A LIE GROUP APPROACH TO RIEMANNIAN BATCH NORMALIZATION

Ziheng Chen[1], **Yue Song**[1*], **Yunmei Liu**[2] **& Nicu Sebe**[1]
[1] University of Trento, [2] University of Louisville
ziheng_ch@163.com, yue.song@unitn.it

ABSTRACT

Manifold-valued measurements exist in numerous applications within computer vision and machine learning. Recent studies have extended Deep Neural Networks (DNNs) to manifolds, and concomitantly, normalization techniques have also been adapted to several manifolds, referred to as Riemannian normalization. Nonetheless, most of the existing Riemannian normalization methods have been derived in an *ad hoc* manner and only apply to specific manifolds. This paper establishes a unified framework for Riemannian Batch Normalization (RBN) techniques on Lie groups. Our framework offers the theoretical guarantee of controlling both the Riemannian mean and variance. Empirically, we focus on Symmetric Positive Definite (SPD) manifolds, which possess three distinct types of Lie group structures. Using the deformation concept, we generalize the existing Lie groups on SPD manifolds into three families of parameterized Lie groups. Specific normalization layers induced by these Lie groups are then proposed for SPD neural networks. We demonstrate the effectiveness of our approach through three sets of experiments: radar recognition, human action recognition, and electroencephalography (EEG) classification. The code is available at https://github.com/GitZH-Chen/LieBN.git.

1 INTRODUCTION

Over the past decade or so, Deep Neural Networks (DNNs) have achieved remarkable progress across various scientific fields (Hochreiter & Schmidhuber, 1997; Krizhevsky et al., 2012; He et al., 2016; Vaswani et al., 2017). Conventionally, DNNs have been developed with the underlying assumption of the Euclidean geometry inherent to input data. Nonetheless, there exists a plethora of applications wherein the latent spaces are defined by non-Euclidean structures such as manifolds (Bronstein et al., 2017). To address this issue, researchers have attempted to extend various types of DNNs to manifolds based on the theories of Riemannian geometry (Huang & Van Gool, 2017; Huang et al., 2017; 2018; Ganea et al., 2018; Chakraborty et al., 2018; Brooks et al., 2019a;e;c;d; Brooks, 2020; Brooks et al., 2020; Chen et al., 2020; Chakraborty et al., 2020; Chakraborty, 2020; Chen et al., 2021; Wang et al., 2022b;a; Nguyen, 2022a;b; Nguyen & Yang, 2023; Chen et al., 2023c;e;b; Wang et al., 2024).

Motivated by the great success of normalization techniques within DNNs (Ioffe & Szegedy, 2015; Ba et al., 2016; Ulyanov et al., 2016; Wu & He, 2018; Chen et al., 2023a), researchers have sought to devise normalization layers tailored for manifold-valued data. Brooks et al. (2019b) introduced Riemannian Batch Normalization (RBN) specifically designed for SPD manifolds, with the ability to regulate the Riemannian mean. This approach was further refined in Kobler et al. (2022b) to extend the control over the Riemannian variance. *However, the above methods are constrained within the affine-invariant metric (AIM) on SPD manifolds, limiting their applicability and generality.* On the other hand, Chakraborty (2020) proposed two distinct Riemannian normalization frameworks, one tailored for Riemannian homogeneous spaces and the other catering to matrix Lie groups. *Nonetheless, the normalization designed for Riemannian homogeneous spaces cannot regulate mean nor variance, while the normalization approach for matrix Lie groups is confined to a specific type of distance (Chakraborty, 2020, Sec. 3.2).* A principled Riemannian normalization framework capable of controlling both Riemannian mean and variance remains unexplored.

*Corresponding author

Given that Batch Normalization (BN) (Ioffe & Szegedy, 2015) serves as the foundational prototype for various types of normalization, our paper only concentrates on RBN currently and can be readily extended to other normalization techniques. As several manifold-valued measurements form Lie groups, including SPD manifolds, Special Orthogonal (SO) groups, and Special Euclidean (SE) groups, we further direct our attention towards Lie groups. We propose a general framework for RBN over Lie groups, referred to as LieBN, and validate our approach in normalizing both the Riemannian mean and variance. On the empirical side, we focus on SPD manifolds, where three distinct types of Lie groups have been recognized in the literature. We generalize these existing Lie groups into parameterized forms through the deformation concept. Then, we showcase our LieBN framework on SPD manifolds under these Lie groups and propose specific normalization layers. Extensive experiments conducted on widely-used SPD benchmarks demonstrate the effectiveness of our framework. We highlight that our work is entirely theoretically different from Brooks et al. (2019b); Kobler et al. (2022a); Lou et al. (2020), and more general than Chakraborty (2020). The previous RBN methods are either designed for a specific manifold or metric (Brooks et al., 2019b; Kobler et al., 2022a; Chakraborty, 2020), or fail to control mean and variance (Lou et al., 2020), while our LieBN guarantees the normalization of mean and variance on general Lie groups. In summary, our main contributions are as follows: **(a)** a general Lie group batch normalization framework with controllable first- and second-order statistical moments, and **(b)** the specific construction of our LieBN layers on SPD manifolds based on three deformed Lie groups and their application on SPD neural networks. Due to page limites, all the proofs are placed in App. I.

2 PRELIMINARIES

This section provides a brief review of the Lie group and the geometries of SPD manifolds. For more in-depth discussions, please refer to Tu (2011); Do Carmo & Flaherty Francis (1992).

Definition 2.1 (Lie Groups)**.** A manifold is a Lie group, if it forms a group with a group operation $\odot$ such that $m(x, y) \mapsto x \odot y$ and $i(x) \mapsto x_{\odot}^{-1}$ are both smooth, where $x_{\odot}^{-1}$ is the group inverse.

Definition 2.2 (Left-invariance)**.** A Riemannian metric g over a Lie group $\{G, \odot\}$ is left-invariant, if for any $x, y \in G$ and $V_1, V_2 \in T_x\mathcal{M}$,

$$g_y(V_1, V_2) = g_{L_x(y)}(L_{x*,y}(V_1), L_{x*,y}(V_2)), \tag{1}$$

where $L_x(y) = x \odot y$ is the left translation by x, and $L_{x*,y}$ is the differential map of L_x at y.

A Lie group is a group and also a manifold. The most natural Riemannian metric on a Lie group is the left-invariant metric[1]. Similarly, one can define the right-invariant metric as Def. 2.2. A bi-invariant Riemannian metric is the one with both left and right invariance. Given the analogous properties of left and right-invariant metrics, this paper focuses on left-invariant metrics.

The idea of pullback is ubiquitous in differential geometry and can be considered as a natural counterpart of bijection in the set theory.

Definition 2.3 (Pullback Metrics)**.** Suppose $\mathcal{M}_1, \mathcal{M}_2$ are smooth manifolds, g is a Riemannian metric on $\mathcal{M}_2$, and $f : \mathcal{M}_1 \to \mathcal{M}_2$ is smooth. Then the pullback of g by f is defined point-wisely,

$$(f^*g)_p(V, W) = g_{f(p)}(f_{*,p}(V), f_{*,p}(W)), \tag{2}$$

where $p \in \mathcal{M}$, $f_{*,p}(\cdot)$ is the differential map of f at p, and $V, W \in T_p\mathcal{M}$. If f^*g is positive definite, it is a Riemannian metric on $\mathcal{M}_1$, called the pullback metric defined by f.

The most common pullback metric is obtained through the pullback of a diffeomorphism f. Additionally, if $\{\mathcal{M}_2, \odot_2\}$ constitutes a Lie group, the diffeomorphism f can pull back the group operation $\odot_2$ to $\odot_1$ on $\mathcal{M}_1$, which is defined as $\forall P, Q \in \mathcal{M}_1, P \odot_1 Q = f^{-1}(f(P) \odot_2 f(Q))$. Henceforth, $\{\mathcal{M}, \odot, g\}$, abbreviated as $\mathcal{M}$, always signify a Lie group with left-invariant metric.

Now, we briefly review the geometry of SPD manifolds. We denote $n \times n$ SPD matrices as $\mathcal{S}^n_{++}$ and $n \times n$ real symmetric matrices as $\mathcal{S}^n$. As shown in Arsigny et al. (2005), $\mathcal{S}^n_{++}$ forms a manifold, known as the SPD manifold, and $\mathcal{S}^n$ is a Euclidean space. SPD manifolds exhibit three Lie group structures, each associated with an invariant metric. These metrics include the Log-Euclidean

[1] Left invariant metric always exists for every Lie group (Do Carmo & Flaherty Francis, 1992).

Metric (LEM) (Arsigny et al., 2005), Affine-Invariant Metric (AIM) (Pennec et al., 2006), and Log-Cholesky Metric (LCM) (Lin, 2019). In Thanwerdas & Pennec (2023), LEM and AIM are generalized into two-parameter families of metrics, denoted as (α, β)-LEM and (α, β)-AIM, respectively. Both (α, β)-LEM and (α, β)-AIM are defined by the $\mathrm{O}(n)$-invariant inner product on the tangent space at the identity matrix, expressed as:

$$\langle V, W \rangle^{(\alpha,\beta)} = \alpha \langle V, W \rangle + \beta \, \mathrm{tr}(V) \, \mathrm{tr}(W), \tag{3}$$

where $\langle \cdot, \cdot \rangle$ is the Frobenius inner product, $V, W \in T_I \mathcal{S}^n_{++} \cong \mathcal{S}^n$, $(\alpha, \beta) \in \mathbf{ST} = \{(\alpha, \beta) \in \mathbb{R}^2 \mid \min(\alpha, \alpha + n\beta) > 0\}$.

In fact, (α, β)-LEM, (α, β)-AIM, and LCM are all pullback metrics. Specifically, (α, β)-LEM is the pullback metric from $\mathcal{S}^n$ (Chen et al., 2023c), while (α, β)-AIM is the pullback metric from a left-invariant metric on the Cholesky manifold (Thanwerdas & Pennec, 2022b). Additionally, as shown in Chen et al. (2023d), LCM is the pullback metric from the Euclidean space $\mathcal{L}^n$ of lower triangular matrices by the diffeomorphism defined as

$$\psi_{\mathrm{LC}}(P) = \lfloor L \rfloor + \mathrm{Dlog}(L), \tag{4}$$

where $P = LL^\top$ represents the Cholesky decomposition, $\lfloor \cdot \rfloor$ is the strictly lower part of a square matrix, and $\mathrm{Dlog}(L)$ is a diagonal matrix consisting of the logarithm of the diagonal element of L.

Let $\{w_{1 \ldots N}\}$ be weights satisfying a convexity constraint, *i.e.*,$\forall i, w_i > 0$ and $\sum_i w_i = 1$. The weighted Fréchet mean (WFM) of a set of SPD matrices $\{P_{i \ldots N}\}$ is defined as

$$\mathrm{WFM}(\{w_i\}, \{P_i\}) = \underset{S \in \mathcal{S}^n_{++}}{\mathrm{argmin}} \sum\nolimits_{i=1}^{N} w_i \, \mathrm{d}^2 \left(P_i, S\right), \tag{5}$$

where $\mathrm{d}(\cdot, \cdot)$ denotes the geodesic distance. When $w_i = 1/N$ for all i, then Eq. (5) is reduced to the Fréchet mean, denoted as $\mathrm{FM}(\{P_i\})$. The Fréchet variance v^2 is the attained value at the minimizer of the Fréchet mean. In this paper, we will interchangeably use the terms Riemannian mean with Fréchet mean, and Riemannian variance with Fréchet variance. For (α, β)-LEM and LCM, the Fréchet mean has a closed-form expression. Moreover, since (α, β)-AIM has non-positive sectional curvature (Thanwerdas & Pennec, 2023, Tab. 5), the Fréchet mean exists uniquely (Berger, 2003, 6.1.5) and can be computed by the Karcher flow algorithm (Karcher, 1977).

Given SPD matrices $P, Q \in \mathcal{S}^n_{++}$ along with tangent vectors $V, W \in T_P \mathcal{S}^n_{++}$, we introduce the following notations. Specifically, we denote $g_P(\cdot, \cdot)$ as the Riemannian metric at P, $\mathrm{Log}_P(\cdot)$ as the Riemannian logarithm at P, $\mathrm{mexp}(\cdot)$ and $\mathrm{mlog}(\cdot)$ as the matrix exponentiation and logarithm, and $\mathrm{d}(\cdot, \cdot)$ as the geodesic distance, respectively. Additionally, $\mathrm{Chol}(\cdot)$ signifies the Cholesky decomposition. We use $L = \mathrm{Chol}(P)$ and $K = \mathrm{Chol}(Q)$ to denote the Cholesky factor of P and Q. $\mathbb{K}$ and $\mathbb{L}$ are diagonal matrices with diagonal elements from K and L. $\mathrm{mlog}_{*,P}$ and $(\mathrm{Chol})^{-1}_{*,L}$ represent the differentials of mlog and Chol^{-1} at P and L. We denote $\| \cdot \|^{(\alpha,\beta)}$ and $\| \cdot \|_{\mathrm{F}}$ as the norm induced by $\langle \cdot, \cdot \rangle$ and the standard Frobenius norm. We summarize all the necessary ingredients in Tab. 1.

