# PSDNorm: Test-Time Temporal Normalization for Deep Learning in Sleep Staging

Théo Gnassounou Université Paris-Saclay, Inria, CEA, 91120 Palaiseau, France theo.gnassounou@inria.fr

Rémi Flamary École Polytechnique, IP Paris, CMAP, UMR 7641, 91120 Palaiseau, France remi.flamary@polytechnique.edu Antoine Collas Université Paris-Saclay, Inria, CEA, 91120 Palaiseau, France antoine.collas@inria.fr

Alexandre Gramfort\*

Université Paris-Saclay, Inria, CEA, 91120 Palaiseau, France alexandre.gramfort@inria.fr

#### ABSTRACT

Distribution shift poses a significant challenge in machine learning, particularly in biomedical applications using data collected across different subjects, institutions, and recording devices, such as sleep data. While existing normalization layers, BatchNorm, LayerNorm and InstanceNorm, help mitigate distribution shifts, when applied over the time dimension they ignore the dependencies and auto-correlation inherent to the vector coefficients they normalize. In this paper, we propose PSDNorm that leverages Monge mapping and temporal context to normalize feature maps in deep learning models for signals. Notably, the proposed method operates as a test-time domain adaptation technique, addressing distribution shifts without additional training. Evaluations with architectures based on U-Net or transformer backbones trained on 10K subjects across 10 datasets, show that PSDNorm achieves state-of-the-art performance on unseen left-out datasets while being 4-times more data-efficient than BatchNorm.

## 1 Introduction

**Data Shift in Physiological Signals** Machine learning techniques have achieved remarkable success in various domains, including computer vision, biology, audio processing, and language understanding. However, these methods face significant challenges when there are distribution shifts between training and evaluation datasets [1]. For example, in biological data, such as electroencephalography (EEG) signals, the distribution of the data can vary significantly. Indeed, data is collected from different subjects, electrode positions, and recording conditions. This paper focuses on sleep staging, a clinical task that consists in classifying periods of sleep in different stages based on EEG signals [2]. Depending on the dataset, the cohort can be composed of different age groups, sex repartition, health conditions, and recording conditions [3, 4, 5]. Such variability brings shift in the distribution making it challenging for the model to generalize to unseen datasets.

**Normalization to Address Data Shift** Normalization layers are widely used in deep learning to improve training stability and generalization. Common layers include BatchNorm [6], LayerNorm [7], and InstanceNorm [8], which respectively compute statistics across the batch, normalize across all features within each sample, and normalize each channel independently within a sample. Some normalization methods target specific tasks, such as EEG covariance matrices [9] or time-series forecasting [10], but they do not fully address spectral distribution shifts reflected in the temporal auto-correlations of signals. In sleep staging, a simple normalization is often applied as preprocessing, e.g.,

<sup>\*</sup>A. Gramfort joined Meta and can be reached at agramfort@meta.com



Figure 1: Description of normalization layers. The input shape is  $(N, c, \ell)$  with batch size N, channels c, and signal length  $\ell$ . BatchNorm estimates the mean  $\hat{\mu}$  and variance  $\hat{\sigma}^2$  over batch and time, and learns parameters  $(\gamma, \beta)$  to normalize the input. PSDNorm estimates PSDs  $\hat{\mathbf{P}}$  over time and accounts for local temporal correlations. It computes the barycenter PSD  $\hat{\mathbf{P}}$ , updates it via a running Riemannian barycenter (6), and applies the filter  $\hat{\mathbf{H}}$  to normalize the input. The hyperparameter F controls the extent of temporal correlation considered, thereby adjusting the strength of the normalization.

standardizing signals over entire nights [11] or short temporal windows [12]. Recent studies [13, 14] highlight the importance of considering temporal correlation and spectral content in normalization, proposing Temporal Monge Alignment (TMA), which aligns Power Spectral Density (PSD) to a common reference using Monge mapping, going beyond simple z-score normalization. However, these methods remain preprocessing steps that cannot be inserted as layers in the network architecture as it is done with BatchNorm, LayerNorm or InstanceNorm.

**Domain Generalization** Sleep staging has been addressed by various neural network architectures, which process raw signals [12, 15, 16], spectrograms [17, 18], or both [19]. More recent approaches involve transformer-based models that handle multimodal [20], spectrogram [21], or heterogeneous inputs [22], offering improved modeling of temporal dependencies. However, most existing models are trained on relatively small cohorts, typically consisting of only a few hundred subjects, which limits their ability to generalize to diverse clinical settings. Notable exceptions include U-Sleep [15], which was trained on a large-scale dataset and incorporates BatchNorm layers to mitigate data variability, and foundational models [23, 24, 25] that achieve strong generalization from vast amount of data but require significant computational resources and are challenging to adapt without fine-tuning. Our focus is on developing smaller, efficient models that balance good generalization with ease of training and deployment in clinical practice.

**Test-time Domain Adaptation** Domain generalization involves training models on data from multiple domains without using domain labels, with the goal of achieving broad generalization. In contrast, domain adaptation (DA) utilizes domain information to improve performance by aligning source and target distributions during training [26, 27]. While DA methods have been applied to address domain shifts in sleep staging [28], they typically require access to source data for each new target, which can be limiting. Test-time DA offers a more flexible approach, adapting the model during inference without requiring access to source data [13]. This makes test-time DA particularly well-suited for clinical applications where retraining or data sharing is impractical.

**Contributions** In this work, we introduce the PSDNorm deep learning layer, a novel approach to address distribution shifts in machine learning for signals. PSDNorm leverages Monge Mapping to incorporate temporal context and normalize feature maps effectively. As a test-time DA method, it adapts to distribution shifts during inference without extra training or source domain access. Compared to a normalization like LayerNorm or InstanceNorm, PSDNorm exploits the sequential nature of the intermediate layer representations as illustrated in Figure 1. We evaluate PSDNorm through extensive experiments on 10 sleep datasets. This evaluation covers 10M of samples across 10K subjects, using a leave-one-dataset-out (LODO) protocol with 3 different random seeds. To the best of our knowledge, such a large-scale and systematic evaluation has never been conducted before. PSDNorm achieves state-of-the-art performance and requires 4 times fewer labeled data to match the accuracy of the best baseline. Results highlight the potential of

PSDNorm as a practical and efficient solution for tackling domain shifts in signals.

The paper is structured as follows: Section 2 discusses existing normalization layers and pre-processing. Section 3 introduces PSDNorm, followed by numerical results in Section 4.

**Notations** Vectors are denoted by small cap boldface letters (e.g., **x**), matrices by large cap boldface letters (e.g., **X**). The element-wise product, power of n and division are denoted  $\odot$ ,  ${}^{\odot n}$  and  $\oslash$ , respectively.  $[\![1,K]\!]$  denotes  $\{1,\ldots,K\}$ . The absolute value is |.|. The discrete circular convolution along the temporal axis operates row-wise as,  $*: \mathbb{R}^{c \times \ell} \times \mathbb{R}^{c \times F} \to \mathbb{R}^{c \times \ell}$  for  $\ell \ge F$ . vec  $: \mathbb{R}^{c \times \ell} \to \mathbb{R}^{c\ell}$  concatenates rows of a time series into a vector.  $x_l = [\mathbf{x}]_l$  refers to the  $l^{\text{th}}$  element of **x**, and  $X_{l,m} = [\mathbf{X}]_{l,m}$  denotes the element of **X** at the  $l^{\text{th}}$  row and  $m^{\text{th}}$  column.  $\mathbf{X}^*$  and  $\mathbf{X}^\top$  are the conjugate and the transpose of **X**, respectively. diag puts the elements of a vector on the diagonal of a matrix.  $\otimes$  is the Kronecker product.  $\mathbf{1}_c$  is the vector of ones of size c.

