CONTROLLING THE FRÉCHET VARIANCE IMPROVES BATCH NORMALIZATION ON THE SYMMETRIC POSITIVE DEFINITE MANIFOLD

Reinmar J. Kobler[1,2]*, Jun-ichiro Hirayama*[1]*, Motoaki Kawanabe*[1,2]

[1]RIKEN Center for Advanced Intelligence Project (RIKEN AIP), Kyoto 619-0288, Japan

[2]Advanced Telecommunications Research Institute International (ATR), Kyoto 619-0288, Japan

ABSTRACT

Symmetric positive definite (SPD) matrices, and in particular covariance matrices as data descriptors find widespread application in various fields but also pure machine learning. SPD matrices form a Riemannian manifold, demanding machine learning methods that take this structure into account. In this work, we extend upon previous works and propose a batch normalization algorithm for the SPD manifold that can be readily combined with SPD neural networks and unlike previous works controls both the Fréchet mean and variance on the SPD manifold. The proposed method is validated in simulations and datasets with small sample sizes from three application domains: action recognition from human motion trajectories, image classification and mental imagery detection of electroencephalographic (EEG) signals. The combined results show a systematic performance increase upon previous works and tangent space approximations, as well as improved robustness to low signal-to-noise ratios and lack of data as often encountered in neuroimaging and brain-computer interfacing (BCI).

Index Terms— batch normalization, SPD manifold, scarce data, neural network, Fréchet variance

1. INTRODUCTION

Covariance matrix descriptors as representations for data have proved their usefulness in a variety of machine learning tasks, including object detection in images [1], connectivity analysis in neuroimaging [2], analysis of time-series data of human behavior for action/emotion recognition [3, 4], and electrophysiology for brain-computer interfaces (BCIs) [5, 6]. Excluding rank-deficient cases, covariance matrices are symmetric and positive definite (SPD). As such, they are constrained to a Riemannian manifold [7]. That is why many standard machine learning techniques, which implicitly assume a Euclidean space, do not perform well for SPD data.

Within the last years, state-of-the-art performance has been reported in various application domains for geometry aware learning methods, ranging from simple nearest neighbor methods [5], up to neural network architectures [1, 8]. In the context of neural networks, Huang and Van Gool [9] proposed SPDNet, an architecture that preserves the SPD manifold structure. The original architecture introduced two principled layers that mimic standard linear and ReLU layers, and one layer that projects data from the SPD manifold to Euclidean space where standard layers can be applied. Since then multiple extensions have been proposed [10, 11].

Recently, Brooks et al. [12] introduced a version of batch normalization [13] that controls the Fréchet mean of a batch of points on the SPD manifold, and demonstrated systematic performance gains - particularly in the regime of scarce data. Later, Lou et al. [8] proposed a batch normalization variant that controls both Fréchet mean and variance on the Hyperboloid and Poincaré ball manifolds. Knowing that controlling the first and second order statistics facilitates learning via smoothing of the energy landscape in Euclidean space [14], we surmise that controlling both the Fréchet mean and variance on the SPD manifold would boost the performance gains of [12]. Unlike [8], we propose and evaluate a batch normalization variant for the SPD manifold and focus on the scarce data regime, often encountered in neuroimaging and BCI datasets where recording data is expensive and time-consuming. In this regime, training is typically done with small minibatches and potentially non-iid data (e.g., different subjects or recording sessions). The performance of standard batch normalization can drop substantially in this setting [15]. To address this issues, our proposed method follows the idea of batch renormalization [15] which relies on running estimates of the first and second order statistics during training and testing. In extensive simulations and experiments with diverse datasets, we show that our proposed method not only improves the performance upon previous methods [9, 12] but also upon standard batch normalization methods in Euclidean space projections.

The remainder of this work is structured as follows: we first recall important geometric operations on the SPD manifold, before the proposed batch normalization method is introduced; then, the baseline SPDNet architecture and batch normalization methods are outlined; finally, the proposed method is experimentally validated against several baseline methods.

2. METHODS

2.1. Riemannian Geometry of the SPD manifold

We start with recalling some notions of geometry on the space of real $d \times d$ SPD matrices $\mathcal{S}_d^+$. The space $\mathcal{S}_d^+ = \{\mathbf{X} \in \mathbb{R}^{d\times d} : \mathbf{X}^T = \mathbf{X}, \mathbf{X} \succ 0\}$ forms a cone shaped Riemannian manifold in $\mathbb{R}^{d\times d}$ [7]. On Riemannian manifolds, distances between two points are naturally measured by the length of the shortest curve (i.e., the geodesic) that connects the points. The geometric distance along geodesics [16, 7], sometimes called as the affine invariant Riemannian metric, is calculated by:

