Automatic Classification of Sleep Stages from EEG Signals Using Riemannian Metrics and Transformer Networks

Mathieu Seraphim[1*], Alexis Lechervy[1], Florian Yger[3], Luc Brun[1], Olivier Etard[2]

[1*]Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France.
[2]Normandie Université, UNICAEN, INSERM, COMETE, CYCERON, CHU Caen, 14000, Caen, France.
[3]LAMSADE, CNRS, PSL University Paris-Dauphine, 75016 Paris, France.

*Corresponding author. E-mail: mathieu.seraphim@unicaen.fr

Abstract

Purpose: In sleep medicine, assessing the evolution of a subject's sleep often involves the costly manual scoring of electroencephalographic (EEG) signals. In recent years, a number of Deep Learning approaches have been proposed to automate this process, mainly by extracting features from said signals. However, despite some promising developments in related problems, such as Brain-Computer Interfaces, analyses of the covariances between brain regions remain underutilized in sleep stage scoring.
Methods: Expanding upon our previous work, we investigate the capabilities of SPDTransNet, a Transformer-derived network designed to classify sleep stages from EEG data through timeseries of covariance matrices. Furthermore, we present a novel way of integrating learned signal-wise features into said matrices without sacrificing their Symmetric Definite Positive (SPD) nature.
Results: Through comparison with other State-of-the-Art models within a methodology optimized for class-wise performance, we achieve a level of performance at or beyond various State-of-the-Art models, both in single-dataset and - particularly - multi-dataset experiments.
Conclusion: In this article, we prove the capabilities of our SPDTransNet model, particularly its adaptability to multi-dataset tasks, within the context of EEG

sleep stage scoring - though it could easily be adapted to any classification task involving timeseries of covariance matrices.

Keywords: Sleep Scoring, Sleep Medicine, SPD Manifold, Transformers

Acknowledgments. This work was granted access to the HPC resources of IDRIS (Institut du Développement et des Ressources en Informatique Scientifique) under the allocation 2022-AD010613618 made by GENCI (Grand Équipement National de Calcul Intensif), and to the computing resources of CRIANN (Centre Régional Informatique et d'Applications Numériques de Normandie, Normandy, France).

1 Introduction

Covariance matrices have long been used to analyze groups of concurrent signals, with applications ranging from financial analysis [1] to hand gesture recognition [2]. By construction, they function as descriptors of the relationships between said signals, but also exhibit exploitable mathematical properties, as they usually exist on the non-Euclidean manifold of Symmetric Positive Definite (SPD) matrices [3]. For instance, SPD matrices are commonly used in structural imagery through Diffusion Tensor Imaging (DTI) [4], and in brain activity estimation tasks.

Said estimations are often done through the lens of functional connectivity, i.e. the correlations of activity between brain regions [5]. By construction, correlation and covariance matrices are well suited to such tasks, and have been derived from functional MRI (fMRI) imagery [6] and electroencephalographic (EEG) signals[1]. EEG-derived connectivity estimations, in particular, are widely used in the field of Brain-Computer Interfaces (BCI), where they are analyzed through geometry-preserving tools [7].

EEG signals are also utilized in sleep medicine as part of the polysomnography (PSG) exam, where they are used to determine the evolution of a subject's sleep over time in order to diagnose sleep disorders. This has traditionally been a time-consuming process, as it has to be done manually by experts. in recent years, a number of approaches have been proposed to automate said process (Section 2.1). However, even though functional connectivity has been shown to be indicative of a subject's sleep stage [8], most of these focus on analyzing individual signal properties, rather than the interactions between signals. Furthermore, SPD analysis seems largely absent from discussions regarding EEG sleep staging, despite the similarities between this problem and BCI.

In response to this under-utilization of functional connectivity for sleep analysis, we previously developed an approach to the automatic classification of sleep stages from EEG data, predominantly through the analysis of EEG-derived covariance matrices [9]. This culminated in **SPDTransNet** [10], a Transformer-based Deep Learning model adapted to analyze sequences of SPD matrices, most notably through **SP-MHA**, a structure-preserving multihead attention mechanism.

[1]Electrical signals acquired from electrodes located around the brain, often non-invasively.

Table 1: Frequency bands inspired by the literature [11]

	Delta	Theta	Alpha	Beta$_{low}$	Beta$_{high}$	Gamma
Hz	[0.5, 4[	[4, 8[	[8, 12[	[12, 22[	[22, 30[	[30, 45[

In this paper, we further analyze the capabilities of SPDTransNet, and introduce a variant that allows for the inclusion of additional EEG-derived learned features within our input matrices. We conduct a thorough examination of our model's capabilities, including its ability to perform in a multi-dataset environment, similarly to what might be encountered in clinical use. Finally, we compare SPDTransNet with various State-of-the-Art (SOA) approaches, showcasing our model's performance and adaptability in both single-dataset and multi-dataset experiments.

Our code is available on GitHub, in the MathieuSeraphim/SPDTransNet_plus repository.

2 Related Work

2.1 Automatic Sleep Stage Scoring

In order to determine the evolution of a subject's sleep over time, the signals originating from a PSG exam are split into fixed-length windows, or "epochs", with each epoch being manually "scored" (i.e. labeled) with a corresponding sleep stage by experts. This scoring is based on the properties of the considered signals in and around said epoch, such as frequential components and distinct punctual events. A selection of relevant frequency bands for PSG signal analysis can be found in Table 1.

The widespread scoring manual of the American Academy of Sleep Medicine (AASM) [11] defines a total of 5 sleep stages - wakefulness, REM sleep, and three stages of non-REM sleep, N1 to N3, along with the corresponding characteristics thereof, and recommends epochs of thirty seconds of length.

Most State-of-the-Art automatic sleep staging models published in recent years take sequences of PSG epochs as input, rather than individual epochs, to better account for contextual information. Said models usually follow one of two classification schemes - sequence-to-element (sometimes called many-to-one) [12, 13], or sequence-to-sequence (a.k.a. many-to-many) [14–18]. In a sequence-to-element classification scheme, the model outputs the classification of a single epoch in the input sequence. By contrast, a sequence-to-sequence scheme outputs one classification per input epoch, and utilize a combination strategy if the input sequences overlap each other (meaning that a given epoch may be classified multiple times, each time in a different position in the sequence). According to a recent survey by Phan et al. [19], sequence-to-sequence approaches tend to yield greater global performance, though this comes at the cost of methodological flexibility (Section 5).

Sequence-based classification can be done in a single step, using a Convolutional Neural Networks (CNN) to extract signal features and deliver a classification [12, 20]. Most notably, both Perslev et al. [21, 22] and Jia et al. [23] treated this scoring as a segmentation problem, and utilized architectures derived from U-Net[24]. By contrast, Vilamala et al. [25] and Dequidt et al. [26] both took advantage of the aforementioned frequential characteristics of EEG signals, and analyzed EEG-derived time-frequency spectra using a standard CNN architecture pretrained for image recognition.

However, as seen in the aforementioned survey, the most common approach is to use a two-step architecture, by first extracting epoch-wise features before comparing said features in a sequence-wise submodel. Here, CNNs are often utilized for epoch-wise feature extraction, with architectures designed for sequence analysis being utilized at the sequence level. The DeepSleepNet model by Supratak et al. [14], often used as a benchmark, uses a Recurrent Neural Network (RNN), as does IITNet by Seo et al. [13], which expands upon DeepSleepNet by extracting features from overlapping subwindows rather than from entire epochs. Going one step further, Phan et al. [15, 17, 18] and Guillot et al. [27, 28] elected to use time-frequency spectra as input of their RNN-based architectures, interpreting the epoch-wise spectra as timeseries of frequency bins and analyzing them with RNNs.

With the advent of Transformer architectures [29], attention mechanisms have proven to be particularly potent in the analysis of temporal sequences. As such, these mechanisms have been applied to many fields, including sleep stage scoring. Some of these approaches still utilize CNNs to extract intra-epoch features from 1D signals [30–32], while others utilize them at both epoch-wise and sequence-wise levels [33], with both SleepTransformer by Phan et al. [16] and MultiChannelSleepNet by Dai et al. [34] taking time-frequency spectra as inputs.

With the exception of Dequidt et al [26] and the pre-DeepSleepNet CNN-only approach of Chambon et al. [12], none of the above approaches utilize large numbers of EEG electrodes, and therefore cannot take advantage of functional connectivity (Section 1). This is unsurprising, as the aforementioned AASM ruleset emphasizes the analysis through signal properties, and the most commonly used open sleep stage scoring datasets - SleepEDF [35] and SHHS [36, 37] - only offer two EEG signals for analysis. That is not to say that no approach has explicitly analyzed functional connectivity between a large number of EEG signals. For instance, Jia et al. [38, 39] - and later Einizade et al [40] - describe each epoch with a learned graph with electrodes as nodes, before comparing them through a sequence-wise spatio-temporal graph neural network (GNN). This approach arguably corresponds to an analysis through functional connectivity, though this encoding is likely to disregard some spatial information inherent to each electrode location, due to the permutation invariance property of GNNs.

It is in this context that we designed the SPDTransNet model, analyzing sequences of EEG epoch for the purpose of sleep stage scoring using a two-step Transformer architecture, through the lens of functional connectivity estimated through covariance matrices. As far as we are aware, ours is the only Machine Learning approach to utilize covariance matrices for automatic sleep stage scoring.

2.2 The Symmetric Positive Definite Manifold

All covariance matrices are Symmetric Positive Semi-Definite (SPSD), i.e. with positive eigenvalues. Furthermore, when computed from linearly independent signals, they are fully Symmetric Positive Definite (SPD) - with strictly positive eigenvalues. Conversely, all SPD matrices can be understood as covariance matrices between linearly independent signals.

The set of $n \times n$ SPD matrices, or $SPD(n)$, is a non-Euclidean, Riemannian (i.e. metric) manifold, meaning that applying regular Euclidean operations on such data seldom preserve its geometric structure, introducing deformations. For instance, when interpolating between SPD matrices of same determinant using Euclidean metrics, the computed intermediate matrices tend to have a higher determinant - a phenomenon known as the "swelling effect" [41].

To counteract this, multiple Riemannian metrics have been introduced, including affine-invariant metrics [42]. As their name suggests, these offer interesting invariance properties, but present computational difficulties. For instance, there is no closed-form formula for the affine-invariant average of SPD matrices, and it has to be estimated.

Alternatively, LogEuclidean metrics offer similar results at a lower computational cost, though they have been shown to be less isotropic [41]. They are defined using the (bijective) matrix logarithm function $log_{mat}(\cdot)$, which maps $SPD(n)$ onto its tangent space $Sym(n)$, the vector space of symmetric matrices:

$$log_{mat}(X) = U \cdot log(\Lambda) \cdot U^T \in Sym(n) \tag{1}$$

with $X \in SPD(n)$, Λ and U respectively the diagonal matrix containing the eigenvalues of X and the corresponding eigenvectors, and $log(\Lambda)$ the matrix resulting from taking the element-wise natural logarithm of Λ's diagonal elements.

