstract.html

[2] A. Das, M. S. Nair, and S. D. Peter, "Sparse representation over learned dictionaries on the riemannian manifold for automated grading of nuclear pleomorphism in breast cancer," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1248–1260, 2018. [Online]. Available: https://doi.org/10.1109/TIP.2018.2877337

[3] R. Chakraborty, J. Bouza, J. Manton, and B. C. Vemuri, "Manifoldnet: A deep neural network for manifold-valued data with applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. [Online]. Available: https://doi.org/10.1109/TPAMI.2020.3003846

[4] O. Yair, M. Ben-Chen, and R. Talmon, "Parallel transport on the cone manifold of SPD matrices for domain adaptation," *IEEE Transactions on Signal Processing*, vol. 67, no. 7, pp. 1797–1811, 2019. [Online]. Available: https://doi.org/10.1109/TSP.2019.2894801

[5] D. Brooks, O. Schwander, F. Barbaresco, J.-Y. Schneider, and M. Cord, "Riemannian batch normalization for SPD neural networks," in *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/6e69ebbfad976d4637bb4b39de261bf7-Abstract.html

[6] R. J. Kobler, J. ichiro Hirayama, Q. Zhao, and M. Kawanabe, "SPD domain-specific batch normalization to crack interpretable unsupervised domain adaptation in EEG," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=pp7onaiM4VB

[7] C. Ju, R. J. Kobler, L. Tang, C. Guan, and M. Kawanabe, "Deep geodesic canonical correlation analysis for covariance-based neuroimaging data," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=PnR1MNen7u

[8] M. Moakher, "On the averaging of symmetric positive-definite tensors," *Journal of Elasticity*, vol. 82, no. 3, pp. 273–296, 2006. [Online]. Available: https://doi.org/10.1007/s10659-005-9035-z

[9] J. Guilleminot and C. Soize, "Generalized stochastic approach for constitutive equation in linear elasticity: a random matrix model," *International Journal for Numerical Methods in Engineering*, vol. 90, no. 5, pp. 613–635, 2012. [Online]. Available: https://doi.org/10.1002/nme.3338

[10] F. López, B. Pozzetti, S. Trettel, M. Strube, and A. Wienhard, "Vector-valued distance and Gyrocalculus on the space of symmetric positive definite matrices," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 350–18 366, 2021. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/98c39996bf1543e974747a2549b3107c-Abstract.html

[11] X. S. Nguyen, "The Gyro-structure of some matrix manifolds," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 26 618–26 630. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/a9ad92a81748a31ef6f2ef68d775da46-Paper-Conference.pdf

[12] W. Zhao, F. Lopez, J. M. Riestenberg, M. Strube, D. Taha, and S. Trettel, "Modeling graphs beyond hyperbolic: Graph neural networks in symmetric positive definite matrices," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2023, pp. 122–139. [Online]. Available: https://doi.org/10.1007/978-3-031-43418-1_8

[13] Z. Huang and L. Van Gool, "A Riemannian network for SPD matrix learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017. [Online]. Available: https://doi.org/10.1609/aaai.v31i1.10866

[14] P. Li, H. Zeng, Q. Wang, S. C. Shiu, and L. Zhang, "High-order local pooling and encoding gaussians over a dictionary of gaussians," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3372–3384, 2017. [Online]. Available: https://doi.org/10.1109/TIP.2017.2695884

[15] W. Wang, R. Wang, Z. Huang, S. Shan, and X. Chen, "Discriminant analysis on riemannian manifold of gaussian distributions for face recognition with image sets," *IEEE Transactions on Image Processing*, vol. 27, no. 1, p. 151, 2018. [Online]. Available: https://doi.org/10.1109/TIP.2017.2746993

[16] S. Qiao, R. Wang, S. Shan, and X. Chen, "Deep heterogeneous hashing for face video retrieval," *IEEE Transactions on Image Processing*, vol. 29, pp. 1299–1312, 2019. [Online]. Available: https://doi.org/10.1109/TIP.2019.2940683