$$\delta_g(\mathbf{P}, \mathbf{X}) = ||\log(\mathbf{P}^{-\frac{1}{2}}\mathbf{X}\mathbf{P}^{-\frac{1}{2}})||_F = \delta_g(\mathbf{I}, \mathbf{P}^{-\frac{1}{2}}\mathbf{X}\mathbf{P}^{-\frac{1}{2}}) \quad (1)$$

where $\mathbf{P}$ and $\mathbf{X}$ are two SPD matrices, $\mathbf{I}$ is the identity matrix, $\log(\cdot)$ denotes the matrix logarithm and $||\cdot||_F$ the Frobenius norm. Points along a geodesic that connects $\mathbf{X}_1$ and $\mathbf{X}_2$ can be interpolated in closed form:

$$\mathbf{X}_{12}(\gamma) = \mathbf{X}_1^{\frac{1}{2}} \left(\mathbf{X}_1^{-\frac{1}{2}}\mathbf{X}_2\mathbf{X}_1^{-\frac{1}{2}}\right)^{\gamma} \mathbf{X}_1^{\frac{1}{2}} \quad (2)$$

E-mail:{reinmar.kobler,motoaki.kawanabe}@riken.jp

The authors thank Gernot Müller-Putz and the Graz BCI racing team for sharing the multi-session EEG dataset.

where the weight $\gamma \in [0, 1]$ defines the step size along the geodesic.

Another important concept of Riemannian manifolds are tangent spaces. A tangent space $\mathcal{T}_{\mathbf{P}}\mathcal{M}$ at point $\mathbf{P} \in \mathcal{M}$ on a Riemannian manifold $\mathcal{M}$, is a Euclidean vector space containing derivatives of curves crossing $\mathbf{P}$. Distances on the tangent space are easy to compute and locally approximate distances on the manifold. Generally, the logarithmic mapping $\mathrm{Log}_{\mathbf{P}} : \mathcal{M} \to \mathcal{T}_{\mathbf{P}}\mathcal{M}$ projects points from the manifold to the tangnet space at point $\mathbf{P}$ and the exponential mapping $\mathrm{Exp}_{\mathbf{P}} : \mathcal{T}_{\mathbf{P}}\mathcal{M} \to \mathcal{M}$ back to the manifold. On $\mathcal{S}_d^+$ both mappings have closed form expressions:

$$\mathrm{Log}_{\mathbf{P}}(\mathbf{X}) = \mathbf{P}^{\frac{1}{2}} \log(\mathbf{P}^{-\frac{1}{2}} \mathbf{X} \mathbf{P}^{-\frac{1}{2}}) \mathbf{P}^{\frac{1}{2}} \tag{3}$$

$$\mathrm{Exp}_{\mathbf{P}}(\mathbf{S}) = \mathbf{P}^{\frac{1}{2}} \exp(\mathbf{P}^{-\frac{1}{2}} \mathbf{S} \mathbf{P}^{-\frac{1}{2}}) \mathbf{P}^{\frac{1}{2}} \tag{4}$$

Later, we will also need notions of mean and dispersion. For a set of SPD points $\{\mathbf{X}_n \in \mathcal{S}_d^+\}_{n \leq N}$, we will use the notion of weighted Fréchet mean $\mathbf{G} \in \mathcal{S}_d^+$ and Fréchet variance $\nu^2 \in \mathbb{R}^+$. The weighted Fréchet mean is defined as the minimizer of the sum of squared distances:

$$\mathbf{G} = \arg\min_{\mathbf{P} \in \mathcal{S}_{\mathrm{d}}^+} \sum_{n=1}^{N} w_n \delta_g^2(\mathbf{P}, \mathbf{X}_n) \tag{5}$$

where the weights w_n are larger than 0 and typically sum to 1. The weighted Fréchet variance ν^2 is the attained value at the minimizer $\mathbf{G}$. The minimizer can be computed iteratively using the Karcher flow algorithm [17]. The algorithm alternates between projecting the points to the tangent space at the previous estimate, averaging, and projecting the result back to form the new estimate. The arithmetic mean is typically used as initial estimate.

