SPD domain-specific batch normalization to crack interpretable unsupervised domain adaptation in EEG

Anonymous Author(s)
Affiliation
Address
email

Abstract

Electroencephalography (EEG) provides access to neuronal dynamics non-invasively with millisecond resolution, rendering it a viable method in neuroscience and healthcare. However, its utility is limited as current EEG technology does not generalize well across domains (i.e., sessions and subjects) without expensive supervised re-calibration. Contemporary methods cast this transfer learning (TL) problem as a multi-source/-target unsupervised domain adaptation (UDA) problem and address it with deep learning or shallow, Riemannian geometry aware alignment methods. Both directions have, so far, failed to consistently close the performance gap to state-of-the-art domain-specific methods based on tangent space mapping (TSM) on the symmetric positive definite (SPD) manifold. Here, we propose a theory-based machine learning framework that enables, for the first time, learning domain-invariant TSM models in an end-to-end fashion. To achieve this, we propose a new building block for geometric deep learning, which we denote SPD domain-specific momentum batch normalization (SPDDSMBN). A SPDDSMBN layer can transform domain-specific SPD inputs into domain-invariant SPD outputs, and can be readily applied to multi-source/-target and online UDA scenarios. In extensive experiments with 6 diverse EEG brain-computer interface (BCI) datasets, we obtain state-of-the-art performance in inter-session and -subject TL with a simple, intrinsically interpretable network architecture, which we denote TSMNet.

1 Introduction

Electroencephalography (EEG) measures multi-channel electric brain activity from the human scalp with millisecond precision [1]. Transient modulations in the rhythmic brain activity can reveal cognitive processes [2], affective states [3] and a persons health status [4]. Unfortunately, these modulations exhibit low signal-to-noise ratio (SNR), domain shifts (i.e., changes in the data distribution) and have low specificity, rendering statistical learning a challenging task - particularly in the context of brain-computer interfaces (BCI) [5] where the goal is to predict a target from a short segment of multi-channel EEG data in real-time.

Under domain shifts, domain adaptation (DA), defined as learning a model from a source domain that performs well on a related target domain, offers principled statistical learning approaches with theoretical guarantees [6, 7]. DA in the BCI field mainly distinguishes inter-session and -subject transfer learning (TL) [8]. In inter-session TL, domain shifts, are expected across sessions mainly due to mental drifts (low specificity) as well as differences in the relative positioning of the electrodes and their impedances. Inter-subject TL is more difficult, as domain shifts are additionally driven by structural and functional differences in brain networks as well as variations in the performed task [9].

These domain shifts are traditionally circumvented by recording labeled calibration data and fitting domain-specific models [10, 11]. As recording calibration data is costly, models that are robust to scarce data with low SNR perform well in practice. Currently, tangent space mapping (TSM) models [12, 13] operating with symmetric, positive definite (SPD) covariance descriptors of preprocessed

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

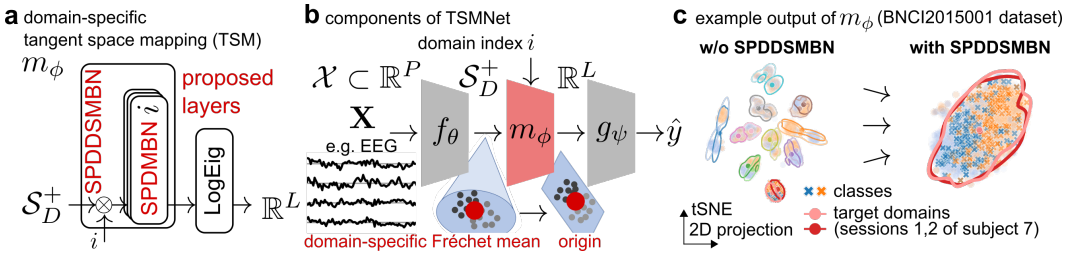

Figure 1: Visualization of the proposed framework around SPD domain-specific momentum batch normalization (SPDDSMBN) (**a**) that learns parameters $\Theta = \{\theta, \phi, \psi\}$ of typical tangent space mapping (TSM) models (**b**) *end-to-end* to crack multi-source/-target unsupervised domain adaptation (MSMTUDA) on $\mathcal{S}_D^+$ for EEG data (**c**, illustrative example). For EEG data, we propose a simple, intrinsically interpretable parametrization of f and g, denoted TSMNet, and obtain SOA performance.

data are considered state-of-the-art (SOA) [10, 14, 15]. They are well suited for EEG data as they exhibit invariances to linear mixing of latent sources [16], and are consistent [13] and intrinsically interpretable [17] estimators for generative models that encode label information with a log-linear relationship in source power modulations.

Competitive, supervised calibration-free methods are one of the long-lasting grand challenges in EEG neurotechnology research [5, 10, 18, 19, 15]. Among the applied transfer learning techniques, including multi-task learning [20] and domain-invariant learning [21–23], unsupervised domain adaptation (UDA) [24] is considered as key to overcome this challenge [10, 19]. Contemporary methods cast the problem as a multiple source and target UDA problem and address it with deep learning [25–28] or shallow, Riemannian geometry aware alignment methods [29–32]. Successful methods must cope with notoriously small and heterogeneous datasets (i.e., dozens of domains with a few dozens observations per domain and class). In a recent, relatively large scale inter-subject and -dataset TL competition with few labeled examples per target domain [19], deep learning approaches that aligned the first and second order statistics either in input [33, 27] or latent space [34] obtained the highest scores. Whereas, in a pure UDA competition [15] with a smaller dataset, Riemannian geometry aware approaches dominated. With the increasing popularity of geometric deep learning [35], Ju and Guan [36] proposed a architecture based on SPD neural networks [37] to align SPD features in latent space and attained similar scores as convolutional neural nets. Despite the tremendous advances in recent years, the field still lacks methods that can consistently close the performance gap to state-of-the-art domain-specific methods.

To close this gap, we propose a theory-based machine learning framework around domain-specific batch normalization on the SPD manifold (Figure 1). We first extend momentum batch normalization (MBN) [32] to SPDMBN that controls the Fréchet mean and variance of SPD data. In a theoretical analysis, we show under reasonable assumptions that SPDMBN can track and converge to the data's true Fréchet mean, enabling, for the first time, end-to-end learning of feature extractors, TSM and tangent space classifiers. Building upon this insight, we combine SPDMBN with domain-specific batch normalization (DSBN) [38] to form SPDDSMBN. A SPDDSMBN layer can transform domain-specific SPD inputs into domain-invariant SPD outputs. Like DSBN, SPDDSMBN easily extends to multi-source, multi-target and online UDA scenarios. Ultimately, the proposed methods are combined in a simple, intrinsically interpretable network architecture, denoted TSMNet. We obtain state-of-the-art performance in inter-session and -subject UDA on small and large scale EEG BCI datasets, and show in an ablation study that the performance increase is driven by performing DSBN on the SPD manifold.

2 Preliminaries

Multi-source multi-target unsupervised domain adaptation (MSMTUDA) Let $\mathcal{X}$ denote the space of input features, $\mathcal{Y}$ a label space, and $\mathcal{I}_d \subset \mathbb{N}$ an index set that contains unique domain identifiers. In the multi-source, multi-target unsupervised domain adaptation scenario considered here, we are given a set $\mathcal{T}^{source} = \{\mathcal{T}_i | i \in \mathcal{I}_d^{source} \subset \mathcal{I}_d\}$ with $|\mathcal{I}_d^{source}| = N$ domains. Each domain $\mathcal{T}_i = \{(\mathbf{X}_{ij}, y_{ij})\}_{j=1}^M \sim P_{XY}^i$ contains M observations of feature ($\mathbf{X} \in \mathcal{X}$) and label ($y \in \mathcal{Y}$) tuples sampled from a joint distribution P_{XY}^i.[1] While the joint distributions can be different

[1]For ease of notation, although not required by our method, we assume that M is the same for each domain.

(but related) across domains, we assume that the class priors are the same (i.e., $P_Y^i = P_Y$). The goal is to learn a predictive function $h : \mathcal{X} \times \mathcal{I}_d \to \mathcal{Y}$ that, once fitted to $\mathcal{T}^{source}$, can generalize to unseen target domains $\mathcal{T}^{target} = \{\mathcal{T}_l | l \in \mathcal{I}_d^{target} \subset \mathcal{I}_d, \mathcal{I}_d^{target} \cap \mathcal{I}_d^{source} = \emptyset\}$ merely based on *unsupervised* adaptation of h to each target domain $\mathcal{T}_l$ once its label l and features $\{\mathbf{X}_{il}\}_{j=1}^M \sim P_X^l$ are revealed.