More generally, this projection onto the tangent space can be parametered by a center of projection $P \in SPD(n)$:

$$log^P_{mat}(X) = P^{1/2} \cdot log_{mat}(P^{-1/2} \cdot X \cdot P^{-1/2}) \cdot P^{1/2} \in Sym(n) \tag{2}$$

The tangent space at point P is in essence a local Euclidean approximation of the non-Euclidean manifold. As such, logarithmically mapping elements of $SPD(n)$ that are distant from the chosen parameter P can introduce deformations in their mapped images.

From this, LogEuclidean metrics are defined thusly:

$$\delta^P_{LE}(X, Y) = \|log_{mat}(P^{-1/2} \cdot X \cdot P^{-1/2}) - log_{mat}(P^{-1/2} \cdot Y \cdot P^{-1/2})\|_* \tag{3}$$

with X, Y and P in $SPD(n)$, and $\|\cdot\|_*$ any Euclidean norm on $Sym(n)$.

This formula can also be written by directly using Equation 2, with $\delta^P_{LE}(X, Y) = \|log^P_{mat}(X) - log^P_{mat}(Y)\|_+$. Here, $\|S\|_+$ is equal to $\|P^{1/2} \cdot S \cdot P^{1/2}\|_*$ for all S in $Sym(n)$ [43, 44].

Like their affine-invariant counterparts, LogEuclidean metrics define geodesics on the SPD manifold, and are immune to the swelling effect. And while their relative anisotropy may affect the performance of any model utilizing them, this can be mitigated by choosing an appropriate center of projection P [44]. More importantly, they provide significant computational advantages, such as a closed-form formula for a

Riemannian weighted sum (and by extension, for a Riemannian average):

$$\mathcal{S}um_{LE}^{P}(\{X\},\{w\}) = \exp_{mat}^{P}\left(\sum_{i=1}^{N} w_i \log_{mat}^{P}(X_i)\right) \tag{4}$$

with every X_i in $\{X\} \subset SPD(n)$, every w_i in $\{w\} \subset \mathbb{R}$, and $\exp_{mat}^{P}(\cdot)$ the inverse of $\log_{mat}^{P}(\cdot)$.

2.3 Deep Learning on the SPD Manifold

Traditional Deep Learning techniques are often implicitly designed to process Euclidean data, such as signals or images. This limitation has led to the development of variants, to utilize these powerful techniques in a non-Euclidean setting. Applied to the SPD manifold, Huang and Van Gool [45] developed Deep Learning layers analogous to those found in CNNs, most notably their SPD-to-SPD Bilinear Mapping (BiMap) layer:

$$\forall X \in SPD(n),\ BiMap(X,W) = W \cdot X \cdot W^T \in SPD(m) \tag{5}$$

with $W \in \mathbb{R}^{m\times n}$ a semi-orthogonal weights matrix.

A similar but distinct approach was taken by Chakraborty et al. [46], with SPD matrices being elements of a feature map, rather than the feature map itself. In this approach, elements of $SPD(n)$ are analogous to scalars in a traditional CNN, and convolution operations are replaced by weighted Riemannian averages (Section 2.2).

Though Riemannian methods have been utilized for a while in Brain-Computer Interfaces [7, 47], those have traditionally been relatively simple, such as Riemannian Minimal Distance to Mean (MDM) or SVM classifiers. Those approaches have at times been enhanced through metric learning, applied to the center of projection and/or Euclidean norm of the LogEuclidean metric [44, 48]. Nevertheless, Riemannian Deep Learning approaches have recently started to emerge [49–52].

That is not to say that Deep Learning approaches are rare in BCI [47]. In particular, as shown by a recent survey by Abibullaev et al. [53], a number of Transformer-based approaches have been applied to BCI tasks, though as far as we are aware, none take advantage of Riemannian geometry.

Beyond BCI, some Transformer-based models *do* make use of Riemannian geometry. For instance, both Konstantinidis et al. [54] and Dong et al. [55] generate SPD matrices as attention maps. By contrast, both He et al. [56] and Li et al. [57] adapted their Transformer architecture to handle manifold-valued data, the manifolds in question being 2D surfaces within 3D space. More generally, Kratsios et al. [58] developed a theoretical framework for the use of Transformers to analyze data from a variety of constrained sets, including Riemannian manifolds.

This said, we have yet to encounter a Transformer-based approach designed to analyze SPD-valued data in a structure-preserving manner, even beyond the scope of functional connectivity.

3 Operations on SPD Matrices

Before their analysis by our model, our approach calls for a number of operations to be applied to the original $n \times n$ SPD covariance matrices - to enrich them and improve our model's performance, without sacrificing their underlying SPD structure.

3.1 Matrix Augmentation

By construction, covariance matrices primarily encode comparative information, with the only monosignal information being the variance over the signal segment. However, as hinted at by the level of performance of single-signal approaches (Section 2.1), it is undeniable that there is relevant information contained in the signals themselves, information that may be lost when only considering covariances. Hence, the addition of signal-wise information to our model's input should be beneficial to said model's classification performance.

Let $X \in SPD(n)$ be a covariance matrix encoding for the relationships between n signal segments. Let $A \in \mathbb{R}^{n \times k}$ be a matrix encoding k features for each of these signal segments, and $\alpha \in \mathbb{R}$ be a factor weighting for the importance of A in the final matrix. These matrices may be combined to create the following SPD matrix:

$$X' = aug_\alpha(X, A) = \left(\begin{array}{c|c} X + \alpha^2 \cdot A \cdot A^T & \alpha \cdot A \\ \hline \alpha \cdot A^T & \mathbb{I}_k \end{array}\right) \in SPD(m) \tag{6}$$

with $m = n + k$.

We refer to this operation as an "augmentation", with the resulting SPD matrix X' being the augmented matrix. The model parameter α (called "augmentation factor" in this paper) controls the relative influence of A within the augmented matrix X'.

A proof of the SPD nature of matrices augmented in this manner can be found in Section 1 of the Supplementary Material.

3.2 Matrix Whitening

When analyzing signals originating from multiple recordings, each computed covariance matrix may encode for information specific to its own recording. As such, multi-subject datasets of biological signals, like the ones considered in this paper, can be prone to significant differences from one subject to the next. These specificities essentially act as noise, and are liable to lower the performance of our model.

To increase comparability between recordings, we eliminate these specificities through the whitening operation [44]. Let $X' \in SPD(m)$ be a matrix computed from a given recording (thereafter referred to as a "recording matrix"), and $G \in SPD(m)$ the matrix estimating said recording's specificities (i.e. the recording's "whitening matrix"). The whitened matrix $M \in SPD(m)$ is obtained thusly:

$$M = whit(X', G) = G^{-1/2} \cdot X' \cdot G^{-1/2} \tag{7}$$

This whitening can be interpreted as a transport operation. For instance, computing a LogEuclidean operation on unwhitened matrices using G as the center of projection (Equation 3) is strictly equivalent to computing the same operation on whitened matrices, but with the identity matrix $\mathbb{I}_m$ as center of projection.

Since our matrices are computed from biological recordings of different individuals, they can encode information specific to their own recording. Therefore, within the SPD manifold, matrices corresponding to a given class and originating from a given recording can be geometrically separated from matrices corresponding to the same class but a different recording. By choosing the proper whitening matrix G for each recording, we can realign the corresponding recording matrices within a unified geometric neighborhood.

3.3 Bijective Tokenization

Let $\{M\}$ be a set of matrices within $SPD(m)$. Let $M_i^* = log_{mat}(M_i)$ (Equation 1) be a bijective mapping of $M_i \in \{M\}$ onto $Sym(m)$, the vector space of symmetric matrices (Section 2.2) and of dimension $d(m) = \frac{m(m+1)}{2}$. Finally, let V_{M_i} be the vector representation of M_i^* within the vector space $\mathbb{R}^{d(m)}$, obtained by describing M_i^* in the canonical basis of $Sym(m)$.

It follows that any weighted sum or linear combination between vectors in $\{V_M\} \subset \mathbb{R}^{d(m)}$ is equivalent to the same operation between the corresponding symmetric matrices in $\{M^*\}$, itself equivalent to a LogEuclidean (and therefore Riemannian) operation between the corresponding SPD matrices within $\{M\}$ (Equation 4), parametered by P equal to the identity matrix $\mathbb{I}_m$.

Therefore, by bijectively mapping matrices in $SPD(m)$ onto $\mathbb{R}^{d(m)}$, we can apply some Riemannian operations between our matrices through Euclidean operations in $\mathbb{R}^{d(m)}$. Equivalently, given any triangular number k (so that $k = \frac{j(j+1)}{2}$), any Euclidean weighted sum between elements of $\mathbb{R}^k$ is equivalent to a Riemannian operation within $SPD(j)$.

We refer to this mapping between $SPD(m)$ and $\mathbb{R}^{d(m)}$ as the "*tokenization*" operation[2], and to $\mathbb{R}^{d(m)}$ as a "*triangular vector space*".

3.4 Triangular Maps

Let $\mathcal{L}_{m,p}(\cdot)$ be a linear map between $\mathbb{R}^{d(m)}$ and $\mathbb{R}^{d(p)}$. As stated above, it can also be understood as a linear map between $Sym(m)$ and $Sym(p)$, with the sets' vector representations as intermediaries.

In [10], we used $\mathcal{L}_{m,p}(\cdot)$ to define a mapping between $SPD(m)$ and $SPD(p)$:

$$\mathcal{L}^{\mathcal{R}}_{m,p}(M) = exp_{mat} \circ \mathcal{L}_{m,p} \circ log_{mat}(M) \in SPD(p) \tag{8}$$

with $M \in SPD(m)$, $log_{mat}(\cdot)$ the matrix logarithm and $exp_{mat}(\cdot)$ the matrix exponential (Section 2.2)[3].

[2]Following the common convention of referring to the vector inputs of Transformer-based architectures as "tokens".

[3]Here, the logarithm is defined from $SPD(m)$ to $Sym(m)$, and the exponential from $Sym(p)$ to $SPD(p)$.

By this definition, applying $\mathcal{L}_{m,p}(\cdot)$ on vectors in $\mathbb{R}^{d(m)}$ is equivalent to applying $\mathcal{L}^{\mathcal{R}}_{m,p}(\cdot)$ on matrices in $SPD(m)$.

Since $\mathcal{L}_{m,p}(\cdot)$ is a continuous linear map, it preserves relations of proximity:

$$\|\mathcal{L}_{m,p}(V_A) - \mathcal{L}_{m,p}(V_B)\|_2 \leq \|W\|_\bullet \cdot \|V_A - V_B\|_2 \tag{9}$$

with $V_A, V_B \in \mathbb{R}^{d(m)}$, $W \in \mathbb{R}^{d(p)\times d(m)}$ the transformation matrix of $\mathcal{L}_{m,p}(\cdot)$, $\|\cdot\|_2$ the L^2 norm for vector representations, and $\|\cdot\|_\bullet$ the matrix norm induced by $\|\cdot\|_2$.