2.2. Batch normalization on the SPD manifold

The standard batch normalization (BN) algorithm [13] implicitly assumes a Gaussian distribution on $\mathbb{R}^d$ with mean $\mathbf{g}_t$ and diagonal covariance matrix $\mathbf{\Sigma}_t^2$ for each batch t. The maximum likelihood estimates are used to normalize the inputs $\mathbf{x}$ before they are scaled and shifted with learnable parameters $\mathbf{\Sigma}_\theta^2$ and $\mathbf{g}_\theta$ to form the outputs $\mathbf{y}$:

$$\mathbf{y} = \mathbf{\Sigma}_\theta \mathbf{\Sigma}_t^{-1} (\mathbf{x} - \mathbf{g}_t) + \mathbf{g}_\theta \tag{6}$$

On $\mathcal{S}_d^+$, several notions of Gaussian density have been proposed [18, 19]. We follow [18] which defines a Gaussian distribution on $\mathcal{S}_d^+$, whose density

$$p(\mathbf{X}|\mathbf{G}, \sigma) = \frac{1}{\zeta(\sigma)} \exp\left(-\frac{\delta_g^2(\mathbf{G}, \mathbf{X})}{2\sigma^2}\right) \tag{7}$$

is parametrized with a mean $\mathbf{G} \in \mathcal{S}_d^+$ and scalar variance $\sigma^2 \in \mathbb{R}^+$. The normalization constant $\zeta(\sigma)$ merely depends on σ. For a set of SPD points, the maximum likelihood estimate of the mean corresponds to their geometric mean (5). There is no closed form solution for σ^2. However, it is a function of the Fréchet variance $\sigma^2 = \Phi(\nu^2)$ with $\Phi(\cdot)$ strictly increasing. Consequently, controlling ν^2 also controls σ^2. To shift a batch of SPD points with Fréchet mean $\mathbf{G}$ to a parametrized mean $\mathbf{G}_\theta$, parallel transport can be used [20, 21, 12]:

$$\Gamma_{\mathbf{G} \to \mathbf{G}_\theta}(\mathbf{X}) = (\mathbf{G}^{-1}\mathbf{G}_\theta)^{\frac{1}{2}} \mathbf{X} (\mathbf{G}_\theta \mathbf{G}^{-1})^{\frac{1}{2}} \tag{8}$$

The Fréchet variance ν^2 can be rescaled efficiently without explicitly computing geometric distances [22]. Using (8) to rewrite (1), we get

Algorithm 1: SPD batch normalization

Input : batch $\mathcal{X}_t = \{\mathbf{X}_n \in \mathcal{S}_d^+\}_{n \leq N}$,
running statistics $(\mathbf{G}_{t-1}, \nu_{t-1})$,
parameters $(\mathbf{G}_\theta, \nu_\theta)$, momentum $\eta \in [0, 1]$
Output: normalized batch $\mathcal{Y}_t = \{\mathbf{Y}_n \in \mathcal{S}_d^+\}_{n \leq N}$

if *training* **then**

$$\mathbf{G}_t = \arg\min_{\mathbf{G} \in \mathcal{S}_{\mathrm{d}}^+} (1-\eta)\delta_g^2(\mathbf{G}, \mathbf{G}_{t-1}) + \frac{\eta}{N} \sum_{n=1}^{N} \delta_g^2(\mathbf{G}, \mathbf{X}_n)$$

$$\nu_t^2 = (1-\eta)\nu_{t-1}^2 + \frac{\eta}{N} \sum_{n=1}^{N} \delta_g^2(\mathbf{G}_t, \mathbf{X}_n) + (1-\eta)\eta\delta_g^2(\mathbf{G}_t, \mathbf{G}_{t-1})$$

else

$(\mathbf{G}_t, \nu_t^2) = (\mathbf{G}_{t-1}, \nu_{t-1}^2)$

end

$\mathcal{Y}_t = \{\Gamma_{\mathbf{I} \to \mathbf{G}_\theta}(\Gamma_{\mathbf{G}_t \to \mathbf{I}}(\mathbf{X}_n)^{\frac{\nu_\theta}{\nu_t}}) : \mathbf{X}_n \in \mathcal{X}_t\}$

$\delta_g(\mathbf{G}, \mathbf{X}) = \delta_g(\mathbf{I}, \Gamma_{\mathbf{G} \to \mathbf{I}}(\mathbf{X}_n))$. After some basic transformations rescaling the minimizer ν^2 of (5) by γ^2 results in

$$\gamma^2 \nu^2 = \sum_{n=1}^{N} w_n \delta_g^2 \left(\mathbf{I}, \Gamma_{\mathbf{G} \to \mathbf{I}} (\mathbf{X}_n)^\gamma\right) \tag{9}$$

which implies that the Fréchet variance can be changed by a factor γ^2 via parallel transport of the points in the batch from $\mathbf{G}$ to $\mathbf{I}$ (8) and moving the resulting points along the geodesics that connect them to $\mathbf{I}$, as defined in (2). Putting everything together, a batch $\mathcal{X}_t = \{\mathbf{X}_n \in \mathcal{S}_d^+\}_{n \leq N}$ with Fréchet mean and variance $(\mathbf{G}_t, \nu_t)$ can be rescaled and shifted to vary around a new mean $\mathbf{G}_\theta$ with variance ν_θ:

$$\mathcal{Y}_t = \{\Gamma_{\mathbf{I} \to \mathbf{G}_\theta}(\Gamma_{\mathbf{G}_t \to \mathbf{I}}(\mathbf{X}_n)^{\frac{\nu_\theta}{\nu_t}}) : \mathbf{X}_n \in \mathcal{X}_t\} \tag{10}$$