Table 1: Lie group structures and the associated Riemannian operators on SPD manifolds.

Metric	(α, β)-LEM	(α, β)-AIM	LCM
$g_P(V, W)$	$\langle \mathrm{mlog}_{*,P}(V), \mathrm{mlog}_{*,P}(W) \rangle^{(\alpha,\beta)}$	$\langle P^{-1}V, WP^{-1} \rangle^{(\alpha,\beta)}$	$\sum_{i>j} V_{ij} W_{ij} + \sum_{j=1}^{n} V_{jj} W_{jj} L_{jj}^{-2}$
$\mathrm{d}(P, Q)$	$\lVert \mathrm{mlog}(P) - \mathrm{mlog}(Q) \rVert^{(\alpha,\beta)}$	$\left\lVert \mathrm{mlog}\left(Q^{-\frac{1}{2}} P Q^{-\frac{1}{2}}\right) \right\rVert^{(\alpha,\beta)}$	$\lVert \psi_{\mathrm{LC}} \circ \mathrm{Chol}(P) - \psi_{\mathrm{LC}} \circ \mathrm{Chol}(Q) \rVert_{\mathrm{F}}$
$Q \odot P$	$\mathrm{mexp}(\mathrm{mlog}(P) + \mathrm{mlog}(Q))$	$KPK^\top$	$\mathrm{Chol}^{-1}(\lfloor L + K \rfloor + \mathbb{K}\mathbb{L})$
$\mathrm{FM}(\{P_i\})$	$\mathrm{mexp}\left(\frac{1}{n} \sum_i \mathrm{mlog}\, P_i\right)$	Karcher Flow	$\psi_{\mathrm{LC}}^{-1}\left(\frac{1}{n} \sum_i \psi_{\mathrm{LC}}(P_i)\right)$
$\mathrm{Log}_P\, Q$	$(\mathrm{mlog}_{*,P})^{-1} \left[\mathrm{mlog}(Q) - \mathrm{mlog}(P)\right]$	$P^{\frac{1}{2}} \mathrm{mlog}\left(P^{\frac{-1}{2}} Q P^{\frac{-1}{2}}\right) P^{\frac{1}{2}}$	$(\mathrm{Chol}^{-1})_{*,L} \left[\lfloor K \rfloor - \lfloor L \rfloor + \mathbb{L}\, \mathrm{Dlog}(\mathbb{L}^{-1}\mathbb{K})\right]$
Invariance	Bi-invariance	Left-invariance	Bi-invariance

3 Revisiting Normalization

3.1 Revisiting Euclidean Normalization

In Euclidean DNNs, normalization stands as a pivotal technique for accelerating network training by mitigating the issue of internal covariate shift (Ioffe & Szegedy, 2015). While various normalization

methods have been introduced (Ioffe & Szegedy, 2015; Ba et al., 2016; Ulyanov et al., 2016; Wu & He, 2018), they all share a common fundamental concept: the regulation of the first and second statistical moments. In this paper, we focus on batch normalization only.

Given a batch of activations $\{x_{i...N}\}$, the core operation of batch normalization can be expressed as:

$$\forall i \leq N, x_i \leftarrow \gamma \frac{x_i - \mu_b}{\sqrt{v_b^2 + \epsilon}} + \beta \tag{6}$$

where μ_b is the batch mean, v_b^2 is the batch variance, γ is the scaling parameter, and β is the biasing parameter.

3.2 Revisiting Existing RBN

Inspired by the remarkable success of normalization techniques in traditional DNNs, endeavors have been made to develop Riemannian normalization approaches tailored for manifolds. Here we recap some representative methods. However, we note that none of the existing methods effectively handle the first and second moments in a principled manner.

Brooks et al. (2019b) introduced RBN over SPD manifolds under AIM, with operations defined as:

$$\text{Centering from geometric mean } M : \forall i \leq N, \bar{P}_i \leftarrow M^{-\frac{1}{2}} P_i M^{-\frac{1}{2}}, \tag{7}$$

$$\text{Biasing towards SPD parameter } B : \forall i \leq N, \hat{P}_i \leftarrow B^{\frac{1}{2}} \bar{P}_i B^{\frac{1}{2}}, \tag{8}$$

where $\{P_{i...N}\}$ are SPD matrices, and M are their Fréchet mean under AIM. Define a map as

$$\Gamma_{P \to Q}(S) = \mathrm{Exp}_Q \left[\mathrm{PT}_{P \to Q} \left(\mathrm{Log}_P(S) \right) \right], \tag{9}$$

where $P, Q, S \in \mathcal{S}^n_{++}$, and $\mathrm{Exp}, \mathrm{Log}, \mathrm{PT}$ are Riemannian exponential map, Riemannian logarithmic map, and parallel transportation along the geodesic, respectively. Then under AIM, Eqs. (7) and (8) can be more generally expressed as

$$\Gamma_{I \to B}[\Gamma_{M \to I}(P_i)]. \tag{10}$$

However, Eqs. (7) and (8) only consider the Riemannian mean[2] and does not consider the Riemannian variance. To remedy this limitation, Kobler et al. (2022b) further improved the RBN to enable control over the Riemannian variance. The key operation is formulated as

$$\forall i \leq N, \bar{P}_i \leftarrow \Gamma_{I \to B}[(\Gamma_{M \to I}(P_i))^{\frac{s}{v}}], \tag{11}$$

where v^2 is the Fréchet variance, $s \in \mathbb{R}$ is a scaling factor. However, this method is still limited to SPD manifolds under AIM. In parallel, Chakraborty (2020, Algs. 1-2) proposed a general framework for Riemannian homogeneous spaces based on Eq. (10) by considering both first and second moments. However, as stated in Chakraborty (2020, Sec. 3.1), Eq. (10) does not generally guarantee the control over Riemannian mean, resulting in agnostic Riemannian statistics. To mitigate this limitation, Chakraborty (2020, Algs. 3-4) further proposed normalization over the matrix Lie group. However, the discussion is limited to a certain distance, limiting the applicability of their method. On the other hand, Lou et al. (2020) proposed an RBN based on a variance of Eq. (10). Although their framework encompasses the standard Euclidean BN when the latent geometry is the standard Euclidean geometry, their approach suffers from the same problem of agnostic Riemannian statistics on general manifolds.

In summary, prevailing Riemannian normalization approaches lack a principled guarantee for controlling the first and second-order statistics. In contrast, as will be elucidated, our method can govern first and second-order statistics for all Lie groups. We summarize the above RBN methods in Tab. 2.

4 Riemannian Normalization on Lie Groups

In this section, the neutral element in the Lie group $\mathcal{M}$ is denoted as E. Notably, the neutral element E is not necessarily the identity matrix. We first clarify the essential properties of Euclidean BN and then present our normalization method, tailored for Lie groups.

[2] Although not mentioned in the original paper, Eqs. (7) and (8) can guarantee the mean of the resulting samples under AIM, as Eqs. (7) and (8) are actions of $\mathrm{GL}(n)$.

Table 2: Summary of some representative RBN methods.

Methods	Involved Statistics	Controllable Mean	Controllable Variance	Application Scenarios
SPDBN (Brooks et al., 2019b)	Mean	✓	N/A	SPD manifolds under AIM
SPDBN (Kobler et al., 2022b)	Mean+Variance	✓	✓	SPD manifolds under AIM
Chakraborty (2020, Algs. 1-2)	Mean+Variance	✗	✗	Riemannian homogeneous space
Chakraborty (2020, Algs. 3-4)	Mean+Variance	✓	✓	A certain Lie group structure and distance
RBN (Lou et al., 2020, Alg. 2)	Mean+Variance	✗	✗	Geodesically complete manifolds
Ours	Mean+Variance	✓	✓	General Lie groups

Two key points regarding Euclidean BN, as expressed by Eq. (6), are worth highlighting: (a) The standard BN (Ioffe & Szegedy, 2015) implicitly assumes a Gaussian distribution and can effectively normalize and transform the latent Gaussian distribution; (b) The centering and biasing operations control the mean, while the scaling controls the variance. Therefore, extending BN into Lie groups requires defining Gaussian distribution, centering, biasing, and scaling on Lie groups.

On manifolds, several notions of Gaussian distribution have been proposed (Pennec, 2004; Chakraborty & Vemuri, 2019; Barbaresco, 2021). In this work, we adopt the definition from Chakraborty & Vemuri (2019), which characterizes a Gaussian distribution on the Lie group $\mathcal{M}$ with a mean parameter $M \in \mathcal{M}$ and variance σ^2. This distribution is denoted as $\mathcal{N}(M, \sigma^2)$, and its Probability Density Function (P.D.F.) is defined as[3]:

$$p\left(X \mid M, \sigma^2\right) = k(\sigma) \exp\left(-\frac{\mathrm{d}(X, M)^2}{2\sigma^2}\right), \tag{12}$$

where $k(\sigma)$ is the normalizing constant, $\exp(\cdot)$ is the scalar exponentiation, and $\mathrm{d}(\cdot, \cdot)$ is the geodesic distance. On Lie groups, the natural counterparts of addition and subtraction in Eq. (6) are group operations. Therefore, centering and biasing on Lie groups can be defined by Lie group left translation. Additionally, scaling can be defined by scaling on the tangent space at the Riemannian mean.

Now, we can extend Eq. (6) into Lie groups $\mathcal{M}$. For a batch of activation $\{P_{i\ldots N} \in \mathcal{M}\}$, we define the key operations of Lie group BN (LieBN) as:

$$\text{Centering to the neutral element } E\text{: } \forall i \leq N, \bar{P}_i \leftarrow L_{M_{\odot}^{-1}}(P_i), \tag{13}$$

$$\text{Scaling the dispersion: } \forall i \leq N, \hat{P}_i \leftarrow \mathrm{Exp}_E\left[\frac{s}{\sqrt{v^2+\epsilon}} \mathrm{Log}_E(\bar{P}_i)\right], \tag{14}$$

$$\text{Biasing towards parameter } B \in \mathcal{M}\text{: } \forall i \leq N, \tilde{P}_i \leftarrow L_B(\hat{P}_i), \tag{15}$$

where M is the Fréchet mean, v^2 is the Fréchet variance, $M_{\odot}^{-1} \in \mathcal{M}$ is the group inverse of M, $L_{M_{\odot}^{-1}}$ and L_G are left translations ($L_G(P_i) = G \odot P_i$), and $s \in \mathbb{R}/\{0\}$ is a scaling parameter. To clarify the effect of our method in controlling mean and variance, we first present the following two propositions, one related to population statistics and the other related to sample statistics.

Proposition 4.1 (Population)**.** [↓] *Given a random point X over $\mathcal{M}$, and the Gaussian distribution $\mathcal{N}(M, v^2)$ defined in Eq.* (12)*, we have the following properties for the population statistics:*

1. *(MLE of M) Given $\{P_{i\ldots N} \in \mathcal{M}\}$ i.i.d. sampled from $\mathcal{N}(M, v^2)$, the maximum likelihood estimator (MLE) of M is the sample Fréchet mean.*
2. *(Homogeneity) Given $X \sim \mathcal{N}(M, v^2)$ and $B \in \mathcal{M}$, $L_B(X) \sim \mathcal{N}(L_B(M), v^2)$*

Proposition 4.2 (Sample)**.** [↓] *Given N samples $\{P_{i\ldots N} \in \mathcal{M}\}$, denoting $\phi_s(P_i) = \mathrm{Exp}_E\left[s \,\mathrm{Log}_E(P_i)\right]$, we have the following properties for the sample statistics:*

$$\textit{Homogeneity of the sample mean: } \mathrm{FM}\{L_B(P_i)\} = L_B(\mathrm{FM}\{P_i\}), \forall B \in \mathcal{M}, \tag{16}$$

$$\textit{Controllable dispersion from } E\textit{: } \sum\nolimits_{i=1}^{N} w_i \,\mathrm{d}^2(\phi_s(P_i), E) = s^2 \sum\nolimits_{i=1}^{N} w_i \,\mathrm{d}^2(P_i, E), \tag{17}$$

where $\{w_{1\ldots N}\}$ are weights satisfying a convexity constraint, i.e.,$\forall i, w_i > 0$ *and* $\sum_i w_i = 1$.

[3]When $\mathcal{M}$ corresponds to $\mathbb{R}$ equipped with the standard Euclidean metric, Eq. (12) reduces to the Euclidean Gaussian distribution.