## 2 Related Works

In this section, we first review fundamental concepts of normalization layers. Then, we recall the Temporal Monge Alignment (TMA) method [13] that aligns the PSD of signals using optimal transport.

**Normalization Layers** Normalization layers enhance training and robustness in deep neural networks. The most common are BatchNorm [6], InstanceNorm [8], and LayerNorm [7]. BatchNorm normalizes feature maps using batch and time statistics, ensuring zero mean and unit variance. The output is adjusted with learnable parameters. InstanceNorm normalizes each channel per sample using its own statistics, independent of the batch (see Fig. 1). Popular in time-series forecasting, it is used in RevIN [10], which reverses normalization after decoding. LayerNorm normalizes across all channels and time steps within each sample, with learnable scaling and shifting. While these normalization layers are widely employed, they operate on vectors ignoring statistical dependence and autocorrelation between their coefficients, which are prevalent when operating on time-series. To address this limitation, the Temporal Monge Alignment (TMA) [13, 14] was introduced as a pre-processing step to align temporal correlations by leveraging the Power Spectral density (PSD) of multivariate signals using Monge Optimal Transport mapping.

**Gaussian Periodic Signals** Consider a multivariate signal  $\mathbf{X} \triangleq [\mathbf{x}_1, \dots, \mathbf{x}_c]^\top \in \mathbb{R}^{c \times \ell}$  of sufficient length. A standard assumption is that this signal follows a centered Gaussian distribution where sensors are uncorrelated and signals are periodic. This periodicity and uncorrelation structure implies that the signal's covariance matrix is block diagonal, with each block having a circulant structure. A fundamental property of symmetric positive definite circulant matrices is their diagonalization [29] with real and positive eigenvalues in the Fourier basis  $\mathbf{F}_{\ell} \in \mathbb{C}^{\ell \times \ell}$  of elements

$$\left[\mathbf{F}_{\ell}\right]_{l,l'} \triangleq \frac{1}{\sqrt{\ell}} \exp\left(-2i\pi \frac{(l-1)(l'-1)}{\ell}\right),\tag{1}$$

where  $l, l' \in [\![1, \ell]\!]$ . Hence, we have  $vec(\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$  with  $\boldsymbol{\Sigma}$  block-diagonal,

$$\boldsymbol{\Sigma} = (\mathbf{I}_c \otimes \mathbf{F}_\ell) \operatorname{diag} \left( \operatorname{vec}(\mathbf{P}) \right) \left( \mathbf{I}_c \otimes \mathbf{F}_\ell^* \right) \in \mathbb{R}^{c\ell \times c\ell}, \tag{2}$$

where  $\mathbf{P} \in \mathbb{R}^{c \times \ell}$  contains positive entries corresponding to the Power Spectral Density of each sensor. In practice, since we only have access to a single realization of the signal, the PSD is estimated with only  $F \ll \ell$  frequencies, *i.e.*,  $\mathbf{P} \in \mathbb{R}^{c \times F}$ . This amounts to considering the local correlation of the signal and neglecting the long-range correlations.

**Power Spectral Density Estimation** The Welch estimator [30] computes the PSD of a signal by averaging the squared Fourier transform of overlapping segments of the signal. Hence, the realization of the signal  $\mathbf{X} \in \mathbb{R}^{c \times \ell}$  is decimated into overlapping segments  $\{\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(L)}\} \subset \mathbb{R}^{c \times F}$  to estimate the PSD. The Welch estimator is defined as

$$\widehat{\mathbf{P}} \triangleq \frac{1}{L} \sum_{l=1}^{L} \left| \left( \left( \mathbf{1}_{c} \mathbf{w}^{\top} \right) \odot \mathbf{X}^{(l)} \right) \mathbf{F}_{F}^{*} \right|^{\odot 2} \in \mathbb{R}^{c \times F} , \qquad (3)$$

where  $\mathbf{w} \in \mathbb{R}^F$  is the window function such that  $\|\mathbf{w}\|_2 = 1$ .

*F*-Monge Mapping Let  $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}^{(s)})$  and  $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}^{(t)})$  be source and target centered Gaussian distributions respectively with covariance matrices following the structure (2) and PSDs denoted by  $\mathbf{P}^{(s)}$  and  $\mathbf{P}^{(t)} \in \mathbb{R}^{c \times F}$ . Given a signal

 $\mathbf{X} \in \mathbb{R}^{c \times \ell}$  such that  $\operatorname{vec}(\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}^{(s)})$ , the *F*-Monge mapping as defined by [13, 14] is

$$m_F\left(\mathbf{X},\mathbf{P}^{(t)}\right) \triangleq \mathbf{X} * \mathbf{H} \in \mathbb{R}^{c \times \ell}, \quad \text{where} \quad \mathbf{H} \triangleq \frac{1}{\sqrt{F}} \left(\mathbf{P}^{(t)} \oslash \mathbf{P}^{(s)}\right)^{\odot \frac{1}{2}} \mathbf{F}_F^* \in \mathbb{R}^{c \times F}.$$
 (4)

In this case, F controls the alignment between the source and target distributions. Indeed, if  $F = \ell$ , then the F-Monge mapping is the classical Monge mapping between Gaussian distributions and the source signal has its covariance matrix equal to  $\Sigma^{(t)}$  after the mapping. If F = 1, then each sensor is only multiplied by a scalar.

**Gaussian Wasserstein Barycenter** For Gaussian distributions admitting the decomposition (2), the Wasserstein barycenter [31] admits an elegant closed-form solution. Consider K centered Gaussian distributions admitting the decomposition (2) of PSDs  $\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(K)}$ . Their barycenter is also a centered Gaussian distribution  $\mathcal{N}(\mathbf{0}, \overline{\Sigma})$  admitting the decomposition (2) with PSD

$$\overline{\mathbf{P}} \triangleq \left(\frac{1}{K} \sum_{k=1}^{K} \mathbf{P}^{(k)^{\odot} \frac{1}{2}}\right)^{\odot 2} \in \mathbb{R}^{c \times F} .$$
(5)

**Temporal Monge Alignement** TMA is a pre-processing method that aligns the PSD of multivariate signals using the F-Monge mapping. Given a source signal  $\mathbf{X}_s$  and a set of target signals  $\mathbf{X}_t = {\mathbf{X}_t^{(1)}, \ldots, \mathbf{X}_t^{(K)}}$ , the TMA method uses the F-Monge mapping between the source and the Wasserstein barycenter of the target signals. Hence, it simply consists of 1) estimating the PSD of all the signals, 2) computing the Wasserstein barycenter of the target signals, and 3) applying the F-Monge mapping to the source signal. TMA, as a preprocessing method, is inherently limited to handling PSD shifts in the raw signals and cannot address more complex distributional changes in the data. This limitation highlights the need for a layer that can effectively capture and adapt to these complex variations during learning and inside deep learning models.

## 3 PSDNorm Layer

The classical normalization layers, such as BatchNorm or InstanceNorm do not take into account the temporal autocorrelation structure of signals. They treat each time sample in the intermediate representations independently. In this section, we introduce the PSDNorm layer that aligns the PSD of each signal onto a barycenter PSD within the architecture of a deep learning model.