Riemannian geometry on $\mathcal{S}_D^+$ We start with recalling notions of geometry on the space of real $D \times D$ symmetric positive definite (SPD) matrices $\mathcal{S}_D^+ = \{\mathbf{Z} \in \mathbb{R}^{D \times D} : \mathbf{Z}^T = \mathbf{Z}, \mathbf{Z} \succ 0\}$. The space $\mathcal{S}_D^+$ forms a cone shaped Riemannian manifold in $\mathbb{R}^{D \times D}$ [39]. A Riemannian manifold $\mathcal{M}$ is a smooth manifold equipped with an inner product on the tangent space $\mathcal{T}_{\mathbf{Z}}\mathcal{M}$ at each point $\mathbf{Z} \in \mathcal{M}$. Tangent spaces have Euclidean structure with easy to compute distances $\mathcal{T}_{\mathbf{Z}}\mathcal{M} \times \mathcal{T}_{\mathbf{Z}}\mathcal{M} \to \mathbb{R}^+$ which locally approximate Riemannian distances on $\mathcal{M}$ induced by an inner product [40]. Logarithmic $\mathrm{Log}_{\mathbf{Z}} : \mathcal{M} \to \mathcal{T}_{\mathbf{Z}}\mathcal{M}$ and exponential $\mathrm{Exp}_{\mathbf{Z}} : \mathcal{T}_{\mathbf{Z}}\mathcal{M} \to \mathcal{M}$ mappings project points to and from tangent spaces.

Equipping $\mathcal{S}_D^+$ with the inner product $\langle \mathbf{S}_1, \mathbf{S}_2 \rangle_{\mathbf{Z}} = ||\mathbf{Z}^{-1}\mathbf{S}_1\mathbf{Z}^{-1}\mathbf{S}_2||_F$ for points $\mathbf{S}_1, \mathbf{S}_2$ in the tangent space $\mathcal{T}_{\mathbf{Z}}\mathcal{S}_D^+$ (i.e., the space of real symmetric $D \times D$ matrices) results in a globally defined affine invariant Riemannian metric (AIRM) on $\mathcal{S}_D^+$ [41, 39], which can be computed in closed form:

$$\delta_{AIRM}(\mathbf{Z}_1, \mathbf{Z}_2) = ||\log(\mathbf{Z}_1^{-\frac{1}{2}}\mathbf{Z}_2\mathbf{Z}_1^{-\frac{1}{2}})||_F \tag{1}$$

where $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are two SPD matrices, $\log(\cdot)$ denotes the matrix logarithm[2], $||\cdot||_F$ the Frobenius norm. Due to affine invariance, we have $\delta_{AIRM}(\mathbf{A}\mathbf{Z}_1\mathbf{A}^T, \mathbf{A}\mathbf{Z}_2\mathbf{A}^T) = \delta_{AIRM}(\mathbf{Z}_1, \mathbf{Z}_2)$ for any invertible affine transformation matrix $\mathbf{A}$. The exponential and logarithmic mapping are also globally defined in closed form as

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

$$\mathrm{Exp}_{\mathbf{Z}}(\mathbf{S}) = \mathbf{Z}^{\frac{1}{2}}\exp(\mathbf{Z}^{-\frac{1}{2}}\mathbf{S}\mathbf{Z}^{-\frac{1}{2}})\mathbf{Z}^{\frac{1}{2}} \tag{3}$$

For a set of SPD points $\mathcal{Z} = \{\mathbf{Z}_j \in \mathcal{S}_d^+\}_{j \leq M}$, we will use the notion of Fréchet mean $\mathbf{G} \in \mathcal{S}_D^+$ and Fréchet variance $\nu^2 \in \mathbb{R}^+$. The Fréchet mean $\mathbf{G}$ is defined as the minimizer of the average squared distances

$$\mathbf{G} = \mathrm{FM}(\mathcal{Z}) = \arg\min_{\mathbf{P} \in \mathcal{S}_D^+} \frac{1}{M} \sum_{j=1}^{M} \delta_{AIRM}^2(\mathbf{P}, \mathbf{Z}_j) \tag{4}$$

For $M = 2$, there is a closed form solution expressed as

$$\mathrm{FM}(\{\mathbf{Z}_1, \mathbf{Z}_2\}; \gamma) = \mathbf{Z}_1 \#_\gamma \mathbf{Z}_2 = \mathbf{Z}_1^{\frac{1}{2}} \left(\mathbf{Z}_1^{-\frac{1}{2}}\mathbf{Z}_2\mathbf{Z}_1^{-\frac{1}{2}}\right)^\gamma \mathbf{Z}_1^{\frac{1}{2}} \tag{5}$$

with weight $\gamma = 0.5$. Choosing $\gamma \in [0, 1]$ computes weighted means along the geodesic (i.e., the shortest curve) that connects both points. For $M > 2$, (4) can be solved using the Karcher flow algorithm [42], which iterates between projecting the data to the tangent space (2) at the current estimate, arithmetic averaging, and projecting the result back (3) to obtain a new estimate. The Fréchet variance ν^2 is defined as the attained value at the minimizer $\mathbf{G}$

$$\nu^2 = \mathrm{FV}(\mathcal{Z}) = \mathrm{Var}_{\mathbf{G}}(\mathcal{Z}) = \frac{1}{M} \sum_{j=1}^{M} \delta_{AIRM}^2(\mathbf{G}, \mathbf{Z}_j) \tag{6}$$

3 Domain-specific batch normalization on $\mathcal{S}_D^+$

In this section, we briefly review batch normalization (BN) [43] variants with a focus on $\mathcal{S}_D^+$ as well as domain-specific batch normalization (DSBN) [38]. We then present SPDDSMBN and show in a theoretical analysis that the running estimates converge to the true domain-specific Fréchet means under reasonable assumptions.

[2]For SPD matrices, powers, logarithms and exponentials can be computed via eigen decomposition.

Batch normalization Batch normalization (BN) [43] is a widely used training technqiue in deep learning as BN layers speed up convergence and improve generalization via smoothing of the engery landscape [44, 32]. A standard BN layer applies slightly different transformations during training and testing to independent and identically distributed (iid) observations $\mathbf{x}_j \in \mathbb{R}^d$ within a minibatch $\mathcal{B}$ of size M drawn from a dataset $\mathcal{T}$. During training, the data are whitened using the batch mean $\mathbf{g}_\mathcal{B} \in \mathbb{R}^d$ and variance $\mathbf{s}^2_\mathcal{B} \in \mathbb{R}^d_+$, and then scaled and shifted to have a parametrized mean $\mathbf{g}_\phi$ and variance $\mathbf{s}^2_\phi$. The BN layer input output relation is expressed as

$$\mathrm{BN}(\mathbf{x}_j; \mathbf{g}_\phi, \mathbf{s}_\phi, \varepsilon, \gamma) = \mathbf{s}_\phi \oslash (\mathbf{s}_\mathcal{B} + \varepsilon\, \mathbf{1}_d) \odot (\mathbf{x} - \mathbf{g}_\mathcal{B}) + \mathbf{g}_\phi \ , \quad \forall \mathbf{x}_j \in \mathcal{B} \tag{7}$$

where the operators $\odot$ and $\oslash$ perform element-wise multiplication and division, $\varepsilon \in \mathbb{R}_+$ is a small stabilizing constant, and γ a momentum parameter that is used internally to update running estimates of the dataset's statistics $(\mathbf{g}_\mathcal{T}, \mathbf{s}^2_\mathcal{T})$; they are used in (7) during testing.

Using batch statistics to whiten observations during training rather than running estimates introduces noise whose level depends on the batch size [32]; smaller batch sizes raise the noise level. The introduced noise regularizes the training process, which can help to escape poor local minima in initial learning but also lead to underfitting. Momentum BN (MBN) [32] allows small batch sizes while avoiding underfitting. Like batch renormalization (BreN) [45], MBN uses running estimates in (7) during training and testing. The key difference is that MBN keeps two sets of running statistics; one for training and one for testing. The latter are updated conventionally, while the former are updated with momentum parameter $\gamma_{train}(k)$ that decays over training steps k. MBN can, therefore, quickly escape poor local minima during initial learning and avoid underfitting at later stages [32].