[17] X. S. Nguyen, "Geomnet: A neural network based on Riemannian geometries of SPD matrix space and Cholesky space for 3D skeleton-based interaction recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2021, pp. 13 379–13 389. [Online]. Available: https://doi.org/10.1109/ICCV48922.2021.01313

[18] Y. Song, N. Sebe, and W. Wang, "Why approximate matrix square root outperforms accurate SVD in global covariance pooling?" in *Proceedings of the IEEE International Conference on Computer Vision*, 2021, pp. 1115–1123. [Online]. Available: https://doi.org/10.1109/ICCV48922.2021.00115

[19] X. S. Nguyen, "A Gyrovector space approach for symmetric positive semi-definite matrix learning," in *Proceedings of the European Conference on Computer Vision*, 2022, pp. 52–68. [Online]. Available: https://doi.org/10.1007/978-3-031-19812-0_4

[20] Y. Song, N. Sebe, and W. Wang, "Fast differentiable matrix square root and inverse square root," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. [Online]. Available: https://doi.org/10.1109/TPAMI.2022.3216339

[21] D. Wei, X. Shen, Q. Sun, and X. Gao, "Discrete metric learning for fast image set classification," *IEEE Transactions on Image Processing*, vol. 31, pp. 6471–6486, 2022. [Online]. Available: https://doi.org/10.1109/TIP.2022.3212284

[22] Z. Chen, Y. Song, Y. Liu, and N. Sebe, "A Lie group approach to Riemannian batch normalization," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=okYdj8Ysru

[23] Z. Chen, Y. Song, G. Liu, R. R. Kompella, X. Wu, and N. Sebe, "Riemannian multiclass logistics regression for SPD neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2024.

[24] X. Pennec, P. Fillard, and N. Ayache, "A Riemannian framework for tensor computing," *International Journal of Computer Vision*, vol. 66, no. 1, pp. 41–66, 2006. [Online]. Available: https://doi.org/10.1007/s11263-005-3222-z

[25] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Fast and simple computations on tensors with log-Euclidean metrics." Ph.D. dissertation, INRIA, 2005. [Online]. Available: https://doi.org/10.1007/11566465_15

[26] Z. Lin, "Riemannian geometry of symmetric positive definite matrices via Cholesky decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 40, no. 4, pp. 1353–1370, 2019. [Online]. Available: https://doi.org/10.1137/18M1221084

[27] R. Wang, H. Guo, L. S. Davis, and Q. Dai, "Covariance discriminative learning: A natural and efficient approach to image set classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2496–2503. [Online]. Available: https://doi.org/10.1109/CVPR.2012.6247965

[28] 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 *International Conference on Machine Learning*. PMLR, 2015, pp. 720–729. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3045118.3045196

[29] Z. Huang, R. Wang, S. Shan, and X. Chen, "Face recognition on large-scale video in the wild with hybrid Euclidean-and-Riemannian metric learning," *Pattern Recognition*, vol. 48, no. 10, pp. 3113–3124, 2015. [Online]. Available: https://doi.org/10.1016/j.patcog.2015.03.011

[30] M. Harandi, M. Salzmann, and R. Hartley, "Dimensionality reduction on SPD manifolds: The emergence of geometry-aware methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 1, pp. 48–62, 2018. [Online]. Available: https://doi.org/10.1109/TPAMI.2017.2655048

[31] Z. Chen, T. Xu, X.-J. Wu, R. Wang, and J. Kittler, "Hybrid Riemannian graph-embedding metric learning for image set classification," *IEEE Transactions on Big Data*, 2021. [Online]. Available: https://doi.org/10.1109/TBDATA.2021.3113084

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012. [Online]. Available: https://doi.org//10.1145/3065386

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[35] Y.-T. Pan, J.-L. Chou, and C.-S. Wei, "MAtt: a manifold attention network for EEG decoding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 116–31 129, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/c981fd12b1d5703f19bd8289da9fc996-Abstract-Conference.html