**PSDNorm as a Drop-in Replacement** As discussed in Section 2, BatchNorm and InstanceNorm layers are commonly applied after each non-linearity in deep learning models, such as U-Time, to normalize feature maps. In contrast, PSDNorm offers a novel approach by aligning the PSD of feature maps to a barycenter PSD, providing a powerful alternative to classical normalization layers. Designed as a drop-in replacement, PSDNorm is optimized for modern hardware accelerators like GPUs, ensuring efficient execution. Once the deep-learning model is trained, PSDNorm operates as a test-time domain adaptation technique, allowing it to adapt to new data without additional training or access to training data. We define the normalized feature map as  $\widetilde{\mathbf{G}} \triangleq \text{PSDNorm}(\mathbf{G})$ . The following sections introduce the core components of PSDNorm and its implementation.

#### 3.1 Core Components of the layer

In the following, we formally define PSDNorm and present each of its three main components: 1) PSD estimation, 2) running Riemannian barycenter estimation, and 3) *F*-Monge mapping computation. Given a batch  $\mathcal{B} = {\mathbf{G}^{(1)}, \ldots, \mathbf{G}^{(N)}}$  of *N* pre-normalization feature maps, PSDNorm outputs a normalized batch  $\tilde{\mathcal{B}} = {\tilde{\mathbf{G}}^{(1)}, \ldots, \tilde{\mathbf{G}}^{(N)}}$  with normalized PSD. Those three steps are detailed in the following and illustrated in the right part of Figure 1.

**PSD Estimation** First, the estimation of the PSD of each feature map is performed using the Welch method. The per-channel mean  $\hat{\mu}^{(j)}$  is computed for each feature map  $\mathbf{G}^{(j)}$  as  $\hat{\mu}^{(j)} \triangleq \frac{1}{\ell} \sum_{l=1}^{\ell} [\mathbf{G}^{(j)}]_{:,l} \in \mathbb{R}^{c}$ . Then, the PSD of the centered feature map  $\mathbf{G}^{(j)} - \hat{\mu}^{(j)} \mathbf{1}_{\ell}^{\top}$ , denoted  $\hat{\mathbf{P}}^{(j)}$ , is estimated as described in Equation (3). This centering step is required as feature maps are typically non-centered due to activation functions and convolution biases but they are assumed to have a stationary mean. The Welch estimation involves segmenting the centered feature map into overlapping windows, computing the Fourier transform of each window and then averaging them.

**Geodesic and Running Riemanian Barycenter** The PSD-Norm aligns the PSD of each feature map to a barycenter PSD. This barycenter is computed during training by interpolating between the batch Wasserstein barycenter and the current running Riemanian barycenter using the geodesic associated with the Bures metric [32]. The batch barycenter is first computed from the current batch PSDs  $\{\widehat{\mathbf{P}}^{(1)}, \dots, \widehat{\mathbf{P}}^{(N)}\}$  using Equation (5). To ensure gradual adaptation, the running barycenter is updated via an exponential geodesic average with  $\alpha \in [0, 1]$ :

$$\widehat{\overline{\mathbf{P}}} \leftarrow \left( (1-\alpha) \widehat{\overline{\mathbf{P}}}^{\odot \frac{1}{2}} + \alpha \widehat{\overline{\mathbf{P}}}_{\mathcal{B}}^{\odot \frac{1}{2}} \right)^{\odot 2} \in \mathbb{R}^{c \times F} .$$
 (6)

A proof of the geodesic is provided in Appendix A.1.



PREPRINT

Figure 2: Description of the running Riemanian barycenter. The barycenter of the batch  $\widehat{\overline{\mathbf{P}}}_{\mathcal{B}}$  is estimated from the PSD of each batch sample. Then the running Riemanian barycenter is updated through an exponential average along the geodesic (- -), parameterized by  $\alpha \in [0, 1]$ .

**PSD** Adaptation with *F*-Monge Mapping The final step of the PSDNorm is the application of the *F*-Monge mapping to each feature map after subtracting the per-channel mean. Indeed, for all  $j \in [\![1, N]\!]$ , it is defined as

$$\widetilde{\mathbf{G}}^{(j)} = m_F \left( \mathbf{G}^{(j)} - \widehat{\boldsymbol{\mu}}^{(j)} \mathbf{1}_{\ell}^{\top}, \widehat{\overline{\mathbf{P}}} \right) = \left( \left( \mathbf{G}^{(j)} - \widehat{\boldsymbol{\mu}}^{(j)} \mathbf{1}_{\ell}^{\top} \right) * \widehat{\mathbf{H}}^{(j)} \right) \in \mathbb{R}^{c \times \ell}$$
(7)

where  $\widehat{\mathbf{H}}^{(j)}$  is the Monge mapping filter computed as

$$\widehat{\mathbf{H}}^{(j)} \triangleq \frac{1}{\sqrt{F}} \left( \widehat{\overline{\mathbf{P}}} \oslash \widehat{\mathbf{P}}^{(j)} \right)^{\odot \frac{1}{2}} \mathbf{F}_F^* \in \mathbb{R}^{c \times f}$$
(8)

where  $\widehat{\mathbf{P}}^{(j)}$  is the estimated PSD of  $\mathbf{G}^{(j)} - \widehat{\boldsymbol{\mu}}^{(j)} \mathbf{1}_{\ell}^{\top}$ .

#### **3.2** Implementation details

**Overall Algorithm** The forward computation of the proposed layer is outlined in Algorithm 1. At train time, the PSDNorm performs three main operations: 1) PSD estimation, 2) running Riemannian barycenter update, and 3) Monge mapping application. At test time, the PSDNorm operates similarly, except it does not update the running barycenter. The PSDNorm is fully differentiable and can be integrated into any deep learning model. Similarly to classical normalization layers, a stop gradient operation is applied to the running barycenter to prevent the backpropagation of the gradient computation through the barycenter. PSDNorm has a unique additional hyperparameter F which is the filter size. It controls the alignment between each feature map and the running barycenter PSD and it is typically chosen in our experiments between 1 and 17. In prac-

## Algorithm 1 Forward pass of PSDNorm

- 1: Input: Batch  $\mathcal{B} = \{\mathbf{G}^{(1)}, \dots, \mathbf{G}^{(N)}\}$ , running barycenter  $\widehat{\overline{\mathbf{P}}}$ , filter-size F, momentum  $\alpha$ , training flag
- 2: **Output:** Normalized batch  $\left\{ \widetilde{\mathbf{G}}^{(1)}, \dots, \widetilde{\mathbf{G}}^{(N)} \right\}$
- 3: **for** j = 1 to *N* **do**
- 4:  $\hat{\mu}^{(j)} \leftarrow$  Mean estimation
- 5:  $\widehat{\mathbf{P}}^{(j)} \leftarrow \text{PSD est. from } \widetilde{\mathbf{G}}^{(j)} \widehat{\boldsymbol{\mu}}^{(j)} \mathbf{1}_{\ell}^{\top} \text{ with eq. (3)}$
- 6: end for
- 7: **if** training **then**
- 8:  $\widehat{\overline{\mathbf{P}}}_{\mathcal{B}} \leftarrow$  Batch bary. from  $\{\widehat{\mathbf{P}}^{(j)}\}_j$  with eq. (5)
- 9:  $\widehat{\overline{\mathbf{P}}} \leftarrow \text{Running bary. up. from } \widehat{\overline{\mathbf{P}}}, \widehat{\overline{\mathbf{P}}}_{\mathcal{B}} \text{ with eq. (6)}$
- 10: end if
- 11: **for** j = 1 to *N* **do**
- 12:  $\widehat{\mathbf{H}}^{(j)} \leftarrow \text{Filter estimation from } \widehat{\mathbf{P}}^{(j)}, \overline{\mathbf{P}} \text{ with eq. (8)}$
- 13:  $\widetilde{\mathbf{G}}^{(j)} \leftarrow F$ -Monge mapping with eq. (7)
- 14: **end for**

tice, the Fourier transforms are efficiently computed using the Fast Fourier Transform (FFT) algorithm. Because of the estimation of PSDs, the complexity of the PSDNorm, both at train and test times, is  $O(Nc\ell F \log(F))$ , where N is the batch size, c the number of channels,  $\ell$  the signal length, and F the filter size.

