Expressivity of congruence-based architectures for DNNs on positive-definite matrices

Antonin Oswald
ICTEAM, UCLouvain
Louvain-la-Neuve, Belgium
antonin.oswald@uclouvain.be

Estelle Massart
ICTEAM, UCLouvain
Louvain-la-Neuve, Belgium
estelle.massart@uclouvain.be

***Abstract*—This work studies neural architectures for classifying symmetric positive-definite matrices, focusing on congruence-like layers, in which the input matrix is multiplied on the left and right by a (possibly rectangular) weight matrix W and its transpose. Such layers lie at the core of the celebrated SPDNet and have also been employed independently for dimensionality reduction on positive-definite data. We show that the (semi)-orthogonality constraint commonly imposed on W limits the expressivity of these layers: for certain activation functions, the resulting architecture collapses to a one-hidden-layer equivalent. This lack of expressivity follows from a loss of spectral diversity in congruence-like layers for semi-orthogonal W and is a direct consequence of Poincaré's separation theorem. We then examine the choice of the final classifier, comparing several Riemannian classifiers and discussing their compatibility with the feature maps produced by congruence-like layers.**

***Index Terms*—Deep learning, Symmetric positive-definite matrices, Riemannian geometry, Machine learning.**

I. Introduction

Symmetric and positive-definite (SPD) matrices arise naturally in a wide range of scientific and engineering applications, where they encode essential second-order information such as correlations between signals or covariances of random variables. Applications range from medical imaging and brain-computer interfaces [4], [22] to pedestrian recognition [30], image set classification [18], [32] and radar data processing [11]. This ubiquity of SPD matrices motivated the design of machine learning algorithms specifically dedicated to this type of data. In particular, the set of SPD matrices naturally inherits a Riemannian manifold structure [26], allowing for the use of Riemannian extensions of usual machine learning techniques.

Various classification methods were proposed for SPD data. The first ones either rely on local linearizations of the manifold [31], or assign each point to the class with the closest Fréchet mean, for several choices of metrics [4], [11], [22]. More recent methods include Riemannian extensions of Gaussian mixture models [27], kernel functions [13], [19], [29] and deep neural networks (DNNs) [17], [20], [23], [24].

Among DNNs, the celebrated SPDNet architecture was widely adopted by the community [17]. This architecture is made of three different types of layers. The first is the BiMap layer, which applies a congruence-like operation to its incoming matrix $X \in \mathrm{SPD}(n)$, where $\mathrm{SPD}(n)$ refers to the manifold of $n \times n$ positive-definite matrices:

$$X \mapsto WXW^\top, \tag{1}$$

for some full-rank $W^\top \in \mathbb{R}^{n\times p}$ ($p \leq n$). To mitigate optimization instabilities caused by the fact that the set of full-rank $n \times p$ matrices is open [8, Sec 2.6], [17] enforces a semi-orthogonality constraint on the weight matrix W, i.e., $WW^\top = I_p$. The second type of layer is the ReEig layer, which applies a ReLU activation function on the eigenvalues of the incoming matrix:

$$X \mapsto U\phi(\Lambda)U^\top, \tag{2}$$

where $X = U\Lambda U^\top$ is an eigenvalue decomposition and $\phi(\Lambda) : \Lambda \mapsto \max(\epsilon I, \Lambda)$ is the ReLU function (shifted by ϵ), applied to the eigenvalues. These two layers are alternated several times, and followed by a LogEig layer, which maps its incoming SPD matrix to a symmetric matrix by replacing the eigenvalues by their logarithm. This matrix is then fed into classical Euclidean network layers for classification.

The main feature of the BiMap and ReEig layers is the fact that they preserve the SPD structure of the data. These layers can thus be seen as a structure-preserving feature extraction block. Note that the congruence-like transformation (1) was already proposed as a dimensionality reduction mechanism on SPD matrices [14], [16], where the parameter W was trained in order to discriminate as well as possible the data, which could also be seen as a (shallow) structure-preserving feature extraction mechanism. On the other hand, and as highlighted in [33], the expressivity of the SPDNet architecture is still poorly understood, which is the focus of this paper.