Prop. 4.1 and Eq. (16) imply that our centering and biasing in Eqs. (13) and (15) can control the sample and population mean. As the post-centering mean is E, Eq. (17) implies that Eq. (14) can control the dispersion. Although the population variance after Eq. (14) is generally agnostic, in some cases such as SPD manifolds under LEM and LCM, Eq. (14) can normalize the population variance. Please refer to App. C for technical details.

Similar to Ioffe & Szegedy (2015), we use moving averages to update running statistics. Now, we present our general framework of LieBN in Alg. 1. Importantly, when $\mathcal{M} = \mathbb{R}^n$, Alg. 1 is reduced to the standard Euclidean BN. More details are exposed in App. D.

Remark 4.3. The MLE of the mean of the Gaussian distribution in Eq. (12) have been examined in several previous works (Said et al., 2017; Chakraborty & Vemuri, 2019; Chakraborty, 2020). However, these studies primarily focus on particular manifolds or specific metrics. ***In contrast, our contribution lies in presenting a universally applicable result for all Lie groups with left-invariant metrics.*** While Eq. (12) briefly appeared in Kobler et al. (2022b), the authors only focus on SPD manifolds under AIM. The transformation of the population under their proposed RBN remains unexplored as well. Besides, while Chakraborty (2020) analyzed the population properties for their RBN over matrix Lie groups, their results were confined within a specific distance. In contrast, our work provides a more extensive examination, encompassing both population and sample properties of our LieBN in a general manner. Besides, all the discussion about our LieBN can be readily transferred to right-invariant metrics. This paper focuses on LieBN based on left-invariant metrics.

Algorithm 1: Lie Group Batch Normalization (LieBN) Algorithm

Input : A batch of activations $\{P_{1\ldots N} \in \mathcal{M}\}$, a small positive constant ϵ, and momentum $\gamma \in [0, 1]$
running mean $M_r = E$, running variance $v_r^2 = 1$,
biasing parameter $B \in \mathcal{M}$, scaling parameter $s \in \mathbb{R}/\{0\}$,

Output : Normalized activations $\{\tilde{P}_{1\ldots N}\}$

if *training* **then**
  Compute batch mean M_b and variance v_b^2 of $\{P_{1\ldots N}\}$;
  Update running statistics $M_r \leftarrow \mathrm{WFM}(\{1-\gamma, \gamma\}, \{M_r, M_b\})$, $v_r^2 \leftarrow (1-\gamma)v_r^2 + \gamma v_b^2$;
end
if *training* **then** $M \leftarrow M_b, v^2 \leftarrow v_b^2$;
else $M \leftarrow M_r, v^2 \leftarrow v_r^2$;
for $i \leftarrow 1$ **to** N **do**
  Centering to the neutral element E: $\bar{P}_i \leftarrow L_{M_{\odot}^{-1}}(P_i)$
  Scaling the dispersion: $\hat{P}_i \leftarrow \mathrm{Exp}_E\left[\frac{s}{\sqrt{v^2+\epsilon}}\mathrm{Log}_E(\bar{P}_i)\right]$
  Biasing towards parameter B: $\tilde{P}_i \leftarrow L_B(\hat{P}_i)$
end

5 LieBN on the Lie Groups of SPD Manifolds

This section showcases our Alg. 1 on SPD manifolds. Firstly, we extend the current Lie groups on SPD manifolds by the concept of deformation, resulting in three families of parameterized Lie groups. Subsequently, we construct LieBN layers based on these generalized Lie groups.

5.1 Deformed Lie Groups of SPD Manifolds

As shown in Tab. 1, there are three types of Lie groups on SPD manifolds, each equipped with a left-invariant metric. These metrics include (α, β)-AIM, (α, β)-LEM, and LCM. For clarity, we denote the group operations w.r.t. (α, β)-AIM, (α, β)-LEM and LCM as $\odot^{\mathrm{AI}}$, $\odot^{\mathrm{LE}}$ and $\odot^{\mathrm{LC}}$, respectively.

In Thanwerdas & Pennec (2019b), (α, β)-AIM is further extended into three-parameters families of metrics by the pullback of matrix power function $\mathrm{P}_\theta(\cdot)$ and scaled by $\frac{1}{\theta^2}$, denoted as (θ, α, β)-AIM. The power function serves as a deformation wherein (θ, α, β)-AIM encompasses (α, β)-AIM when $\theta = 1$, and includes (α, β)-LEM as θ approaches 0 (Thanwerdas & Pennec, 2019a). Inspired by the deforming utility of the power function, we define the power-deformed metrics of (α, β)-LEM and LCM as the pullback metrics by P_θ and scaled by $\frac{1}{\theta^2}$. We denote these two metrics as (θ, α, β)-LEM and θ-LCM, respectively. We have the following results w.r.t. the deformation.

Proposition 5.1 (Deformation). [↓] *(θ,α,β)-LEM is equal to (α,β)-LEM. θ-LCM interpolates between $\tilde{g}$-LEM ($\theta=0$) and LCM ($\theta=1$), with $\tilde{g}$-LEM defined as*

$$\langle V,W\rangle_P=\tilde{g}(\mathrm{mlog}_{*,P}(V),\mathrm{mlog}_{*,P}(W)),\forall P\in\mathcal{S}^n_{++},\forall V,W\in T_P\mathcal{S}^n_{++}, \tag{18}$$

where $\tilde{g}(V_1,V_2)=\frac{1}{2}\langle V_1,V_2\rangle-\frac{1}{4}\langle\mathbb{D}(V_1),\mathbb{D}(V_2)\rangle$, $\mathbb{D}(V_i)$ is a diagonal matrix consisting of the diagonal elements of V_i, and $\mathrm{mlog}_{*,P}$ is the differential map at P.

As (θ,α,β)-LEM is equal to (α,β)-LEM, in the following, we focus on (α,β)-LEM, (θ,α,β)-AIM, and θ-LCM. As a diffeomorphism, P_θ also can pull back the group operation $\odot^{\mathrm{AI}}$ and $\odot^{\mathrm{LC}}$, denoted as $\odot^{\theta\text{-AI}}$ and $\odot^{\theta\text{-LC}}$. We have the following proposition on the invariance.

Proposition 5.2 (Invariance). [↓] *(θ,α,β)-AIM is left-invariant w.r.t. $\odot^{\theta\text{-AI}}$, while θ-LCM is bi-invariant w.r.t. $\odot^{\theta\text{-LC}}$.*

5.2 LieBN on SPD Manifolds

Now, we showcase our LieBN framework illustrated in Alg. 1 on SPD manifolds. As discussed in Sec. 5.1, there are three families of left-invariant metrics, namely (θ,α,β)-AIM, (α,β)-LEM, and θ-LCM. Since all three metric families are pullback metrics, the LieBN based on these metrics can be simplified and calculated in the co-domain. We denote Alg. 1 as

$$\mathrm{LieBN}(P_i;B,s,\epsilon,\gamma),\forall p_i\in\{P_{1\dots N}\in\mathcal{M}\}. \tag{19}$$

Then we can obtain the following theorem.

Theorem 5.3. [↓] *Given a Lie group $\mathcal{M}_1$, a Lie group $\mathcal{M}_2$ with a left-invariant metric g^2, and a diffeomorphism $f:\mathcal{M}_1\to\mathcal{M}_2$, then f induces a left-invariant metric g^1 on $\mathcal{M}_1$, denoted as $g^1=f^*g^2$. For a batch of activation $\{P_{1\dots N}\}$ in $\mathcal{M}_1$, $\mathrm{LieBN}(P_i;B,s,\epsilon,\gamma)$ in $\mathcal{M}_1$ can be calculated in $\mathcal{M}_2$ by the following process:*

$$\textit{Mapping data into } \mathcal{M}_2: \bar{P}_i=f(P_i),\bar{B}=f(B), \tag{20}$$

$$\textit{Calculating LieBN in } \{\mathcal{M}_2,g^2\}: \hat{P}_i=\mathrm{LieBN}(\bar{P}_i;\bar{B},s,\epsilon,\gamma), \tag{21}$$

$$\textit{Mapping the normalized data back to } \mathcal{M}_1: \tilde{P}_i=f^{-1}(\hat{P}_i), \tag{22}$$

Given a metric g on $\mathcal{S}^n_{++}$, the power-deformed metric $\tilde{g}=\frac{1}{\theta^2}\,\mathrm{P}_\theta^*\,g$ is equal to $\mathrm{P}_\theta^*(\frac{1}{\theta^2}g)$. Therefore, the LieBN under $\tilde{g}$ can be calculated by the LieBN under $\frac{1}{\theta^2}g$. Besides, as the Christoffel symbols remain the same under constant scaling, the LieBNs under $\frac{1}{\theta^2}g$ and g only differ in the variance. We denote $g^{(\alpha,\beta)\text{-AI}}$ and $g^{(\theta,\alpha,\beta)\text{-AI}}$ as the metric tensors of (α,β)-AIM and (θ,α,β)-AIM, respectively Based on the above discussions, the computations of the LieBN under $g^{(\theta,\alpha,\beta)\text{-AI}}$ are reduced to the LieBN under $\frac{1}{\theta^2}g^{(\alpha,\beta)\text{-AI}}$. Furthermore, as shown in Chen et al. (2023c), (α,β)-LEM is a pullback metric from the Euclidean space $\mathcal{S}^n$ of symmetric matrices, while the proof of Prop. 5.2 indicates that θ-LCM is a pullback metric from the Euclidean space $\mathcal{L}^n$ of lower triangular matrices. Note that in the Euclidean space $\mathcal{S}^n$ or $\mathcal{L}^n$, as shown in App. D, Eq. (21) simplifies to the standard Euclidean BN. We denote P,Q,P_1 and P_2 as points in the codomain ($\mathcal{S}^n_{++}$ with (α,β)-AIM for (θ,α,β)-AIM, $\mathcal{S}^n$ for (α,β)-LEM, and $\mathcal{L}^n$ for θ-LCM, respectively). We summarize all the necessary ingredients in Tab. 3 for calculating LieBN on SPD manifolds. Note that for (θ,α,β)-AIM, our scaling operation defined in Eq. (14) encompasses the scaling operation in Kobler et al. (2022b, Eq. (9)) as a special case, when $(\theta,\alpha,\beta)=(1,1,0)$.

6 Experiments

In this section, we implement our three families of LieBN to SPD neural networks. Following the previous work (Huang & Van Gool, 2017; Brooks et al., 2019b; Kobler et al., 2022a), we adopt three different applications: radar recognition on the Radar dataset (Brooks et al., 2019b), human action recognition on the HDM05 (Müller et al., 2007) and FPHA (Garcia-Hernando et al., 2018) datasets, and EEG classification on the Hinss2021 dataset (Hinss et al., 2021). More details on datasets and hyper-parameters are exposed in App. G. Besides SPD neural networks, we also implement LieBN on special orthogonal groups and present some preliminary experiments (see App. H).

Implementation details: Note that our LieBN layers are architecture-agnostic and can be applied to any existing SPD neural network. In this paper, we focus on two network architectures: SPDNet

Table 3: Key operators in calculating LieBN on SPD manifolds.

Metric		(θ,α,β)-AIM	(α,β)-LEM	θ-LCM
Pullback Map		P_θ	mlog	$\mathrm{P}_\theta \circ \psi_{\mathrm{LC}}$
Codomain		$\{\mathcal{S}^n_{++}, \odot^{\mathrm{AI}}, \frac{1}{\theta^2} g^{(\alpha,\beta)\text{-AI}}\}$	$\{\mathcal{S}^n, \langle\cdot,\cdot\rangle^{(\alpha,\beta)}\}$	$\{\mathcal{L}^n, \frac{1}{\theta^2}\langle\cdot,\cdot\rangle\}$
Rieman-nian and Lie group operators in the codomain	$L_Q(P)$	$KPK^\top$	$P+Q$	$P+Q$
	$L_{Q^{-1}_{\odot}}(P)$	$K^{-1}PK^{-\top}$	$P-Q$	$P-Q$
	$\mathrm{Exp}_E\left[s\,\mathrm{Log}_E(P)\right]$	P^s	sP	sP
	FM	Karcher Flow	Arithmetic average	Arithmetic average
	$\mathrm{WFM}(\{\gamma, 1-\gamma\},\{P_1,P_2\})$	$P_2^{\frac{1}{2}}\left(P_2^{-\frac{1}{2}}P_1P_2^{-\frac{1}{2}}\right)^\gamma P_2^{\frac{1}{2}}$	Arithmetic weighted average	Arithmetic weighted average

Table 4: 10-fold average results of SPDNet with and without SPDBN or LieBN on the Radar, HDM05, and FPHA datasets. For simplicity, LieBN-Metric-(θ) is abbreviated as Metric-(θ). For the LieBN under each metric, if the LieBN induced by the standard metric ($\theta = 1$) is not saturated, we report the LieBN under the deformed metric in the rightmost columns of the table.