Batch normalization on $\mathcal{S}^+_D$ It is intractable to compute the Fréchet mean $\mathbf{G}_\mathcal{B}$ and variance $\nu^2_\mathcal{B}$ for each minibatch $\mathcal{B} = \{\mathbf{Z}_j \in \mathcal{S}^+_D\}^M_{j=1}$, as there is no efficient algorithm to solve (4). Brooks et al. [46] proposed Riemannian Batch Normalization (RBN) as a tractable approximation. RBN approximateley solves (4) by aborting the iterative Karcher flow algorithm after one iteration. To transform $\mathbf{Z}_j \in \mathcal{B}$ with estimated mean $\hat{\mathbf{G}}_\mathcal{B}$ to vary around $\mathbf{G}_\phi$, parallel transport can be used [47, 31, 46].[3] The RBN input output transformation is then expressed as

$$\mathrm{RBN}(\mathbf{Z}_j; \mathbf{G}_\phi, \gamma) = \Gamma_{\hat{\mathbf{G}}_\mathcal{B} \to \mathbf{G}_\phi}(\mathbf{Z}_j) = \mathbf{E}^T \mathbf{Z}_j \mathbf{E} \ , \ \ \mathbf{E} = (\hat{\mathbf{G}}^{-1}_\mathcal{B} \mathbf{G}_\phi)^{\frac{1}{2}} \ , \quad \forall \mathbf{Z}_j \in \mathcal{B} \tag{8}$$

Using (5), the running estimate of the dataset's Fréchet mean can be computed in closed form

$$\mathbf{G}_\mathcal{T}(k) = \mathbf{G}_\mathcal{T}(k-1) \#_\gamma \hat{\mathbf{G}}_\mathcal{B} \tag{9}$$

Kobler et al. [48] proposed an extension to RBN, denoted SPD batch renormalization (SPDBN) that controls both Fréchet mean and variance. Like BreN, SPDBN uses running estimates $\mathbf{G}_\mathcal{T}(k)$ and $\nu_\mathcal{T}(k)$ during training and testing. To transform $\mathbf{Z}_j \in \mathcal{B}$ to vary around $\mathbf{G}_\phi$ with variance ν^2_ϕ, each observation is first transported to vary around the identity matrix $\mathbf{I}$, rescaled via computing matrix powers and finally transported to vary around $\mathbf{G}_\phi$. The sequence of operations can be expressed as

$$\mathrm{SPDBN}(\mathbf{Z}_j; \mathbf{G}_\phi, \nu^2_\phi, \varepsilon, \gamma) = \Gamma_{\mathbf{I} \to \mathbf{G}_\phi} \circ \Gamma_{\mathbf{G}_\mathcal{T}(k) \to \mathbf{I}}(\mathbf{Z}_j)^{\frac{\nu_\phi}{\nu_\mathcal{T}(k) + \varepsilon}} \ , \quad \forall \mathbf{Z}_j \in \mathcal{B} \tag{10}$$

The standard backpropagation framework with extensions for structured matrices [49] and manifold-constrained gradients [40] can be used to propagate gradients through RBN and SPDBN layers and learn the parameters $(\mathbf{G}_\phi, \nu_\phi)$.

Domain-specific batch normalization on $\mathcal{S}^+_D$ Considering a MSUDA scenario, Chang et al. [38] proposed a domain-specific BN (DSBN) layer which simply keeps multiple parallel BN layers and distributes observations according to the associated domains. Formally, we consider minibatches $\mathcal{B}$ that form the union of $N_\mathcal{B} \leq |\mathcal{I}_d|$ domain-specific minibatches $\mathcal{B}_i$ drawn from distinct domains $i \in \mathcal{I}_\mathcal{B} \subseteq \mathcal{I}_d$. As before, each $\mathcal{B}_i$ contains $j = 1, ..., M/N_\mathcal{B}$ iid observations $\mathbf{x}_j$. A DSBN layer mapping $\mathbb{R}^d \times \mathcal{I}_d \to \mathbb{R}^d$ can then be expressed as

$$\mathrm{DSBN}(\mathbf{x}_j, i) = \mathrm{BN}_i(\mathbf{x}_j; \mathbf{g}_{\phi_i}, \mathbf{s}_{\phi_i}, \varepsilon, \gamma) \ , \quad \forall \mathbf{x}_j \in \mathcal{B}_i \ , \quad \forall i \in \mathcal{I}_\mathcal{B} \tag{11}$$

In practice, the batch size M is typically fixed. The particular choice is influenced by resource availability and the desired noise level introduced by minibatch based stochastic gradient descent. A drawback of DSBN is that for a fixed batch size M and an increasing number of source domains $N_\mathcal{B}$,

[3] On $\mathcal{S}^+_D$ parallel transport can be used to transport points on the tangent space and manifold.

Algorithm 1: SPD momentum batch normalization (SPDMBN)

Input : batch $\mathcal{B} = \{\mathbf{Z}_j \in \mathcal{S}_D^+\}_{j=1}^M$ at training step k,
running mean $\bar{\mathbf{G}}_\mathcal{T}(k-1)$, $\bar{\mathbf{G}}_\mathcal{T}(0) = \mathbf{I}$ and variance $\bar{\nu}_\mathcal{T}^2(k-1)$, $\bar{\nu}_\mathcal{T}(0) = 1$ for training,
running mean $\tilde{\mathbf{G}}_\mathcal{T}(k-1)$, $\tilde{\mathbf{G}}_\mathcal{T}(0) = \mathbf{I}$ and variance $\tilde{\nu}_\mathcal{T}^2(k-1)$, $\tilde{\nu}_\mathcal{T}(0) = 1$ for testing,
learnable parameters $(\mathbf{G}_\phi, \nu_\phi)$, and momentum for training and testing $\gamma_{train}(k), \gamma \in [0, 1]$

Output : normalized batch $\{\tilde{\mathbf{Z}}_j = \mathrm{SPDMBN}(\mathbf{Z}_j) \in \mathcal{S}_D^+ \mid \mathbf{Z}_j \in \mathcal{B}\}$

if *training* **then**

$\hat{\mathbf{G}}_\mathcal{B} =$ `karcher_flow` $(\mathcal{B}$, `steps = 1`$)$; `// approx. solve problem (4)`

$\bar{\mathbf{G}}_\mathcal{T}(k) = \bar{\mathbf{G}}_\mathcal{T}(k-1) \#_{\gamma_{train}(k)} \hat{\mathbf{G}}_\mathcal{B}$; `// update running stats for training`

$\bar{\nu}_\mathcal{T}^2(k) = (1 - \gamma_{train}(k))\bar{\nu}_\mathcal{T}^2(k-1) + \gamma_{train}(k)\mathrm{Var}_{\bar{\mathbf{G}}_\mathcal{T}(k)}(\mathcal{B})$

$\tilde{\mathbf{G}}_\mathcal{T}(k) = \tilde{\mathbf{G}}_\mathcal{T}(k-1) \#_\gamma \hat{\mathbf{G}}_\mathcal{B}$; `// update running stats for testing`

$\tilde{\nu}_\mathcal{T}^2(k) = (1 - \gamma)\tilde{\nu}_\mathcal{T}^2(k-1) + \gamma\mathrm{Var}_{\tilde{\mathbf{G}}_\mathcal{T}(k)}(\mathcal{B})$

end

$(\mathbf{G}_\mathcal{T}(k), \nu_\mathcal{T}^2(k)) = (\bar{\mathbf{G}}_\mathcal{T}(k), \bar{\nu}_\mathcal{T}(k))$ **if** *training* **else** $(\tilde{\mathbf{G}}_\mathcal{T}(k), \tilde{\nu}_\mathcal{T}(k))$

$\tilde{\mathbf{Z}}_j = \Gamma_{\mathbf{I} \to \mathbf{G}_\phi} \circ \Gamma_{\mathbf{G}_\mathcal{T}(k) \to \mathbf{I}}(\mathbf{Z}_j)^{\frac{\nu_\phi}{\nu_\mathcal{T}(k) + \varepsilon}}$; `// use (10) to whiten, rescale and rebias`

the effective batch size declines for the BN layers within DSBN. Since small batch sizes increase the noise level introduced by BN, increasing the number of domains per batch can lead to underfitting [32].