As a last ingredient for our SPD BN algorithm, we need to keep running estimates of $\mathbf{G}_t$ and ν_t. Following [23], we adopt a recursive Fréchet mean and variance estimator. For each new batch t, we use a momentum $\eta \in [0, 1]$ and (5) to update $\mathbf{G}_t$. The previous geometric mean $\mathbf{G}_{t-1}$ receives the weight $1 - \eta$, while the weight η is shared across points in the batch. To speed up learning, we run one iteration of the Karcher flow algorithm [12]. With a sufficiently small learning rate, the data distribution changes only marginally between successive batches, effectively resulting in multiple Karcher flow iterations across batches. We now have all the ingredients for the proposed SPD BN algorithm 1.

Following [12], we used algorithm 1 in conjunction with the SPDNet architecture [9]. The architecture defines three basic layers that exploit the geometry of SPD matrices. The first layer, denoted BiMap, is parametrized with a matrix $\mathbf{W}_\theta$ on the Stiefel manifold $\{\mathbf{W}_\theta \in \mathbb{R}^{d \times f} : \mathbf{W}_\theta^T \mathbf{W}_\theta = \mathbf{I}\}$ and performs a bilinear projection $\mathbf{Y} = \mathbf{W}_\theta^T \mathbf{X} \mathbf{W}_\theta$. The second layer, denoted ReEig, rectifies all eigenvalues of a matrix, lower than a threshold $\mathbf{Y} = \mathbf{U} \max(\mathbf{\Sigma}, \epsilon\mathbf{I}) \mathbf{U}^T$ with $[\mathbf{\Sigma}, \mathbf{U}] = \mathrm{eig}(\mathbf{X})$. The purpose of the third layer, denoted LogEig, is to project matrices on $\mathcal{S}_d^+$ to the tangent space at the identity matrix, a Euclidean vector space. We computed the layer output as $\mathbf{y} = \mathrm{upper}(\mathrm{Log}_{\mathbf{I}}(\mathbf{X})) \in \mathbb{R}^{d(d+1)/2}$. After this layer, any standard neural network layer can be used. As in previous works [4, 9, 12], we used a linear layer with softmax activation. We also used the standard back-propagation framework with extensions for structured matrices [24] and Riemannian manifolds [25] to learn all network parameters in an end-to-end fashion.

BN method	space	operation	stats	df
none [9]	$\mathcal{S}_d^+$	$\mathbf{Y} = \mathbf{X}$	-	-
RBN [12]		$\mathbf{Y} = \Gamma_{\mathbf{I}\to\mathbf{G}_\theta}(\Gamma_{\mathbf{G}_t\to\mathbf{I}}(\mathbf{X}))$	batch	k
SPDBN(m)		algorithm 1, $\nu_\theta = \nu_t = 1$	running	k
SPDBN(m+v)		algorithm 1		k+1
TSBN(m)	$\mathbb{R}^k$	$\mathbf{y} = \mathbf{x} - \mathbf{g}_t + \mathbf{g}_\theta$		k
TSBN(m+v)		$\mathbf{y} = \frac{\sigma_\theta}{\sigma_t}(\mathbf{x} - \mathbf{g}_t) + \mathbf{g}_\theta$	running	k+1
TSBN(renorm)		$\mathbf{y} = \mathbf{\Sigma}_\theta \mathbf{\Sigma}_t^{-1}(\mathbf{x} - \mathbf{g}_t) + \mathbf{g}_\theta$		$2k$

Table 1. Considered SPD batch normalization methods (first 4 rows) and tangnet space (TS) approximations (last 3 rows). The listed degrees of freedom (df) use $k = d(d+1)/2$. As in [15], TSBN(renorm) assumes diagonal covariance matrices $\mathbf{\Sigma}_\theta, \mathbf{\Sigma}_t$.

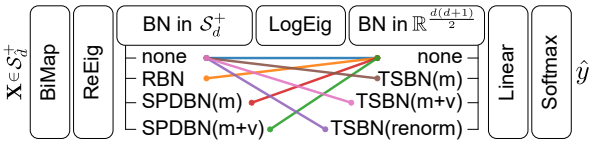

Fig. 1. SPDNet architecture with batch normalization (BN), mapping inputs $\mathbf{X} \in \mathcal{S}_d^+$ to class labels $\hat{y}$. The considered configurations are connected via colored lines.