(a) Radar dataset.

Method	SPDNet	SPDNetBN	AIM-(1)	LEM-(1)	LCM-(1)	LCM-(-0.5)
Fit Time (s)	0.98	1.56	1.62	1.28	1.11	1.43
Mean±STD	93.25±1.10	94.85±0.99	**95.47±0.90**	94.89±1.04	93.52±1.07	94.80±0.71
Max	94.4	96.13	96.27	**96.8**	95.2	95.73

(b) HDM05 and FPHA datasets.

Method		SPDNet	SPDNetBN	AIM-(1)	LEM-(1)	LCM-(1)	AIM-(1.5)	LCM-(0.5)
HDM05	Fit Time (s)	0.57	0.97	1.14	0.87	0.66	1.46	1.01
	Mean±STD	59.13±0.67	66.72±0.52	67.79±0.65	65.05±0.63	66.68±0.71	68.16±0.68	**70.84±0.92**
	Max	60.34	67.66	68.75	66.05	68.52	69.25	**72.27**
FPHA	Fit Time (s)	0.32	0.62	0.80	0.55	0.39	1.03	0.65
	Mean±STD	85.59±0.72	89.33±0.49	89.70±0.51	86.56±0.79	77.64±1.00	**90.39±0.66**	86.33±0.43
	Max	86	90.17	90.5	87.83	79	**92.17**	87

(Huang & Van Gool, 2017) for the Radar, HDM05, and FPHA datasets; and TSMNet (Kobler et al., 2022a) for the Hinss2021 dataset. For the SPDNet architecture, we compare our LieBN with SPDNetBN (Brooks et al., 2019b), which applies the SPDBN (Eqs. (7) and (8)) to SPDNet. Consistent with SPDNetBN, we apply our LieBN after each transformation layer (BiMap layer in App. B). In the EEG application, the state-of-the-art Riemannian method is TSMNet with SPD domain-specific momentum batch normalization (TSMNet+SPDDSMBN) (Kobler et al., 2022a), which is a domain adaptation version of Kobler et al. (2022b). For a fair comparison, we also implement a domain-specific momentum LieBN, referred to as DSMLieBN (detailed in App. E). Following Kobler et al. (2022a), we apply our DSMLieBN before the LogEig layer (detailed in Appendix B) in TSMNet. We use the standard cross-entropy loss as the training objective and optimize the parameters with the Riemannian AMSGrad optimizer (Bécigneul & Ganea, 2018). The network architectures are represented as $\{d_0, d_1, \ldots, d_L\}$, where the dimension of the parameter in the i-th BiMap layer is $d_i \times d_{i-1}$. The experiments are conducted with a learning rate of $5e^{-3}$, batch size of 30, and training epoch of 200 on the Radar, HDM05, and FPHA datasets. For the Hinss2021 dataset, following Kobler et al. (2022a), we use a learning rate of $1e^{-3}$ with a weight decay of $1e^{-4}$, a batch size of 50, and a training epoch of 50. All experiments use an Intel Core i9-7960X CPU with 32GB RAM and an NVIDIA GeForce RTX 2080 Ti GPU. Evaluation methods are explained in App. G.3.

6.1 Experimental Results

For each family of LieBN or DSMLieBN, we report two representatives: the standard one induced from the standard metric ($\theta = 1$), and the one induced from the deformed metric with selected θ. *If the standard one is already saturated, we only report the results of the standard ones.*

Application to SPDNet: As SPDNet is the most classic SPD network, we apply our LieBN to SPDNet on the Radar, HDM05, and FPHA datasets. Additionally, we compare our method with SPDNetBN, which applies the SPDBN in Eqs. (7) and (8) to SPDNet. Following Brooks et al.

Table 5: Cross-validation results of TSMNet with SPDDSMBN and DSMLieBN on the Hinss dataset. For simplicity, DSMLieBN-Metric-(θ) is abbreviated as Metric-(θ). For the DSMLieBN under each metric, if the DSMLieBN induced by the standard metric ($\theta = 1$) is not saturated, we report the DSMLieBN under the deformed metric at the bottom rows of the table.

(a) Inter-session classification

Method	Fit Time (s)	Mean±STD
SPDDSMBN	0.16	54.12±9.87
AIM-(1)	0.16	**55.10±7.61**
LEM-(1)	0.13	54.95±10.09
LCM-(1)	0.10	51.54±6.88
LCM-(0.5)	0.15	53.11±5.65

(b) Inter-subject classification

Method	Fit Time (s)	Mean±STD
SPDDSMBN	7.74	50.10±8.08
AIM-(1)	6.94	50.04±8.01
LEM-(1)	4.71	50.95±6.40
LCM-(1)	3.59	51.86±4.53
AIM-(-0.5)	8.71	**53.97±8.78**

(2019b); Chen et al. (2023d), we use the architectures of $\{20, 16, 8\}$, $\{93, 30\}$, and $\{63, 33\}$ for the Radar, HDM05 and FPHA datasets, respectively. The 10-fold average results, including the average training time (s/epcoh), are summarized in Tab. 4. We have three key observations regarding the choice of metrics, deformation, and training efficiency. **The choice of metrics:** The metric that yields the most effective LieBN layer differs for each dataset. Specifically, the optimal LieBN layers on these three datasets are the ones induced by AIM-(1), LCM-(0.5), and AIM-(1.5), respectively, **which improves the performance of SPDNet by 2.22%, 11.71%, and 4.8%**. Additionally, although the LCM-based LieBN performs worse than other LieBN variants on the Radar and FPHA datasets, it exhibits the best performance on the HDM05 dataset. These observations highlight the advantage of the generality of our LieBN approach. **The effect of deformation:** Deformation patterns also vary across datasets. Firstly, the standard AIM is already saturated on the Radar dataset. Secondly, as indicated in Tab. 4, an appropriate deformation factor θ can further enhance the performance of LieBN. Notably, even though the LieBN induced by LCM-(1) impedes the learning of SPDNet on the FPHA datasets, it can improve performance when an appropriate deformation factor θ is applied. These findings highlight the utility of the deforming geometry of the SPD manifold. **Efficiency:** Our LieBN achieves comparable or even better efficiency than SPDNetBN, although compared with SPDNetBN, our LieBN places additional consideration on variance. Particularly, the LieBN induced by standard LEM or LCM exhibits better efficiency than SPDNetBN. Even with deformation, the LCM-based LieBN is still comparable with SPDNetBN in terms of efficiency. This phenomenon could be attributed to the fast and simple computation of LCM and LEM.

Application to EEG classification: We evaluate our method on the architecture of TSMNet for two tasks, inter-session and inter-subject EEG classification. Following Kobler et al. (2022a), we adopt the architecture of $\{40, 20\}$. Compared to the SPDDSMBN, TSMNet+DSMLieBN-AIM obtains the highest average scores of 55.10% and 53.97% in inter-session and -subject transfer learning, improving the SPDDSMBN by 0.98% and 3.87%, respectively. In the inter-subject scenario, the advantage of the efficiency of our LieBN over SPDDSMBN is more obvious. Specifically, both the LEM- and LCM-based DSMLieBN achieve similar or better performance compared to SPDDSMBN, while requiring considerably less training time. For example, DSMLieBN-LCM-(1) achieves better results with only half the training time of SPDDSMBN on inter-subject tasks. Interestingly, under the standard AIM, the sole difference between SPDDSMBN and our DSMLieBN is the way of centering and biasing. SPDDSMBN applies the inverse square root and square root to fulfill centering and biasing, while AIM-induced LieBN uses more efficient Cholesky decomposition. As such, the DSMLieBN induced by the standard AIM is more efficient than SPDDSMBN, particularly on the inter-subject task.

7 Conclusions

This paper proposes a novel framework called LieBN, enabling batch normalization over Lie groups. Our LieBN can effectively normalize both the sample and population statistics. Besides, we generalize the existing Lie groups on SPD manifolds and showcase our framework on the parameterized Lie groups of SPD manifolds. Extensive experiments demonstrate the advantage of our LieBN.

There are several other types of Lie groups in machine learning, such as special Euclidean groups. As a future avenue, we shall extend our LieBN to other Lie groups.

ACKNOWLEDGMENTS

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

REFERENCES

Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. *Fast and simple computations on tensors with log-Euclidean metrics.* PhD thesis, INRIA, 2005. URL https://doi.org/10.1007/11566465_15.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450.

Frédéric Barbaresco. Gaussian distributions on the space of symmetric positive definite matrices from souriau's gibbs state for siegel domains by coadjoint orbit and moment map. In *Geometric Science of Information: 5th International Conference, GSI 2021, Paris, France, July 21–23, 2021, Proceedings 5*, pp. 245–255. Springer, 2021. URL https://doi.org/10.1007/978-3-030-80209-7_28.

Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. *arXiv preprint arXiv:1810.00760*, 2018. URL https://arxiv.org/abs/1810.00760.

Marcel Berger. *A panoramic view of Riemannian geometry*. Springer, 2003. URL https://doi.org/10.1007/978-3-642-18245-7.

Rajendra Bhatia. *Matrix analysis*, volume 169. Springer Science & Business Media, 2013.

Victoria Bloom, Dimitrios Makris, and Vasileios Argyriou. G3D: A gaming action dataset and real time action recognition evaluation framework. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 7–12. IEEE, 2012. URL https://doi.org/10.1109/CVPRW.2012.6239175.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. URL https://doi.org/10.1109/MSP.2017.2693418.

Daniel Brooks. *Deep Learning and Information Geometry for Time-Series Classification*. PhD thesis, Sorbonne université, 2020. URL https://theses.hal.science/tel-03984879.

Daniel Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. A hermitian positive definite neural network for micro-doppler complex covariance processing. In *2019 International Radar Conference (RADAR)*, pp. 1–6. IEEE, 2019a. URL https://doi.org/10.1109/RADAR41533.2019.171277.

Daniel Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Riemannian batch normalization for SPD neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019b. URL https://arxiv.org/abs/1909.02414.

Daniel Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Second-order networks in pytorch. In *Geometric Science of Information: 4th International Conference, GSI 2019, Toulouse, France, August 27–29, 2019, Proceedings 4*, pp. 751–758. Springer, 2019c. URL https://doi.org/10.1007/978-3-030-26980-7_78.

Daniel Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Deep learning and information geometry for drone micro-doppler radar classification. In *2020 IEEE Radar Conference (RadarConf20)*, pp. 1–6. IEEE, 2020. URL https://doi.org/10.1109/RadarConf2043947.2020.9266689.

Daniel A Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Complex-valued neural networks for fully-temporal micro-doppler classification. In *2019 20th International Radar Symposium (IRS)*, pp. 1–10. IEEE, 2019d. URL https://doi.org/10.23919/IRS.2019.8768161.

Daniel A Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Exploring complex time-series representations for riemannian machine learning of radar data. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3672–3676. IEEE, 2019e. URL https://doi.org/10.1109/ICASSP.2019.8683056.

Rudrasis Chakraborty. ManifoldNorm: Extending normalizations on Riemannian manifolds. *arXiv preprint arXiv:2003.13869*, 2020. URL https://arxiv.org/abs/2003.13869.

Rudrasis Chakraborty and Baba C Vemuri. Statistics on the Stiefel manifold: theory and applications. *The Annals of Statistics*, 47(1):415–438, 2019. URL https://doi.org/10.1214/18-AOS1692.

Rudrasis Chakraborty, Chun-Hao Yang, Xingjian Zhen, Monami Banerjee, Derek Archer, David Vaillancourt, Vikas Singh, and Baba Vemuri. A statistical recurrent model on the manifold of symmetric positive definite matrices. *Advances in Neural Information Processing Systems*, 31, 2018. URL https://arxiv.org/abs/1805.11204.

Rudrasis Chakraborty, Jose Bouza, Jonathan Manton, and Baba C Vemuri. Manifoldnet: A deep neural network for manifold-valued data with applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. URL https://doi.org/10.1109/TPAMI.2020.3003846.

Kai-Xuan Chen, Jie-Yi Ren, Xiao-Jun Wu, and Josef Kittler. Covariance descriptors on a Gaussian manifold and their application to image set classification. *Pattern Recognition*, 107:107463, 2020. doi: 10.1016/j.patcog.2020.107463.

Kaixuan Chen, Shunyu Liu, Tongtian Zhu, Ji Qiao, Yun Su, Yingjie Tian, Tongya Zheng, Haofei Zhang, Zunlei Feng, Jingwen Ye, et al. Improving expressivity of GNNs with subgraph-specific factor embedded normalization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 237–249, 2023a. doi: 10.1145/3580305.3599388.

Kaixuan Chen, Jie Song, Shunyu Liu, Na Yu, Zunlei Feng, Gengshi Han, and Mingli Song. Distribution knowledge embedding for graph pooling. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):7898–7908, 2023b. doi: 10.1109/TKDE.2022.3208063.