**PSDNorm as a generalization of InstanceNorm** InstanceNorm applies a per-channel z-score over time, subtracting the mean and dividing by the standard deviation—equivalent to whitening under an i.i.d. assumption over time. In contrast, PSDNorm explicitly accounts for temporal structure by estimating the PSD and whitening/re-coloring in the frequency domain. InstanceNorm is recovered as a special case of PSDNorm by setting the filter size to F = 1 and using the identity as the re-coloring transform instead of the barycentric PSD.

#### **4** Numerical Experiments

In this section, we evaluate the proposed method through a series of experiments designed to highlight its effectiveness and robustness on the clinically relevant task of sleep staging. We first describe the datasets and training setup employed, followed by a performance comparison with existing normalization techniques. Next, we assess the efficiency of PSDNorm by training over varying numbers of subjects per dataset. Finally, we analyze the robustness of PSDNorm against domain shift by focusing on subject-wise performance and different architectures. The code will be available on GitHub upon acceptance. The anonymized code is available in the supplementary material. All numerical experiments were conducted using a total of 1500 GPU hours on NVIDIA H100 GPUs.

#### 4.1 Experimental Setup

**Datasets** To evaluate the effect of normalization layers, we use ten datasets of sleep staging described in Table 1. ABC [33], CCSHS [34], CFS [35], HPAP [36], MROS [37], SHHS [4], CHAT [5], and SOF [38] are publicly available sleep datasets with restricted access from National Sleep Research Resource (NSRR) [39]. PHYS [40] and MASS [3] are two other datasets publicly available. Every 30 s epoch is labeled with one of the five sleep stages: Wake, N1, N2, N3, and REM. These datasets are unbalanced in terms of age, sex, number of subjects, and have been recorded with different sensors in different institutions which makes the sleep staging task challenging. We now describe the pre-processing steps and splits of the datasets.

Table 1: Characteristics of the datasets.

PREPRINT

Dataset	Subj.	Rec.	Age $\pm$ std	Sex (F/M)
ABC	44	117	$48.8\pm9.8$	43%/57%
CCSHS	515	515	$17.7\pm0.4$	50%/50%
CFS	681	681	$41.7\pm20.0$	55%/45%
HPAP	166	166	$46.5\pm11.9$	43%/57%
MROS	2101	2698	$76.4\pm5.5$	0%/100%
PHYS	70	132	$58.8\pm22.0$	33%/67%
SHHS	5730	8271	$63.1\pm11.2$	52%/48%
MASS	61	61	$42.5\pm18.9$	55%/45%
CHAT	1230	1635	$6.6\pm1.4$	52%/48%
SOF	434	434	$82.8\pm3.1$	100%/0%
Total	11032	14710	_	—

Data Pre-processing We follow a standard pre-

processing pipeline used in the field [41, 42]. The datasets

vary in the number and type of available EEG and electrooculogram (EOG) channels. To ensure consistency, we use two bipolar EEG channels, as some datasets lack additional channels. For dataset from NSRR, we select the channels C3-A2 and C4-A1. For signals from Physionet and MASS, we use the only available channels Fpz-Cz and Pz-Oz. The EEG signals are low-pass filtered with a 30 Hz cutoff frequency and resampled to 100 Hz. All data extraction and pre-processing steps are implemented using MNE-BIDS [43] and MNE-Python [44].

**Leave-One-Dataset-Out (LODO) Setup and Balancing** We evaluate model performance using a leave-one-datasetout (LODO) protocol: in each fold, one dataset is held out for testing, and the model is trained on the union of the remaining datasets. From the training data, 80% of subjects are used for training and 20% for validation, which is used for early stopping. The full held-out dataset is used for testing. To assess performance in low-data regimes, we also evaluate a variant in which we subsample at most N subjects per dataset, promoting balanced contributions across training sources. We refer to this configuration as **balanced**@N, with N ranging from 40 to 400. The exact number of subjects per dataset in each case is listed in Appendix Table 3.

**Architecture and Training** Sleep staging has inspired a variety of neural architectures, from early CNN-based models [41, 42, 19] to recent attention-based approaches [21, 17, 20]. We evaluate two architectures: **U-Sleep** [45, 15], a state-of-the-art temporal CNN model designed for robustness and large-scale training, and a newly introduced architecture, **CNNTransformer**. CNNTransformer combines a lightweight convolutional encoder with a Transformer applied to epoch-level embeddings. It is specifically tailored for two-channel EEG and designed to scale efficiently to large datasets, while remaining minimal in implementation (under 100 lines of code) and training cost (Appendix A.3). Its design draws inspiration from recent transformer-based models for time series [46], with an emphasis on simplicity and practicality.

We use the Adam optimizer [47] with a learning rate of  $10^{-3}$  to minimize the weighted cross-entropy loss, where class weights are computed from the training set distribution. Training is performed with a batch size of 64, and early stopping is applied based on validation loss with a patience of 3 epochs. Each input corresponds to a sequence of 17'30s, with a stride of 10'30s between sequences along the full-night recording.

**Evaluation** At test time, the model similarly processes sequences of 17'30s with a stride of 10'30s. Performance is evaluated using the balanced accuracy score (BACC), computed on the central 10'30s of each prediction window.

Table 2: **Balanced Accuracy (BACC) scores on the left-out datasets.** The top section reports results in the **large-scale** setting (using all available subjects), while the bottom section presents results in the **medium-scale** setting (balanced@400). For each row, the best score is highlighted in **bold**, and standard deviations reflect training variability across 3 random seeds.