3. EXPERIMENTS

In our experiments, we evaluated the performance gain of the proposed SPD BN method against several baseline methods on different tasks: action recognition from motion capture data, image classification, and mental imagery classification from EEG data. The considered methods are listed in Table 1 and include the original SPDNet without BN [9], Riemannian batch normalization (RBN) [12], the proposed method SPDBN(m+v), and a variant of the proposed method that only controls the mean (SPDBN(m)). We additionally included three methods that operate in Euclidean space after the LogEig layer (Figure1). As such, they operate in the tangent space (TS) of the identity matrix, approximating the geometric distances on $\mathcal{S}_d^+$. The TS approximations either controlled only the mean (TSBN(m)), mean and global variance (TSBN(m+v)) or each feature individually (TSBN(renorm)) as in [15]. The considered SPDNet architectures are summarized in Figure 1. All networks were trained to minimize the weighted cross-entropy error for 50 epochs using the Riemannian Adam optimizer [26], batch sizes of 30 or 50 and a learning rate of $1e^{-2}$ or $5e^{-3}$. As in [9, 12], the ReEig layer threshold ϵ was set to $1e^{-4}$ and the BN layer momentum η to 0.1. The batch normalization parameters $(\mathbf{G}_{t-1}, \nu_{t-1})$ were initialized to $\mathbf{I}$ and 1.

3.1. Simulations

As a first experiment, we simulated a 2D, imbalanced (10/1 ratio), binary classification problem for different SNR levels. The data (2×2 SPD matrices) were generated as a linear mixture of 1 encoding and 1 noise source; the joint and marginal distributions are displayed in Figure 2a (SNR = 0dB). Since the data of one class followed a bimodal distribution, we replaced the final linear layer in the model architecture with two parallel linear layers and max pooling of their activations. The results for 2000 observations (50% train/test split) are displayed in Figure 2b,c. As expected, all methods that control the variance (SPDBN(m+v), TSBN(m+v), TSBN(renorm)) clearly outperformed the other methods across the considered SNR levels (Figure 2c). Among the variance controlling methods, SPDBN(m+v) was affected least by declines in the SNR; the decline in balanced accuracy from 0dB to -20dB was 2.4% compared to 4.5% for TSBN(m+v) and TSBN(renorm). As

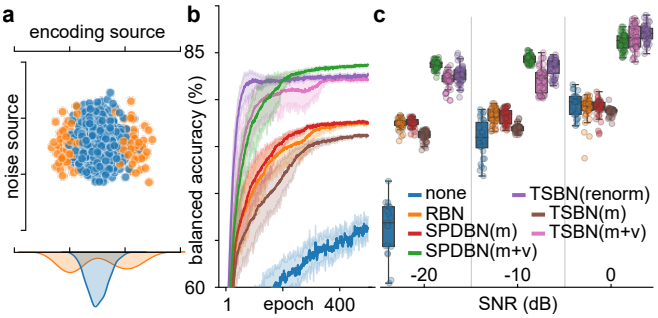

Fig. 2. Simulation results. **a,** Joint distribution of the noise and encoding sources (top), and marginal distribution of the encoding source (bottom) for SNR = 0 dB. **b,** Test set performance over training epochs (SNR = -20 dB). The methods are color-coded. Solid lines summarize the average balanced accuracy of up to 50 repetitions that converged. Shaded areas show the bootstrapped 95% confidence interval. **c,** Test set performance at epoch 500 for SNR = {-20, -10, 0}dB.

BN method	accuracy (%)	df	time per epoch (s)
none [9]	60.7±1.6	56,847	0.60
RBN [12]	57.7±1.2	57,312	0.78
SPDBN(m)	61.1±2.8	57,312	0.81
SPDBN(m+v)	**65.9±1.1**	57,313	0.92
TSBN(renorm)	59.9±0.8	57,777	0.61

Table 2. HDM05 dataset results, comprising test set accuracy (mean±std, 10 repetitions), degrees of freedom (df) and the time per training epoch for a computer with a 32-core CPU and 64 GB RAM.

batch normalization typically improves the invariance to random weight initializations [13, 14], we also computed convergence ratios, defined as the fraction of repetitions (out of 50) for which the mean training accuracy was larger than 65% after 100 epochs. For SPDNet without BN, the convergence ratio declined from an already low level of 54% (0dB) to 38% (-20dB), while it remained invariant to the SNR level for the methods that controlled the mean (87%) or mean and variance (98%). A 11% higher convergence ratio underlines the importance of controlling both mean and variance.