Note also that, while SPDNet was subsequently endowed with a batch normalization strategy [10] and is at the core of recent extensions of U-Nets and auto-encoders to SPD data [7], [34], several works proposed concurrent architectures by leveraging the hyperbolic geometric structure of SPD data through gyrovector space theory [20], [23], [24]. The authors of [12] generalized convolutional neural networks to manifold–valued data by defining convolution via weighted Fréchet means. More broadly, the research on DNN architecture for manifold-valued data is an active topic, and falls under the umbrella of *geometric deep learning* [9].

Contributions: This paper studies the congruence-like transformation (1), and characterizes how classically recommended constraints on its weight matrices affect expressivity. In Sections III and IV, we show that for certain activation functions, restricting W to be (semi-)orthogonal makes the architecture equivalent to a one-hidden-layer network regardless of depth, a degeneracy that follows from the loss of spectral diversity implied by Poincaré's separation theorem. In Section V, we then study the final classifier, comparing several candidates, including Riemannian ones, through the discriminatory properties of congruence-like layers under the associated metrics.
Notations: $\mathrm{SPD}(n)$, $\mathrm{GL}(n)$ and $\mathcal{O}(n)$ are respectively the sets of $n \times n$ SPD, invertible and orthogonal matrices. We write $\mathrm{St}(n,p) := \left\{Y \in \mathbb{R}^{n\times p} \mid Y^\top Y = I_p\right\}$, with $p \leq n$, the Stiefel manifold. We introduce the notation $\mathcal{I}_{\mathrm{spec}}(A) := [\lambda_{\min}(A), \lambda_{\max}(A)]$, for the interval between the smallest and largest eigenvalue of A. The Frobenius norm is denoted as $\|A\|_F = \left(\mathrm{tr}\left(A^\top A\right)\right)^{1/2}$.

II. Problem formulation

We address the following classification problem. Given a collection of training inputs $X_1, \ldots, X_N$, with $X_i \in \mathrm{SPD}(d)$, and associated class label $y_i \in \mathbb{R}^K$ (assumed without loss of generality to be one-hot encoded), our goal is to learn a neural network $\phi : \mathrm{SPD}(d) \to \mathbb{R}^K : X^0 \mapsto \phi(X^0)$, that fits as well as possible these datapoints in regard to some predefined loss function, and whose architecture is characterized by signal propagation equations of the form:

$$\begin{aligned} X^\ell &= g(W_\ell X^{\ell-1} W_\ell^\top), \qquad \ell = 1, \ldots L \\ \phi(X^0) &= f(X^L), \end{aligned} \tag{3}$$

for some weight matrices $W_\ell \in \mathbb{R}^{d^\ell \times d^{\ell-1}}$ for $\ell = 1, \ldots L$, with $d^0 = d$, some classifier $f : \mathrm{SPD}(d_L) \to \mathbb{R}^K$ and an activation function $g : \mathrm{SPD}(n) \to \mathrm{SPD}(n)$, which we assume here to be identical across layers[1]. We assume here this activation function to be a (primary) matrix function[2] as defined in [15], i.e., to satisfy $g(A) = U g(\Lambda) U^\top$, with $U\Lambda U^\top$ an eigenvalue decomposition of A, and

$$g(\Lambda)_{i,j} = \begin{cases} 0 & \text{if } i \neq j \\ \sigma(\Lambda_{i,i}) & \text{otherwise,} \end{cases} \tag{4}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a scalar function. In particular, this encompasses the ReEig layer used in [17].

In other words, and as illustrated on Figure 1, we assume our model to be made of two blocks: a structure-preserving feature extractor block, that alternates between congruence-like transformations and the nonlinear matrix function g, and a final classifier f. For instance, this classifier could be any Euclidean classifier applied to a mapping of X^L onto a linear space (e.g., through the LogEig layer in [17]), or any Riemannian classifier (e.g., the minimum-distance-to-the-mean classifier used in [4], [11], [22]).

[1]While the dimension of the input and output spaces of g vary in general, we keep the same notation throughout the paper for the simplicity of notation.