Domain-specific batch normalization could be extended to $\mathcal{S}_D^+$ by replacing BN in (11) with RBN or SPDBN. However, we believe that the performance can be improved by extending MBN to the SPD manifold. We list the pseudocode of our proposed extension, which we denote SPDMBN, in algorithm 1. As MBN [32], we decay $\gamma_{train}(k)$ with a clamped exponential decay schedule

$$\gamma_{train}(k) = 1 - \gamma_{min}^{\frac{1}{K-1}\max(K-k,0)} + \gamma_{min} \tag{12}$$

where K defines the training step at which $\gamma_{min} \in [0, 1]$ should be attained. The proposed layer is then formally defined as

$$\mathrm{SPDDSMBN}(\mathbf{Z}_j, i) = \mathrm{SPDMBN}_i(\mathbf{Z}_j; \mathbf{G}_{\phi_i}, \nu_{\phi_i}, \varepsilon, \gamma, \gamma_{train}(k)) \ , \ \forall \mathbf{Z}_j \in \mathcal{B}_i \subset \mathcal{S}_D^+ \ , \ \forall i \in \mathcal{I}_\mathcal{B} \tag{13}$$

SPDDSMBN can be readily adapted to new domains, as new SPDMBN layers can be added on the fly. If the entire data of a domain becomes available, the domain-specific Fréchet mean and variance can be estimated by solving (4), otherwise, the update rules in algorithm 1 can be used.

The running mean in SPDMBN converges to the Fréchet mean So far we have introduced SPDMBN which uses approximations of batch-specific Fréchet means to update running estimates of the dataset's Fréchet mean. In the following, we consider models that apply a SPDMBN layer to latent observations generated by a feature extractor $f_\theta : \mathcal{X} \to \mathcal{S}_D^+$ with learnable parameters θ.

We define a dataset containing the latent observations generated with feature set θ_k as $\mathcal{T}_{\theta_k} = \{f_{\theta_k}(\mathbf{x}) | \mathbf{x} \in \mathcal{T}\}$, and a minibatch of M iid samples drawn from $\mathcal{T}_{\theta_k}$ at training step k as $\mathcal{B}_k$. We denote the Fréchet mean of $\mathcal{T}_{\theta_k}$ as $\mathbf{G}_{\theta_k}$, the Fréchet variance as $\mathrm{Var}_{\theta_k}(\mathcal{T}_{\theta_k})$, and the estimated Fréchet mean, defined in (9), as $\mathbf{G}_k$.

Let us assume that the variance of the latent dataset $\mathcal{T}_{\theta_{k-1}}$ at the previous training step with respect to $\mathbf{G}_{\theta_k}$ at the current training step is bounded by the current Fréchet variance and the norm of the differences in the parameters

$$\mathrm{Var}_\theta(\mathcal{T}_{\theta_{k-1}}) \leq (1 + ||\theta_k - \theta_{k-1}||)\mathrm{Var}_\theta(\mathcal{T}_{\theta_k}) \tag{14}$$

This is, for instance, the case for functions f_θ that are smooth in the parameters (i.e., $\exists L > 0 : \delta_{AIRM}(f_\theta(\mathbf{x}), f_{\tilde{\theta}}(\mathbf{x})) \leq L||\theta - \tilde{\theta}|| \ \forall \mathbf{x} \in \mathcal{T} \ , \ \forall \theta, \tilde{\theta}$) and small step sizes.

Proposition 1 (Bound for SPDBN) *Consider the setting defined above, and assumption (14) holds true. Then, the variance* $\mathrm{Var}_\theta(\mathbf{G}_k)$ *of the running mean is non-increasing over training steps* k *if*

$$||\theta_k - \theta_{k-1}|| \leq \frac{1 - \gamma^2}{(1 - \gamma)^2} - 1 \tag{15}$$

holds true.

The proof is provided in appendix A.1 of the supplementary material and relies on the proof of the geometric law of large numbers [50]. For a fixed $\gamma \in (0, 1)$, Proposition 1 is fulfilled, if the parameter updates are chosen sufficiently small.

Proposition 2 (Convergence of SPDMBN) *If we additionally assume that the parameters θ_k converge to a stationary point θ^* (i.e., $||\theta_k - \theta^*|| \leq 1/k^\beta$), the momentum parameter is decayed with schedule $\gamma(k) = 1/k^\alpha$, and $\beta > \alpha > 0$, then the running mean $\mathbf{G}_k$ converges to the Fréchet mean $\mathbf{G}_{\theta^*}$ in probability as $k \to \infty$.*

The proof reduces to the proof of the theorem of large numbers on $\mathcal{S}_D^+$, provided in [50], since in the limit $k \to \infty$ the variance introduced by the parameter updates vanishes faster than the variance introduced by the noisy estimates.

SPDMBN to learn tangent space mapping at Fréchet means Typical TSM models for classification [12] and regression [13] first use (2) to project $\mathbf{Z} \in \mathcal{T} \subset \mathcal{S}_D^+$ to the tangent space at the Fréchet mean $\mathbf{G}_\mathcal{T}$, then use (8) to transport the result to vary around $\mathbf{I}$, and finally extract elements in the upper triangular part[4] to reduce feature redundancy. The invertible mapping $\mathcal{P}_{\mathbf{G}_\mathcal{T}} : \mathcal{S}_D^+ \to \mathbb{R}^{D(D+1)/2}$ is expressed as:

$$\mathcal{P}_{\mathbf{G}_\mathcal{T}}(\mathbf{Z}) = \text{upper} \circ \Gamma_{\mathbf{G}_\mathcal{T} \to \mathbf{I}} \circ \text{Log}_{\mathbf{G}_\mathcal{T}}(\mathbf{Z}) = \text{upper}(\log(\mathbf{G}_\mathcal{T}^{-\frac{1}{2}} \mathbf{Z} \mathbf{G}_\mathcal{T}^{-\frac{1}{2}})) \tag{16}$$

We propose to use a SPDMBN layer followed by a LogEig layer [37] to compute a similar mapping m_ϕ (Figure 1a). A LogEig layer simply computes the matrix logarithm and vectorizes the result so that the norm is preserved. If the parametrized mean of SPDMBN is fixed to the identify matrix ($\mathbf{G}_\phi = \mathbf{I}$), the composition computes

$$\begin{aligned} m_\phi(\mathbf{Z}) &= \text{LogEig} \circ \text{SPDMBN}(\mathbf{Z}) = \text{upper} \circ \log \circ \Gamma_{\mathbf{G}_\mathcal{T}(k) \to \mathbf{I}}(\mathbf{Z})^{\frac{\nu_\phi}{\nu_\mathcal{T} + \varepsilon}} \\ &= \text{upper}\left(\frac{\nu_\phi}{\nu_\mathcal{T}(k) + \varepsilon} \log\left(\mathbf{G}_\mathcal{T}^{-\frac{1}{2}}(k) \mathbf{Z} \mathbf{G}_\mathcal{T}^{-\frac{1}{2}}(k) \right) \right) \end{aligned} \tag{17}$$

where $(\mathbf{G}_\mathcal{T}(k), \nu_\mathcal{T}^2(k))$ are the estimated mean and variance at training step k, and $\phi = \{\nu_\phi\}$ the learnable parameters. If the assumptions of proposition 2 are fulfilled, $\mathbf{G}_\mathcal{T}(k)$ converges to $\mathbf{G}$ and, in turn, m_ϕ to a scaled version of $\mathcal{P}_\mathbf{G}$, since upper is linear.

The mapping m_ϕ offers several advantageous properties. First, the features are projected to a Euclidean vector space where standard layers can be applied and distances are cheap to compute. Second, distances between the projected features locally approximate δ_{AIRM} and, therefore, inherit its invariance properties (e.g., affine mixing) [41]. This improves upon a LogEig layer [37] which projects features to the tangent space at the identity matrix. As a result, distances between LogEig projected features correspond to distances measured with the log-Euclidean Riemannian metric (LERM) [51] which is not invariant to affine mixing. Third, controlling the Fréchet variance in (17) empirically speeds up learning and improves generalization [48].

4 SPDDSMBN to crack interpretable MSMTUDA for EEG data

With SPDDSMBN introduced in the previous section, we focus on a specific application domain, namely, MSMTUDA for EEG-based BCIs and propose an intrinsically interpretable architecture which we denote TSMNet.