		BatchNorm	LayerNorm	InstanceNorm	PSDNorm(F=5)	PSDNorm(F=9)	PSDNorm(F=17)
	ABC	$78.49 \pm 0.42$	$77.94 \pm 0.31$	$\textbf{78.83} \pm \textbf{0.59}$	$78.56 \pm 0.67$	$78.73 \pm 0.32$	$78.60\pm0.28$
	CCSHS	$\textbf{88.79} \pm \textbf{0.21}$	$87.51\pm0.77$	$88.75\pm0.04$	$88.56\pm0.36$	$88.48 \pm 0.07$	$88.52\pm0.19$
	CFS	$84.97\pm0.37$	$84.29\pm0.67$	$\textbf{85.73} \pm \textbf{0.29}$	$85.42\pm0.09$	$85.25\pm0.19$	$85.28\pm0.12$
cts	CHAT	$64.72 \pm 3.94$	$64.36\pm0.40$	$68.86 \pm 2.49$	$70.57 \pm 1.24$	$70.36 \pm 2.22$	$\textbf{70.71} \pm \textbf{2.15}$
jeć	HOMEPAP	$76.39\pm0.29$	$75.23\pm0.78$	$76.70\pm0.35$	$76.72\pm0.27$	$76.93 \pm 0.10$	$\textbf{77.02} \pm \textbf{0.32}$
sut	MASS	$\textbf{73.71} \pm \textbf{0.62}$	$71.39\pm3.00$	$72.12\pm0.70$	$72.51 \pm 1.68$	$71.34 \pm 2.68$	$71.61\pm0.71$
	MROS	$81.30\pm0.25$	$80.44 \pm 0.29$	$81.49 \pm 0.18$	$\textbf{81.57} \pm \textbf{0.34}$	$81.35\pm0.17$	$81.30\pm0.13$
A	PhysioNet	$76.13\pm0.57$	$75.12\pm0.22$	$76.15\pm0.52$	$75.96 \pm 1.02$	$\textbf{76.35} \pm \textbf{0.27}$	$76.02\pm0.47$
	SHHS	$77.97 \pm 1.46$	$75.98 \pm 0.48$	$79.05\pm0.89$	$79.14 \pm 1.01$	$\textbf{79.33} \pm \textbf{0.62}$	$79.12\pm0.11$
	SOF	$81.33\pm0.54$	$81.82\pm0.79$	$81.98 \pm 0.22$	$\textbf{82.50} \pm \textbf{0.34}$	$82.15 \pm 1.00$	$81.94 \pm 0.65$
	Mean	$78.38\pm0.47$	$77.41\pm0.28$	$78.97 \pm 0.11$	$\textbf{79.15} \pm \textbf{0.14}$	$79.03\pm0.10$	$79.01 \pm 0.36$
	ABC	$78.26 \pm 1.33$	$75.29 \pm 0.81$	$\textbf{78.73} \pm \textbf{0.42}$	$78.18\pm0.68$	$78.18\pm0.91$	$77.76 \pm 1.00$
	CCSHS	$87.42 \pm 0.16$	$85.20\pm0.48$	$\textbf{87.62} \pm \textbf{0.42}$	$87.58 \pm 0.30$	$87.35\pm0.52$	$\textbf{87.62} \pm \textbf{0.48}$
	CFS	$84.32\pm0.57$	$81.66 \pm 1.36$	$\textbf{84.72} \pm \textbf{0.33}$	$84.29\pm0.36$	$84.06\pm0.10$	$84.46\pm0.06$
9	CHAT	$66.55\pm0.88$	$61.19 \pm 1.16$	$64.43 \pm 4.41$	$\textbf{70.28} \pm \textbf{1.70}$	$68.11 \pm 3.94$	$69.88 \pm 0.46$
@ 7	HOMEPAP	$75.25 \pm 0.50$	$74.86\pm0.25$	$76.47\pm0.63$	$\textbf{76.83} \pm \textbf{0.61}$	$76.61\pm0.74$	$76.49\pm0.45$
ede	MASS	$70.00\pm1.91$	$68.56\pm3.33$	$71.52 \pm 1.13$	$72.77 \pm 1.09$	$\textbf{73.07} \pm \textbf{1.30}$	$72.23\pm2.40$
nc	MROS	$80.37\pm0.20$	$78.05\pm0.22$	$80.28 \pm 0.21$	$80.26\pm0.11$	$80.32\pm0.22$	$\textbf{80.70} \pm \textbf{0.42}$
ala	PhysioNet	$\textbf{75.81} \pm \textbf{0.13}$	$71.82\pm2.12$	$74.68\pm0.55$	$74.82\pm2.11$	$73.77 \pm 1.73$	$75.09\pm0.97$
В	SHHS	$76.44 \pm 0.92$	$75.12\pm0.39$	$78.68 \pm 0.37$	$\textbf{78.88} \pm \textbf{0.68}$	$77.28 \pm 0.91$	$78.41 \pm 0.49$
	SOF	$81.08 \pm 1.14$	$78.70\pm0.50$	$80.68 \pm 1.38$	$79.49\pm0.41$	$\textbf{81.44} \pm \textbf{0.97}$	$81.07\pm0.66$
	Mean	$77.55 \pm 0.34$	$75.05\pm0.28$	$77.78\pm0.46$	$78.34 \pm 0.42$	$78.02\pm0.67$	$\textbf{78.37} \pm \textbf{0.38}$

Each experiment is repeated three times with different random seeds, and we report the mean and standard deviation of BACC.

**Normalization Strategies** We compare the proposed PSDNorm with three normalization strategies: BatchNorm, LayerNorm, and InstanceNorm. Note that InstanceNorm corresponds to a special case of PSDNorm with F = 1 and a fixed identity mapping instead of a learned running barycenter. In the following experiments, the BatchNorm layers in the first three convolutional layers are replaced with either PyTorch's default implementations of LayerNorm, InstanceNorm [48], or PSDNorm. To preserve the receptive field, the filter size F of PSDNorm is used in the first layer and progressively halved in the following ones. We fix the momentum  $\alpha$  to  $10^{-2}$ .

## 4.2 Numerical Results

This section presents results from large-scale sleep stage classification experiments. The analysis begins with a comparison of PSDNorm against standard normalization layers—BatchNorm, LayerNorm, and InstanceNorm—on the full datasets. Then, the data efficiency of each method is evaluated under limited training data regimes. Finally, robustness to distribution shift is assessed via subject-wise performance across multiple neural network architectures.

**Performance Comparison on Full Datasets** Table 2 (top) reports the LODO BACC of U-Sleep across all datasets, averaged over three random seeds. PSDNorm consistently outperforms all baseline normalization layers—BatchNorm, LayerNorm, and InstanceNorm—achieving the highest mean BACC of 79.15%, which exceeds BatchNorm (78.38%), InstanceNorm (78.97%), and LayerNorm (77.41%) by at least one standard deviation. This performance holds across tested filter sizes, with PSDNorm reaching 79.03% for F=9 and 79.01% for F=17. In total, PSDNorm ranks first on 6 out of the 10 datasets. On the challenging CHAT dataset, where all methods struggle, PSDNorm with F=5 outperforms all other normalizations by more than 2 percentage points, highlighting its robustness under strong distribution shifts. Although InstanceNorm is a strong baseline—outperforming BatchNorm and LayerNorm by at least one standard deviation on average—it is consistently surpassed by PSDNorm in average performance. In contrast, LayerNorm underperforms across the board, achieving the lowest average BACC and never ranking first, confirming its limited suitability for this task.

Efficiency: Performance with 4× Less Data The PSDNorm layer improves model performance when trained on the full dataset ( $\sim 10000$  subjects), but such large-scale data availability is not always the case. In many real-world scenarios-such as rare disease studies, pediatric populations, or data collected in constrained clinical settings-labeled recordings are scarce, expensive to annotate, or restricted due to privacy concerns. Evaluating model robustness under these constraints is therefore essential. To this end, we train all models using the balanced@400 setup, which reduces the training data by a factor of 4 compared to the full-data setting. In this lower-data regime, PSDNorm continues to outperform all baseline normalization strategies on 6 out of 10 datasets and achieves



Figure 3: Critical Difference (CD) diagram for two architectures on datasets balanced @400. Average ranks across datasets and subjects for USleep and CNNTransformer. Black lines connect methods that are not significantly different.

higher average BACC across all tested values of F. The performance improvement of PSDNorm over the best baseline is more pronounced in this setting: for F=5, the BACC gain reaches +0.56%, compared to +0.18% in the full-data setting. Again, the gains exceed one standard deviation. To assess statistical significance, we conducted a critical difference (CD) test [49]. Figure 3 (top) reports the average rank of each method and the corresponding statistical comparisons. The results confirm that PSDNorm significantly outperforms the baselines, underscoring the value of incorporating temporal structure into normalization for robust and data-efficient generalization. The same trend is observed for U-Sleep trained on all subjects (see in Appendix Figure 6). The following experiments focus on the balanced@400 setup.

**Robustness Across Architectures** PSDNorm is a plug-and-play normalization layer that can be seamlessly integrated into various neural network architectures. To demonstrate this flexibility, we evaluate its performance on both the U-Sleep and CNNTransformer models. Figure 3 reports the average rank of each normalization method across datasets and subjects for both architectures using datasets balanced@400. In both architectures, PSDNorm with F=5 achieves the best overall ranking and demonstrates statistically significant improvements over both BatchNorm, LayerNorm and InstanceNorm. Notably, it is also the only method to rank among the top two across both architectures. The results confirm that PSDNorm generalizes well beyond a single architecture and can provide consistent improvements in diverse modeling setups. InstanceNorm performs competitively in some cases but is never significantly better than PSDNorm. The figure also illustrates that PSDNorm with F=17 does not yield additional benefit in this medium-scale setting, further supporting the conclusion that moderate temporal context is sufficient at this data scale. Detailed numerical scores for CNNTransformer are reported in the supplementary material (Table 7).