[2]Note that this differs from applying a scalar activation function $\sigma : \mathbb{R} \to \mathbb{R}$ to all entries of A, which is another way to define matrix functions that we do not consider here.

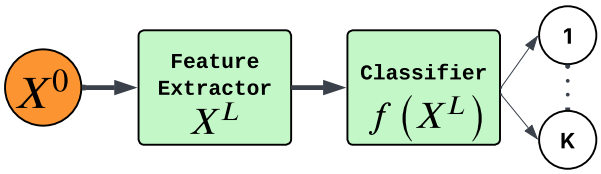

Fig. 1. Illustration of the architecture (3) of depth L.

III. Model expressivity for orthogonal weight matrices

In this section, we prove that, if all layers are square (i.e., $d^\ell = d$ for $\ell = 1, \ldots, L$) and if the weight matrices $W_1, \ldots, W_L$ are orthogonal matrices, there exists a one-hidden-layer neural network $\tilde{\phi} : \mathrm{SPD}(d) \to \mathbb{R}^K$ that is equivalent to (3), i.e., such that $\phi(X) = \tilde{\phi}(X)$ for all $X \in \mathrm{SPD}(d)$.

Proposition 1. *Assume that $d^\ell = d$ and $W_\ell \in \mathcal{O}(d)$ for $\ell = 1, \ldots, L$ in (3). Let us define the one-hidden-layer network $\tilde{\phi} : \mathrm{SPD}(d) \to \mathbb{R}^K : X^0 \mapsto \tilde{\phi}(X^0)$ such that*

$$\begin{aligned} \tilde{X}^1 &= g^{(L)}\left(\tilde{W}_L X^0 \tilde{W}_L^\top\right), \\ \tilde{\phi}(X^0) &= f\left(\tilde{X}^1\right), \end{aligned} \tag{5}$$

where $\tilde{W}_L = (W_L \ldots W_2 W_1) \in \mathcal{O}(d)$ and with activation function $g^{(L)}$ being $g(\cdot)$ composed L times with itself. Then,

$$\phi(X) = \tilde{\phi}(X) \qquad \forall X \in \mathrm{SPD}(d).$$

Proof. Note that there suffices to show that, for all L and for all $X^0 \in \mathrm{SPD}(d)$,

$$X^L = \tilde{X}^1,$$

where X^L and $\tilde{X}^1$ are defined in (3) and (5), respectively. The proof is by induction. Trivially, the result holds for $L = 1$. For the induction step, assuming that the claim holds for a depth L, there holds by (3)

$$\begin{aligned} X^{L+1} &= g\left(W_{L+1} X^L W_{L+1}^\top\right) \\ &= g\left(W_{L+1} g^{(L)}\left(\tilde{W}_L X^0 \tilde{W}_L^\top\right) W_{L+1}^\top\right) \\ &= g^{(L+1)}\left(\tilde{W}_{L+1} X^0 \tilde{W}_{L+1}^\top\right), \end{aligned}$$

where the last equality results from our definition of g as matrix function, which implies that g is invariant under congruence with orthogonal matrices; see [15, Thm. 1.13]. □

As a consequence, when all layers in (3) are square and all weight matrices are orthogonal, the expressivity of the neural network architecture (3) always reduces to the one of a one-hidden-layer neural network, with depth-depending activation function $g^{(L)}$. The following remark illustrates a possible drop in expressivity for some choices of activation functions.

Remark 1. *Assume that $\sigma : \mathbb{R} \to \mathbb{R}$, defined in* (4)*, is a contractive function, i.e.,*

$$|\sigma(x) - \sigma(y)| \leq c|x - y| \qquad \forall x, y \in \mathbb{R}, \tag{6}$$

for some positive constant $c < 1$. Then, by the Banach fixed-point theorem, $\lim_{L\to\infty} \sigma^{(L)}(x) = a$ for some unique fixed-point $a \in \mathbb{R}$. Our definition of matrix function g in (4) *implies then that the spectrum of X^L defined in* (3) *becomes more and more uniform as L increases, converging ultimately to some constant value. This will occur for instance when σ is the sigmoid, i.e., the mapping $x \mapsto (1 + \exp(-x))^{-1}$, which satisfies* (6) *for $c = 0.25$.*

A stronger version of Proposition 1 can be derived for idempotent activation functions, which satisfy the property $g(g(X)) = g(X)$ for all X in their domain. Note that such idempotent functions can be obtained by choosing σ in (4) as a scalar idempotent function, which includes the *ReLU* (as in the ReEig layer in [17]), and *hard tanh* [25].