By the definition of LogEuclidean metrics (Equation 3), and with $A, B \in SPD(m)$ the matrices that tokenize into V_A and V_B, we can rewrite Equation 9 thusly:

$$\delta^{\mathbb{I}_p}_{LE}(\mathcal{L}^{\mathcal{R}}_{m,p}(A), \mathcal{L}^{\mathcal{R}}_{m,p}(B)) \leq \|W\|_\bullet \cdot \delta^{\mathbb{I}_m}_{LE}(A, B) \tag{10}$$

Hence, the SPD-to-SPD mapping $\mathcal{L}^{\mathcal{R}}_{m,p}(\cdot)$ preserves proximity between elements of the source manifold - and by extension, the underlying SPD structure of the input data.

Note that unlike the BiMap SPD-to-SPD linear mapping (Section 2.3), our $\mathcal{L}^{\mathcal{R}}_{m,p}(\cdot)$ mapping does not require restrictions to be imposed on the weights matrix W.

4 The SPDTransNet Model

The SPDTransNet model, first introduced in [10], can be seen Figure 3. As stated in Section 1, it expends upon an earlier iteration of our approach [9], itself based on the SleepTransformer architecture of Phan et al. [16], and follows a Transformer-based [29] architecture, as those are particularly well suited to the analysis of vector sequences.

To better account for contextual information, as with much of the SOA approaches (Section 2.1), our model takes a sequence of epochs as input, and follows a two-step architecture: an intra-epoch step to extract epoch-wise features, followed by an inter-epoch (or sequence-wise) step to compare them before the final classification. In particular, SPDTransNet takes a sequence of length $L = 2 \cdot \ell + 1$, composed of a central epoch flanked by its ℓ preceding and following epochs (Figure 3).

4.1 Data Preparation

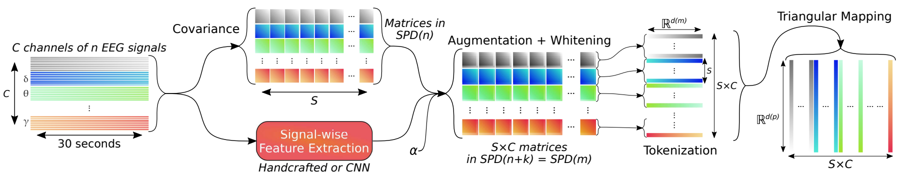

Fig. 1: Our data preparation pipeline, with $S = 30$, $C = 7$ and $n = 8$.

Using the concepts developed in Section 3, for each EEG epoch, we process a collection of signals into a sequence of tokens with an underlying SPD structure, as seen in Figure 1.

4.1.1 Building SPD matrices

After selecting n signals to compose our matrices, our first step is to apply a z-score normalization on each signal, to ensure comparability between all signals, both between recordings and within each recording. Additionally, for each of the six frequency bands defined in Table 1, we apply a fourth order Butterworth bandpass filter, to isolate different frequential components of the signal. This results in a total of $C = 7$ input channels, six derived from filtered signals and one from the unfiltered signal, corresponding to the input in Figure 1.

For each channel, the signals are subdivided into fixed-length segments, without overlap between segments. We chose to fix the segments' length to one second, as this roughly corresponds to the duration of events indicative of a given sleep stage within the signal [11]. We then compute an $n \times n$ covariance matrix for each segment of each channel, netting us a grid of 30×7 such matrices per epoch.

4.1.2 Enriching our Matrices

Once computed, the matrices are then enriched through augmentation and whitening (Sections 3.1 and 3.2), with distinct whitening matrices computed for each channel (Figure 1).

4.1.2.1 Augmentation Strategies

Let $\{X\} \subset SPD(n)$ be the set of all covariance matrices for a given channel of a given recording. As stated in Section 3.1, the goal of the augmentation operation is to add signal-specific information to each covariance matrix $X_i \in \{X\}$, through the augmentation matrix $A_i \in \mathbb{R}^{n \times k}$ weighted by $\alpha \in \mathbb{R}$. With X_i a constant, and for a given value of α, the output of the augmentation operation is entirely determined by the matrix A_i. To compute it, we have developed two strategies.

Firstly, we can compute A_i through signal-wise handcrafted features. That is to say, for each of the n signal segments used to construct X_i, we compute k segment-wise statistics. In this paper, we utilize the average power spectral density (PSD) as the relevant statistic, as we have found it to be the best performing amongst a number of tested statistics in our previous work [9]. Having tested the use of a combination of statistics to construct A_i (i.e. $k > 1$), we have found no improvement in performance, and have therefore elected to only use the average PSD of each signal as handcrafted features (keeping $k = 1$).

Our second strategy is to learn the augmentation matrix A_i using a dedicated pre-trained submodel, integrating said submodel within our architecture and finetuning it during training. As a first iteration, we have chosen to utilize the epoch-wise feature extraction CNN developed by Seo et al. for IITNet [13], as it is designed to analyze subwindows of each input epoch, delivering a local feature vector for each subwindow. We have modified it to take our n signals over C channels per input (Section 4.1.1). The CNN's weights are shared between the n signals in a given channel, but not between channels, since our channel-wise filtering necessarily changes the signals' spectral configuration between channels - and therefore the required CNN embedding.

More details about the learned augmentation submodel can be found in Section 2 of the Supplementary Material.

When utilizing this strategy, in order to not drown the covariance information within the augmented matrix, we impose $k = 3$.

4.1.2.2 Composition with Whitening

As stated in Section 3.2, the whitening operation can be used to transport matrices computed from different recordings onto the same neighborhood within the SPD manifold. As such, we have elected to operate the whitening operation after the augmentation, regardless of whichever augmentation strategy is chosen.

Let I denote the set of epochs for a given recording and channel, and let $\{X_i\}_{i\in I}$ and $\{A_i\}_{i\in I}$ be the computed covariance matrices and their augmentation matrices, respectively. A first definition of their whitening matrix is as follows:

$$G = \mathcal{A}vg_{AI}\left(\{aug_\alpha(X_i, A_i)\}_{i\in I}\right) \tag{11}$$

where $\mathcal{A}vg_{AI}(\cdot,\cdot)$ denotes the standard affine-invariant average[4] (Section 2.2), and $aug_\alpha(\cdot,\cdot)$ is the augmentation operator (Equation 6). We refer to this composition strategy as "direct average whitening" (DAW).

Whitening operations performed through Equations 7 and 11 shift all the relevant augmented matrices around the identity matrix $\mathbb{I}_m$. In other words, the affine-invariant average of all matrices enriched through DAW is $\mathbb{I}_m$. While this shift has been shown to improve performance in some contexts [43], the introduction of augmentation raises some issues. Namely, that Equation 11 computes in a single step an average value derived from both matrices in $\{X_i\}_{i\in I}$ and in $\{A_i\}_{i\in I}$. While $\{X_i\}_{i\in I}$ is composed of SPD matrices, and is therefore nicely resumed by an affine-invariant average, $\{A_i\}_{i\in I}$ is composed of matrices of attributes, and would be better represented by an Euclidean mean.

Based on this, we have proposed two different alternative scheme to compute each whitening matrix, starting with our "mirrored augmentation whitening" (MAW) composition strategy, which defines G thusly:

$$\begin{gathered} G' = \mathcal{A}vg_{AI}(\{X_i\}_{i\in I})\ ;\ A'_G = \mathcal{A}vg_E(\{A_i\}_{i\in I}) \\ G = aug_\alpha(G', A'_G) \end{gathered} \tag{12}$$

with $\mathcal{A}vg_E(\cdot)$ the standard Euclidean average.

Both DAW and MAW configurations augment their matrices prior to whitening. For the sake of comparison, we elected to test the reverse configuration. We define the resulting "whitening prior to augmentation" (WPA) thusly:

$$\begin{gathered} G' = \mathcal{A}vg_{AI}(\{X_i\}_{i\in I})\ ;\ \{X_i^*\}_{i\in I} = whit(\{X_i\}_{i\in I}, G') \\ \{M_i\}_{i\in I} = aug_\alpha(\{X_i^*\}_{i\in I}, \{A_i\}_{i\in I}) \end{gathered} \tag{13}$$

[4]As implemented in [59].

Shown in Figure 2 are subject-wise average matrices resulting from all three enrichment strategies, as obtained for each strategy's highest performing configuration (entries 1, 2 and 4 in Table 3). As such, the displayed WPA-enriched average matrix utilizes learned augmentation features, while the other two utilize handcrafted ones.

As we can see in the figure, although the DAW strategy leads to the closest augmented matrix to the identity, with no non-diagonal element greater than 10^{-6} in absolute value, the MAW and WPA strategy still leads to matrices relatively close to said identity.

As this closeness to the identity remains true for other recordings and channels[5], we may empirically conclude that all three composition strategies transport our matrices upon the same neighborhood within the SPD manifold.

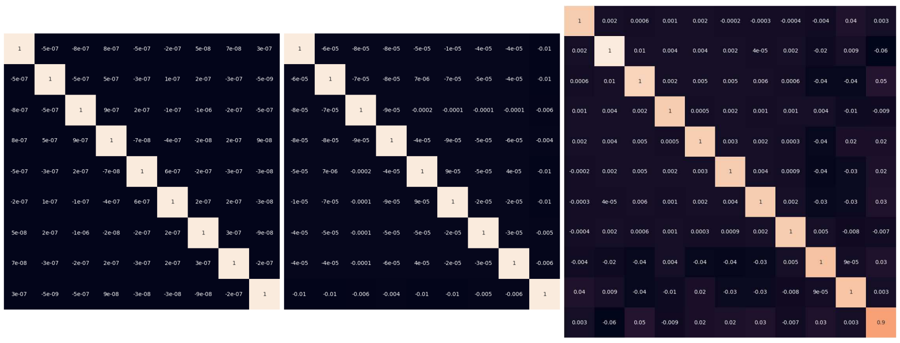

Fig. 2: Estimation of the affine-invariant average of recording-wise enriched matrices in heatmap form, computed from unfiltered signals on the MASS-SS3 recording of index 42. From left to right, these correspond to DAW, MAW and WPA enrichment, respectively.

4.1.3 Final Input Processing

Following the matrix enrichment operations, from the original covariance matrices in $SPD(n)$, we obtain matrices in $SPD(m)$ $(m \geq n)$. These are then tokenized (Section 3.3), yielding a total of 30×7 tokens of $\mathbb{R}^{d(m)}$ per epoch. In order to combine all channels into a single dimension, as seen in Figure 1, the tokens are rearranged into a single sequence of length 210, with the thirty tokens of channel 0 being followed by the thirty tokens of channel 1, etc.

As we have found that relatively small token sizes lead to worse performance[6], we linearly map our tokens (Section 3.4) onto a larger dimension $\mathbb{R}^{d(p)}$ $(p > m)$ before going forward to the intra-epoch component.

[5]As can be seen in the equivalent matrices computed for each recording and channel, available in our GitHub repository.