Generative model of EEG EEG signals $\mathbf{x}(t) \in \mathbb{R}^P$ capture voltage fluctuations on P channels. A EEG record (=domain) is uniquely identified by a subject and session identifier. After standard pre-processing steps, each domain i contains $j = 1, ..., M$ labeled observations with features $\mathbf{X}_{ij} \in \mathcal{X} \subset \mathbb{R}^{P \times T}$ where T is the number of temporal samples. Due to linearity of Maxwell's equations and Ohmic conductivity of tissue layers in the frequency ranges relevant for EEG [52], a domain-specific linear instantaneous mixture of sources model is a valid generative model:

$$\mathbf{X}_{ij} = \mathbf{A}_i \mathbf{S}_{ij} + \mathbf{N}_{ij} \tag{18}$$

where $\mathbf{S}_{ij} \in \mathbb{R}^{Q \times T}$ represents the activity of Q latent sources, $\mathbf{A}_i \in \mathbb{R}^{P \times Q}$ a domain-specific mixing matrix and $\mathbf{N}_{ij} \in \mathbb{R}^{P \times T}$ additive noise. Both $\mathbf{A}_i$ and $\mathbf{S}_{ij}$ are unknown which demands making assumptions on $\mathbf{A}_i$ (e.g., anatomical prior knowledge [53]) and/or $\mathbf{S}_{ij}$ (e.g., statistical independence [54]) to extract interesting sources.

[4] To preserve the norm, the off diagonal elements are scaled by $\sqrt{2}$.

Interpretable MSMTUDA for EEG data As label information is available for the source domains, our goal is to identify discriminative oscillatory sources shared across domains. Our approach relies on TSM models with linear classifiers [12], as they are consistent [13] and intrinsically interpretable [17] estimators for generative models with log-linear relationships between the target y and variance $\mathrm{Var}\{s^{(k)}(t)\}$ of $k = 1, ..., K \leq Q$ discriminative sources. We, therefore, constrain our hypothesis class $\mathcal{H}$ to functions $h : \mathcal{X} \times \mathcal{I}_d \to \mathcal{Y}$ that can be decomposed into a composition of a shared linear feature extractor with covariance pooling $f_\theta : \mathcal{X} \to \mathcal{S}_D^+$, domain-specific tangent space mapping $m_\phi : \mathcal{S}_D^+ \times \mathcal{I}_d \to \mathbb{R}^{D(D+1)/2}$, and a shared linear classifier $g_\psi : \mathbb{R}^{D(D+1)/2} \to \mathcal{Y}$ with parameters $\Theta = \{\theta, \phi, \psi\}$.

TSMNet with SPDDSMBN Unlike previous approaches which learn f_θ, m_ϕ, g_ψ sequentially [29, 31, 13, 30], we parametrize $h = g_\psi \circ m_\phi \circ f_\theta$ as a neural network and learn the entire model in an end-to-end fashion (Figure 1b). Details of the proposed architecture, denoted TSMNet, are provided in appendix A.2.4. In a nutshell, we parametrize f_θ as the composition of the first two linear convolutional layers of ShConvNet [55], covariance pooling [56], BiMap [37], and ReEig [37] layers. A BiMap layer applies a linear subspace projection, and a ReEig layer thresholds eigenvalues of symmetric matrices so that the output is SPD. We used the default threshold (10^{-4}) and found that it was never active in the trained models. Hence, after training, f_θ fulfilled the hypothesis class constraints. In order for m_ϕ to align the domain data and compute TSM, we use SPDDSMBN (13) with shared parameters (i.e., $\mathbf{G}_{\phi_i} = \mathbf{G}_\phi = \mathbf{I}, \nu_{\phi_i} = \nu_\phi$) in (17). Finally, the classifier g_ψ was parametrized as a linear layer with softmax activations. We use the standard-cross entropy loss as training objective, and optimized the parameters with the Riemannian ADAM optimizer [57].

5 Experiments with EEG data

In the following, we apply our method to classify target labels from short segments of EEG data. We consider two BCI applications, namely, mental imagery [5] and mental workload estimation [58]. Both applications have high potential to aid society but have, currently, limited practical value because of poor generalization across sessions and subjects [19, 15].

Datasets and preprocessing The considered mental imagery datasets were BNCI2014001 [59] (9 subjects/2 sessions/4 classes), BNCI2015001 [60] (12/2-3/2), Lee2019 [61] (54/2/2), Lehner2020 [62] (1/7/2), Stieger2021 [63] (62/4-8/4) and Hehnberger2021 [64] (1/26/4). For mental workload estimation, we used a recent competition dataset [65] (12/2/3). A detailed description of the datasets is provided in appendix A.2.1. Altogether, we analyzed a total of 603 sessions of 158 human subjects whose data was acquired in previous studies that obtained the subjects' informed consent and the right to share anonymized data.
Python packages (moabb [14], mne [66]) were used to preprocess the datasets. The applied steps comprise resampling the EEG signals to 250/256 Hz, applying temporal filters to extract oscillatory EEG activity in the 4 to 36 Hz range (spectrally resolved if required by a method) and finally extract short segments ($\leq 3s$) associated to a class label (details provided in appendix A.2.2).

Evaluation We evaluated TSMNet against several baseline methods implementing direct transfer or multi-source (-target) UDA strategies. They can be broadly categorized as component based [67, 64], Riemannian geometry aware [12, 17, 30, 68] or deep learning [55, 69, 25]. All models were fit and evaluated with a randomized leave 5% of the sessions (inter-session TL) or subjects (inter-subject TL) out cross-validation (CV) scheme. For inter-session TL, the models were only provided with data of the associated subject. When required, inner train/test splits (neural nets) or CV (shallow methods) were used to optimize hyper parameters (e.g., early stopping, regularization parameters). The dataset Hehenberger2021 was used to fit the hyper parameters of TSMNet, and is, therefore, omitted in the presented results. Balanced accuracy (i.e., the average recall across classes) was used as scoring metric.

Soft- and hardware We either used publicly available python code or implemented the methods in python using the packages torch [70], scikit-learn [71], skorch [72], geoopt [73], mne [66], pyriemann [74], pymanopt [75]. We ran the experiments on standard computation PCs equipped with 32 core CPUs with 128 GB of RAM and used up to 1 GPU (24 GRAM). Depending on the dataset size, fitting and evaluating TSMNet varied from a few seconds to minutes.

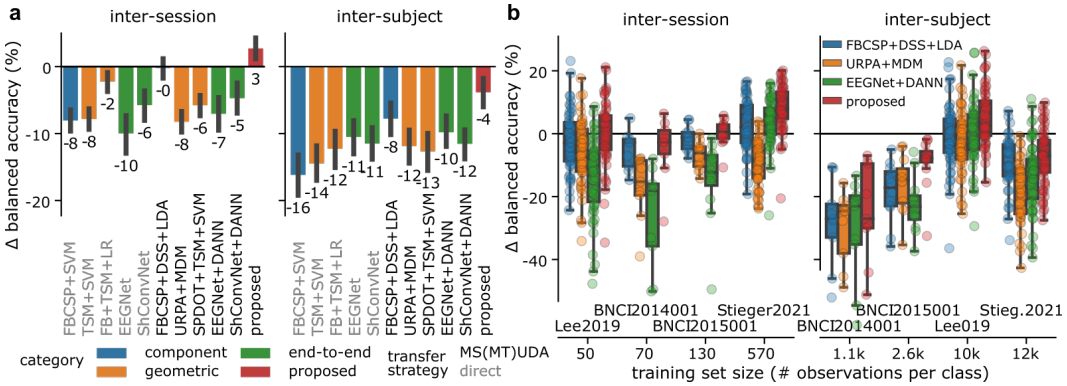

Figure 2: Mental imagery results (5 datasets). BCI test set score (balanced accuracy) for inter-session/-subject transfer learning methods relative to a SOA domain-specific reference model (=FB+TSM+LR; 80%/20% chronological train/test split). **a**, Barplots summarize the grand average (573 sessions, 138 subjects) results. Errorbars indicate bootstrapped (1e3 repetitions) 95% confidence intervals (over subjects). **b**, Box and scatter plots summarize the dataset-specific results for selected methods from each category. Datasets are ordered according to the training set size. Each dot summarizes the score for one subject. Lehner2021 is not displayed as it contains only 1 subject.