Corollary 1. *Under the assumptions of Proposition 1, let us define the one-hidden-layer network $\tilde{\phi} : \mathrm{SPD}(d) \to \mathbb{R}^K : X^0 \mapsto \tilde{\phi}(X^0)$ such that*

$$\tilde{X}^1 = g\left(\tilde{W}_L X^0 \tilde{W}_L^\top\right),$$
$$\tilde{\phi}(X^0) = f\left(\tilde{X}^1\right),$$

where $\tilde{W}_L = (W_L \dots W_2 W_1) \in \mathcal{O}(d)$. If g is idempotent, then

$$\phi(X) = \tilde{\phi}(X) \qquad \forall X \in \mathrm{SPD}(d).$$

Proof. This results from Proposition 1 combined with the fact that $g(g(X)) = g(X)$ for all $X \in \mathrm{SPD}(d)$. □

When g is idempotent, the expressivity of (3) is thus the expressivity of a one-hidden-layer neural network with the same activation function.

Remark 2. *Although depth does not increase expressivity in this setting, it may still affect optimization and the final accuracy of the model. The resulting overparameterization could act as a preconditioner during training (i.e., accelerating the convergence of the weights), or as a regularization mechanism; see [1] and [2] for related works in the Euclidean setting.*

IV. Model Expressivity for semi-orthogonal weight matrices

We next address the case where the weight matrices in (3) are semi-orthogonal. The following classical result characterizes the action of semi-orthogonal matrices on SPD data through the congruence-like transformation (1).

Theorem 1 (Poincaré Separation Theorem, see [5])**.** *Let A be an $n \times n$ real symmetric matrix, and $B \in \mathrm{St}(n, p)$. Let λ_j be the j-th eigenvalue of A for $j = 1, \dots, n$, and μ_i the i-th eigenvalue of $B^\top AB$ for $i = 1, \dots, p$ in descending order. Then,*

$$\lambda_i \geq \mu_i \geq \lambda_{i+n-p}, \quad i = 1, \dots, p.$$

That is, $\mathcal{I}_{\mathrm{spec}}\left(B^\top AB\right) \subseteq \mathcal{I}_{\mathrm{spec}}(A)$, where $\mathcal{I}_{\mathrm{spec}}(A)$ is the interval between the smallest and largest eigenvalue of A.

In other words, the congruence-like transformation (1) modifies the spectrum of the matrices in a controlled way, which implies that the expressivity of this architecture does not benefit from depth for some choices of activation functions.

Proposition 2. *Assume that the scalar activation function $\sigma : \mathbb{R} \to \mathbb{R}$ in* (4) *is the identity on some interval $\mathcal{I}$, i.e.,*

$$\sigma(x) = x \qquad \forall x \in \mathcal{I},$$

and let $\mathcal{X} \subseteq \mathrm{SPD}(d)$ be the set of matrices whose spectrum is contained in $\mathcal{I}$, i.e.,

$$\mathcal{X} = \{X \in \mathrm{SPD}(d) : \mathcal{I}_{\mathrm{spec}}(X) \subseteq \mathcal{I}\}.$$