3.2. Action recognition

We used the HDM05 motion capture dataset [27] and the same experimental setup as [12] to classify actions from human movement trajectories. The preprocessed data, provided in [12], contains 2086 observations and defines 130 action classes. Each observation is described by a label and a 93×93 dimensional covariance matrix, summarizing the second order statistics of 3D position trajectories for 31 joints. As in [12], 50% of the observations were used to train the model parameters, and the BiMap layer was configured to project the data to a 30×30 dimensional subspace. We additionally applied weight decay of $1e^{-5}$ to all trainable parameters. Table 2 lists the classification accuracy, degrees of freedom (df) and the time per epoch across models. We observed the highest classification accuracy for SPDBN(m+v). The gap to the second best method (SPDBN(m)) was 4.8%. The performance increase came at the cost of 0.11s longer computations per epoch.

3.3. Image classification

Here, we used the standard MNIST dataset [28] to detect 10 handwritten digits from 28×28 greyscale images. The original datset contains 60k train and 10k test images. For our experiments we

trained the networks on the first 0.5/1/2k training images. To transform the images to SPD matrices, we extracted region covariance matrices (RCMs) [29]. We considered 6 regions: the entire image (28×28), the four quadrants (14×14) and a segment around the center (14×14). As in [30], the regions were summarized with 8×8 dimensional covariance descriptors, which were stacked to form a 48×48 dimensional block-diagonal SPD matrix for each image. The extracted features included 2D positions, the pixel intensity, and first and second order derivatives. The BiMap layer reduced the dimensions from 48×48 to 24×24.

Apart from fixed SPD features, we also considered feature learning via combining CNNs, covariance pooling and SPDNet as explored in [4]. The CNN architecture corresponded to the first 3 layers of the classical LeNet5 [31], namely, convolution (5×5 kernels, 20 channels, ReLU activation), max-pooling (2×2), and convolution (5×5, 50, ReLU). We applied covariance pooling [4] to the feature activations of the final CNN layer ($\mathbf{a} \in \mathbb{R}^{50 \times h \cdot w}$) and two additional channels that encoded the horizontal and vertical positions, resulting in 52×52 dimensional covariance matrixes as inputs for SPDNet. The BiMap layer was configured to reduce the dimensions to 20×20.

Figure 3 summarizes the results. Compared to fixed features (Figure 3a), feature learning (Figure 3b) clearly improved the performance of all SPD/TS BN methods; the gap was on average (methods, training set size) 5.5%. For the fixed features (Figure 3a), SPDBN(m+v) attained the highest accuracy across dataset sizes. It was followed by TSBN(renorm) with an average gap of 0.4%.

For the feature learning approach (Figure 3b), the gap between SPDBN(m+v) and the other SPD/TS BN methods was small (0.1%). Here, we also compared the SPDNet-based methods (avg. df = 29k, avg. time/epoch = 0.32 s) to LeNet5 [31] (431k, 0.19 s), LeNet5 with batch normalization before the final linear layer (432k, 0.24 s) and a fast matrix power norm covariance pooling method (iSQRT-COV) [32] (29k, 0.30 s), achieving state-of-the-art performance in large-scale image classification tasks [1]. The additional methods exhibited a performance gap to the SPDNet-based methods that declined as the training data size increased, suggesting that the combination of SPDNet on top of CNN feature extractors performs favorably in the scarce data regime. Due to a high degree of freedom, the performance of both LeNet5 variants substantially dropped as the training data decreased. For 0.5k observations, the performance of LeNet5 even declined to the same level as SPDBN(m+v) with fixed SPD features. Since the operations of iSQRT-COV merely approximate operations on the SPD manifold in favor of faster computations (quadratic complexity), a higher performance of SPDNet-based methods can be expected at the cost of longer computations (cubic complexity).

3.4. Mental imagery classification

We used a long-term (15 months, 26 sessions) dataset containing EEG data of a tetraplegic person during a trial-based paradigm with 4 mental tasks (=classes) [33], and the same experimental setup as [6]. In a nutshell, we extracted EEG activity in 4 frequency bands and computed covariance matrices for 32 EEG channels during the task period (2-s segment, 1 per trial), yielding a total of 2876 labeled observations. The network architecture considered individual SPDNets per frequency band, whose BiMap layers projected the 32×32 dimensional data to 10×10 dimensional subspaces. After the LogEig layers, the features of all 4 frequency bands were concatenated and submitted to the final linear classification layer. As in [33, 6] we evaluated the methods with regard to their performance in a leave-one-session-out cross-validation (CV) scheme.

The results are summarized in Table 3 and distinguish between a 2-class problem with the 2 most discriminative classes and the full 4-class problem. In either case, we observed a large variance (∼5%) of the classification accuracy across CV runs. Nonetheless, we could identify small to moderate effects. In the 2-class problem, we observed a significantly higher accuracy of SPDBN(m+v) compared to no batch normalization. The difference between SPDBN(m+v) and RBN remained a trend. We observed similar, but slightly stronger effects in terms of t-values for the 4-class problem.