[6]Given a token size d and a number of attention heads h for a Transformer encoder, we found that imposing $\frac{d}{h} \geq 32$ yielded better results.

Note that the most time-consuming operation in our training pipeline is the eigenvalue decomposition necessary to compute the matrix logarithm, itself a component of our bijective tokenization (Section 3.3). We implement this operation through a singular value decomposition (SVD) algorithm. More precisely, for the purposes of backpropagation, we approximate the SVD operator's gradient using Taylor polynomials, as this computation tends to be numerically unstable otherwise [60].

In order to speed up model training, whenever the enrichment process is invariant (i.e. the augmentation factor α is non-trainable and handcrafted features are utilized, cf. Section 4.1.2.1), we tokenize our matrices prior to the start of the learning process, rather than at each training step, which results in a significant speed-up.

4.2 Model Architecture

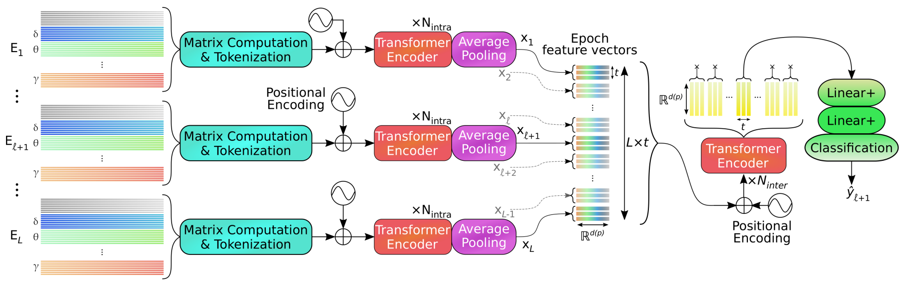

Fig. 3: Architecture of SPDTransNet, with $t = 3$ feature tokens per epoch, $L = 2\cdot\ell+1$ the length of the input epoch sequence, and $\ell + 1$ the index of the central epoch. The "Matrix Computation & Tokenization" component is further developed in Figure 1.

As presented in Figure 3, after a learnable positional encoding [61], our tokens pass trough an intra-epoch Transformer encoder, whose output sequence is then averaged into t epoch feature tokens. While only $t = 1$ (single feature token) and $t = 7$ (one feature token per input channel) have interpretations connecting the output of the average pooling layer and the input of the encoder, we tested other values of t for the sake of completeness.

The epoch-wise features are then combined into a $L \times t$ sequence, and compared through the inter-epoch Transformer encoder, after which the t tokens corresponding to the central epoch pass through two linear (i.e. fully connected) layers with dropout and ReLU activation (labeled "Linear+" in Figure 3), followed by a third and final linear layer outputting the classification vector. Finally, the training performance is ascertained through a cross-entropy loss function (with label smoothing [62] for additional regularization), complete with application of a logarithmic softmax function to the classification vector.

As such, our model follows a sequence-to-element classification scheme (for reasons explained in Section 5), but can easily be adapted to follow a sequence-to-sequence scheme instead (Section 2.1).

4.3 Model Structural Preservation

As seen in Section 3.4, weighted sums and triangular linear maps involving tokenized matrices don't cause issue when it comes to SPD structural preservation. Hence, standard fully connected / linear neural network layers are permissible, as long as the output length is triangular. Furthermore, the addition of a bias term doesn't cause any additional issue, as it boils down to a weighted sum between vector representations.

This said, our SPDTransNet model contains a number of other operations, whose structure preservation properties need to be further justified.

4.3.1 Structure-Preserving Multihead Attention (SP-MHA)

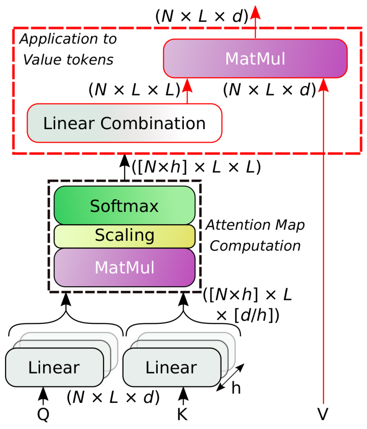

Fig. 4: Our SP-MHA component. The computation of attention maps (small-dashed black rectangles) is identical to the original MHA, whereas the application of said attention maps to the tokens within the **V**alue tensor (large-dashed red rectangle) has been modified to avoid any projection and subsequent concatenation.

Classically, Transformers utilize a component called Linear Multihead Attention (L-MHA), first introduced by Vaswani et al. [29] Unfortunately, this component calls for the concatenation of output tokens for each attention head, an operation that is not interpretable using our framework of tokenized SPD matrices.

To counteract this, in our previous work [10], we defined a structure-preserving alternative to L-MHA, called SP-MHA (as seen in Figure 4). We proved that not only did our SP-MHA preserve the structure of our input tokens, but it did so without requiring linear maps, aside for the attention map computation (the small-dashed black rectangle in the figure) - which remains identical to L-MHA.

4.3.2 Additional Guarantees

Aside from the final classification layer and the computation of attention maps (Section 4.3.1), all linear mappings of our tokens are triangular (Section 3.4). Notably, this includes the two mappings contained in the Feed-Forward (FF) component of Transformer encoders [29].

As sums and linear combinations of tokens are permissible, most other model components - i.e. positional encodings, average poolings and in-encoder layer normalizations - do not cause a loss of structure. Same for the ReLU and dropout layers in the Transformer encoders' FF components, as setting values within a token to zero won't remove the corresponding matrix from $Sym(m)$. The only issue remaining is the first Linear+ layer, which combines the t feature tokens output by the inter-epoch encoder by flattening them - hence losing the interpretability of its output as SPD matrices. However, as these layers are intended to translate the embedding within the Transformer encoders into the final classification, rather than being part of said embedding themselves, this doesn't cause issue.

Therefore, we claim structure preservation for the data going through the SPDTransNet model up to the final classification layer for $t = 1$, and up to the first Linear+ layer for $t > 1$.

5 Experimental Methodology

Due to the high degree of imbalance within sleep stage scoring datasets, along with low performance of SOA methods for some sleep stages (Section 7), we have elected to orient our methodology towards maximizing per-stage performance. As such, we use the macro-averaged (i.e. unweighted average of a binary statistic computed for each class) F1 score, or MF1, as main performance metric, as it is insensitive to class imbalance and is often used in the literature in this context.

In addition, to ensure that the influence of the least represented classes isn't eclipsed during the learning process, we rebalance our training sets through oversampling - resulting in the same number of examples for each class. This requires the assignment of a unique label to each input; as such, we cannot use a (multi-label) sequence-to-sequence classification scheme (Section 2.1). For each configuration tested, we evaluate our model on a given dataset through cross-validation, preceded by a hyperparameter research on a single fold.

In Section 7, we compare our model to a number of sequence-based State-of-the-Art approaches. However, the corresponding models do not share our input sequence definition. In order to counteract border effects and ensure that the same epochs are classified in all models within the reported results, we ignore some epochs at the beginning and end of each test set recording, as done in our previous work [9, 10]. More precisely, we ensure that the first and last 24 epochs are ignored[7]. For SPDTransNet, it means that the first and last 24 - ℓ epochs (Section 4) in each test set recording will not ever be used, even as context epochs. We refer to this as "test set clipping" in subsequent sections.

[7]The largest possible number of epochs ignored by DeepSleepNet [14] at the end of a test set recording.

Table 2: Overview of the tested datasets. Subject average age shown with standard deviation.

Dataset	Avg. age (yrs)	# epochs	% N3	% N2	% N1	% REM	% Awake
MASS-SS1	63.6 ± 5.3	51292	6.64	43.22	13.87	12.41	23.87
MASS-SS3	42.5 ± 18.9	59317	12.90	50.24	8.16	17.84	10.86
Dreem DOD-H	35.32 ± 7.51	24662	14.25	48.17	6.10	19.17	12.31

5.1 SPDTransNet Configuration

As with SleepTransformer [16], we have chosen a default input sequence length of $L = 21$ for SPDTransNet, corresponding to a context size of $\ell = 10$ (Section 4). To avoid cluttering the hyperparameter research process, we chose constant values for some low-impact hyperparameters, for instance setting all dropout rates to 0.1, and the hidden dimension of our Transformer encoders' feedforward components [29] to $d(x) = 903$ (i.e. $x = 42$, cf. Section 3.3).

For the hyperparameter research (Section 5), we utilize the Optuna tool [63], with five simultaneous runs and 50 total runs per configuration (or eight simultaneous runs if the augmentation features are learned, as this significantly lengthens the learning process - see thereafter). Through it, the token size $d(p)$ following the first triangular map (Section 4.1.3) is chosen in $\{351, 378\}$ (i.e. $p \in \{26, 27\}$), and the h parameter of each Transformer encoder (Section 4.3.1) in $\{3, 9\}$ - a restricted choice, since $\frac{d(p)}{h}$ must remain an integer whatever the chosen combination. In turn, the number of epoch feature tokens t (Section 4.2) among $\{1, 3, 5, 7, 10\}$. Other parameters, like the learning rate and augmentation factor α (Sections 3.1 and 4.1.2), are chosen from a range using log-uniform sampling.

In our testing, we have found that moderate variations in the factor α have a small impact on classification when the network is configured for augmentation through handcrafted features (Section 4.1.2.1). As such, when allowed to be learned, α remains quasi-constant. To avoid unnecessary computations at each step, in this configuration, we set α as constant, enabling us to speed up training times by pre-tokenizing our matrices (Section 4.1.3). This is not the case when utilizing learned features for augmentation, and α is kept learnable in this context.

More details can be found in our GitHub repository.

5.2 Datasets used

To best compare ourselves to the literature, we selected publicly available sleep stage scoring datasets that had a relatively large and varied number of EEG signals to choose from, since our model is based on comparison of brain activity between different regions.

The Montreal Archive of Sleep Studies (MASS) [64] is a dataset composed of five subsets (SS1 to SS5), containing a large number of common EEG signals (16 common signals in all but MASS-SS4) acquired with a common reference electrode[8]. Of these, we selected eight signals corresponding to electrodes F3, F4, C3, C4, T3, T4, O1 and

[8]All MASS-SS3 subjects use a linked-ear reference (LER), as do some MASS-SS1 subjects, the others using a computed linked-ear (CLE) reference.

O2 - i.e. frontal, central, temporal and occipital locations on both hemispheres, giving us a reasonably wide range of sampled localized cerebral activity. We selected for analysis only the subsets scored with the AASM ruleset:

MASS-SS3 is the largest and most widely used of the tested datasets, with 62 healthy subjects. With it, we use the seemingly random 31 folds generated by Seo et al. [13], with each fold having a 50/10/2 division of subjects dedicated to the training, validation and test sets respectively. This follows a leave-two-out cross-validation scheme, with the union of all test sets corresponding to the dataset in its entirety.