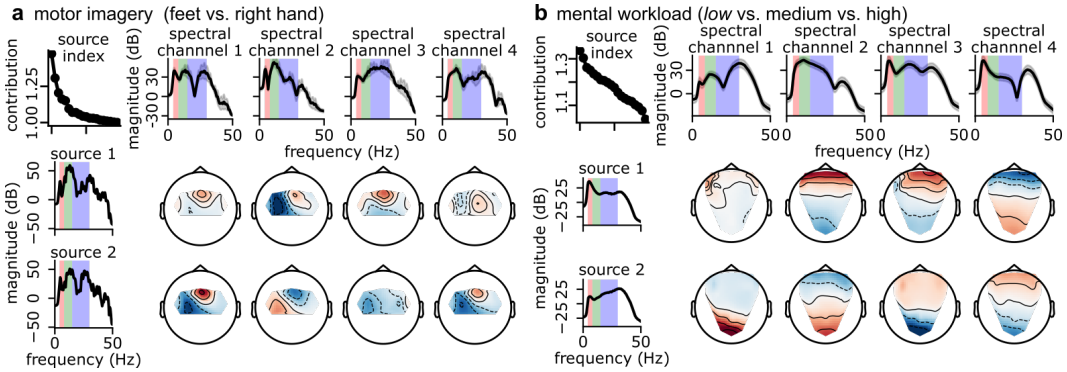

Figure 3: Model interpretation results. Patterns extracted from a fitted TSMNet(SPDDSMBN). **a**, Motor imagery dataset (BNCI2015001, inter-subject TL). The top, left panel lists the contribution of each extracted source (n=20) to the target class. Panels in the left column summarize the spectral patterns of extracted sources. For visualization purposes, only the 2 most discriminative sources are displayed. Panels in the top row summarize the frequency profile of each spectral channel (output of 4 temporal convolution layers in f_θ). Topographic plots summarize the spatial mixing coefficients for each source (=rows) and spectral channel (=columns). **b**, As in **a** but for the mental workload estimation dataset and class *low*.

5.1 Mental imagery

TSMNet closes the gap to domain-specific methods Figure 2 summarizes the mental imagery results. It displays test set scores of the considered TL methods relative to the score of a SOA domain-specific reference method. Combining the results of all subjects across datasets (Figure 2a), it becomes apparent that TSMNet is the only method that can significantly reduce the gap to the reference method (inter-subject) or even exceed its performance (inter-session). Figure 2b displays the results resolved across datasets (for details see appendix A.3.1). We make two important observations. First, concerning inter-session TL, TSMNet meets or exceeds the score of the reference method consistently across datasets. Second, concerning inter-subject TL, we found that all considered methods tend to reduce the performance gap as the dataset size (# subjects) increases, and that TSMNet is consistently the top or among the top methods. As a fitted TSMNet corresponds to a typical TSM model with a linear classifier, we can transform the fitted parameters into interpretable patterns [17]. Figure 3a displays extracted patterns for the BNCI2015001 dataset (inter-subject TL). It is clearly visible that TSMNet infers the target label from neurophysiologically plausible sources (rows in Figure 3a). As expected [2], the source with highest contribution has spectral peaks in the alpha and beta bands, and originates in contralateral and central sensorimotor cortex.

Table 1: Ablation study. Grand average (5 mental imagery datasets, 138 subjects, inter-session TL) score for the test data relative to the proposed method, and training fit time (50 epochs). Standard-deviation is used to report the variability across subjects. Permutation t-tests (1e4 perms, df=137, 4 tests with t-max adjustment) were used to identify significant effects.

			Δ balanced accuracy (%)		fit time (s)
			mean (std)	t-val (p-val)	mean (std)
$\mathcal{S}_D^+$	DSBN	BN method			
yes	yes	SPDMBN (algo. 1) *(proposed)*	-	-	16.9 (1.0)
	yes	SPDBN [48]	-1.6 (2.2)	-7.8 (0.0001)	20.3 (1.6)
	no	SPDMBN (algo. 1)	-3.9 (4.4)	-10.7 (0.0001)	11.3 (0.5)
no	yes	MBN [32]	-4.5 (3.8)	-10.1 (0.0001)	6.6 (0.2)
	no	MBN [32]	-6.9 (4.8)	-13.4 (0.0001)	4.4 (0.1)

DSBN on $\mathcal{S}_D^+$ drives the success of TSMNet Since TSMNet combines several advances, we present the results of an ablation study in Table 1. It summarizes the grand average inter-session TL test scores relative to TSMNet with SPDDSMBN for n = 138 subjects. We observed three significant effects. The largest effect can be attributed to $\mathcal{S}_D^+$, as we observed the largest performance decline if the architecture would be modified[5] to omit the SPD manifold (4.5% with DSBN, 3% w/o DSBN). The performance gain comes at the cost of a 2.6x longer time to fit the parameters. The second largest effect can be attributed to DSBN; without DSBN the performance dropped by 3.9% (with $\mathcal{S}_D^+$) and 2.4% (w/o $\mathcal{S}_D^+$). The smallest, yet significant effect can be attributed to SPDMBN.

5.2 Mental workload estimation

Compared to the baseline methods, TSMNet obtained the highest average scores of 54.7% (7.3%) and 52.4% (8.8%) in inter-session and -subject TL (for details see appendix A.3.1). Interestingly, the inter-session TL score of TSMNet matches the score (54.3%) of the winning method in last year's competition [15]. To shed light on the sources utilized by TSMNet, we show patterns for a fitted model in Figure 3b. For the low mental workload class, the top contributing source's activity peaked in the theta band and originated in pre-frontal areas. The second source's activity originated in occipital cortex with non-focal spectral profile. Our results agree with the findings of previous research, as both areas and the theta band have been implicated in mind wandering and effort withdrawal [58].

6 Discussion

In this contribution, we proposed a machine learning framework around (domain-specific) momentum batch normalization on $\mathcal{S}_D^+$ to learn tangent space mapping (TSM) models and feature extractors in an end-to-end fashion. In a theoretical analysis, we provided error bounds as well as convergence guarantees under reasonable assumptions. We then applied the framework, to a multi-source multi-target unsupervised domain adaptation problem, namely, inter-session and -subject transfer learning for EEG data and obtained or attained state-of-the art performance with a simple, intrinsically interpretable model, denoted TSMNet, in a total of 6 diverse BCI datasets (138 human subjects, 573 sessions). In the case of mental imagery, we found that TSMNet significantly reduced (inter-subject TL) or even exceeded (inter-session TL) the performance gap to a SOA domain-specific method. Although our framework could be readily extended to online UDA for unseen target domains, we limited this study to offline evaluations and leave actual BCI studies to future work. A limitation of our framework, and also any other method that involves eigen decomposition, is the computational complexity, which limits its application to high-dimensional SPD features (e.g., fMRI connectivity matrices with fine spatial granularity). Altogether, the presented results demonstrate the utility of our framework and in particular TSMNet as it not only achieves highly competitive results but is also intrinsically interpretable. While we do not foresee any immediate negative societal impacts, we provide direct contributions towards the scalability and acceptability of EEG-based healthcare and consumer technologies.

[5] To do so, we replaced the covariance pooling, BiMap, ReEig, SPD(DS)MBN, LogEig layers with variance pooling, elementwise log activations followed by (DS)MBN. Note that the resulting architecture is similar to ShConvNet.

References

[1] Donald L. Schomer and Fernando H. Lopes da Silva. *Niedermeyer's Electroencephalography: basic principles, clinical applications, and related fields.* Lippincott Williams & Wilkins, 7 edition, 2018. ISBN 978-0-19-022848-4. doi: 10.1212/WNL.0b013e31822f0490.

[2] G Pfurtscheller and F H Lopes. Event-related EEG / MEG synchronization and. *Clinical Neurophysiology*, 110:10576479, 1999.

[3] Josef Faller, Jennifer Cummings, Sameer Saproo, and Paul Sajda. Regulation of arousal via online neurofeedback improves human performance in a demanding sensory-motor task. *Proceedings of the National Academy of Sciences*, 116(13):6482–6490, 2019. doi: 10.1073/pnas.1817207116.

[4] Yu Zhang, Wei Wu, Russell T. Toll, Sharon Naparstek, Adi Maron-Katz, Mallissa Watts, Joseph Gordon, Jisoo Jeong, Laura Astolfi, Emmanuel Shpigel, Parker Longwell, Kamron Sarhadi, Dawlat El-Said, Yuanqing Li, Crystal Cooper, Cherise Chin-Fatt, Martijn Arns, Madeleine S. Goodkind, Madhukar H. Trivedi, Charles R. Marmar, and Amit Etkin. Identification of psychiatric disorder subtypes from functional connectivity patterns in resting-state electroencephalography. *Nature Biomedical Engineering*, 5(4):309–323, 2021. doi: 10.1038/s41551-020-00614-8.