[36] R. Wang, X.-J. Wu, Z. Chen, T. Xu, and J. Kittler, "Learning a discriminative SPD manifold neural network for image set classification," *Neural networks*, vol. 151, pp. 94–110, 2022. [Online]. Available: https://doi.org/10.1016/j.neunet.2022.03.012

[37] ——, "DreamNet: A deep Riemannian manifold network for SPD matrix learning," in *Proceedings of the Asian Conference on Computer Vision*, 2022, pp. 3241–3257. [Online]. Available: https://openaccess.thecvf.com/content/ACCV2022/html/Wang_DreamNet_A_Deep_Riemannian_Manifold_Network_for_SPD_Matrix_Learning_ACCV_2022_paper.html

[38] Z. Chen, T. Xu, X.-J. Wu, R. Wang, Z. Huang, and J. Kittler, "Riemannian local mechanism for SPD neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, pp. 7104–7112. [Online]. Available: https://doi.org/10.1609/aaai.v37i6.25867

[39] X. S. Nguyen and S. Yang, "Building neural networks on matrix manifolds: A Gyrovector space approach," *arXiv preprint arXiv:2305.04560*, 2023. [Online]. Available: https://proceedings.mlr.press/v202/nguyen23f.html

[40] R. Wang, X.-J. Wu, Z. Chen, C. Hu, and J. Kittler, "SPD manifold deep metric learning for image set classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2024. [Online]. Available: https://doi.org/10.1109/TNNLS.2022.3216811

[41] Y. Thanwerdas and X. Pennec, "Theoretically and computationally convenient geometries on full-rank correlation matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 43, no. 4, pp. 1851–1872, 2022. [Online]. Available: https://doi.org/10.1137/22M1471729

[42] ——, "O (n)-invariant Riemannian metrics on SPD matrices," *Linear Algebra and its Applications*, vol. 661, pp. 163–201, 2023. [Online]. Available: https://doi.org/10.1016/j.laa.2022.12.009

[43] I. Katsman, E. Chen, S. Holalkere, A. Asch, A. Lou, S. N. Lim, and C. M. De Sa, "Riemannian residual neural networks," *Advances in Neural Information Processing Systems*, vol. 36, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/c868aa7437dc9b29e674cd2e25689021-Abstract-Conference.html

[44] R. Chakraborty, "ManifoldNorm: Extending normalizations on Riemannian manifolds," *arXiv preprint arXiv:2003.13869*, 2020. [Online]. Available: https://arxiv.org/abs/2003.13869

[45] T. Ando, C.-K. Li, and R. Mathias, "Geometric means," *Linear algebra and its applications*, vol. 385, pp. 305–334, 2004. [Online]. Available: https://doi.org/10.1016/j.laa.2003.11.019

[46] S. Sternberg, *Lectures on differential geometry*. American Mathematical Soc., 1999, vol. 316. [Online]. Available: https://www.ams.org/journals/bull/1965-71-02/S0002-9904-1965-11286-1/S0002-9904-1965-11286-1.pdf

[47] X. Zhen, R. Chakraborty, N. Vogt, B. B. Bendlin, and V. Singh, "Dilated convolutional neural networks for sequential manifold-valued data," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 10 621–10 631. [Online]. Available: https://doi.org/10.1109/ICCV.2019.01072

[48] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix backpropagation for deep networks with structured layers," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2965–2973. [Online]. Available: https://doi.org/10.1109/ICCV.2015.339

[49] R. Bhatia, *Positive Definite Matrices*. Princeton University Press, 2009. [Online]. Available: https://doi.org/10.1515/9781400827787

[50] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008. [Online]. Available: https://doi.org/10.1515/9781400830244

[51] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber, "Documentation mocap database HDM05," Universität Bonn, Technical Report, 2007. [Online]. Available: https://resources.mpi-inf.mpg.de/HDM05/

[52] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 409–419. [Online]. Available: https://doi.org/10.1109/CVPR.2018.00050

[53] A. Dhall, A. Kaur, R. Goecke, and T. Gedeon, "Emotiw 2018: Audio-video, student engagement and group-level affect prediction," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, 2018, pp. 653–656. [Online]. Available: https://doi.org/10.1145/3242969.3264993