Ziheng Chen, Tianyang Xu, Xiao-Jun Wu, Rui Wang, and Josef Kittler. Hybrid Riemannian graph-embedding metric learning for image set classification. *IEEE Transactions on Big Data*, 2021. URL https://doi.org/10.1109/TBDATA.2021.3113084.

Ziheng Chen, Yue Song, Gaowen Liu, Ramana Rao Kompella, Xiaojun Wu, and Nicu Sebe. Riemannian multiclass logistics regression for SPD neural networks. *arXiv preprint arXiv:2305.11288*, 2023c. URL https://arxiv.org/abs/2305.11288.

Ziheng Chen, Tianyang Xu, Zhiwu Huang, Yue Song, Xiao-Jun Wu, and Nicu Sebe. Adaptive Riemannian metrics on SPD manifolds. *arXiv preprint arXiv:2303.15477*, 2023d. URL https://arxiv.org/abs/2303.15477.

Ziheng Chen, Tianyang Xu, Xiao-Jun Wu, Rui Wang, Zhiwu Huang, and Josef Kittler. Riemannian local mechanism for SPD neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7104–7112, 2023e. URL https://doi.org/10.1609/aaai.v37i6.25867.

Manfredo Perdigao Do Carmo and J Flaherty Francis. *Riemannian Geometry*, volume 6. Springer, 1992. URL https://link.springer.com/book/9780817634902.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *Advances in Neural Information Processing Systems*, 31, 2018. URL https://arxiv.org/abs/1805.09112.

Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun 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*, pp. 409–419, 2018. URL https://arxiv.org/abs/1704.02463.

Alexandre Gramfort. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7, 2013. URL https://doi.org/10.3389/fnins.2013.00267.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. URL https://arxiv.org/abs/1512.03385.

Marcel F. Hinss, Ludovic Darmet, Bertille Somon, Emilie Jahanpour, Fabien Lotte, Simon Ladouce, and Raphaëlle N. Roy. An EEG dataset for cross-session mental workload estimation: Passive BCI competition of the Neuroergonomics Conference 2021, 2021. URL https://doi.org/10.5281/zenodo.5055046.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Zhiwu Huang and Luc Van Gool. A Riemannian network for SPD matrix learning. In *Thirty-first AAAI Conference on Artificial Intelligence*, 2017. URL https://arxiv.org/abs/1608.04233.

Zhiwu Huang, Chengde Wan, Thomas Probst, and Luc Van Gool. Deep learning on Lie groups for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6099–6108, 2017. URL https://arxiv.org/abs/1612.05877.

Zhiwu Huang, Jiqing Wu, and Luc Van Gool. Building deep networks on Grassmann manifolds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. URL https://doi.org/10.1609/aaai.v32i1.11725.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015. URL https://proceedings.mlr.press/v37/ioffe15.html.

Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2965–2973, 2015. URL https://arxiv.org/abs/1509.07838.

Vinay Jayaram and Alexandre Barachant. MOABB: trustworthy algorithm benchmarking for BCIs. *Journal of Neural Engineering*, 15(6):066011, 2018. URL https://doi.org/10.1088/1741-2552/aadea0.

Hermann Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30(5):509–541, 1977. URL https://doi.org/10.1002/cpa.3160300502.

Reinmar Kobler, Jun-ichiro Hirayama, Qibin Zhao, and Motoaki Kawanabe. SPD domain-specific batch normalization to crack interpretable unsupervised domain adaptation in EEG. *Advances in Neural Information Processing Systems*, 35:6219–6235, 2022a. URL https://arxiv.org/abs/2206.01323.

Reinmar J Kobler, Jun-ichiro Hirayama, and Motoaki 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)*, pp. 3863–3867. IEEE, 2022b. URL https://doi.org/10.1109/ICASSP43922.2022.9746629.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012. URL https://doi.org//10.1145/3065386.

Zhenhua Lin. Riemannian geometry of symmetric positive definite matrices via Cholesky decomposition. *SIAM Journal on Matrix Analysis and Applications*, 40(4):1353–1370, 2019. URL https://arxiv.org/abs/1908.09326.

Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa. Differentiating through the Fréchet mean. In *International Conference on Machine Learning*, pp. 6393–6403. PMLR, 2020. URL https://proceedings.mlr.press/v119/lou20a.html.

Jonathan H Manton. A globally convergent numerical algorithm for computing the centre of mass on compact lie groups. In *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004.*, volume 3, pp. 2211–2216. IEEE, 2004. URL https://doi.org/10.1109/ICARCV.2004.1469774.

Meinard Müller, Tido Röder, Michael Clausen, Bernhard Eberhardt, Björn Krüger, and Andreas Weber. Documentation mocap database HDM05. Technical report, Universität Bonn, 2007. URL https://resources.mpi-inf.mpg.de/HDM05/.

Iain Murray. Differentiation of the Cholesky decomposition. *arXiv preprint arXiv:1602.07527*, 2016. URL https://arxiv.org/abs/1602.07527.

Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017. URL https://doi.org/10.1201/9781315136370.

Xuan Son Nguyen. The Gyro-structure of some matrix manifolds. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26618–26630, 2022a. URL https://openreview.net/forum?id=eyE9Fb2AvOT.

Xuan Son Nguyen. A Gyrovector space approach for symmetric positive semi-definite matrix learning. In *Proceedings of the European Conference on Computer Vision*, pp. 52–68, 2022b. URL https://doi.org/10.1007/978-3-031-19812-0_4.

Xuan Son Nguyen and Shuo Yang. Building neural networks on matrix manifolds: A Gyrovector space approach. *arXiv preprint arXiv:2305.04560*, 2023. URL https://arxiv.org/abs/2305.04560.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019. URL https://arxiv.org/abs/1912.01703.

Xavier Pennec. *Probabilities and statistics on Riemannian manifolds: A geometric approach*. PhD thesis, INRIA, 2004. URL https://hal.science/inria-00071490.

Xavier Pennec and Nicholas Ayache. Uniform distribution, distance and expectation problems for geometric features processing. *Journal of Mathematical Imaging and Vision*, 9:49–67, 1998. URL https://doi.org/10.1023/A:1008270110193.

Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006. URL https://doi.org/10.1007/s11263-005-3222-z.

Salem Said, Lionel Bombrun, Yannick Berthoumieu, and Jonathan H Manton. Riemannian gaussian distributions on the space of symmetric positive definite matrices. *IEEE Transactions on Information Theory*, 63(4):2153–2170, 2017. URL https://doi.org/10.1109/TIT.2017.2653803.

Yann Thanwerdas and Xavier Pennec. Is affine-invariance well defined on SPD matrices? a principled continuum of metrics. In *Geometric Science of Information: 4th International Conference, GSI 2019, Toulouse, France, August 27–29, 2019, Proceedings 4*, pp. 502–510. Springer, 2019a. URL https://arxiv.org/abs/1906.01349.

Yann Thanwerdas and Xavier Pennec. Exploration of balanced metrics on symmetric positive definite matrices. In *Geometric Science of Information: 4th International Conference, GSI 2019, Toulouse, France, August 27–29, 2019, Proceedings 4*, pp. 484–493. Springer, 2019b. URL https://arxiv.org/abs/1909.03852.

Yann Thanwerdas and Xavier Pennec. The geometry of mixed-Euclidean metrics on symmetric positive definite matrices. *Differential Geometry and its Applications*, 81:101867, 2022a. URL https://arxiv.org/abs/2111.02990.

Yann Thanwerdas and Xavier Pennec. Theoretically and computationally convenient geometries on full-rank correlation matrices. *SIAM Journal on Matrix Analysis and Applications*, 43(4): 1851–1872, 2022b. URL https://arxiv.org/abs/2201.06282.

Yann Thanwerdas and Xavier Pennec. O (n)-invariant Riemannian metrics on SPD matrices. *Linear Algebra and its Applications*, 661:163–201, 2023. URL https://arxiv.org/abs/2109.05768.

Loring W.. Tu. *An introduction to manifolds*. Springer, 2011. URL https://doi.org/10.1007/978-1-4419-7400-6_3.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. URL https://arxiv.org/abs/1607.08022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3D skeletons as points in a Lie group. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 588–595, 2014. URL https://doi.org/10.1109/CVPR.2014.82.

Rui Wang, Xiao-Jun Wu, and Josef Kittler. SymNet: A simple symmetric positive definite manifold deep learning method for image set classification. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):2208–2222, 2021. URL https://doi.org/10.1109/tnnls.2020.3044176.

Rui Wang, Xiao-Jun Wu, Ziheng Chen, Tianyang Xu, and Josef Kittler. DreamNet: A deep Riemannian manifold network for SPD matrix learning. In *Proceedings of the Asian Conference on Computer Vision*, pp. 3241–3257, 2022a. URL https://arxiv.org/abs/2206.07967.

Rui Wang, Xiao-Jun Wu, Ziheng Chen, Tianyang Xu, and Josef Kittler. Learning a discriminative SPD manifold neural network for image set classification. *Neural Networks*, 151:94–110, 2022b. URL https://doi.org/10.1016/j.neunet.2022.03.012.

Rui Wang, Xiao-Jun Wu, Ziheng Chen, Cong Hu, and Josef Kittler. SPD manifold deep metric learning for image set classification. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2024. doi: 10.1109/TNNLS.2022.3216811.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018. URL https://arxiv.org/abs/1803.08494v3.

Or Yair, Mirela Ben-Chen, and Ronen Talmon. Parallel transport on the cone manifold of SPD matrices for domain adaptation. *IEEE Transactions on Signal Processing*, 67(7):1797–1811, 2019. URL https://doi.org/10.1109/TSP.2019.2894801.

Hongwei Yong, Jianqiang Huang, Deyu Meng, Xiansheng Hua, and Lei Zhang. Momentum batch normalization for deep learning with small batch size. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pp. 224–240. Springer, 2020. URL https://doi.org/10.1007/978-3-030-58610-2_14.

APPENDIX CONTENTS

A Notations **16**

B Basic Layers in SPDNet and TSMNet **16**

C Statistical Results of Scaling in the LieBN **16**

D LieBN as a Natural Generalization of Euclidean BN **19**

E Domain-specific Momentum LieBN for EEG Classification **19**

F Backpropagation of Matrix Functions **20**

G Additional Details and Experiments of LieBN on SPD manifolds **20**

G.1 Datasets and Preprocessing 21

G.2 Hyper-Parameters 21

G.3 Evaluation Methods 21

G.4 Empirical Insights on the Hyper-parameters in LieBN on SPD Manifolds 21

G.4.1 The Effect of β in SPD LieBN 22

H Preliminary Experiments on Rotation Matrices **22**

H.1 Geometry on Rotation Matrices 22

H.2 Datasets and Preprocessing 22

H.3 Implementation Details 22

H.4 Results 23

I Proofs of the Lemmas and Theories in the Main Paper **23**

A NOTATIONS

For better clarity, we summarize all the notations used in this paper in Tab. 6.

Table 6: Summary of notations.