Consider the neural network ϕ defined in (3)*, assuming that $d^0 = d \geq d^1 \geq \cdots \geq d^L$, that $W_\ell^\top \in \mathrm{St}(d^{\ell-1}, d^\ell)$ for all $\ell = 1, \dots, L$ in* (3)*, and that g is the matrix activation function associated to σ by* (4)*. Let us define the alternative neural network $\tilde{\phi} : \mathrm{SPD}(d) \to \mathbb{R}^K : X^0 \mapsto \tilde{\phi}(X^0)$ by*

$$\begin{aligned} \tilde{X}^1 &= \tilde{W}_L X^0 \tilde{W}_L^\top, \\ \tilde{\phi}(X^0) &= f\left(\tilde{X}^1\right), \end{aligned} \tag{7}$$

where $\tilde{W}_L^\top = (W_L \dots W_2 W_1)^\top \in \mathrm{St}(d, d^L)$. Then,

$$\phi(X) = \tilde{\phi}(X), \qquad \forall X \in \mathcal{X}.$$

Proof. As for Proposition 1, it is sufficient to prove that for all $X^0 \in \mathcal{X}$,

$$X^L = \tilde{X}^1,$$

where X^L and $\tilde{X}^1$ are defined in (3) and (7), respectively. The proof is by induction on L. We first check the result for $L = 1$. Let $X^0 \in \mathcal{X}$, and let $W^\top \in \mathrm{St}(d, d^1)$. Then, by definition of $\mathcal{X}$, $\mathcal{I}_{\mathrm{spec}}\left(X^0\right) \subseteq \mathcal{I}$, and by Theorem 1,

$$\mathcal{I}_{\mathrm{spec}}\left(WX^0W^\top\right) \subseteq \mathcal{I}_{\mathrm{spec}}\left(X^0\right) \subseteq \mathcal{I}.$$

This implies that $g\left(WX^0W^\top\right) = WX^0W^\top$, which proves the claim for $L = 1$. Assuming that the claim holds for a depth L, we prove that it holds for depth $L + 1$. Since, by Theorem 1,

$$\mathcal{I}_{\mathrm{spec}}\left(W_{L+1}X^L W_{L+1}^\top\right) \subseteq \mathcal{I}_{\mathrm{spec}}\left(X^L\right) \subseteq \mathcal{I},$$

there holds

$$\begin{aligned} X^{L+1} &= g\left(W_{L+1}X^L W_{L+1}^\top\right) \\ &= W_{L+1}X^L W_{L+1}^\top \\ &= \tilde{W}_{L+1}X^0 \tilde{W}_{L+1}^\top. \end{aligned}$$

□

The following result extends this proposition to an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ whose image is contained in the interval $\mathcal{I}$ on which it operates as the identity (as the ReLU activation). In this case, all layers except the first one collapse to a single congruence-like transformation of the form (1) for all $X \in \mathrm{SPD}(d)$.

Corollary 2. *Let the assumptions of Proposition 2 hold, and assume that the image of σ is included in the interval $\mathcal{I}$ on which its acts as the identity (note that such a σ is a special case of idempotent function). Let $\tilde{\phi} : \mathrm{SPD}(d) \to \mathbb{R}^K : X^0 \mapsto \tilde{\phi}(X^0)$ be such that*

$$\tilde{X}^1 = \tilde{W}_L g\left(W_1 X^0 W_1^\top\right) \tilde{W}_L^\top,$$
$$\tilde{\phi}(X^0) = f\left(\tilde{X}^1\right),$$

with $\tilde{W}_L^\top = \left(W_L \dots W_2 W_1\right)^\top \in \mathrm{St}\left(d^2, d^L\right)$ and $L \geq 2$. Then

$$\phi(X) = \tilde{\phi}(X) \qquad \forall X \in \mathrm{SPD}(d).$$

Proof. This follows from Proposition 2, replacing X^0 by X^1, whose spectrum belongs to $\mathcal{I}$ due to the assumption on σ. □

To conclude this section, let us mention that the redundancy of the ReEig layers in SPDNet with (semi-)orthogonal weight matrices was numerically observed in [35, Sec. 3.6].

V. Interaction between the feature extractor block and the final classifier

This last section addresses the choice of the final classifier f in (3). In the SPDNet architecture, f is a Euclidean classifier applied to the output of the transformation $X^L \mapsto \log\left(X^L\right)$, where $\log$ is the principal logarithm[3]. An alternative strategy[4] is to classify directly the extracted features using a shallow classifier on the manifold $\mathrm{SPD}(d^L)$, e.g., the Minimum Riemannian Distance to Mean (MRDM) classifier used in [4], [11], [22]. This classifier assigns each point to the class whose Fréchet mean is the closest, and relies on a notion of distance between two SPD matrices. In this section, we discuss the compatibility between the choice of the distance and the feature extraction block. We show that several celebrated distances between SPD matrices are invariant under congruence with orthogonal/invertible matrices, and classifiers relying on these distances may thus be less indicated when using the congruence-like transformation (1) for feature extraction.