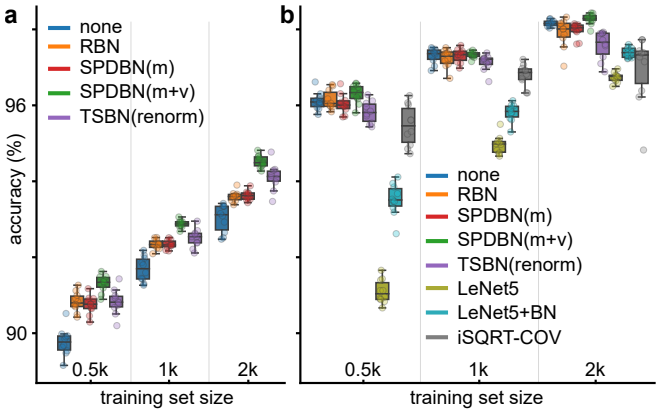

Fig. 3. MNIST dataset results. Test set accuracy (mean±std, 10 repetitions) as a function of training set size for fixed SPD features (**a**) and CNN-based feature learning (**b**).

BN method	2 classes			4 classes		
	accuracy	t	p	accuracy	t	p
none [9]	91.2±5.6	2.6	**0.050**	53.4±6.3	3.1	**0.022**
RBN [12]	91.4±5.5	2.3	*0.088*	54.2±5.8	2.7	*0.054*
SPDBN(m)	92.0±5.1	1.5	0.391	**54.9±6.2**	-0.1	1.000
SPDBN(m+v)	**92.5±4.6**	-	-	**54.9±6.3**	-	-
TSBN(renorm)	91.4±5.1	1.8	0.247	54.1±6.0	1.5	0.404
FBCSP+LR [6]	91.0±4.3	-	-	51.6±6.8	-	-
FBTSM+LR [6]	92.7±5.2	-	-	52.2±6.1	-	-
FBCSP+LDA [33]	-	-	-	46.4±3.8	-	-

Table 3. Multi-session EEG mental imagery dataset results. Test set accuracies (mean±std) for a leave-one-session-out CV scheme (26 sessions). Two-sided, paired, permutation (1e4 permutations) t-tests were computed to identify significant differences between SPDBN(m+v) and the other SPD/TS BN methods [34]. P values were corrected for multiple comparisons with the t-max method. The remaining rows list previously reported results.

4. CONCLUSION

We proposed a batch normalization algorithm for SPD data that makes use of the manifold structure of SPD matrices and can be readily combined with SPD neural networks [4, 9, 11, 10]. Unlike an earlier method that merely controls the Fréchet mean [12], the proposed method additionally controls the Fréchet variance. In simulations and experiments with diverse datasets, we observed a systematic performance increase, in some cases considerable, compared to previous works [4, 9, 12, 32] and tangent space batch normalization approximations. We demonstrated that the proposed method makes the SPDNet architecture more robust to low signal-to-noise ratios and a lack of data, even when combined with CNNs as feature extractors. Altogether, the performance of our proposed SPDBN(m+v) in combination with SPDNet makes it a suitable candidate in learning scenarios with poor SNR and scarce data where the cubic computation cost of eigen-decomposition is manageable.

5. REFERENCES

[1] Qilong Wang, Jiangtao Xie, Wangmeng Zuo, Lei Zhang, and Peihua Li, "Deep cnns meet global covariance pooling: Better representation and generalization," *IEEE TPAMI*, vol. 43, no. 8, pp. 2582–2597, 2021.

[2] Kisung You and Hae-Jeong Park, "Re-visiting Riemannian geometry of symmetric positive definite matrices for the analysis of functional connectivity," *NeuroImage*, vol. 225, 2021.

[3] Mehrtash Harandi, Mathieu Salzmann, and Richard Hartley, "Dimensionality Reduction on SPD Manifolds: The Emergence of Geometry-Aware Methods," *IEEE TPAMI*, vol. 40, no. 1, pp. 48–62, 2018.

[4] Dinesh Acharya, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool, "Covariance Pooling for Facial Expression Recognition," in *IEEE CVPR Workshops*, 2018.

[5] Marco Congedo, Alexandre Barachant, and Rajendra Bhatia, "Riemannian geometry for EEG-based brain-computer interfaces; a primer and a review," *Brain-Computer Interfaces*, vol. 4, no. 3, pp. 155–174, 2017.

[6] Reinmar Kobler, Jun-Ichiro, Hirayama, Lea, Hehenberger, et al., "On the interpretation of linear Riemannian tangent space model parameters in M/EEG," in *IEEE EMBC*, 2021.

[7] Rajendra Bhatia, *Positive definite matrices*, Princeton university press, 2015.