MASS-SS1 is made of 53 older subjects (Table 2), including 15 that had been diagnosed with mild cognitive impairment (MCI), making it harder to classify. We randomly generated 26 cross-validation folds, with a 42/9/2 split for all but one fold, said fold having a 42/8/3 split instead. As with MASS-SS3, we ensure that the union of test sets corresponds to the entire dataset.

Finally, we also tested our approach on the Dreem dataset [27], an openly available dataset composed of subjects both healthy and suffering from sleep apnea. The recordings in its **Dreem DOD-H** subset, corresponding to all healthy subjects, contain 12 EEG-derived signals, including five obtained from an EEG electrode and a reference[9]: F3, F4, C3, Fp1 and Fp2 - the latter two being pre-frontal. The remaining derivations are acquired between pairs of EEG electrodes, but we can recover the signals corresponding to O1 and O2 from them. We have elected to only utilize these seven EEG signals in this paper, as we wish to compute our matrices (and estimate functional connectivity) between signals corresponding only to localized brain activity. As with MASS-SS1, we have randomly generated our folds, this time 25 folds following a leave-one-out scheme and a 20/4/1 split.

As shown in Table 2, our training sets contain at most 59k epochs, without accounting for border effects or test set clipping (Section 5). This is a relatively small dataset for training a Transformer-based model of this size, a limitation due to the relative scarcity of publicly available sleep stage scoring datasets having a sufficient amount of EEG electrodes included. As a consequence, the cross-validation folds of each dataset are relatively small (see thereafter). Combined with a high level of inter-subject (and thus inter-recording) variability due to the biological nature of our data, fold-wise results tend to differ significantly, resulting in inflated values of standard deviation.

For the purposes of hyperparameter researches (Section 5.1), when training on MASS-SS3, we use the same randomly selected fold as our previous work [9, 10]. For each of the other two datasets, a new fold was randomly selected for this purpose.

6 Ablation Study

In order to better understand the impact of SPDTransNet's components and configurations the model's performance, we evaluated and compared a number of model variations. These experiments utilize exclusively the MASS-SS3 dataset (Section 5.2), due to its size and widespread use.

For all tables introduced in this section and Section 7, the term after the $\pm$ symbol corresponds to the standard deviation between the results obtained from each relevant

[9]Located behind the ear in the opposite hemisphere.

Table 3: Comparison of enrichment configurations for SPDTransNet, using MASS-SS3

Augmentation	Whitening	MF1	N3 F1	N2 F1	N1 F1
Handcrafted	DAW	79.66 ± 3.68	78.44 ± 10.93	87.77 ± 2.42	56.47 ± 8.74
Handcrafted	MAW	81.24 ± 3.29	78.81 ± 11.17	89.36 ± 2.36	**60.50** ± 6.18
Handcrafted	WPA	80.97 ± 3.06	79.79 ± 10.27	**89.73** ± 2.04	59.21 ± 7.06
Learned	WPA	**81.64** ± 2.88	**80.15** ± 10.46	89.54 ± 2.46	60.29 ± 5.52

cross-validation fold. As stated in Section 5.2, this value is somewhat inflated, and doesn't exactly correspond to an uncertainty value. In addition, only the class-wise F1 scores for sleep stages N1 to N3 are displayed, as those are particularly relevant - with N2 being the easiest to classify, showcasing peak class-wise performance, and N1 and/or N3 being the worst performing. Full results are available on our GitHub repository.

6.1 Enrichment configurations

As presented in Section 4.1.2, prior to tokenization, our SPD matrices are enriched through a combination of augmentation and whitening, adding additional information to said matrices while ensuring their comparability by transporting matrices from different recordings and channels onto the same neighborhood.

As we can see in Table 3, when using handcrafted augmentation features, the MAW strategy is the best performing, though WPA isn't significantly behind. By contrast, the DAW strategy performed poorly, particularly in the N1 sleep stage. This confirms that the single-step averaging of combined SPD covariance matrices and Euclidean augmentation matrices would lead to data degradation, hurting the model's performance.

Due to the learning process shifting the weights used to compute learned augmentation features at every learning step, utilizing the DAW strategy in a learned augmentation configuration would add a layer of complexity to the already inflated cost of this approach (Section 5.1). Given this fact, compounded with the low performance differential between MAW and WPA, we have elected to only test the learned augmentation configuration with the WPA strategy.

As seen in the table, this results in a performance improvement of .4 points - relatively minimal when compared to the standard deviation of both variants. Since the above experiments do not alter the main, post-tokenization structure-preserving model, we have elected to utilize the handcrafted augmentation and MAW strategy as our baseline enrichment configuration in Section 6.2.

6.2 Further Ablation Experiments

Starting with the baseline configuration of SPDTransNet (Section 6.1), we further analyze our model through punctual, non-cumulative modifications to its structure and to our methodology, with results presented in Table 4.

Though the different composition strategies for augmentation and whitening were investigated in Section 6.1, their individual contribution to classification was not. In

Table 4: Analysis of various modifications to the SPDTransNet baseline, using MASS-SS3

Configuration	MF1	N3 F1	N2 F1	N1 F1
Baseline	**81.24** $\pm$ 3.29	**78.81** $\pm$ 11.17	**89.36** $\pm$ 2.36	**60.50** $\pm$ 6.18
Zero-valued augmentation	80.58 $\pm$ 3.86	78.45 $\pm$ 11.52	89.03 $\pm$ 2.38	59.09 $\pm$ 7.28
Global covariance whitening	79.46 $\pm$ 3.06	77.35 $\pm$ 11.59	88.56 $\pm$ 2.70	57.40 $\pm$ 5.78
Classic MHA	80.82 $\pm$ 3.40	76.96 $\pm$ 12.79	88.92 $\pm$ 2.09	60.16 $\pm$ 7.20
Input length $L = 13$	81.06 $\pm$ 3.49	78.79 $\pm$ 11.13	88.71 $\pm$ 2.31	60.39 $\pm$ 6.77
Input length $L = 29$	80.45 $\pm$ 3.87	77.70 $\pm$ 11.83	89.26 $\pm$ 2.36	59.57 $\pm$ 5.86

the case of handcrafted augmentation features, we know that variations in the factor α have a negligible effect on said performance (Section 5.1), putting into question the augmentation's influence on our classification's performance. To learn more, we modified the baseline configuration to augment our matrices with vectors of zeros. As seen in the table, this **zero-valued augmentation** leads to a moderate performance drop, though one slightly larger than when using the WPA strategy (Table 3). Hence, the addition of handcrafted features through augmentation *does* result in an increase in performance.

In an earlier publication [9], we computed the MAW whitening matrices G' (Equation 12) by taking the **global covariance matrix** of each recording and channel. This is equivalent to the Euclidean average of said recording's one-second matrices, and therefore ill-suited to an analysis based on Riemannian structural preservation, as evidenced by the performance drop seen in Table 4 - hence our decision to utilize the affine-invariant average instead.

A major component of SPDTransNet is the replacement of our Transformer encoders' standard L-MHA component with our own SP-MHA (Section 4.3.1). To better understand its contribution to our classification accuracy, we tested our model's performance when equipped with the more **classic MHA**. As seen in Table 4, this change leads to a slight drop in performance. Hence, though our lead when using SP-MHA is not significant, it does mean that this component does not underperform when compared to L-MHA. As such, the added interpretability afforded by our model's structure-preserving operations throughout the analysis (Section 4.3.2) comes at no additional performance cost.

Finally, we have elected to study the input sequence length L (Section 4) on SPDTransNet. As stated in Section 5.1, our initial choice of context size (i.e. $\ell = 10$) is semi-arbitrary. Hence, we implement a **change in input length**, testing both $L = 13$ (i.e. $\ell = 6$) and $L = 29$ (i.e. $\ell = 14$). However, as seen in the table, both alterations yield a reduction in performance, though the negative impact of increasing contextual information seems to outweigh that of reducing it. The baseline value of $\ell = 10$ therefore seems to be a good compromise.

7 Comparison to the State-of-the-Art

To better contextualize our approach's capabilities, we test a number of previously published approaches (Section 2.1) - more specifically, **DeepSleepNet** by Supratak et al. [14], **IITNet** by Seo et al. [13], and the approach taken by Dequidt et al. [26], labeled **SleepVGG16** in this paper.

We compare them to the **SPDTransNet baseline**, as defined in Section 6.2, as well as the variant of our model trained using learned augmentation features (Section 6.1), labeled **SPDTransNet+** in this section.

To ensure a fair comparison, we re-trained these models with our own methodology whenever possible, while keeping as much of the original authors' published implementation as possible. Due to the absence of published hyperparameter research guidelines, we only use their published hyperparameters (or, if absent, those present in their published code) in our experiments.

Furthermore, DeepSleepNet already implements a rebalancing through oversampling in their pretrained epoch-wise component, before including it to an end-to-end sequence-to-sequence model. We have not modified this classification scheme.

Note that since we are optimizing for class-wise performance through the MF1 score when these approaches for the most part optimized the global (unweighted) accuracy score, our obtained results tend to be lower than those the original authors published. This is true even for SleepVGG16, as even though the original approach also optimized through the MF1 score, it didn't include our test set clipping (Section 5). We have found that this clipping tended to reduce the test set performance measures, presumably because epochs in the beginning and end of recordings (i.e. mostly corresponding to the Awake stage) are less ambiguous than other instances of the same stage.

7.1 Learning From Scratch (LFS)

For each dataset presented in Section 5.2, we trained every considered model - i.e. both the baseline SPDTransNet model and the models presented in Section 7. This allows us to directly compare our model's performance with State-of-the-Art approaches when using our class-wise performance optimization methodology. The obtained results are displayed in Table 5.

As seen in the table, the size and relative homogeneity of MASS-SS3 (Section 5.2) translated into the best performance overall for each considered model. Interestingly, the higher degree of inter-subject variability in MASS-SS1 meant that even though this set has more than double the epoch count of Dreem DOD-H, the latter yielded better overall results than MASS-SS1, though the single-recording test sets led to a greater standard deviation in the aggregated results.

Model-wise, the baseline SPDTransNet performs similarly to or slightly better than SleepVGG16, with the learned augmentation variant outperforming both on SS3. Both SPDTransNet and SleepVGG16 outperform DeepSleepNet and IITNet, though with a lead of around 3 points for SS3, 4 points for Dreem DOD-H and 6+ points for SS1.

This resiliency to increased inter-subject variability that both top models exhibit is most likely due to their common multi-signal approach, rather than the reliance on single-channel EEG analysis favored by DeepSleepNet and IITNet.