Definition 1. *We recall the definitions of the following classical distances/divergences on $\mathrm{SPD}(n)$.*

- *The (classical) affine-invariant distance (AI, see [26]) between $A, B \in \mathrm{SPD}(n)$ is defined by*

$$\delta_{\mathrm{AI}}(A, B) = \left\| \log\left(A^{-\frac{1}{2}} B A^{-\frac{1}{2}}\right) \right\|_F.$$

- *The log-Euclidean distance (LE, see [3]) between $A, B \in \mathrm{SPD}(n)$ is defined by*

$$\delta_{LE}(A, B) = \left\| \log(A) - \log(B) \right\|_F.$$

- *The Stein divergence (S, see [28]) between $A, B \in \mathrm{SPD}(n)$ is defined as*

$$\delta_S(A, B) = \left(\log\det\left(\frac{A+B}{2}\right) - \frac{1}{2}\log\det(AB) \right)^{\frac{1}{2}}.$$

- *The Bures-Wasserstein distance (BW, see [21]) between $A, B \in \mathrm{SPD}(n)$ is defined as*[5]

$$\delta_{\mathrm{BW}}(A, B) = \left(\mathrm{tr}(A) + \mathrm{tr}(B) - 2\mathrm{tr}\left(A^{\frac{1}{2}} B A^{\frac{1}{2}}\right)^{\frac{1}{2}} \right)^{\frac{1}{2}}.$$

- *Finally, we recall the Euclidean (E) distance between $A, B \in \mathrm{SPD}(n)$, which is simply defined as*

$$\delta_{\mathrm{E}}(A, B) = \|A - B\|_{\mathrm{F}}.$$

We next discuss the (possibly lack of) invariances of these distances under congruence transformations with orthogonal and invertible weight matrices, respectively, and comment on the consequences on the expressivity of the feature extraction blocks relying on the congruence-like transformation (1), when combined with a simple (e.g., MRDM-like) classifier; see Table I.

a) Congruence with orthogonal weight matrices: we show that, for all the distances and divergences in Definition 1, there holds:

$$\delta(A, B) = \delta\left(WAW^\top, WBW^\top\right) \tag{8}$$

for all $A, B \in \mathrm{SPD}(n)$ and for all $W \in \mathcal{O}(n)$. This result is well-known for the log-Euclidean and Euclidean distances, see [3, Prop. 3.11], and [6, Section 4.2], respectively. The affine-invariant distance and the Stein divergence are invariant under congruence with arbitrary (invertible) matrices: for all $W \in \mathrm{GL}(n)$ and $A, B \in \mathrm{SPD}(n)$, $\delta(A, B) = \delta\left(WAW^\top, WBW^\top\right)$. These results can be found in [26, Sec. 3.2] and in [28, Prop. 2.3] for the affine-invariant distance and the Stein divergence, respectively. As a result, invariance under congruence also trivially holds when W is orthogonal. Finally, the invariance of the Bures-Wasserstein distance is a consequence of the invariance under similarity of the trace and matrix square root.

There follows that, regardless of the nonlinearity, the congruence-like layers (1), with orthogonal weight matrices, do not result in any gain of expressivity: they preserve both the spectrum of the matrices (hence, do not affect the result of successive activation functions, see Proposition 1), and the distance separating any two points from the dataset.

b) Congruence with arbitrary non-singular matrices: For $W \in \mathrm{GL}(n)$, (8) does not hold for the log-Euclidean, Bures-Wasserstein and Euclidean distances. As stated above, (8) still holds for the affine-invariant distance and the Stein divergence, so that, for these two distances, congruence-like layers (1) do not allow to separate points from different classes when used in isolation. The expressivity gains of these layers are then limited to the resulting modifications of the spectrum

[3]The principal logarithm is defined as $\log(A) := U\mathrm{diag}\left(\log\lambda_1, \dots, \log\lambda_n\right) U^\top$, for $A = U\mathrm{diag}\left(\lambda_1, \dots, \lambda_n\right) U^\top$ an eigenvalue decomposition.