[54] D. Meng, X. Peng, K. Wang, and Y. Qiao, "Frame attention networks for facial expression recognition in videos," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 3866–3870. [Online]. Available: https://doi.org/10.1109/ICIP.2019.8803603

[55] G. Becigneul and O.-E. Ganea, "Riemannian adaptive optimization methods," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1eiqi09K7

[56] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A large scale dataset for 3D human activity analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1010–1019. [Online]. Available: https://doi.org/10.1109/CVPR.2016.115

[57] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html

[58] R. J. Kobler, J.-i. Hirayama, and M. Kawanabe, "Controlling the Fréchet variance improves batch normalization on the symmetric positive definite manifold," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3863–3867. [Online]. Available: https://doi.org/10.1109/ICASSP43922.2022.9746629

[59] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," *Advances in neural information processing systems*, vol. 31, 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/dbab2adc8f9d078009ee3fa810bea142-Abstract.html

[60] R. Bhatia, T. Jain, and Y. Lim, "On the Bures-Wasserstein distance between positive definite matrices," *Expositiones Mathematicae*, vol. 37, no. 2, pp. 165–191, 2019. [Online]. Available: https://doi.org/10.1016/j.exmath.2018.01.002

Ziheng Chen received the B.A. degree in logistics management from Shandong University, Jinan, China, and M.S. degree in computer science and technology from Jiangnan University, Wuxi, China. He is currently working toward the Ph.D. degree with the Multimedia and Human Understanding Group (MHUG), University of Trento, Trento, Italy. His research interests are machine learning, geometric deep learning, matrix manifolds, and matrix Lie groups.

Yue Song received the B.Sc. cum laude from KU Leuven, Belgium and the joint M.S. summa cum laude from the University of Trento, Italy and KTH Royal Institute of Technology, Sweden. He is currently working toward the Ph.D. degree with the Multimedia and Human Understanding Group (MHUG), University of Trento, Trento, Italy. His research interests are computer vision, deep learning, and numerical analysis and optimization.

Tianyang Xu received the B.Sc. degree in electronic science and engineering from Nanjing University, Nanjing, China, in 2011. He received his Ph.D. degree at the School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, China, in 2019. He is currently an Associate Professor at the School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, China. His research interests include visual tracking and deep learning.

Zhiwu Huang received the B.Sc. degree in computer science and technology from Huaqiao University, Quanzhou, Fujian, China, in 2007, and the M.S. degree in computer software and theory from Xiamen University, Xiamen, Fujian, China, in 2010, and the Ph.D. degree in computer science and technology from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2015. He is currently a Lecturer affiliated with the Vision, Learning, and Control (VLC) group in the School of Electronics and Computer Science (ECS) at the University of Southampton. His research interests include computer vision, Riemannian computing, metric learning, and deep learning.

Xiao-Jun Wu received the B.Sc. degree in mathematics from Nanjing Normal University, Nanjing, China, in 1991. He received the M.S. degree and the Ph.D. degree in pattern recognition and intelligent systems from Nanjing University of Science and Technology, Nanjing, China, in 1996 and 2002, respectively. He is a Professor in artificial intelligence and pattern recognition at the Jiangnan University, Wuxi, China. His research interests include pattern recognition, computer vision, fuzzy systems, neural networks, and intelligent systems. He has won several domestic and international awards because of his research achievements. He served as an associate editor for several international journals. He is currently a Fellow of IAPR and AAIA.

Nicu Sebe is a professor at the University of Trento, Italy, leading the research in the areas of multimedia information retrieval and human behavior understanding. He was the General Co-Chair of the IEEE FG Conference 2008 and ACM Multimedia 2013; the Program Chair of the International Conference on Image and Video Retrieval in 2007 and 2010, ACM Multimedia 2007 and 2011, and the ICCV 2017 and ECCV 2016; and the General Chair of ACM ICMR 2017. He is a fellow of the IAPR.