Table 5: Learning from scratch on considered datasets

Dataset	Model	MF1	N3 F1	N2 F1	N1 F1
SS3	DeepSleepNet	78.14 ± 4.12	80.38 ± 9.35	89.25 ± 3.12	53.52 ± 8.24
	IITNet	78.48 ± 3.15	81.97 ± 8.91	88.15 ± 2.84	56.02 ± 6.54
	SleepVGG16	81.23 ± 2.56	**82.02** ± 8.76	**90.57** ± 2.65	58.29 ± 5.01
	SPDTransNet	81.24 ± 3.29	78.81 ± 11.17	89.36 ± 2.36	**60.50** ± 6.18
	SPDTransNet+	**81.64** ± 2.88	80.15 ± 10.46	89.54 ± 2.46	60.29 ± 5.52
SS1	DeepSleepNet	68.49 ± 6.12	53.47 ± 21.59	83.74 ± 4.30	50.81 ± 8.19
	IITNet	70.82 ± 6.09	56.40 ± 21.76	78.79 ± 6.20	52.29 ± 6.47
	SleepVGG16	77.31 ± 5.46	**66.75** ± 20.76	**86.51** ± 3.82	62.59 ± 5.34
	SPDTransNet	**77.75** ± 5.41	63.08 ± 20.96	85.28 ± 4.55	**63.29** ± 6.02
DOD-H	DeepSleepNet	73.07 ± 13.66	75.98 ± 23.58	84.50 ± 15.20	48.40 ± 16.86
	IITNet	73.55 ± 9.36	76.67 ± 24.21	86.32 ± 6.27	49.68 ± 12.53
	SleepVGG16	75.80 ± 10.60	76.87 ± 24.15	85.99 ± 11.72	51.30 ± 12.00
	SPDTransNet	**77.48** ± 8.74	**77.75** ± 24.04	**87.40** ± 5.42	**56.99** ± 11.93

7.2 Direct Transfer (DT)

To be used in a clinical setting, a sleep stage scoring model would need to retain a good level of performance when utilized with newly acquired data. As such, we have elected to compare the ability of all considered models to adapt to other datasets. These experiments are not undertaken using the Dreem DOD-H dataset, as it differs from both SS1 and SS3 in the number and nature of the considered electrodes.

For each model, we utilize the LFS weights obtained on each fold of MASS-SS3 (Section 5), and test on all recordings of the MASS-SS1 dataset. The results are displayed in Table 6.

One first observation is that for each model, the standard deviations tend to be lower than their LFS equivalent, both on SS1 and SS3, particularly for our model and SleepVGG16. As stated in Section 6, the high standard deviations in Table 5 are a consequence of our cross-validation strategy. Since test sets between folds are relatively small and non-overlapping, inter-test-set variability tends to be high, inflating the resulting standard deviation when aggregating results. By using a unified test set for each fold, we remove this factor entirely, yielding lower standard deviations than for the LFS experiments, and increasing our confidence in the significance of the obtained results.

The most striking result, however, is the dramatic reduction in performance between LFS and DT results for both single-channel models, with a loss of 15+ points on DeepSleepNet and IITNet compared to their LFS performance on MASS-SS1 - not to mention their relatively high standard deviations, indicative of inter-fold instability.

By contrast, SleepVGG16 lost slightly less than 3 points, and the baseline SPDTransNet, less than one point. Hence, it would seem that our approach through functional connectivity does outperform SleepVGG16 in resiliency when in a multi-dataset context.

Interestingly, the SPDTransNet+ configuration outperforms even our baseline LFS results on SS1. This would indicate that the feature extraction through 1D CNN strategy utilized by DeepSleepNet and IITNet isn't the main source of their lack of flexibility, as SPDTransNet+ was able to utilize the epoch-wise feature extraction

Table 6: Direct transfer, i.e. testing models trained on MASS-SS3 with recordings from another dataset

Model	MF1	N3 F1	N2 F1	N1 F1
DeepSleepNet	52.41 ± 3.15	55.43 ± 5.95	61.17 ± 6.44	32.11 ± 3.75
IITNet	55.27 ± 4.08	61.05 ± 4.39	69.82 ± 6.71	36.54 ± 4.42
SleepVGG16	74.68 ± 1.82	70.82 ± 3.80	**86.72** ± 0.62	55.79 ± 2.36
SPDTransNet	76.99 ± 0.94	67.59 ± 2.08	86.52 ± 0.54	57.92 ± 1.85
SPDTransNet+	**78.23** ± 0.62	**68.71** ± 2.19	86.43 ± 0.91	**59.68** ± 1.72

Table 7: Finetuning weights trained on MASS-SS3 using another dataset

Model	MF1	N3 F1	N2 F1	N1 F1
DeepSleepNet	67.65 ± 7.47	55.15 ± 22.58	81.03 ± 5.42	47.48 ± 7.03
IITNet	70.95 ± 5.85	56.51 ± 21.63	80.46 ± 6.46	53.85 ± 5.90
SleepVGG16	78.90 ± 4.61	**68.88** ± 18.53	**86.41** ± 3.91	64.15 ± 6.19
SPDTransNet	78.99 ± 4.45	64.65 ± 18.69	86.19 ± 5.09	64.60 ± 5.46
SPDTransNet+	**79.41** ± 4.40	65.90 ± 18.21	85.90 ± 5.02	**65.16** ± 6.24

submodel of IITNet (Section 4.1.2.1) to great extent. Rather, it would seem that said inflexibility is once again due to their monosignal nature.

7.3 Finetuning (FT)

In addition to DT (Section 7.2), and for the same reasons, we tested the considered models' ability to improve classification through transfer learning, loading weights trained on MASS-SS3 (Section 7.1) and finetuning them by re-training on each fold of the other considered datasets. For the MASS-SS3 weights, we selected those trained using the fold used in hyperparameter researches (Section 5.2).

As seen in Table 7, in this configuration, SPDTransNet is able to improve its classification performance when compared to learning from scratch on MASS-SS1, with SleepVGG16 not far behind, and the SPDTransNet+ variant once again outperforming both. However, the same cannot be said for DeepSleepNet and IITNet, whose performance was comparable to the one obtained in the LFS configuration - meaning that prior learning on SS3 had the same impact as random weight initialization when it comes to learn on SS1.

Combined with the results from Section 7.2, this tells us that differences between EEG sleep staging datasets can be large enough to make non-LFS learning strategies irrelevant, particularly for models trained on few EEG signals. However, multi-signal models, such as our SPDTransNet network and SleepVGG16, have proven to be robust to such changes, allowing it them retain decent performance in DT and even improve in FT. Additionally, SPDTransNet's learned augmentation variant has proved particularly well-performing across the board, with the handcrafted augmentation baseline performing on par with the SleepVGG16 model, and outperforming it when exposed to an unknown dataset.

8 Conclusion

In this paper, we have explored the capabilities of our SPDTransNet model, a Transformer-based model designed for the structure-preserving analysis of timeseries of Symmetric Positive Definite covariance matrices, in the classification of sleep stages from a collection of electroencephalographic signals.

Beyond once again proving the relevance of an analysis through functional connectivity for this task, we have showcased our model's flexibility in multi-dataset environment. In particular, we outperformed an otherwise equivalent model when classifying signals from an unknown dataset, with minimal loss of performance compared to training on said dataset.

Furthermore, we have expanded upon our signal processing pipeline, enriching our covariance matrices through the inclusion of learned signal-specific features, further improving both our model's baseline performance and flexibility.

Supplementary Material. Specifics regarding some portions of this paper have been included as Supplementary Material, and are located at the end of this document.

Statements and Declarations

Competing Interests. The authors declare that they have no conflict of interest.

Funding Information. This work was granted access to computing resources of IDRIS and CRIANN (see Acknoledgements). The first author is supported by the French National Research Agency (ANR) and Region Normandie under grant HAISCoDe.

Author contribution. All authors contributed to the study conception and design. Experiments were performed by Mathieu Seraphim, under the supervision of Alexis Lechervy, Luc Brun and Olivier Etard. The first draft of the manuscript was written by Mathieu Seraphim and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data Availability. Our code is publicly available in the following repository: github.com/MathieuSeraphim/SPDTransNet_plus. The Montreal Archive of Sleep Studies (MASS) dataset is available upon request, pending approval from the Ethics Review Board (ERB) of the MASS host [64]. The Dreem DOD-H dataset is freely available online, and downloading instructions can be found in the authors' GitHub repository [27].

Research Involving Human and /or Animals. Both utilized datasets are composed of anonymized recordings of human biological signals. The MASS recordings were anonymized and distributed in accordance with guidelines from the MASS research team's ethics board [64]. The Dreem DOD-H dataset was compiled and made publicly available with approval from the Committees of Protection of Persons (CPP), and declared and carried out in accordance with French law [27]. No further biological data was utilized in the production of this paper.

Informed Consent. Not applicable (see above).

References

[1] Sun, R., Ma, T., Liu, S., Sathye, M.: Improved covariance matrix estimation for portfolio risk measurement: A review. Journal of Risk and Financial Management **12**(1) (2019) https://doi.org/10.3390/jrfm12010048

[2] Nguyen, X.S., Brun, L., Lézoray, O., Bougleux, S.: A neural network based on spd manifold learning for skeleton-based hand gesture recognition. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 12028–12037 (2019)

[3] Bhatia, R.: Positive Definite Matrices. Princeton University Press, ??? (2007). http://www.jstor.org/stable/j.ctt7rxv2 Accessed 2024-01-12

[4] Pennec, X.: Manifold-valued image processing with SPD matrices. In: Riemannian Geometric Statistics in Medical Image Analysis, pp. 75–134. Elsevier, ??? (2020). https://doi.org/10.1016/B978-0-12-814725-2.00010-8 . https://inria.hal.science/hal-02341958

[5] Eickhoff, S.B., Müller, V.I.: Functional connectivity. In: Toga, A.W. (ed.) Brain Mapping, pp. 187–201. Academic Press, Waltham (2015)

[6] You, K., Park, H.-J.: Re-visiting riemannian geometry of symmetric positive definite matrices for the analysis of functional connectivity. NeuroImage **225**, 117464 (2021) https://doi.org/10.1016/j.neuroimage.2020.117464

[7] Yger, F., Berar, M., Lotte, F.: Riemannian approaches in brain-computer interfaces: A review. IEEE Transactions on Neural Systems and Rehabilitation Engineering **25**(10), 1753–1762 (2017)

[8] Bouchard, M., Lina, J.-M., Gaudreault, P.-O., Dubé, J., Gosselin, N., Carrier, J.: EEG connectivity across sleep cycles and age. Sleep **43**(3) (2019)

[9] Seraphim, M., Dequidt, P., Lechervy, A., Yger, F., Brun, L., Etard, O.: Temporal sequences of eeg covariance matrices for automated sleep stage scoring with attention mechanisms. In: Tsapatsoulis, N., Lanitis, A., Pattichis, M., Pattichis, C., Kyrkou, C., Kyriacou, E., Theodosiou, Z., Panayides, A. (eds.) Computer Analysis of Images and Patterns, pp. 67–76. Springer, Cham (2023)

[10] Seraphim, M., Lechervy, A., Yger, F., Brun, L., Etard, O.: Structure-Preserving Transformers for Sequences of SPD Matrices (2023)