Figure 4: **Performance of PSDNorm and BatchNorm with varying training set sizes.** The BACC score is plotted against the number of training subjects used with U-Sleep.

**Sensitivity to Filter Size** The choice of *F* in PSDNorm controls the intensity of the normalization: larger F provide stronger normalization, while smaller F allow more flexibility in the model. In Figure 4, we evaluate its impact across different training set sizes and observe a clear trend: when trained on fewer subjects, larger filter sizes yield better performance (*i.e.*, F = 17), whereas smaller filter sizes are more effective with larger datasets (*i.e.*, F = 5). This suggests that with limited data, stronger normalization helps prevent overfitting, while with more data, a more flexible model is preferred. On average, PSD-Norm with F = 5 offers a good compromise, achieving one of the best performances across all training set sizes.

Performance on the most challenging subjects Performance variability across subjects is a key challenge in biomedical applications where ensuring consistently high performance-even for the most challenging subjects-is critical. To highlight the robustness of PSDNorm, Figure 5 presents a scatter plot of subject-wise BACC scores comparing BatchNorm or InstanceNorm vs. PSDNorm across two selected target datasets. CHAT and MASS are two challenging datasets, where the prediction performance is significantly lower than the other datasets. For CHAT, most of the dots are below the diagonal, indicating that PSDNorm improves performance for 91% of subjects against Batch-Norm and 99% of subjects against InstanceNorm, with the largest gains observed for the hardest subjects, reinforcing its ability to handle challenging cases. For MASS, PSD-Norm improves performance for 75% of subjects against BatchNorm and 69% against InstanceNorm. This demonstrates that PSDNorm is not only effective in improving overall performance but also excels in enhancing the performance of the most challenging subjects.



Figure 5: Subject-wise BACC comparison on MASS and CHAT (balanced @400). Blue dot means improvement with PSDNorm.

## 5 Conclusion, Limitations, and Future Work

This paper introduced PSDNorm, a normalization layer that aligns the power spectral density (PSD) of each signal to a geodesic barycenter. By leveraging temporal correlations, PSDNorm offers a principled alternative to standard normalization layers. Experiments on large-scale sleep staging datasets show that PSDNorm consistently improves performance, robustness, and data efficiency, especially under domain shift and limited-data settings—outperforming BatchNorm, LayerNorm, and InstanceNorm across architectures.

While the results are promising, some limitations remain. PSDNorm introduces a filter size hyperparameter (F) that controls normalization strength; although we provide default values that perform well across datasets, selecting it automatically in adaptive settings could be challenging.

Despite these limitations, PSDNorm is flexible and easy to integrate into existing models. Future work includes extending it to other signals such as audio and other biomedical applications.

## Acknowledgements

This work was supported by the grants ANR-22-PESN-0012 to AC under the France 2030 program, ANR-20-CHIA-0016 and ANR-20-IADJ-0002 to AG while at Inria, and ANR-23-ERCC-0006 to RF, all from Agence nationale de la recherche (ANR). This project has also received funding from the European Union's Horizon Europe research and innovation programme under grant agreement 101120237 (ELIAS).

This project received funding from the Fondation de l'École polytechnique

This project was provided with computer and storage resources by GENCI at IDRIS thanks to the grant 2025-AD011016052 and 2025-AD011016067 on the supercomputer Jean Zay's the V100 & H100 partitions.

This work was conducted at Inria, AG is presently employed by Meta Platforms. All the datasets used for this work were accessed and processed on the Inria compute infrastructure.

All the datasets used for this work were accessed and processed on the Inria compute infrastructures. Numerical computation was enabled by the scientific Python ecosystem: Matplotlib [50], Scikit-learn [51], Numpy [52], Scipy [53], PyTorch [48] and MNE [44].

PREPRINT

- J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition* 45 no. 1, (2012) 521–530.
- [2] S. Stevens and G. Clark, "Chapter 6 polysomnography," in *Sleep Medicine Secrets*, D. STEVENS, ed., pp. 45–63. Hanley & Belfus, 2004.
- [3] C. O'Reilly, N. Gosselin, and J. Carrier, "Montreal archive of sleep studies: an open-access resource for instrument benchmarking and exploratory research," *Journal of sleep research* **23** (06, 2014).
- [4] S. Quan, B. Howard, *et al.*, "The sleep heart health study: Design, rationale, and methods," *Sleep* **20** (01, 1998) 1077–85.
- [5] C. L. Marcus, R. H. Moore, *et al.*, "A randomized trial of adenotonsillectomy for childhood sleep apnea," *The New England Journal of Medicine* **368** no. 25, (June, 2013) 2366–2376.
- [6] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning -Volume 37*, ICML'15, p. 448–456. JMLR.org, 2015.
- [7] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization." 2016. http://arxiv.org/abs/1607.06450.
- [8] D. Ulyanov, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022* (2016).
- [9] R. Kobler, J.-i. Hirayama, Q. Zhao, and M. Kawanabe, "Spd domain-specific batch normalization to crack interpretable unsupervised domain adaptation in EEG," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., vol. 35, pp. 6219–6235. Curran Associates, Inc., 2022.
- [10] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *International Conference on Learning Representations*. 2021.
- [11] A. Apicella, F. Isgrò, A. Pollastro, and R. Prevete, "On the effects of data normalization for domain adaptation on EEG data," *Engineering Applications of Artificial Intelligence* **123** (2023) 106205.
- [12] S. Chambon, M. N. Galtier, P. J. Arnal, G. Wainrib, and A. Gramfort, "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series," *IEEE Transactions on Neural Systems* and Rehabilitation Engineering 26 no. 4, (2018) 758–769.
- [13] T. Gnassounou, R. Flamary, and A. Gramfort, "Convolutional monge mapping normalization for learning on biosignals," in *Neural Information Processing Systems (NeurIPS)*. 2023.
- [14] T. Gnassounou, A. Collas, R. Flamary, K. Lounici, and A. Gramfort, "Multi-source and test-time domain adaptation on multivariate signals using spatio-temporal monge alignment," arXiv preprint arXiv:2407.14303 (2024).
- [15] M. Perslev, S. Darkner, L. Kempfner, M. Nikolic, P. Jennum, and C. Igel, "U-Sleep: resilient high-frequency sleep staging," *npj Digital Medicine* **4** (04, 2021) 72.
- [16] A. Guillot and V. Thorey, "RobustSleepNet: Transfer learning for automated sleep staging at scale," arXiv:2101.02452 [stat.ML].
- [17] H. Phan, K. P. Lorenzen, E. Heremans, O. Y. Chén, M. C. Tran, P. Koch, A. Mertins, M. Baumert, K. B. Mikkelsen, and M. De Vos, "L-SeqSleepNet: Whole-cycle Long Sequence Modeling for Automatic Sleep Staging," *IEEE Journal of Biomedical and Health Informatics* 27 no. 10, (Oct., 2023) 4748–4757. https://ieeexplore.ieee.org/document/10210638/?arnumber=10210638. Conference Name: IEEE Journal of Biomedical and Health Informatics.
- [18] H. Phan, F. Andreotti, N. Cooray, O. Y. Chén, and M. D. Vos, "SeqSleepNet: End-to-End Hierarchical Recurrent Neural Network for Sequence-to-Sequence Automatic Sleep Staging." Feb., 2019. http://arxiv.org/abs/1809.10932. arXiv:1809.10932.
- [19] H. Phan, O. Y. Chen, M. C. Tran, P. Koch, A. Mertins, and M. D. Vos, "XSleepNet: Multi-view sequential model for automatic sleep staging," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 no. 09, (Sep, 2022) 5903–5915.
- [20] J. Wang, S. Zhao, H. Jiang, Y. Zhou, Z. Yu, T. Li, S. Li, and G. Pan, "CareSleepNet: A hybrid deep learning network for automatic sleep staging,". https://ieeexplore.ieee.org/document/10595067/.