Notation	Explanation
$\{\mathcal{M}, \odot, g\}$ or abbreviated as $\mathcal{M}$	A Lie group with a group operation $\odot$ and left-invariant Riemannian metric g
$P_{\odot}^{-1}$	Group inverse of $P \in \mathcal{M}$
$T_P\mathcal{M}$	The tangent space at $P \in \mathcal{M}$
$g_p(\cdot,\cdot)$ or $\langle \cdot,\cdot \rangle_P$	The Riemannian metric at $P \in \mathcal{M}$
$\Vert \cdot \Vert_P$	The norm induced by $\langle \cdot,\cdot \rangle_P$ on $T_P\mathcal{M}$
$\mathrm{d}(\cdot,\cdot)$	Geodesic distance
FM	Fréchet mean
WFM	Weighted Fréchet mean
Log_P	The Riemannian logarithm at P
$\mathrm{PT}_{P\to Q}$	The Riemannian parallel transportation along the geodesic connecting P and Q
L_P	The Lie group left translation by $P \in \mathcal{M}$
$f_{*,P}$	The differential map of the smooth map f at $P \in \mathcal{M}$
f^*g	The pullback metric by f from g
$\mathcal{S}_{++}^n$	The SPD manifold
$\mathcal{S}^n$	The Euclidean space of symmetric matrices
$\mathcal{L}^n$	The Euclidean space of lower triangular matrices
$\langle \cdot,\cdot \rangle$	The standard Frobenius inner product
$\Vert \cdot \Vert_{\mathrm{F}}$	The norm induced by the standard Frobenius inner product
$\mathbf{ST}$	$\mathbf{ST} = \{(\alpha,\beta) \in \mathbb{R}^2 \mid \min(\alpha, \alpha + n\beta) > 0\}$
$\langle \cdot,\cdot \rangle^{(\alpha,\beta)}$	The $\mathrm{O}(n)$-invariant Euclidean inner product
$\Vert \cdot \Vert^{(\alpha,\beta)}$	The norm induced by $\mathrm{O}(n)$-invariant Euclidean inner product
$g^{(\alpha,\beta)\text{-AI}}$	The Riemannian metric tensor of (α,β)-AIM
$g^{(\theta,\alpha,\beta)\text{-AI}}$	The Riemannian metric tensor of (θ,α,β)-AIM
$\mathcal{N}(M,\sigma^2)$	Gaussian distribution
exp	Scalar exponential map
mlog	Matrix logarithm
mexp	Matrix exponentiation
Chol	Cholesky decomposition
Dlog	The diagonal element-wise logarithm
ψ_{LC}	$\mathrm{Dlog} \circ \mathrm{Chol}$
$\lfloor \cdot \rfloor$	The strictly lower triangular part of a square matrix
$\mathbb{D}(\cdot)$	A diagonal matrix with diagonal elements from a square matrix
$\mathrm{P}_\theta(\cdot)$ or $(\cdot)^\theta$	Matrix power function

B BASIC LAYERS IN SPDNET AND TSMNET

SPDNet (Huang & Van Gool, 2017) is the most classic SPD neural network. SPDNet mimics the conventional densely connected feedforward network, consisting of three basic building blocks

$$\text{BiMap layer: } S^k = W^k S^{k-1} W^{k\top}, \text{ with } W^k \text{ semi-orthogonal,} \tag{23}$$

$$\text{ReEig layer: } S^k = U^{k-1} \max(\Sigma^{k-1}, \epsilon I_n) U^{k-1\top}, \text{ with } S^{k-1} = U^{k-1}\Sigma^{k-1}U^{k-1\top}, \tag{24}$$

$$\text{LogEig layer: } S^k = \log(S^{k-1}). \tag{25}$$

where $\max(\cdot)$ is element-wise maximization. BiMap and ReEig mimic transformation and non-linear activation, while LogEig maps SPD matrices into the tangent space at the identity matrix for classification.

TSMNet (Kobler et al., 2022a) can be illustrate as $f_{tc} \to f_{sc} \to f_{BiMap} \to f_{ReEig} \to f_{LogEig}$, where f_{tc} and f_{sc} denote temporal and spatial convolution, respectively.

C STATISTICAL RESULTS OF SCALING IN THE LIEBN

In this section, we will show the effect of our scaling (Eq. (14)) on the population. We will see that while the resulting population variance is generally agnostic, it becomes analytic under certain circumstances, such as SPD manifolds under LEM or LCM. As a result, Eq. (14) can normalize and transform the latent Gaussian distribution.

To simplify, let $\phi_s(P) = \operatorname{Exp}_E[s \operatorname{Log}_E(P)]$. Similar to the main paper, $\mathcal{M}$ denotes a Lie group with a left-invariant metric. First, we present a lemma on the resulting P.D.F. of a random point transformed by ϕ_s.

Lemma C.1. *Given a random point X distributed over $\mathcal{M}$ with P.D.F. p_X, the P.D.F of $Y = \phi_s(X)$ is given by:*

$$p_Y(Q) = \Delta p_X(\phi_s^{(-1)}(Q)). \tag{26}$$

where $\Delta = \frac{|\phi_{s*}^{-1}|}{\left|L_{\phi_s^{-1}(Q)\odot Q^{-1}*}\right|}$. Here $|\cdot|$ denotes the determinant, and ϕ_{s*}^{-1} and $L_{\phi_s^{-1}(Q)\odot Q^{-1}*}$ are the differentials.

Proof. For the sake of simplicity, we will denote ϕ_s as ϕ throughout this proof. The volume element w.r.t. a left-invariant metric is the Haar measure (Pennec & Ayache, 1998, Sec. 3.2):

$$\mathrm{d}_L\, \mathcal{M}(P) = \frac{dP}{|L_{P*,E}|}, \tag{27}$$

where $|L_{P*,E}|$ is the determinant[4] of the differential of L_P at the neutral element E. Then we have

$$\begin{aligned}
\mathrm{d}_L\, \mathcal{M}(\phi^{-1}(Q)) &= \frac{\mathrm{d}\,\phi^{-1}(Q)}{|L_{\phi^{-1}(Q)*,E}|} \\
&= |(L_{\phi^{-1}(Q)\odot Q^{-1}} \circ L_Q)_{*,E}|^{-1} |\phi_*^{-1}|\, \mathrm{d}\, Q \\
&= \frac{|\phi_*^{-1}|}{|L_{\phi^{-1}(Q)\odot Q^{-1}*}|}\, \mathrm{d}_L\, \mathcal{M}(Q) \\
&= \Delta\, \mathrm{d}_L\, \mathcal{M}(Q).
\end{aligned} \tag{28}$$

The probability of $Q = \phi(P)$ in a set $\mathcal{Y} \subset \mathcal{M}$ is

$$\begin{aligned}
F(\phi(P) \in \mathcal{Y}) &= F(P \in \phi^{-1}(\mathcal{Y})) \\
&= \int_{\phi^{-1}(\mathcal{Y})} p_X(P) \cdot d_L\mathcal{M}(P) \\
&= \int_{\mathcal{Y}} p_X(\phi^{(-1)}(Q)) d_L\mathcal{M}(\phi^{(-1)}(Q)) \\
&= \int_{\mathcal{Y}} \Delta p_X(\phi^{(-1)}(Q)) d_L\mathcal{M}(Q).
\end{aligned} \tag{29}$$

Therefore, the density of $Y = \phi(X)$ is

$$p_Y(Q) = \Delta p_X(\phi^{(-1)}(Q)). \tag{30}$$

□

The above lemma implies that when Δ is a constant, Y also follows a Gaussian distribution.

Corollary C.2. *Following the notations in Lem. C.1, if $\Delta = c$ is a constant and $X \sim \mathcal{N}(E, \sigma^2)$, then Y also follows a Gaussian distribution,* i.e.,$Y \sim \mathcal{N}(E, s^2\sigma^2)$

Proof.

$$\begin{aligned}
p_Y(Q) &= ck(\sigma) \exp\left(-\frac{\mathrm{d}(\phi_s^{-1}(Q), E)^2}{2\sigma^2}\right) \\
&= k'(\delta) \exp\left(-\frac{\mathrm{d}(\operatorname{Exp}_E {}^1/_s \operatorname{Log}_E(Q), E)^2}{2\sigma^2}\right) \\
&= k'(\delta) \exp\left(-\frac{\|\operatorname{Log}_E(Q)\|_E^2}{2s^2\sigma^2}\right) \\
&= k'(\delta) \exp\left(-\frac{\mathrm{d}(Q, E)}{2s^2\sigma^2}\right),
\end{aligned} \tag{31}$$

[4]This should be more precisely understood as the determinant of the matrix representation of $L_{P*,E}$ under a local coordinate

where $\|\cdot\|_E$ is the norm of the tangent space at the neutral element E. □

Cor. C.2 implies that when $\Delta = c$, ϕ_s can scale the population variance and further transform the Gaussian distribution. Simple computations show that in the standard Euclidean space $\mathbb{R}^n$, $\Delta = 1/s$. Therefore, it is natural to expect that the pullback of $\mathbb{R}^n$ also enjoys constant Δ.

Proposition C.3. *Consider an n-dimensional Lie group $\mathcal{M}$ pulled back from the standard Euclidean space $\mathbb{R}^n$ by the diffeomorphism $\psi : \mathcal{M} \to \mathbb{R}^n$. In other words, the group operations and Riemannian metric on $\mathcal{M}$ are defined by ψ from $\mathbb{R}^n$. Then Δ remains constant on $\mathcal{M}$.*

Proof. To simplify notation, we denote ϕ_s as ϕ. Under the given assumption, the group addition and Riemannian metric on $\mathcal{M}$ are defined as follows:

$$\begin{aligned} \forall P, Q \in \mathcal{M}, P \odot Q &= \psi^{-1}(\psi(P) + \psi(Q)) \\ g &= \psi^* g^{\mathrm{E}}, \end{aligned} \tag{32}$$

where g^{E} is the standard Euclidean metric. Therefore, ϕ can be simplified as

$$\begin{aligned} \phi(P) &= \mathrm{Exp}_E\left[s\,\mathrm{Log}_E(P)\right] \\ &= \psi^{-1}\left(\tilde{\mathrm{Exp}}_0\left[\psi_{*,E}\left(s\psi_{*,0}^{-1}\tilde{\mathrm{Log}}_0\psi(P)\right)\right]\right) \\ &= \psi^{-1}(s\psi(P)), \end{aligned} \tag{33}$$

where $\tilde{\mathrm{Exp}}$ and $\tilde{\mathrm{Log}}$ are the Riemannian exponential and logarithmic maps in $\mathbb{R}^n$, which are reduced to vector addition and subtraction, respectively. Therefore, the inverse of ϕ is

$$\phi^{-1}(P) = \psi^{-1}\left(1/s\psi(P)\right). \tag{34}$$

Besides, $L_{\phi^{-1}(Q)\odot Q^{-1}}$ can also be further simplified:

$$L_{\phi^{-1}(Q)\odot Q^{-1}}(P) = \psi^{-1}\left(1/s\psi(Q) - \psi(Q) + \psi(P)\right) \tag{35}$$

The differentials of Eqs. (34) and (35) at Q are

$$\phi_{*,Q}^{-1} = \frac{1}{s}\psi_{*,1/s\psi(Q)}^{-1} \circ \psi_{*,Q}, \tag{36}$$

$$L_{\phi^{-1}(Q)\odot Q^{-1}*,Q} = \psi_{*,1/s\psi(Q)}^{-1} \circ \psi_{*,Q}. \tag{37}$$

Therefore, $\Delta = 1/s$ for all $Q \in \mathcal{M}$. □

By Prop. C.3, we can directly obtain the following corollary.

Corollary C.4. *Given a Lie group $\mathcal{M}$ pulled back from the Euclidean space, and a random point $X \sim \mathcal{N}(E, \sigma^2)$ over $\mathcal{M}$, $Y = \phi_s(X) \sim \mathcal{N}(E, s^2\sigma^2)$*

In machine learning, several Lie groups are derived by the pullback from the standard Euclidean space. As shown in Chen et al. (2023c) and the proof of Prop. 5.2, (α, β)-LEM and θ-LCM are pullback metrics from the Euclidean metric. Therefore, for the Lie groups of SPD manifolds w.r.t. (α, β)-LEM and θ-LCM, Eq. (14) can transform the Gaussian distribution. Specifically, given a random point $X \sim \mathcal{N}(M, \sigma^2)$, Eqs. (13) to (15) transform the Gaussian distribution as:

$$\mathcal{N}(M, \sigma^2) \to \mathcal{N}(E, \sigma^2) \to \mathcal{N}(E, s^2) \to \mathcal{N}(B, s^2), \tag{38}$$

where M and σ are employed to normalize X, and ϵ in Eq. (14) is omitted. The above process exactly mirrors the transformation of Gaussian distributions within the framework of standard BN (Ioffe & Szegedy, 2015).

Remark C.5. A similar result to our Cor. C.4 was also presented in Chakraborty (2020, Prop. 3). However, in his proof, the author did not account for the Haar measure and only considered the P.D.F., casting doubt on the validity of their results. Additionally, their discussion is limited to matrix Lie groups, specifically under the distance $\mathrm{d}(P, Q) = \|\,\mathrm{mlog}(P^{-1}Q)\|_{\mathrm{F}}$. In contrast, we rectify their proof and consider general Lie groups.

D LieBN as a Natural Generalization of Euclidean BN

The centering and biasing in Euclidean BN correspond to the group action of $\mathbb{R}$. From a geometric perspective, the standard Euclidean metric is invariant under this group operation. Consequently, it is not surprising that our LieBN algorithm formulated in Alg. 1 serves as a natural generalization of standard Euclidean batch normalization. We formalize this fact in the following proposition.

Proposition D.1. *The LieBN algorithm presented in Alg. 1 is equivalent to the standard Euclidean BN when $\mathcal{M} = \mathbb{R}^n$, both during the training and testing phases.*

Proof. The core of this proof lies in the fact that on $\mathbb{R}^n$, (1) the Fréchet mean and variance are reduced to the familiar Euclidean statistics. (2) the calculation of the running mean becomes the weighted arithmetic mean. (3) Eqs. (13) to (15) become Eq. (6); We prove these three points one by one.

As stated in Lou et al. (2020, Prop. G.1 and Cor. G.2), from the view of the product manifold, the element-wise Fréchet mean and variance on $\mathbb{R}^n$ are equivalent to the vector-valued Euclidean variance and mean.