[11] Berry, R.B., Brooks, R., Gamaldo, C., Harding, S.M., Lloyd, R.M., Quan, S.F., Troester, M.T., Vaughn, B.V.: Aasm scoring manual updates for 2017 (version 2.4). Journal of Clinical Sleep Medicine **13**(05), 665–666 (2017) https://doi.org/10.5664/jcsm.6576 https://jcsm.aasm.org/doi/pdf/10.5664/jcsm.6576

[12] Chambon, S., Galtier, M., Arnal, P.J., Wainrib, G., Gramfort, A.: A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series. IEEE Transactions on Neural Systems and Rehabilitation Engineering **26**(4), 17683810 (2018)

[13] Seo, H., Back, S., Lee, S., Park, D., Kim, T., Lee, K.: Intra- and inter-epoch temporal context network (iitnet) using sub-epoch features for automatic sleep scoring on raw single-channel eeg. Biomedical Signal Processing and Control **61**, 102037 (2020)

[14] Supratak, A., Dong, H., Wu, C., Guo, Y.: Deepsleepnet: a model for automatic sleep stage scoring based on raw single-channel eeg. IEEE Transactions on Neural Systems and Rehabilitation Engineering **25**(11), 1998–2008 (2017)

[15] Phan, H., Andreotti, F., Cooray, N., Chén, O.Y., De Vos, M.: Seqsleepnet: End-to-end hierarchical recurrent neural network for sequence-to-sequence automatic sleep staging. IEEE Transactions on Neural Systems and Rehabilitation Engineering **27**(3), 400–410 (2019)

[16] Phan, H., Mikkelsen, K., Chén, O.Y., Koch, P., Mertins, A., De Vos, M.: Sleeptransformer: Automatic sleep staging with interpretability and uncertainty quantification. IEEE Transactions on Biomedical Engineering **69**(8), 2456–2467 (2022)

[17] Phan, H., Chén, O.Y., Tran, M.C., Koch, P., Mertins, A., De Vos, M.: Xsleepnet: Multi-view sequential model for automatic sleep staging. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(9), 5903–5915 (2022)

[18] Phan, H., Lorenzen, K.P., Heremans, E., Chén, O.Y., Tran, M.C., Koch, P., Mertins, A., Baumert, M., Mikkelsen, K.B., De Vos, M.: L-seqsleepnet: Whole-cycle long sequence modelling for automatic sleep staging. IEEE Journal of Biomedical and Health Informatics, 1–10 (2023) https://doi.org/10.1109/JBHI.2023.3303197

[19] Phan, H., Mikkelsen, K.: Automatic sleep staging of eeg signals: recent development, challenges, and future directions. Physiological Measurement **43**(4), 04–01 (2022) https://doi.org/10.1088/1361-6579/ac6049

[20] Phan, H., Andreotti, F., Cooray, N., Chén, O., Vos, M.: Joint classification and prediction cnn framework for automatic sleep stage classification. IEEE Transactions on Biomedical Engineering **66**, 1285–1296 (2019)

[21] Perslev, M., Jensen, M., Darkner, S., Jennum, P.J.r., Igel, C.: U-time: A fully convolutional network for time series segmentation applied to sleep staging. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc., ??? (2019). https://proceedings.neurips.cc/paper_files/paper/2019/

file/57bafb2c2dfeefba931bb03a835b1fa9-Paper.pdf

[22] Perslev, M., Darkner, S., Kempfner, L., Nikolic, M., Jennum, P.J., Igel, C.: U-sleep: resilient high-frequency sleep staging. npj Digital Medicine **4**(1), 72 (2021) https://doi.org/10.1038/s41746-021-00440-5

[23] Jia, Z., Lin, Y., Wang, J., Wang, X., Xie, P., Zhang, Y.: Salientsleepnet: Multimodal salient wave detection network for sleep staging, pp. 2614–2620 (2021). https://doi.org/10.24963/ijcai.2021/360

[24] Ronneberger, O., P.Fischer, Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI). LNCS, vol. 9351, pp. 234–241. Springer, ??? (2015). (available on arXiv:1505.04597 [cs.CV]). http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a

[25] Vilamala, A., Madsen, K.H., Hansen, L.K.: Deep convolutional neural networks for interpretable analysis of eeg sleep stage scoring. In: 2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6 (2017). IEEE

[26] Dequidt, P., Seraphim, M., Lechervy, A., Gaez, I.I., Brun, L., Etard, O.: Automatic sleep stage classification on eeg signals using time-frequency representation. In: Juarez, J.M., Marcos, M., Stiglic, G., Tucker, A. (eds.) Artificial Intelligence in Medicine, pp. 250–259. Springer, Cham (2023)

[27] Guillot, A., Sauvet, F., During, E.H., Thorey, V.: Dreem open datasets: Multi-scored sleep datasets to compare human and automated sleep staging. IEEE Transactions on Neural Systems and Rehabilitation Engineering **28**(9), 1955–1965 (2020) https://doi.org/10.1109/TNSRE.2020.3011181

[28] Guillot, A., Thorey, V.: Robustsleepnet: Transfer learning for automated sleep staging at scale. IEEE Transactions on Neural Systems and Rehabilitation Engineering **29**, 1441–1451 (2021) https://doi.org/10.1109/TNSRE.2021.3098968

[29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., ??? (2017)

[30] Qu, W., Wang, Z., Hong, H., Chi, Z., Feng, D.D., Grunstein, R., Gordon, C.: A residual based attention model for eeg based sleep staging. IEEE journal of biomedical and health informatics **24**(10), 2833–2843 (2020)

[31] Zhu, T., Luo, W., Yu, F.: Convolution-and Attention-Based Neural Network for Automated Sleep Stage Classification. Int J Environ Res Public Health **17**(11)

(2020)

[32] Eldele, E., Chen, Z., Liu, C., Wu, M., Kwoh, C.-K., Li, X., Guan, C.: An attention-based deep learning approach for sleep stage classification with single-channel eeg. IEEE Transactions on Neural Systems and Rehabilitation Engineering **29**, 809–818 (2021) https://doi.org/10.1109/TNSRE.2021.3076234

[33] Pradeepkumar, J., Anandakumar, M., Kugathasan, V., Suntharalingham, D., Kappel, S.L., Silva, A.C.D., Edussooriya, C.U.S.: Towards interpretable sleep stage classification using cross-modal transformers. ArXiv **abs/2208.06991** (2022)

[34] Dai, Y., Li, X., Liang, S., Wang, L., Duan, Q., Yang, H., Zhang, C., Chen, X., Li, L., Li, X., Liao, X.: Multichannelsleepnet: A transformer-based model for automatic sleep stage classification with psg. IEEE Journal of Biomedical and Health Informatics **27**(9), 4204–4215 (2023) https://doi.org/10.1109/JBHI.2023.3284160

[35] Kemp, B., Zwinderman, A.H., Tuk, B., Kamphuisen, H.A.C., Oberye, J.J.L.: Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg. IEEE Transactions on Biomedical Engineering **47**(9), 1185–1194 (2000) https://doi.org/10.1109/10.867928

[36] Zhang, G.-Q., Cui, L., Mueller, R., Tao, S., Kim, M., Rueschman, M., Mariani, S., Mobley, D., Redline, S.: The national sleep research resource: towards a sleep data commons. J Am Med Inform Assoc **25**(10), 1351–1358 (2018)

[37] Quan, S.F., Howard, B.V., Iber, C., Kiley, J.P., Nieto, F.J., O'Connor, G.T., Rapoport, D.M., Redline, S., Robbins, J., Samet, J.M., Wahl, P.W.: The sleep heart health study: design, rationale, and methods. Sleep **20**(12), 1077–1085 (1997)

[38] Jia, Z., Lin, Y., Wang, J., Zhou, R., Ning, X., He, Y., Zhao, Y.: Graphsleepnet: Adaptive spatial-temporal graph convolutional networks for sleep stage classification. In: IJCAI, pp. 1324–1330 (2020)

[39] Jia, Z., Lin, Y., Wang, J., Ning, X., He, Y., Zhou, R., Zhou, Y., Lehman, L.-w.H.: Multi-view spatial-temporal graph convolutional networks with domain generalization for sleep stage classification. IEEE Transactions on Neural Systems and Rehabilitation Engineering **29**, 1977–1986 (2021)

[40] Einizade, A., Nasiri, S., Sardouie, S.H., Clifford, G.D.: Productgraphsleepnet: Sleep staging using product spatio-temporal graph learning with attentive temporal aggregation. Neural Networks **164**, 667–680 (2023) https://doi.org/10.1016/j.neunet.2023.05.016

[41] Arsigny, V., Fillard, P., Pennec, X., Ayache, N.: Log-euclidean metrics for fast

and simple calculus on diffusion tensors. Magnetic Resonance in Medicine **56**(2), 411–421 (2006)

[42] Pennec, X., Fillard, P., Ayache, N.: A riemannian framework for tensor computing. International Journal of Computer Vision **66**(1), 41–66 (2006)

[43] Barachant, A., Bonnet, S., Congedo, M., Jutten, C.: Classification of covariance matrices using a riemannian-based kernel for bci applications. Neurocomputing **112**, 172–178 (2013) https://doi.org/10.1016/j.neucom.2012.12.039 . Advances in artificial neural networks, machine learning, and computational intelligence

[44] Yger, F., Sugiyama, M.: Supervised LogEuclidean Metric Learning for Symmetric Positive Definite Matrices. arXiv (2015)

[45] Huang, Z., Van Gool, L.: A riemannian network for spd matrix learning. In: Association for the Advancement of Artificial Intelligence (AAAI) (2017)

[46] Chakraborty, R., Bouza, J., Manton, J.H., Vemuri, B.C.: Manifoldnet: A deep neural network for manifold-valued data with applications. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(2), 799–810 (2022) https://doi.org/10.1109/TPAMI.2020.3003846

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

[48] Huang, Z., Wang, R., Shan, S., Li, X., Chen, X.: Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 720–729. PMLR, Lille, France (2015). https://proceedings.mlr.press/v37/huanga15.html

[49] Tang, Y., Chen, D., Wu, J., Tu, W., Monaghan, J.J.M., Sowman, P., Mcalpine, D.: Functional connectivity learning via siamese-based spd matrix representation of brain imaging data. Neural Networks **163**, 272–285 (2023) https://doi.org/10.1016/j.neunet.2023.04.004

[50] Peng, Z., Li, H., Zhao, D., Pan, C.: Reducing the dimensionality of spd matrices with neural networks in bci. Mathematics **11**(7) (2023) https://doi.org/10.3390/math11071570

[51] Lu, J., Tian, Y., Zhang, Y., Ge, J., Sheng, Q.Z., Zheng, X.: LGL-BCI: A Lightweight Geometric Learning Framework for Motor Imagery-Based Brain-Computer Interfaces (2023)

[52] Ouahidi, Y.E., Gripon, V., Pasdeloup, B., Bouallegue, G., Farrugia, N., Lioi, G.: A Strong and Simple Deep Learning Baseline for BCI MI Decoding (2024)

[53] Abibullaev, B., Keutayeva, A., Zollanvari, A.: Deep learning in eeg-based bcis: A comprehensive review of transformer models, advantages, challenges, and applications. IEEE Access **11**, 127271–127301 (2023) https://doi.org/10.1109/ACCESS.2023.3329678