[5] Jonathan R Wolpaw, Niels Birbaumer, Dennis J McFarland, Gert Pfurtscheller, and Theresa M Vaughan. Brain-computer interfaces for communication and control. *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology*, 113:767–791, 2002. doi: doi:10.1016/s1388-2457(02)00057-3.

[6] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79 (1-2):151–175, 2010. doi: 10.1007/s10994-009-5152-4.

[7] Judy Hoffman, Mehryar Mohri, and Ningshan Zhang. Algorithms and theory for multiple-source adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[8] Dongrui Wu, Yifan Xu, and Bao-Liang Lu. Transfer Learning for EEG-Based Brain-Computer Interfaces: A Review of Progress Made Since 2016. *IEEE Transactions on Cognitive and Developmental Systems*, pages 1–1, 2020. doi: 10.1109/TCDS.2020.3007453.

[9] Christa Neuper, Reinhold Scherer, Miriam Reiner, and Gert Pfurtscheller. Imagery of motor actions: Differential effects of kinesthetic and visual–motor mode of imagery in single-trial EEG. *Cognitive Brain Research*, 25(3):668–677, 2005. doi: 10.1016/j.cogbrainres.2005.08.014.

[10] F Lotte, L Bougrain, A Cichocki, M Clerc, M Congedo, A Rakotomamonjy, and F Yger. A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update. *Journal of Neural Engineering*, 15(3):031005, 2018. doi: 10.1088/1741-2552/aab2f2.

[11] Karina Statthaler, Andreas Schwarz, David Steyrl, Reinmar J. Kobler, Maria Katharina Höller, Julia Brandstetter, Lea Hehenberger, Marvin Bigga, and Gernot Müller-Putz. Cybathlon experiences of the Graz BCI racing team Mirage91 in the brain-computer interface discipline. *Journal of NeuroEngineering and Rehabilitation*, 14(1):129, 2017. doi: 10.1186/s12984-017-0344-9.

[12] Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Multiclass brain-computer interface classification by Riemannian geometry. *IEEE transactions on bio-medical engineering*, 59(4):920–928, 2012. doi: 10.1109/TBME.2011.2172210.

[13] David Sabbagh, Pierre Ablin, Gaël Varoquaux, Alexandre Gramfort, and Denis A Engemann. Manifold-regression to predict from MEG/EEG brain signals without source modeling. In *Advances in Neural Information Processing Systems*, pages 7323–7334, 2019.

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

[15] Raphaëlle N. Roy, Marcel F. Hinss, Ludovic Darmet, Simon Ladouce, Emilie S. Jahanpour, Bertille Somon, Xiaoqi Xu, Nicolas Drougard, Frédéric Dehais, and Fabien Lotte. Retrospective on the First Passive Brain-Computer Interface Competition on Cross-Session Workload Estimation. *Frontiers in Neuroergonomics*, 3:838342, 2022. doi: 10.3389/fnrgo.2022.838342.

[16] Marco Congedo, Alexandre Barachant, and Rajendra Bhatia. Riemannian geometry for EEG-based brain-computer interfaces; a primer and a review. *Brain-Computer Interfaces*, 4(3): 155–174, 2017. doi: 10.1080/2326263X.2017.1297192.

[17] Reinmar J. Kobler, Jun-Ichiro Hirayama, Lea Hehenberger, Catarina Lopes-Dias, Gernot Müller-Putz, and Motoaki Kawanabe. On the interpretation of linear Riemannian tangent space model parameters in M/EEG. In *Proceedings of the 43rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2021. doi: 10.1109/EMBC46164.2021.9630144.

[18] Stephen H. Fairclough and Fabien Lotte. Grand Challenges in Neurotechnology and System Neuroergonomics. *Frontiers in Neuroergonomics*, 1:602504, 2020. doi: 10.3389/fnrgo.2020.602504.

[19] Xiaoxi Wei, A. Aldo Faisal, Moritz Grosse-Wentrup, Alexandre Gramfort, Sylvain Chevallier, Vinay Jayaram, Camille Jeunet, Stylianos Bakas, Siegfried Ludwig, Konstantinos Barmpas, Mehdi Bahri, Yannis Panagakis, Nikolaos Laskaris, Dimitrios A. Adamos, Stefanos Zafeiriou, William C. Duong, Stephen M. Gordon, Vernon J. Lawhern, Maciej Śliwowski, Vincent Rouanne, and Piotr Tempczyk. 2021 BEETL Competition: Advancing Transfer Learning for Subject Independence & Heterogenous EEG Data Sets. *arXiv:2202.12950 [cs, eess]*, 2022.

[20] Vinay Jayaram, Morteza Alamgir, Yasemin Altun, Bernhard Scholkopf, and Moritz Grosse-Wentrup. Transfer Learning in Brain-Computer Interfaces Abstract\uFFFDThe performance of brain-computer interfaces (BCIs) improves with the amount of avail. *IEEE Computational Intelligence Magazine*, 11(1):20–31, 2016. doi: 10.1109/MCI.2015.2501545.

[21] Siamac Fazli, Florin Popescu, Márton Danóczy, Benjamin Blankertz, Klaus-Robert Müller, and Cristian Grozea. Subject-independent mental state classification in single trials. *Neural Networks*, 22(9):1305–1312, 2009. doi: 10.1016/j.neunet.2009.06.003.

[22] Wojciech Samek, Motoaki Kawanabe, and Klaus Robert Müller. Divergence-based framework for common spatial patterns algorithms. *IEEE Reviews in Biomedical Engineering*, 7:50–72, 2014. doi: 10.1109/RBME.2013.2290621.

[23] O-Yeon Kwon, Min-Ho Lee, Cuntai Guan, and Seong-Whan Lee. Subject-Independent Brain–Computer Interfaces Based on Deep Convolutional Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(10):3839–3852, 2020. doi: 10.1109/TNNLS.2019.2946869.

[24] Sicheng Zhao, Bo Li, Colorado Reed, Pengfei Xu, and Kurt Keutzer. Multi-source Domain Adaptation in the Deep Learning Era: A Systematic Survey, 2020.

[25] Ozan Ozdenizci, Ye Wang, Toshiaki Koike-Akino, and Deniz Erdogmus. Learning Invariant Representations From EEG via Adversarial Inference. *IEEE Access*, 8:27074–27085, 2020. doi: 10.1109/ACCESS.2020.2971600.

[26] Lichao Xu, Zhen Ma, Jiayuan Meng, Minpeng Xu, Tzyy-Ping Jung, and Dong Ming. Improving Transfer Performance of Deep Learning with Adaptive Batch Normalization for Brain-computer Interfaces *. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 5800–5803, Mexico, 2021. IEEE. ISBN 978-1-72811-179-7. doi: 10.1109/EMBC46164.2021.9629529.

[27] Lichao Xu, Minpeng Xu, Yufeng Ke, Xingwei An, Shuang Liu, and Dong Ming. Cross-Dataset Variability Problem in EEG Decoding With Deep Learning. *Frontiers in Human Neuroscience*, 14:103, 2020. doi: 10.3389/fnhum.2020.00103.

[28] Ravikiran Mane, Effie Chew, Karen Chua, Kai Keng Ang, Neethu Robinson, A. P. Vinod, Seong-Whan Lee, and Cuntai Guan. FBCNet: A Multi-view Convolutional Neural Network for Brain-Computer Interface. *arXiv:2104.01233 [cs, eess]*, 2021.

[29] Paolo Zanini, Marco Congedo, Christian Jutten, Salem Said, and Yannick Berthoumieu. Transfer Learning: A Riemannian Geometry Framework With Applications to Brain–Computer Interfaces. *IEEE Transactions on Biomedical Engineering*, 65(5):1107–1116, 2018. doi: 10.1109/TBME.2017.2742541.

[30] Pedro Luiz Coelho Rodrigues, Christian Jutten, and Marco Congedo. Riemannian Procrustes Analysis: Transfer Learning for Brain–Computer Interfaces. *IEEE Transactions on Biomedical Engineering*, 66(8):2390–2401, 2019. doi: 10.1109/TBME.2018.2889705.

[31] 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. doi: 10.1109/TSP.2019.2894801.