[8] Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa, "Differentiating through the Fréchet Mean," in *ICML*, 2020, pp. 6393–6403.

[9] Zhiwu Huang and Luc Van Gool, "A Riemannian Network for SPD Matrix Learning," in *AAAI*, 2017, pp. 2036–2042.

[10] Rudrasis Chakraborty, Chun-Hao Yang, Xingjian Zhen, Monami Banerjee, Derek Archer, David Vaillancourt, Vikas Singh, and Baba C. Vemuri, "A Statistical Recurrent Model on the Manifold of Symmetric Positive Definite Matrices," in *NeurIPS*, 2018, pp. 8897–8908.

[11] Rudrasis Chakraborty, Jose Bouza, Jonathan Manton, and Baba C Vemuri, "ManifoldNet: A Deep Neural Network for Manifold-valued Data with Applications," *IEEE TPAMI*, pp. 1–1, 2020.

[12] Daniel Brooks, Olivier Schwander, Frederic Barbaresco, Jean-Yves Schneider, and Matthieu Cord, "Riemannian batch normalization for SPD neural networks," in *NeurIPS*, 2019.

[13] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456.

[14] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Mądry, "How Does Batch Normalization Help Optimization?," in *NeurIPS*, 2018, pp. 2488–2498.

[15] Sergey Ioffe, "Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models," in *NeurIPS*, 2017, pp. 1942–1950.

[16] Xavier Pennec, Pierre Fillard, and Nicholas Ayache, "A Riemannian framework for tensor computing," *International Journal of Computer Vision*, vol. 66, no. 1, pp. 41–66, 2006.

[17] Hermann Karcher, "Riemannian center of mass and mollifier smoothing," *Communications on pure and applied mathematics*, vol. 30, no. 5, pp. 509–541, 1977.

[18] Salem Said, Lionel Bombrun, Yannick Berthoumieu, and Jonathan H. Manton, "Riemannian Gaussian Distributions on the Space of Symmetric Positive Definite Matrices," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2153–2170, 2017.

[19] Frédéric Barbaresco, "Jean-Louis Koszul and the elementary structures of Information Geometry," in *Geometric Structures of Information*, pp. 333–392. Springer, 2019.

[20] Shun-ichi Amari, *Information geometry and its applications*, Springer Japan, 2016.

[21] Or Yair, Mirela Ben-Chen, and Ronen Talmon, "Parallel Transport on the Cone Manifold of SPD Matrices for Domain Adaptation," *IEEE Trans. on Signal Processing*, vol. 67, no. 7, pp. 1797–1811, 2019.

[22] Pedro Rodrigues, Christian Jutten, and Marco Congedo, "Riemannian Procrustes Analysis: Transfer Learning for Brain–Computer Interfaces," *IEEE Trans. on Biomedical Engineering*, vol. 66, no. 8, pp. 2390–2401, 2019.

[23] Jeffrey Ho, Guang Cheng, Hesamoddin Salehian, and Baba Vemuri, "Recursive Karcher Expectation Estimators And Geometric Law of Large Numbers," in *AISTATS*, 2013, vol. 31, pp. 325–332.

[24] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu, "Matrix Backpropagation for Deep Networks with Structured Layers," in *IEEE ICCV*, 2015, pp. 2965–2973.

[25] P-A Absil, Robert Mahony, and Rodolphe Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, 2009.

[26] Gary Bécigneul and Octavian-Eugen Ganea, "Riemannian Adaptive Optimization Methods," in *ICLR*, 2018.

[27] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber, "Documentation Mocap Database HDM05," Tech. Rep., Universität Bonn, 2007.

[28] Yann LeCun, Corinna Cortes, and Christopher Burges, "MNIST handwritten digit database," 2010.

[29] Oncel Tuzel, Fatih Porikli, and Peter Meer, "Region Covariance: A Fast Descriptor for Detection and Classification," in *ECCV*, 2006, pp. 589–600.

[30] Hiroyuki Kasai and Bamdev Mishra, "Riemannian joint dimensionality reduction and dictionary learning on symmetric positive definite manifolds," in *IEEE EUSIPCO*, 2018, pp. 2010–2014.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[32] Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao, "Towards faster training of global covariance pooling networks by iterative matrix square root normalization," in *IEEE CVPR*, 2018, pp. 947–955.

[33] Lea Hehenberger, Reinmar Kobler, Catarina Lopes-Dias, et al., "Long-term mutual training for the CYBATHLON BCI Race with a tetraplegic pilot: a case study on inter-session transfer and intra-session adaptation," *Front. in Humn. Neurosc.*, 2021.

[34] T. Nichols and A. Holmes, "Nonparametric permutation tests for functional neuroimaging: A primer with examples," *Human Brain Mapping*, vol. 15, no. 1, pp. 1–25, 2002.