[4]Note that SPDNet is directly related to the log-Euclidean metric, as Euclidean distances between the outputs of the log operator coincide with log-Euclidean distances between the original SPD matrices.

[5]The matrix square root is defined as $A^{\frac{1}{2}} := U\mathrm{diag}\left(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n}\right) U^\top$, with $U\mathrm{diag}\left(\lambda_1, \dots, \lambda_n\right) U^\top$ and eigenvalue decomposition.

	AI	LE	Stein	BW	E
$W \in \mathrm{GL}(n)$	Yes	No	Yes	No	No
$W \in \mathcal{O}(n)$	Yes	Yes	Yes	Yes	Yes

TABLE I
VALIDITY OF (8) FOR THE DISTANCES CONSIDERED.

of the matrices before the application of the nonlinearity in the feature extraction block.

VI. CONCLUSION

We explored DNN architectures for SPD matrices that involve a structure-preserving feature extraction block made of congruence-like layers and nonlinear matrix functions, followed by a final classifier. We showed that imposing (semi)-orthogonal constraints on congruence weights can collapse deep architectures to one-hidden-layer networks for certain activation function, reducing expressivity. We also noted that several common distances between SPD matrices are invariant under congruences, preventing the resulting classifiers to separate data from different classes. Future work could explore the design of new activation functions to improve expressivity.

ACKNOWLEDGMENT

This work was supported by the *GRAVIT-AI* Concerted Research Action (ARC) of the Fédération Wallonie-Bruxelles. E. M. work is also partly funded by the FRS-FNRS Research Project NTTN (grant number TW02223).

REFERENCES

[1] S. Arora, N. Cohen, and E. Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[2] S. Arora, N. Cohen, W. Hu, and Y. Luo. Implicit Regularization in Deep Matrix Factorization. In *Proceedings of the 33rd Conf. on Neural Information Processing Systems*, 2019.

[3] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Geometric Means in a Novel Vector Space Structure on Symmetric Positive-Definite Matrices. *SIAM J. on Matrix Analysis and Applications*, 29(1):328–347, 2007.

[4] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten. Multiclass brain-computer interface classification by Riemannian geometry. *IEEE Tr. on Biomedical Engineering*, 59(4):920–928, 2012.

[5] R. Bellman. *Introduction to Matrix Analysis, Second Edition*. SIAM, 1997.

[6] R. Bhatia. *Matrix Analysis*, volume 169 of *Graduate Texts in Mathematics*. Springer, 1997.

[7] C. Boucherie, T. de Surrel, and F. Yger. SPDNet-AE: a Compact SPD Representation through Riemannian Autoencoding. In *34th European Symposium on Artificial Neural Networks*, 2026.

[8] N. Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[9] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(3):18–42, 2017.

[10] D. Brooks, O. Schwander, F. Barbaresco, J.-Y. Schneider, and M. Cord. Riemannian batch normalization for SPD neural networks. In *Proceedings of the 33rd International Conf. on Neural Information Processing Systems*, 2019.

[11] Y. Cabanes, F. Barbaresco, M. Arnaudon, and J. Bigot. Toeplitz Hermitian positive definite matrix machine learning based in Fisher metric. In *Proceedings of Geometric Science of Information*, 2019.

[12] R. Chakraborty, J. Bouza, J. H. Manton, and B. C. Vemuri. ManifoldNet: A Deep Neural Network for Manifold-Valued Data with Applications. *IEEE Tr. on Pattern Analysis and Machine Intelligence*, 44(2):799–810, 2022.

[13] M. Harandi and M. Salzmann. Riemannian coding and dictionary learning: Kernels to the rescue. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3926–3935, 2015.

[14] M. Harandi, M. Salzmann, and R. Hartley. Dimensionality Reduction on SPD Manifolds: The Emergence of Geometry-Aware Methods. *IEEE Tr. on Pattern Analysis and Machine Intelligence*, 40(1):48–62, 2018.