Besides, by similar proof as in Lou et al. (2020, Prop. G.1), the weighted Fréchet mean on $\mathbb{R}^n$ is simplified as the weighted arithmetic average. Therefore, on $\mathbb{R}^n$, the calculation of running statistics in our Alg. 1 becomes the familiar moving average.

Thirdly, on $\mathbb{R}^n$, we know that $L_x(y) = x + y$, $\operatorname{Exp}_x v = x + v$, $\operatorname{Log}_x y = y - x$, and the neutral element is 0. Since statistics, as well as the Euclidean BN, are calculated element-wisely, we can safely consider a single element, *i.e.,*$\mathbb{R}^n = \mathbb{R}$. For a batch of activations $\{x_{i\ldots N} \in \mathbb{R}\}$, where the batch mean and batch variance are denoted as μ_b and v_b^2, then Eqs. (13) to (15) are rewrote as:

$$L_\beta\left(\operatorname{Exp}_0\left[\frac{\gamma}{\sqrt{v_b^2+\epsilon}}\operatorname{Log}_0(L_{-\mu_b}(x_i))\right]\right) = \gamma\frac{x_i-\mu_b}{\sqrt{v_b^2+\epsilon}} + \beta. \tag{39}$$

The above equation is the exact core computation of the standard Euclidean BN. □

E Domain-specific Momentum LieBN for EEG Classification

Algorithm 2: Momentum LieBN (MLieBN) Algorithm

Input : A batch of activations $\{P_{1\ldots N} \in \mathcal{M}\}$, and a small positive constant ϵ
running mean $\bar{M}_r = E$, running variance $\bar{v}_r^2 = 1$ for training
running mean $\tilde{M}_r = E$, running variance $\tilde{v}_r^2 = 1$ for testing
biasing parameter $B \in \mathcal{M}$, scaling parameter $s \in \mathbb{R}/\{0\}$,
momentum for training and testing $\gamma_{train}, \gamma \in [0, 1]$

Output : Normalized activations $\{\tilde{P}_{1\ldots N}\}$

if *training* **then**
  Compute batch mean M_b and variance v_b^2 of $\{P_{1\ldots N}\}$;
  $\bar{M}_r \leftarrow \mathrm{WFM}(\{1-\gamma_{train}, \gamma_{train}\}, \{\bar{M}_r, M_b\})$;
  $\bar{v}_r^2 \leftarrow (1-\gamma_{train})\bar{v}_r^2 + \gamma_{train} v_b^2$;
  $\tilde{M}_r \leftarrow \mathrm{WFM}(\{1-\gamma, \gamma\}, \{\tilde{M}_r, M_b\})$;
  $\tilde{v}_r^2 \leftarrow (1-\gamma)\tilde{v}_r^2 + \gamma v_b^2$;
end
if *training* **then** $M \leftarrow \bar{M}_r, v^2 \leftarrow \bar{v}_r^2$;
else $M \leftarrow \tilde{M}_r, v^2 \leftarrow \tilde{v}_r^2$;
for $i \leftarrow 1$ **to** N **do**
  Centering to the neutral element E: $\bar{P}_i \leftarrow L_{M_\odot^{-1}}(P_i)$
  Scaling the dispersion: $\hat{P}_i \leftarrow \operatorname{Exp}_E\left[\frac{s}{\sqrt{v^2+\epsilon}}\operatorname{Log}_E(\bar{P}_i)\right]$
  Biasing towards parameter B: $\tilde{P}_i \leftarrow L_B(\hat{P}_i)$
end

Kobler et al. (2022a) proposed SPD domain-specific momentum batch normalization (SPDDSMBN) as a domain adaptation approach for EEG classification. SPDDSMBN, based on Eq. (11), performed normalization of mean and variance on SPD manifolds under the specific AIM. Additionally, SPDDSMBN utilized separate momentums for updating training and testing running statistics, inspired by the work of Yong et al. (2020). Following Kobler et al. (2022a, Alg. 1), we also present a momentum LieBN (MLieBN) in Alg. 2. Here γ is fixed and γ_{train} is defined as

$$\gamma_{train} = 1 - \rho^{\frac{1}{K-1}\max(K-k,0)} + \rho, \text{ where } \rho = \frac{1}{domains_per_batch} \tag{40}$$

Furthermore, in line with Kobler et al. (2022a), we adopt multi-channel mechanisms for domain-specific MLieBN (DSMLieBN), where each domain has its own MLieBN layer. Similar to Kobler et al. (2022a), we set the biasing parameter equal to the neutral element, and the scaling factor is shared across all domains. We denote Alg. 2 as $\mathrm{MLieBN}(P_j|M, s, \epsilon, \gamma, \gamma_{train})$. Then our DSMLieBN follows

$$\mathrm{DSMLieBN}(P_j, i) = \mathrm{MLieBN}_i(P_j|E, s, \epsilon, \gamma, \gamma_{train}), \forall P_j \in \{P_{1\ldots N}\}, \tag{41}$$

where i is the index of the domain. We follow the official code of SPDDSMBN[5] to implement our DSMLieBN. In a word, the only difference between DSMLieBN and SPDDSMBN is the different way of normalization.

Analogous to Thm. 5.3, computations for DSMLieBN under pullback metrics can also be performed by mapping, calculating, and then remapping.

F Backpropagation of Matrix Functions

Our implementation of LieBN on SPD manifolds involves several matrix functions. Thus, we employ matrix backpropagation (BP) (Ionescu et al., 2015) for gradient computation. These matrix operations can be divided into Cholesky decomposition and the functions based on Eigendecomposition.

The differentiation of the Cholesky decomposition can be found in Murray (2016, Eq. 8) or Lin (2019, Props. 4). Besides, our homemade BP of the Cholesky decomposition yields a similar gradient to the one generated by autograd of `torch.linalg.cholesky`. Therefore, during the experiments, we use `torch.linalg.cholesky`.

The second type of matrix functions is based on Eigendecomposition, such as matrix exponential, logarithm, and power. Although torch (Paszke et al., 2019) supports autograd of Eigendecomposition, it requires the computation of $\frac{1}{\delta_i - \delta_j}$ (Ionescu et al., 2015, Props. 1), where δ_i and δ_j denote eigenvalues. This might trigger numerical instability when δ_i approximates δ_j. Following Brooks et al. (2019b), we use the Daleckĭi-Kreĭn formula (Bhatia, 2013, Thm. V.3.3) to calculate the BP of Eigen-based matrix functions. In detail, for a matrix function defined as $X = f(S) = Uf(\Sigma)U^\top$, with $S = U\Sigma U^\top$ as the eigendecomposition of an SPD matrix, its BP is expressed as

$$\nabla_S L = U[K \odot (U^T(\nabla_X L)U)]U^T. \tag{42}$$

where $\nabla_X L$ is the Euclidean gradient of the loss function L w.r.t. X. Matrix K is defined as

$$K_{ij} = \begin{cases} \frac{f(\sigma_i) - f(\sigma_j)}{\sigma_i - \sigma_j} & \text{if } \sigma_i \neq \sigma_j \\ f'(\sigma_i) & \text{otherwise} \end{cases} \tag{43}$$

where $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_d)$. Eq. (43) demonstrates the numerical stability of Daleckĭi-Kreĭn formula.

G Additional Details and Experiments of LieBN on SPD manifolds

We use the official code of SPDNetBN[6] (Brooks et al., 2019b) and TSMNet[7] (Kobler et al., 2022a) to implement our experiments on the SPDNet and TSMNet backbones.

[5] https://github.com/rkobler/TSMNet

[6] https://proceedings.neurips.cc/paper_files/paper/2019/file/6e69ebbfad976d4637bb4b39de261bf7-Supplemental.zip

[7] https://github.com/rkobler/TSMNet

G.1 Datasets and Preprocessing

Radar dataset (Brooks et al., 2019b) contains 3,000 synthetic radar signals. Following the protocol in Brooks et al. (2019b), each signal is split into windows of length 20, resulting in 3,000 covariance matrices of the size 20×20 equally distributed in 3 classes. **HDM05** dataset (Müller et al., 2007) consists of 2,273 skeleton-based motion capture sequences executed by different actors. Each frame consists of 3D coordinates of 31 joints, allowing the representation of each sequence as a 93×93 covariance matrix. In line with Brooks et al. (2019b), we trim the dataset down to 2086 instances scattered throughout 117 classes by removing some under-represented clips. **FPHA** (Garcia-Hernando et al., 2018) includes 1,175 skeleton-based first-person hand gesture videos of 45 different categories with 600 clips for training and 575 for testing. Following Wang et al. (2021), we represent each sequence as a 63×63 covariance matrix. **Hinss2021** dataset (Hinss et al., 2021) is a recently released competition dataset containing EEG signals for mental workload estimation. The dataset is employed for two tasks, namely inter-session and inter-subject, which are treated as domain adaptation problems. Geometry-aware methods (Yair et al., 2019; Kobler et al., 2022a) have demonstrated promising performance in EEG classification. We follow Kobler et al. (2022a) for data preprocessing. In detail, the python package MOABB (Jayaram & Barachant, 2018) and MNE (Gramfort, 2013) are used to preprocess the datasets. The applied steps include resampling the EEG signals to 250/256 Hz, applying temporal filters to extract oscillatory EEG activity in the 4 to 36 Hz range, extracting short segments ($\leq$ 3s) associated with a class label, and finally obtaining 40×40 SPD covariance matrices.

G.2 Hyper-Parameters

We implement the SPD LieBN and DSMLieBN induced by three standard left-invariant metrics, namely AIM, LEM, and LCM, along with their parameterized metrics. Therefore, our method has a maximum of three hyper-parameters, *i.e.,*(θ, α, β), where θ controls deformation. In our LieBN, (α, β) only affects variance calculation. Therefore, we set $(\alpha, \beta) = (1, 0)$ and only tune the deformation factor θ from the candidate values of ± 0.5, ± 1, and ± 1.5. We denote [Baseline]+[BN_Type]+[Metric]-[θ] as the baseline endowed with a specific LieBN, such as SPDNet+LieBN-AIM-(1) and TSMNet+DSMLieBN-LCM-(1).

G.3 Evaluation Methods

In line with the previous work (Brooks et al., 2019b; Kobler et al., 2022a), we use accuracy as the scoring metric for the Radar, HDM05, and FPHA datasets, and balanced accuracy (*i.e.,*the average recall across classes) for the Hinss2021 dataset. Ten-fold experiments on the Radar, HDM05, and FPHA datasets are carried out with randomized initialization and split (split is officially fixed for the FPHA dataset), while on the Hinss2021 dataset, models are fit and evaluated with a randomized leave 5% of the sessions (inter-session) or subjects (inter-subject) out cross-validation scheme.

G.4 Empirical Insights on the Hyper-parameters in LieBN on SPD Manifolds

Our SPD LieBN has at most three types of hyper-parameters: Riemannian metric, deformation factor θ, and $\mathrm{O}(n)$-invariance parameters (α, β). The general order of importance should be Riemannian metric $> \theta > (\alpha, \beta)$.

The most significant parameter is the choice of Riemannian metric, as all the geometric properties are sourced from a metric. A safe choice would start with AIM, and then decide whether to explore other metrics further. The most important reason is the property of affine invariance of AIM, which is a natural characteristic of covariance matrices. In our experiments, the LieBN-AIM generally achieves the best performance. However, AIM is not always the best metric. As shown in Tab. 4b, the best result on the HDM05 dataset is achieved by LCM-based LieBN, which improves the vanilla SPDNet by 11.71%. Therefore, when choosing Riemannian metrics on SPD manifolds, a safe choice would start with AIM and extend to other metrics. Besides, if efficiency is an important factor, one should first consider LCM, as it is the most efficient one.

The second one is the deformation factor θ. As we discussed in Sec. 5.1, θ interpolates between different types of metrics ($\theta = 1$ and $\theta \to 0$). Inspired by this, we select θ around its deformation boundaries (1 and 0). In this paper we roughly select θ from $\{\pm 0.5, \pm 1, \pm 1.5\}$

The less important parameters are (α, β). Recalling Alg. 1. and Tab. 1, (α, β) only affects the calculation of variance, which should have less effects compared with the above two parameters. Therefore, we simply set $(\alpha, \beta) = (1, 0)$ during experiments.

G.4.1 THE EFFECT OF β IN SPD LIEBN

Recalling Eq. (3), β controls the relative importance of the trace part against the inner product. Therefore, we set the candidate values of β as $\{1, 1/n, 1/n^2, 0, -1/n+\epsilon, -1/n^2\}$, where n is the input dimension of LieBN, and ϵ is a small positive scalar to ensure $\mathrm{O}(n)$-invariance, *i.e.,*$(\alpha, \beta) \in \mathbf{ST}$. $1/n^2$ and $1/n$ means averaging the trace in Eq. (3), while the sign of β denotes suppressing (-), enhancing (+), or neutralizing (0) the trace.