- [21] H. Phan, K. Mikkelsen, O. Y. Chén, P. Koch, A. Mertins, and M. De Vos, "Sleeptransformer: Automatic sleep staging with interpretability and uncertainty quantification," *IEEE Transactions on Biomedical Engineering* 69 no. 8, (2022) 2456–2467.
- [22] Y. Guo, M. Nowakowski, and W. Dai, "Flexsleeptransformer: a transformer-based sleep staging model with flexible input channel configurations," *Scientific Reports* 14 (11, 2024).
- [23] R. Thapa, M. R. Kjær, *et al.*, "A multimodal sleep foundation model developed with 500k hours of sleep recordings for disease predictions,". https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11838666/.
- [24] B. Fox, J. Jiang, S. Wickramaratne, P. Kovatch, M. Suarez-Farinas, N. A. Shah, A. Parekh, and G. N. Nadkarni, "A foundational transformer leveraging full night, multichannel sleep study data accurately classifies sleep stages,".
- [25] G. Deng, M. Niu, *et al.*, "A unified flexible large polysomnography model for sleep staging and mental disorder diagnosis,". https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11661386/.
- [26] B. Sun and K. Saenko, "Deep CORAL: Correlation Alignment for Deep Domain Adaptation." July, 2016. http://arxiv.org/abs/1607.01719. arXiv:1607.01719 [cs].
- [27] B. B. Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty, "Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 447–463. 2018.
- [28] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. Kwoh, X. Li, and C. Guan, "Adast: Attentive cross-domain EEG-based sleep staging framework with iterative self-training," *IEEE Transactions on Emerging Topics in Computational Intelligence* 7 (2021) 210–221.
- [29] R. M. Gray, "Toeplitz and circulant matrices: A review," *Foundations and Trends*® *in Communications and Information Theory* **2** no. 3, (2006) 155–239. http://dx.doi.org/10.1561/010000006.
- [30] P. Welch, "The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on audio and electroacoustics* **15** no. 2, (1967) 70–73.
- [31] M. Agueh and G. Carlier, "Barycenters in the wasserstein space," SIAM Journal on Mathematical Analysis 43 no. 2, (2011) 904–924.
- [32] R. Bhatia, T. Jain, and Y. Lim, "On the bures-wasserstein distance between positive definite matrices," *Expositiones Mathematicae* 37 no. 2, (2019) 165–191.
- [33] B. Jessie P., T. Ali, R. Michael, W. Wei, A. Robert, M. Atul, O. Robert L., A. Amit, D. Katherine, and P. Sanya R., "Gastric Banding Surgery versus Continuous Positive Airway Pressure for Obstructive Sleep Apnea: A Randomized Controlled Trial," *American journal of respiratory and critical care medicine* 197 no. 8, (Apr., 2018) . https://pubmed.ncbi.nlm.nih.gov/29035093/. Publisher: Am J Respir Crit Care Med.
- [34] C. L. Rosen, E. K. Larkin, H. L. Kirchner, J. L. Emancipator, S. F. Bivins, S. A. Surovec, R. J. Martin, and S. Redline, "Prevalence and risk factors for sleep-disordered breathing in 8- to 11-year-old children: association with race and prematurity," *The Journal of Pediatrics* 142 no. 4, (Apr., 2003) 383–389.
- [35] S. Redline, P. V. Tishler, T. D. Tosteson, J. Williamson, K. Kump, I. Browner, V. Ferrette, and P. Krejci, "The familial aggregation of obstructive sleep apnea," *American Journal of Respiratory and Critical Care Medicine* 151 no. 3 Pt 1, (Mar., 1995) 682–687.
- [36] C. L. Rosen, D. Auckley, R. Benca, N. Foldvary-Schaefer, C. Iber, V. Kapur, M. Rueschman, P. Zee, and S. Redline, "A multisite randomized trial of portable sleep studies and positive airway pressure autotitration versus laboratory-based polysomnography for the diagnosis and treatment of obstructive sleep apnea: the HomePAP study," *Sleep* 35 no. 6, (June, 2012) 757–767.
- [37] T. Blackwell, K. Yaffe, S. Ancoli-Israel, S. Redline, K. E. Ensrud, M. L. Stefanick, A. Laffan, K. L. Stone, and Osteoporotic Fractures in Men Study Group, "Associations between sleep architecture and sleep-disordered breathing and cognition in older community-dwelling men: the Osteoporotic Fractures in Men Sleep Study," *Journal of the American Geriatrics Society* 59 no. 12, (Dec., 2011) 2217–2225.
- [38] A. P. Spira, T. Blackwell, K. L. Stone, S. Redline, J. A. Cauley, S. Ancoli-Israel, and K. Yaffe, "Sleep-disordered breathing and cognition in older women," *Journal of the American Geriatrics Society* 56 no. 1, (Jan., 2008) 45–50.
- [39] G.-Q. Zhang, L. Cui, R. Mueller, S. Tao, M. Kim, M. Rueschman, S. Mariani, D. Mobley, and S. Redline, "The National Sleep Research Resource: towards a sleep data commons," *Journal of the American Medical Informatics Association: JAMIA* 25 no. 10, (Oct., 2018) 1351–1358.
- [40] A. Goldberger, L. Amaral, *et al.*, "Components of a new research resource for complex physiologic signals," *PhysioNet* **101** (01, 2000) .

- ging PREPRINT
- [41] S. Chambon, M. Galtier, P. Arnal, G. Wainrib, and A. Gramfort, "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series," arXiv:1707.03321 [stat.ML].
- [42] J. B. Stephansen, A. N. Olesen, *et al.*, "Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy," *Nature Communications* **9** no. 1, (Dec, 2018).
- [43] S. Appelhoff, M. Sanderson, *et al.*, "MNE-BIDS: Organizing electrophysiological data into the BIDS format and facilitating their analysis," *Journal of Open Source Software* **4** no. 44, (2019) 1896.
- [44] A. Gramfort, M. Luessi, *et al.*, "MEG and EEG data analysis with MNE-Python," *Frontiers in Neuroscience* 7 no. 267, (2013) 1–13.
- [45] M. Perslev, M. Jensen, S. Darkner, P. J. Jennum, and C. Igel, "U-time: A fully convolutional network for time series segmentation applied to sleep staging," Advances in Neural Information Processing Systems 32 (2019).
- [46] C. Yang, M. Westover, and J. Sun, "Biot: Biosignal transformer for cross-data learning in the wild," in Advances in Neural Information Processing Systems, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds., vol. 36, pp. 78240–78260. Curran Associates, Inc., 2023. https://proceedings.neurips. cc/paper\_files/paper/2023/file/f6b30f3e2dd9cb53bbf2024402d02295-Paper-Conference.pdf.
- [47] D. P. Kingma, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980 (2014).
- [48] A. Paszke, S. Gross, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035. Curran Associates, Inc., 2019.
- [49] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," Journal of Machine Learning Research 7 no. 1, (2006) 1–30. http://jmlr.org/papers/v7/demsar06a.html.
- [50] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering* **9** no. 3, (2007) 90–95.
- [51] F. Pedregosa, G. Varoquaux, et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research 12 (2011) 2825–2830.
- [52] C. Harris, K. Millman, et al., "Array programming with NumPy," Nature 585 no. 7825, (2020) 357-362.
- [53] P. Virtanen, R. Gommers, et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods* 17 (2020) 261–272.
- [54] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, pp. 234–241, Springer. 2015.
- [55] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball, "Deep learning with convolutional neural networks for EEG decoding and visualization," *Human Brain Mapping* **38** no. 11, (Nov., 2017) 5391–5420. http://arxiv.org/abs/1703.05051. arXiv:1703.05051 [cs].