[15] N. J. Higham. *Functions of Matrices*. SIAM, 2008.

[16] I. Horev, F. Yger, and M. Sugiyama. Geometry-aware principal component analysis for symmetric positive definite matrices. In *Proceedings of the Asian Conf. on Machine Learning*, pages 1–16, 2022.

[17] Z. Huang and L. Van Gool. A Riemannian network for SPD matrix learning. In *Proceedings of the 31st AAAI Conf. on Artificial Intelligence*, page 2036–2042, 2017.

[18] C. Ionescu, J. Carreira, and C. Sminchisescu. Iterated second-order label sensitive pooling for 3d human pose estimation. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1661–1668, 2014.

[19] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi. Kernel Methods on the Riemannian Manifold of Symmetric Positive Definite Matrices. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, page 73–80, 2013.

[20] F. López, B. Pozzetti, S. Trettel, M. Strube, and A. Wienhard. Vector-valued distance and gyrocalculus on the space of symmetric positive definite matrices. In *Proceedings of the 35th International Conf. on Neural Information Processing Systems*, 2021.

[21] E. Massart and P.-A. Absil. Quotient geometry with simple geodesics for the manifold of fixed-rank positive-semidefinite matrices. *SIAM J. on Matrix Analysis and Applications*, 41(1):171–198, 2020.

[22] E. Massart and S. Chevallier. Inductive means and sequences applied to online classification of EEG. In *Proceedings of Geometric Science of Information*, 2017.

[23] X. S. Nguyen and S. Yang. Building Neural Networks on Matrix Manifolds: A Gyrovector Space Approach. In *Proceedings of the 40th International Conf. on Machine Learning*, 2023.

[24] X.S Nguyen, S. Yang, and A. Histace. Matrix manifold neural networks++. In *Proceedings of the 12th International Conf. on Learning Representations*, 2024.

[25] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[26] X. Pennec, P. Fillard, and N. Ayache. A Riemannian Framework for Tensor Computing. *International J. of Computer Vision*, 66(1):41–66, 2006.

[27] S. Said, L. Bombrun, Y. Berthoumieu, and J. H. Manton. Riemannian Gaussian distributions on the Space of Symmetric Positive Definite Matrices. *IEEE Tr. on Information Theory*, 63:2153–2170, 2017.

[28] S. Sra. A new metric on the manifold of kernel matrices with application to matrix geometric means. In *Proceedings of the 26th Conf. on Neural Information Processing Systems*, 2012.

[29] F. Steinert, S. Said, and C. Mostajeran. Universal Kernels via Harmonic Analysis on Riemannian Symmetric Spaces. In *Proceedings of Geometric Science of Information*, 2025.

[30] D. Tosato, M. Farenzena, M. Spera, V. Murino, and M. Cristani. Multi-class Classification on Riemannian Manifolds for Video Surveillance. In *Proceedings of the 11th European Conf. on Computer Vision*, pages 378–391, 2010.

[31] O. Tüzel, F. Porikli, and P. Meer. Pedestrian Detection via Classification on Riemannian Manifolds. *IEEE Tr. on Pattern Analysis and Machine Intelligence*, 30(10):1713–1727, 2008.

[32] 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 Conf. on Computer Vision and Pattern Recognition*, pages 2496–2503, 2012.

[33] R. Wang, X.-J. Wu, Z. Chen, T. Xu, and J. Kittler. DreamNet: A Deep Riemannian Manifold Network for SPD Matrix Learning. In *Proceedings of the Asian Conf. on Computer Vision*, pages 3241–3257, 2022.

[34] R. Wang, X.-J. Wu, T. Xu, C. Hu, and J. Kittler. U-SPDNet: An SPD manifold learning-based neural network for visual classification. *Neural Networks*, 161:382–396, 2023.

[35] D. Wilson, R. T. Schirrmeister, L. A. W. Gemein, and T. Ball. Deep Riemannian Networks for end-to-end EEG decoding. *Imaging Neuroscience*, 3, 2025.