[54] Konstantinidis, D., Papastratis, I., Dimitropoulos, K., Daras, P.: Multi-manifold attention for vision transformers. IEEE Access **11**, 123433–123444 (2023) https://doi.org/10.1109/ACCESS.2023.3329952

[55] Dong, Z., Wang, Q., Zhu, P.: Multi-head second-order pooling for graph transformer networks. Pattern Recognition Letters **167**, 53–59 (2023) https://doi.org/10.1016/j.patrec.2023.01.017

[56] He, L., Dong, Y., Wang, Y., Tao, D., Lin, Z.: Gauge equivariant transformer. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 27331–27343. Curran Associates, Inc., ??? (2021). https://proceedings.neurips.cc/paper_files/paper/2021/file/e57c6b956a6521b28495f2886ca0977a-Paper.pdf

[57] Li, Z., TANG, X., Xu, Z., Wang, X., Yu, H., Chen, M., Wei, X.: Geodesic self-attention for 3d point clouds. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022). https://openreview.net/forum?id=2ndfW2bw4mi

[58] Kratsios, A., Zamanlooy, B., Liu, T., Dokmanić, I.: Universal approximation under constraints is possible with transformers. In: International Conference on Learning Representations (2022). https://openreview.net/forum?id=JGO8CvG5S9

[59] Barachant, A., Barthélemy, Q., King, J.-R., Gramfort, A., Chevallier, S., Rodrigues, P.L.C., Olivetti, E., Goncharenko, V., Berg, G.W., Reguig, G., Lebeurrier, A., Bjäreholt, E., Yamamoto, M.S., Clisson, P., Corsi, M.-C.: pyRiemann/pyRiemann: V0.5. https://doi.org/10.5281/zenodo.8059038 . https://doi.org/10.5281/zenodo.8059038

[60] Song, Y., Sebe, N., Wang, W.: Why approximate matrix square root outperforms accurate svd in global covariance pooling? In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1095–1103 (2021). https://doi.org/10.1109/ICCV48922.2021.00115

[61] Wang, G., Lu, Y., Cui, L., Lv, T., Florencio, D., Zhang, C.: A simple yet effective learnable positional encoding method for improving document transformer model. In: He, Y., Ji, H., Li, S., Liu, Y., Chang, C.-H. (eds.) Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022, pp. 453–463. Association for

Computational Linguistics, Online only (2022). https://aclanthology.org/2022.findings-aacl.42

[62] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. (2016). https://doi.org/10.1109/CVPR.2016.308

[63] Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2623–2631 (2019)

[64] O'reilly, C., Gosselin, N., Carrier, J., Nielsen, T.: Montreal archive of sleep studies: an open-access resource for instrument benchmarking and exploratory research. Journal of sleep research **23**(6), 628–635 (2014)

Supplementary Material

1 Proof of SPD Preservation through Augmentation

Let $X \in SPD(n)$, $A \in \mathbb{R}^{n \times k}$ and $x \in \mathbb{R}^{n+k}$ be arbitrary elements of their respective set. In addition, let $x_1 \in \mathbb{R}^n$ and $x_2 \in \mathbb{R}^k$ be defined so that:

$$x^T = (x_1^T,\ x_2^T)$$

In other words, x is the concatenation of x_1 and x_2.

Let $X' = aug_\alpha(X, A) \in \mathbb{R}^{m \times m}$, with $m = n + k$, be the result of the augmentation of X by A (Section 3.1 of the paper). Since we make no assumption on the value taken by A, we set $\alpha = 1$, without loss of generality[10].

By definition,

$$X' \in SPD(m) \Leftrightarrow \forall x \neq 0_m,\ x^T \cdot X' \cdot x > 0$$

with $0_m \in \mathbb{R}^m$ a vector of zeros.

$$\begin{aligned}
x^T \cdot X' \cdot x &= (x_1^T,\ x_2^T) \cdot \left(\begin{array}{c|c} X + A \cdot A^T & A \\ \hline A^T & \mathbb{I}_k \end{array}\right) \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
&= (x_1^T \cdot X + x_1^T \cdot A \cdot A^T + x_2^T \cdot A^T,\ x_1^T \cdot A + x_2^T \cdot \mathbb{I}_k) \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
&= x_1^T \cdot X \cdot x_1 + x_1^T \cdot A \cdot A^T \cdot x_1 + x_2^T \cdot A^T \cdot x_1 + x_1^T \cdot A \cdot x_2 + x_2^T \cdot \mathbb{I}_k \cdot x_2 \\
&= x_1^T \cdot X \cdot x_1 + (x_1^T \cdot A) \cdot (x_1^T \cdot A)^T + (x_1^T \cdot A) \cdot x_2 + x_2^T \cdot (x_1^T \cdot A)^T + x_2^T \cdot x_2
\end{aligned}$$

Therefore:

$$x^T \cdot X' \cdot x = x_1^T \cdot X \cdot x_1 + (x_1^T \cdot A + x_2^T) \cdot (x_1^T \cdot A + x_2^T)^T$$

with $x_1^T \cdot A$ and x_2^T line vectors of length k.

$\rightarrow (x_1^T \cdot A + x_2^T) \cdot (x_1^T \cdot A + x_2^T)^T \geq 0$

However:

- If $x_1 \neq 0_n$, $x_1^T \cdot X \cdot x_1 > 0$ (by definition, as $X \in SPD(n)$)
- If $x_1 = 0_n$ but $x_2 \neq 0_k$, $(x_1^T \cdot A + x_2^T) \cdot (x_1^T \cdot A + x_2^T)^T = x_2^T \cdot x_2 > 0$

[10] If $\alpha \notin \{0, 1\}$, replace A with $A' = \frac{1}{\alpha} \cdot A$. If $\alpha = 0$, replace it with $0_{n \times k}$.

In conclusion, whichever the values of x_1 and x_2, and therefore for any arbitrary vector $x \in \mathbb{R}^m$, we have:

$$x^T \cdot X' \cdot x = 0 \Leftrightarrow x = 0_m$$

Which proves that $X' = aug_1(X, A)$ is SPD, and more generally, that the augmentation operation presented in Equation 6 of the paper preserves the SPD nature of its input X.

2 The Signal-Wise Feature Extraction Submodel

As stated in Section 4.1.2.1 of the paper, we utilize the epoch-wise feature extraction CNN utilized by Seo et al. in IITNet[13]. It is derived from the ResNet-50 architecture, modified to analyze 1D signals, and which outputs N feature vectors of length 128. Each feature vector is derived from a section of the input signal, with the number of vectors N and their receptive fields being determined by the length of the input signal. Since EEG epochs have a fixed duration of 30 seconds (Section 2.1 of the paper), these will depend on the sampling frequency of the signals.

As seen in Seo et al.'s original publication, an input of length 3000 (corresponding to an epoch sampled at 100Hz) results in $N = 47$ feature vectors of $\mathbb{R}^{128}$. With MASS-SS1 and MASS-SS3, both sampled at 256Hz, this becomes $N = 120$. Given our subdivision of each epoch in 30 one-second segments (Section 4.1.1 of the paper), this equates to 4 vectors of $\mathbb{R}^{128}$ (i.e. 512 features) per segment, channel and signal.

We modify this CNN in two ways. Firstly, we adapt it to handle multi-channel, multi-signal inputs. As seen in Figure 1, for each epoch, features from each EEG signal and each channel are computed in parallel, sharing weights between signals in the same channel, but not between channels. Secondly, we reduce the 512 features per one-second segment into k features through a fully connected layer, obtaining a timeseries of 30 vectors of $\mathbb{R}^k$ per channel and signal.

Combining the n signals for a given segment and channel, we obtain the $n \times k$ feature matrix that we use to augment our covariance matrices. As seen in Table 1, the receptive field corresponding to each feature vector of $\mathbb{R}^k$ always includes the time segment on which the corresponding matrix is computed, with additional overlap with neighboring segments.

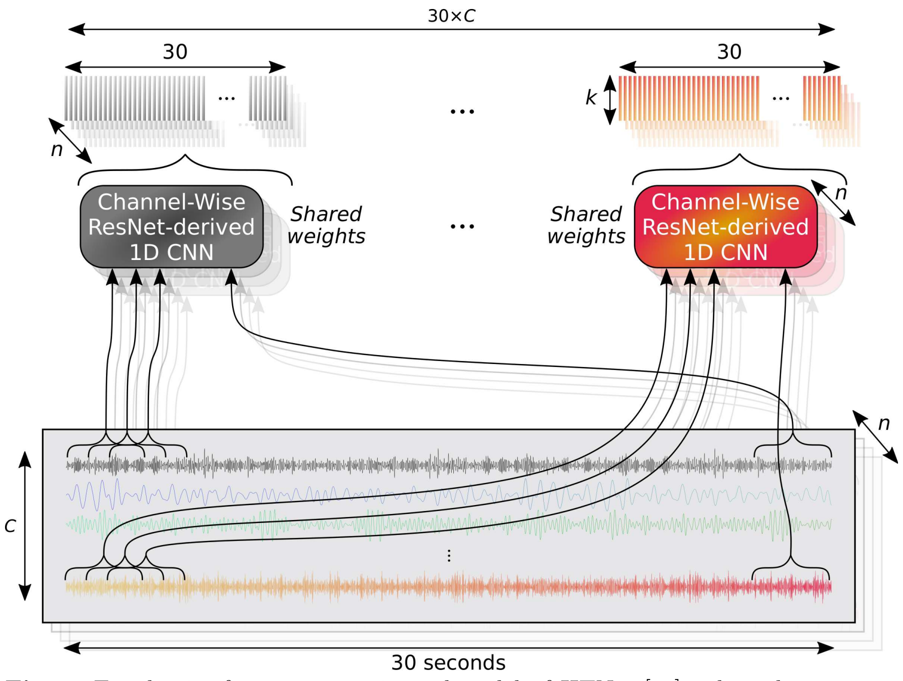
30×C
30
30
k
n
Channel-Wise ResNet-derived 1D CNN
Shared weights
Shared weights
Channel-Wise ResNet-derived 1D CNN
n
n
C
30 seconds

Fig. 1: Epoch-wise feature extraction submodel of IITNet [13], adapted to extract augmentation features - with n the number of EEG signals, C the number of channels, and k the size of each signal-wise feature vector.

Table 1: Receptive fields for each of the 30 augmentation feature vectors extracted, for epochs sampled at 256 Hz.

ID	Lower sample	Upper sample	Starting time (s)	Duration (s)
0	0	589	0	2.3
1	0	845	0	3.3
2	131	1101	0.51	3.79
3	387	1357	1.51	3.79
4	643	1613	2.51	3.79
...	...	...	...	...
25	6019	6986	23.51	3.79
26	6275	7245	24.51	3.79
27	6531	7501	25.51	3.79
28	6787	7679	26.51	3.48
29	7043	7679	27.51	2.48