[32] Hongwei Yong, Jianqiang Huang, Deyu Meng, Xiansheng Hua, and Lei Zhang. Momentum Batch Normalization for Deep Learning with Small Batch Size. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, volume 12357, pages 224–240. Springer International Publishing, Cham, 2020. ISBN 978-3-030-58609-6 978-3-030-58610-2. doi: 10.1007/978-3-030-58610-2_14.

[33] He He and Dongrui Wu. Transfer Learning for Brain–Computer Interfaces: A Euclidean Space Data Alignment Approach. *IEEE Transactions on Biomedical Engineering*, 67(2):399–410, 2020. doi: 10.1109/TBME.2019.2913914.

[34] Stylianos Bakas, Siegfried Ludwig, Konstantinos Barmpas, Mehdi Bahri, Yannis Panagakis, Nikolaos Laskaris, Dimitrios A. Adamos, and Stefanos Zafeiriou. Team Cogitat at NeurIPS 2021: Benchmarks for EEG Transfer Learning Competition, 2022.

[35] 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. doi: 10.1109/MSP.2017.2693418.

[36] Ce Ju and Cuntai Guan. Tensor-CSPNet: A Novel Geometric Deep Learning Framework for Motor Imagery Classification. *arXiv:2202.02472 [cs, eess]*, 2022.

[37] Zhiwu Huang and Luc Van Gool. A Riemannian Network for SPD Matrix Learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2036–2042. AAAI Press, 2017.

[38] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-Specific Batch Normalization for Unsupervised Domain Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[39] Rajendra Bhatia. *Positive definite matrices*. Princeton university press, 2009.

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

[41] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *International Journal of computer vision*, 66(1):41–66, 2006.

[42] Hermann Karcher. Riemannian center of mass and mollifier smoothing. *Communications on pure and applied mathematics*, 30(5):509–541, 1977.

[43] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[44] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Mądry. How Does Batch Normalization Help Optimization? In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 2488–2498, Red Hook, NY, USA, 2018. Curran Associates Inc.

[45] Sergey Ioffe. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 1942–1950, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

[46] Daniel Brooks, Olivier Schwander, Frederic Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Riemannian batch normalization for SPD neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[47] Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

[48] 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)*, pages 3863–3867, Singapore, Singapore, 2022. IEEE. ISBN 978-1-66540-540-9. doi: 10.1109/ICASSP43922.2022.9746629.

[49] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix Backpropagation for Deep Networks with Structured Layers. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2965–2973, Santiago, Chile, 2015. IEEE. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.339.

[50] Jeffrey Ho, Guang Cheng, Hesamoddin Salehian, and Baba Vemuri. Recursive Karcher Expectation Estimators And Geometric Law of Large Numbers. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 325–332, Scottsdale, Arizona, USA, 2013. PMLR.

[51] Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Geometric Means in a Novel Vector Space Structure on Symmetric Positive-Definite Matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(1):328–347, 2007. doi: 10.1137/050637996.

[52] Paul L. Nunez and Ramesh Srinivasan. *Electric Fields of the Brain*. Oxford University Press, 2006. ISBN 978-0-19-505038-7. doi: 10.1093/acprof:oso/9780195050387.001.0001.

[53] Christoph M. Michel, Micah M. Murray, Göran Lantz, Sara Gonzalez, Laurent Spinelli, and Rolando Grave De Peralta. EEG source imaging. *Clinical Neurophysiology*, 115(10), 2004. doi: 10.1016/j.clinph.2004.06.001.

[54] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000. doi: 10.1016/S0893-6080(00)00026-5.

[55] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for EEG decoding and visualization: Convolutional Neural Networks in EEG Analysis. *Human Brain Mapping*, 38 (11):5391–5420, 2017. doi: 10.1002/hbm.23730.

[56] Dinesh Acharya, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool. Covariance Pooling for Facial Expression Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[57] Gary Becigneul and Octavian-Eugen Ganea. Riemannian Adaptive Optimization Methods. In *International Conference on Learning Representations*, 2019.

[58] Frédéric Dehais, Alex Lafont, Raphaëlle Roy, and Stephen Fairclough. A Neuroergonomics Approach to Mental Workload, Engagement and Human Performance. *Frontiers in Neuroscience*, 14:268, 2020. doi: 10.3389/fnins.2020.00268.

[59] Michael Tangermann, Klaus-Robert Müller, Ad Aertsen, Niels Birbaumer, Christoph Braun, Clemens Brunner, Robert Leeb, Carsten Mehring, Kai J Miller, Gernot Müller-Putz, Guido Nolte, Gert Pfurtscheller, Hubert Preissl, Gerwin Schalk, Alois Schlögl, Carmen Vidaurre, Stephan Waldert, and Benjamin Blankertz. Review of the BCI Competition IV. *Frontiers in Neuroscience*, 6, 2012.

[60] Josef Faller, Carmen Vidaurre, Teodoro Solis-Escalante, Christa Neuper, and Reinhold Scherer. Autocalibration and recurrent adaptation: towards a plug and play online ERD-BCI. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 20(3):313–319, 2012.

[61] Min-Ho Lee, O-Yeon Kwon, Yong-Jeong Kim, Hong-Kyung Kim, Young-Eun Lee, John Williamson, Siamac Fazli, and Seong-Whan Lee. EEG dataset and OpenBMI toolbox for three BCI paradigms: an investigation into BCI illiteracy. *GigaScience*, 8(5):giz002, 2019. doi: 10.1093/gigascience/giz002.

[62] Rea Lehner, Neethu Robinson, Tushar Chouhan, Mihelj Mihelj, Ernest, Kratka Kratka, Paulina, Frédéric Debraine, Cuntai Guan, and Nicole Wenderoth. Design considerations for long term non-invasive Brain Computer Interface training with tetraplegic CYBATHLON pilot: CYBATHLON 2020 Brain-Computer Interface Race Calibration Paradigms. 2020. doi: 10.3929/ETHZ-B-000458693. URL `http://hdl.handle.net/20.500.11850/458693`. Medium: application/vnd.openxmlformats-officedocument.wordprocessingml.document,application/octet-stream,application/zip,text/plain,2.93 GB Publisher: ETH Zurich.

[63] James R. Stieger, Stephen A. Engel, and Bin He. Continuous sensorimotor rhythm based brain computer interface learning in a large population. *Scientific Data*, 8(1):98, 2021. doi: 10.1038/s41597-021-00883-1.

[64] Lea Hehenberger, Reinmar J. Kobler, Catarina Lopes-Dias, Nitikorn Srisrisawang, Peter Tumfart, John B. Uroko, Paul R. Torke, and Gernot R. Müller-Putz. Long-term mutual training for the CYBATHLON BCI Race with a tetraplegic pilot: a case study on inter-session transfer and intra-session adaptation. *Frontiers in Human Neuroscience*, 2021. doi: 10.3389/fnhum.2021.635777.

[65] 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. doi: 10.5281/ZENODO.5055046.

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

[67] Kai Keng Ang, Zhang Yang Chin, Haihong Zhang, and Cuntai Guan. Filter Bank Common Spatial Pattern (FBCSP) in Brain-Computer Interface. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 2390–2397, Hong Kong, China, 2008. IEEE. ISBN 978-1-4244-1820-6. doi: 10.1109/IJCNN.2008.4634130.

[68] Or Yair, Felix Dietrich, Ronen Talmon, and Ioannis G. Kevrekidis. Domain Adaptation with Optimal Transport on the Manifold of SPD matrices. *arXiv:1906.00616 [cs, stat]*, 2020.

[69] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of Neural Engineering*, 15(5):056013, 2018. doi: 10.1088/1741-2552/aace8c.

[70] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[71] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[72] Marian Tietz, Thomas J. Fan, Daniel Nouri, Benjamin Bossan, and skorch Developers. *skorch: A scikit-learn compatible neural network library that wraps PyTorch*. 2017. URL `https://skorch.readthedocs.io/en/stable/`.

[73] Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian Optimization in PyTorch, 2020. _eprint: 2005.02819.

[74] Barachant Alexandre. pyRiemann, 2022. URL `https://github.com/pyRiemann/pyRiemann`.

[75] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016.

[76] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See paragraph 4 in section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See appendix A.1 in the supplementary material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplementary Material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See paragraph 2 in section 5 for a brief summary and appendix A.2 for details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We provide error bars with respect to the variability across sessions and subjects.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See paragraph 3 in section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See paragraph 1 in section 5.
   (b) Did you mention the license of the assets? [Yes] See appendix A.2.1 in the Supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See paragraph 1 in section 5.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See paragraph 1 in section 5.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]