We focus on AIM-based LieBN on the HDM05 dataset. We set $\theta = 1.5$, as it is the best deformation factor under this scenario. Other network settings remain the same as the main paper. The 10-fold average results are presented in Tab. 7. Note that on this setting, $n = 30$. As expected, β has minor effects on our LieBN.

Table 7: The effect of different β for AIM-based LieBN on the HDM05 dataset.

Beta	$-1/30^2$	-0.03	$1/30^2$	$1/30$	1	0
Mean±STD	68.18±0.86	68.12±0.74	68.20±0.85	68.18±0.85	68.16±0.80	68.16±0.68

H PRELIMINARY EXPERIMENTS ON ROTATION MATRICES

This section implements our LieBN in Alg. 1 on the special orthogonal groups, *i.e.,*$\mathrm{SO}(n)$, also known as rotation matrices. We apply our LieBN to the classic LieNet (Huang & Van Gool, 2017), where the latent space is the special orthogonal group.

H.1 GEOMETRY ON ROTATION MATRICES

Table 8: The associated Riemannian operators on Rotation matrices.

Operators	$\mathrm{d}^2(R,S)$	$\mathrm{Log}_I R$	$\mathrm{Exp}_I(A)$	$\gamma_{(R,S)}(t)$	FM
Expression	$\left\|\mathrm{mlog}\left(R^\top S\right)\right\|_{\mathrm{F}}^2$	$\mathrm{mlog}(R)$	$\mathrm{mexp}(A)$	$R\,\mathrm{mexp}(t\,\mathrm{mlog}(R^\top S))$	Manton (2004, Alg. 1)

We denote $R, S \in \mathrm{SO}(n)$, and $\gamma_{(R,S)}(t)$ as the geodesic connecting R and S. The neutral elements of rotation matrices is the identity matrix. Tab. 8 summarizes all the necessary Riemannian ingredients of the invariant metric on $\mathrm{SO}(n)$.

For the specific $\mathrm{SO}(3)$, the matrix logarithm and exponentiation can be calculated without decomposition (Murray et al., 2017, Exs. A. 11 and A.14).

H.2 DATASETS AND PREPROCESSING

Following LieNet, we validate our LieBN on the **G3D** dataset (Bloom et al., 2012). This dataset (Bloom et al., 2012) consists of 663 sequences of 20 different gaming actions. Each sequence is recorded by 3D locations of 20 joints (i.e., 19 bones). Following Huang & Van Gool (2017), we use the code of Vemulapalli et al. (2014) to represent each skeleton sequence as a point on the Lie group $\mathrm{SO}^{N\times T}(3)$, where N and T denote spatial and temporal dimensions. As preprocessed in Huang & Van Gool (2017), we set T as 100 for each sequence on the G3D.

H.3 IMPLEMENTATION DETAILS

LieNet: The LieNet consists of three basic layers: RotMap, RotPooling, and LogMap layers. The RotMap mimics the convolutional layer, while the RotPooling extends the pooling layers to rotation matrices. The logMap layer maps the rotation matrix into the tangent space at the identity for classification. Note that the official code of LieNet[8] is developed by Matlab. We follow the open-sourced Pytorch code[9] to implement our experiments. To reproduce LieNet more faithfully, we made the following modifications to this Pytorch code. We re-code the LogMap and RotPooling layers to make them consistent with the official Matlab implementation. In addition, we also extend the existing Riemannian optimization package geoopt Bécigneul & Ganea (2018) into $\mathrm{SO}(3)$ to allow for Riemannian version of SGD, ADAM, and AMSGrad on $\mathrm{SO}(3)$, which is missing in the current

[8]https://github.com/zhiwu-huang/LieNet

[9]https://github.com/hjf1997/LieNet

package. However, we find that SGD is the best optimizer for LieNet. Therefore, we adopt SGD during the experiments. We apply our LieBN before the LogMap layer and refer to this network as LieNetLieBN. Note that the dimension of features in LieNet is $B \times N \times T \times 3 \times 3$, we calculate Lie group statistics along the batch and spatial dimensions ($B \times T$), resulting in an $N \times 3 \times 3$ running mean.

Training Details: Following Huang et al. (2017), we focus on the suggested 3Blocks architecture for the G3D dataset. The learning rate is $1e^{-2}$ with a weight decay of $1e^{-5}$. Following LieNet, we adopt a 10-fold cross-subject test setting, where half of the subjects are used for training and the other half are employed for testing.

H.4 RESULTS

The 10-fold results are shown in Tab. 9. Due to different software, our reimplemented LieNet is slightly worse than the performance reported in Huang et al. (2017). However, we still can observe a clear improvement of LieNetLieBN over LieNet.

Table 9: Results of LieNet with or without LieBN on the G3D dataset.

Methods	G3D	
	Mean±STD	Max
LieNet	87.91±0.90	89.73
LieNetLieBN	88.88±1.62	90.67

I PROOFS OF THE LEMMAS AND THEORIES IN THE MAIN PAPER

Proof of Prop. 4.1 . Property 1:

The MLE of M is

$$\begin{aligned} M_{\mathrm{MLE}} &= \operatorname{argmax} \log(K(v)) - \sum_{i=1}^{N} \frac{\mathrm{d}(P_i, M)^2}{2v^2} \\ &= \operatorname{argmin} \sum_{i=1}^{N} \mathrm{d}(P_i, M)^2. \end{aligned} \tag{44}$$

Property 2:

We denote $Y = L_B(X)$, and p_X and p_Y as the density of X and Y, respectively. The density of Y is

$$\begin{aligned} p_Y(Q) &= p_X(L_{B_{\odot}^{-1}}(Q)) \\ &= k(\sigma) \exp\left(-\frac{\mathrm{d}(L_{B_{\odot}^{-1}}(Q), M)^2}{2\sigma^2}\right) \\ &= k(\sigma) \exp\left(-\frac{\mathrm{d}(Q, L_B(M))^2}{2\sigma^2}\right). \end{aligned} \tag{45}$$

The first equation is obtained by (Pennec, 2004, Thm. 7), while the last equation is obtained by the isometry of the left translation. □

Proof of Prop. 4.2 . The isometry of L_B can directly obtain the homogeneity of the sample mean. Now let us focus on Eq. (17). We have the following:

$$\begin{aligned} \sum\nolimits_{i=1}^{N} w_i \, \mathrm{d}^2(\phi_s(P_i), E) &= \sum\nolimits_{i=1}^{N} w_i \| s \operatorname{Log}_E P_i \|_E \\ &= s^2 \sum\nolimits_{i=1}^{N} w_i \| \operatorname{Log}_E P_i \|_E^2 \\ &= s^2 \sum\nolimits_{i=1}^{N} w_i \, \mathrm{d}^2(P_i, E), \end{aligned} \tag{46}$$

where $\| \cdot \|_E$ is the norm on $T_E\mathcal{M}$. □

Proof of Prop. 5.1 . We first prove the case of (θ, α, β)-LEM, and then proceed to the case of θ-LCM.

(θ, α, β)**-LEM:** For clarity, we denote the metric tensor of (θ, α, β)-LEM as

$$g^{(\theta,\alpha,\beta)\text{-LE}} = \frac{1}{\theta^2} \mathrm{P}_\theta^* \, g^{(\alpha,\beta)\text{-LE}}, \tag{47}$$

where $g^{(\alpha,\beta)\text{-LE}}$ is the metric tensor of (α, β)-LEM. Let $P \in \mathcal{S}^n_{++}$ and $V, W \in T_P\mathcal{S}^n_{++}$, then we have

$$\begin{aligned} g_P^{(\theta,\alpha,\beta)\text{-LE}}(V, W) &= \frac{1}{\theta^2} g^{(\alpha,\beta)\text{-LE}}_{\mathrm{P}_\theta(P)} \left(\mathrm{P}_{\theta *,P}(V), \mathrm{P}_{\theta *,P}(W)\right) \\ &= \frac{1}{\theta^2} \langle (\mathrm{mlog} \circ \mathrm{P}_\theta)_{*,P}(V), (\mathrm{mlog} \circ \mathrm{P}_\theta)_{*,P}(W) \rangle^{(\alpha,\beta)} \\ &= \langle \mathrm{mlog}_{*,P}(V), \mathrm{mlog}_{*,P}(W) \rangle^{(\alpha,\beta)} \\ &= g_P^{(\alpha,\beta)\text{-LE}}(V, W). \end{aligned} \tag{48}$$

θ**-LCM:** Let us first review a well-known fact of deformed metrics (Thanwerdas & Pennec, 2022a). Let $\tilde{g} = \frac{1}{\theta^2} \mathrm{P}_\theta^* \, g$ be the power-deformed metric on SPD Then when θ tends to 0, for all $P \in \mathcal{S}^n_{++}$ and all $V \in T_P\mathcal{S}^n_{++}$, we have

$$\tilde{g}_P(V, V) \to g_I(\log_{*,P}(V), \log_{*,P}(V)). \tag{49}$$

By Eq. (49), we can readily obtain the results. □

Proof of Prop. 5.2 . (α, β)-AIM is left-invariant (Thanwerdas & Pennec, 2022b). As the pullback of (α, β)-AIM, (θ, α, β)-AIM is left-invariant as well. Besides, Chen et al. (2023d) shows that LCM is the pullback metric from the Euclidean space of $\mathcal{L}^n$. Therefore, θ-LCM is bi-invariant. □

Proof of Thm. 5.3 . We denote Eqs. (13) to (15) on $\mathcal{M}_i, i = 1, 2$ as the mapping $\xi^i(\cdot|M, v^2, B, s)$. Let $\mathcal{B} = \{P_{1\ldots N}\}$ and $f(\mathcal{B}) = \{f(P_{1\ldots N})\}$.

The core of this proof lies in three points:

1. The Fréchet mean and variance of $\mathcal{B}$ in $\mathcal{M}_1$ correspond to the counterparts of $f(\mathcal{B})$ in $\mathcal{M}_2$.
2. $\xi^1(P_i|M, v^2, B, s)$ in $\mathcal{M}_1$ is equal to $f^{-1}(\xi^2(f(P_i)|f(M), v^2, f(B), s))$.
3. The updates of running statistics in $\mathcal{M}_1$ correspond to the counterparts in $\mathcal{M}_2$.

We denote M as the Fréchet mean of $\mathcal{B}$, and v^2 as the Fréchet variance of $\mathcal{B}$. Then, by the isometry of f, the Fréchet mean and variance of $f(\mathcal{B})$ are $f(M)$ and v^2, respectively.

On $\mathcal{M}_i, i = 1, 2$, we denote $L^i, \odot^i, \mathrm{Exp}^i, \mathrm{Log}^i$ as the Lie group and Riemannian operators, E^i as the neutral element, and Eq. (14) as $\phi^i_s(\cdot)$. With the isometry and Lie group isomorphism of f, we have the following equations:

$$L^1_{M^{-1}_{\odot^1}} = f^{-1} \circ L^2_{f(M)^{-1}_{\odot^2}} \circ f, \tag{50}$$

$$\begin{aligned} \phi^1_s &= \mathrm{Exp}^1_{E^1}\left[s \, \mathrm{Log}^1_{E^1}(\cdot)\right] \\ &= f^{-1}\left(\mathrm{Exp}^1_{E^2}\left[s \, \mathrm{Log}^2_{E^2}(f(\cdot))\right]\right) \\ &= f^{-1} \circ \phi^2_s \circ f, \end{aligned} \tag{51}$$

$$L^1_B = f^{-1} \circ L^2_{f(B)} \circ f. \tag{52}$$

Then we have

$$\xi^1(P_i|M, v^2, B, s) = f^{-1}(\xi^2(f(P_i)|f(M), v^2, f(B), s)) \tag{53}$$

Lastly, we show the correspondence between running statistics. Since the Fréchet variance is the same for both $\mathcal{B}$ and $f(\mathcal{B})$, we focus on the running mean. Let M_r and $f(M_r)$ denote the initial values of the running means in $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively, and WFM^i represent the weighted Fréchet mean in $\mathcal{M}_i$. Then the updated running mean in $\mathcal{M}_1$ is

$$\mathrm{WFM}^1(\{1-\gamma, \gamma\}, \{M_r, M\}) = f^{-1}(\mathrm{WFM}^2(\{1-\gamma, \gamma\}, \{f(M_r), f(M)\})) \tag{54}$$

we can further simply the above equation as

$$\mathrm{WFM}^1 = f^{-1} \circ \mathrm{WFM}^2 \circ f \tag{55}$$

Denoting LieBN^i as the LieBN algorithm on $\mathcal{M}_i$, Eq. (53) and Eq. (55) imply that:

$$\mathrm{LieBN}^1(P_i|B, s, \epsilon, \gamma) = f^{-1}\left[\mathrm{LieBN}^2(f(P_i)|f(B), s, \epsilon, \gamma)\right]. \tag{56}$$

□