## **A** Appendix

#### A.1 Proof of the Bures-Wasserstein geodesic (6) between covariance matrices of structure (2)

**Proposition A.1.** Let  $\Sigma^{(s)}$  and  $\Sigma^{(t)}$  be two covariance matrices in  $\mathbb{R}^{cF \times cF}$  following (2). Let us denote  $\mathbf{P}^{(s)}$  and  $\mathbf{P}^{(t)}$  the corresponding PSD matrices. The geodesic associated with the Bures-Wasserstein metric between  $\Sigma^{(s)}$  and  $\Sigma^{(t)}$  and parameterized by  $\alpha \in [0, 1]$  is  $\Sigma(\alpha)$  following (2) of PSD

$$\mathbf{P}(\alpha) = \left( (1-\alpha) \, \mathbf{P}^{(s)\odot\frac{1}{2}} + \alpha \mathbf{P}^{(t)\odot\frac{1}{2}} \right)^{\odot 2} \, .$$

*Proof.* From [32], the geodesic associated with the Bures-Wasserstein metric between two covariance matrices  $\Sigma^{(s)}$  and  $\Sigma^{(t)}$  is given by

$$\gamma(\alpha) = (1 - \alpha)^2 \Sigma^{(s)} + \alpha^2 \Sigma^{(t)} + \alpha (1 - \alpha) \left[ (\Sigma^{(s)} \Sigma^{(t)})^{\frac{1}{2}} + (\Sigma^{(t)} \Sigma^{(s)})^{\frac{1}{2}} \right].$$
(9)

where

$$(\boldsymbol{\Sigma}^{(s)}\boldsymbol{\Sigma}^{(t)})^{\frac{1}{2}} = \boldsymbol{\Sigma}^{(s)^{\frac{1}{2}}} \left(\boldsymbol{\Sigma}^{(s)^{\frac{1}{2}}}\boldsymbol{\Sigma}^{(t)}\boldsymbol{\Sigma}^{(s)^{\frac{1}{2}}}\right)^{\frac{1}{2}} \boldsymbol{\Sigma}^{(s)^{-\frac{1}{2}}}.$$
(10)

Since  $\Sigma^{(s)}$  and  $\Sigma^{(t)}$  diagonalize in the unitary basis  $\mathbf{I}_c \otimes \mathbf{F}_F$ ,  $\gamma(\alpha)$  also diagonalizes in this basis. Thus, we only have to compute the geodesic between the PSD matrices  $\mathbf{P}^{(s)}$  and  $\mathbf{P}^{(t)}$  and from now on, all operations are element-wise. Let  $\mathbf{P}(\alpha)$  be the PSD of  $\gamma(\alpha)$ , we have

$$\mathbf{P}(\alpha) = (1-\alpha)^{2} \mathbf{P}^{(s)} + \alpha^{2} \mathbf{P}^{(t)} + \alpha (1-\alpha) \left[ (\mathbf{P}^{(s)} \odot \mathbf{P}^{(t)})^{\odot \frac{1}{2}} + (\mathbf{P}^{(t)} \odot \mathbf{P}^{(s)})^{\odot \frac{1}{2}} \right]$$
(11)

$$= (1 - \alpha)^{2} \mathbf{P}^{(s)} + \alpha^{2} \mathbf{P}^{(t)} + 2\alpha (1 - \alpha) (\mathbf{P}^{(s)} \odot \mathbf{P}^{(t)})^{\odot \frac{1}{2}}$$
(12)

$$= \left( (1-\alpha)\mathbf{P}^{(s)\odot\frac{1}{2}} + \alpha\mathbf{P}^{(t)\odot\frac{1}{2}} \right)^{\odot 2}.$$
(13)

This concludes the proof.

#### A.2 Balanced datasets

In the main paper, we report results across different training set sizes. Since the datasets are highly imbalanced (*e.g.*, ABC has 44 subjects, SHHS has 5,730), we create balanced subsets by randomly selecting up to N subjects per dataset. This avoids over-representing the largest dataset and ensures greater diversity in the training data. We consider four values of N: 40, 100, 200, and 400. The average number of subjects in each balanced set is shown in Table 3. Notably, the balanced set with 400 subjects contains roughly four times less data than the full dataset.

#### A.3 U-Time: CNN for time series segmentation

Table 3: Number of samples in the balanced datasets. Average and standard deviation (across LODO) are computed over 10 datasets left-out from the training set.

Balanced datasets	Number of subjects
Balanced@40	$360\pm0$
Balanced@100	$787 \pm 19$
Balanced@200	$1387\pm63$
Balanced@400	$2466 \pm 157$
All subjects	$9929 \pm 1659$

U-Time [45, 15] is a convolutional neural network (CNN) inspired by the U-Net architecture [54], designed for segmenting

temporal sequences. U-Time maps sequential inputs of arbitrary length to sequences of class labels on a freely chosen temporal scale. The architecture is composed of several encoder and decoder blocks, with skip connections between them.

**Encoder blocks** A single encoder block is composed of a convolutional layer, an activation function, a BatchNorm layer, and a max pooling layer. First, the convolution is applied to the input signal, followed by the activation function and the BatchNorm layer. Finally, the max pooling layer downsamples the temporal dimension. In the following, the pre-BatchNorm feature map is denoted G and the post-BatchNorm feature map  $\tilde{G}$ , *i.e.*,  $\tilde{G} \triangleq$  BatchNorm (G). Each encoder block downsamples by 2 the signal length but increases the number of channels.

**Decoder blocks and Segmentation Head** The decoding part of U-Time is symmetrical to the encoding part. Each decoder block doubles the signal length and decreases the number of channels. It is composed of a convolutional layer, an activation function, a BatchNorm layer, an upsampling layer and a concatenation layer of the skip connection of the corresponding encoding block. Finally, the segmentation head applies two convolutional layers with an activation function in between to output the final segmentation. It should be noted that U-Time employs BatchNorm layers but other normalization layers, such as LayerNorm [7] or InstanceNorm [8] are possible.

**Implementation** The architecture is inspired from Braindecode [55]. The implementation is improved to make it more efficient and faster. One epoch of training takes about 30 min on a single H100 GPU.

#### Architecture: CNNTransformer

The CNNTransformer is a hybrid architecture designed for multichannel time series classification inspired by transformers for EEG-Data [20, 21, 46, 23]. It combines convolutional feature extraction with long-range temporal modeling via a Transformer encoder at epoch-level. The model processes an input tensor of shape (B, S, C, T), where B is the batch size, S is the number of temporal segments, C is the number of input channels, and T is the number of time samples per segment. It outputs a tensor of shape  $(B, n_{classes}, S)$ , where  $n_{classes}$  is the number of classes and S is the number of epochs.

The architecture consists of the following components:

- **Reshaping:** The input is first permuted and reshaped to a 3D tensor of shape  $(B, C, S \cdot T)$  to be compatible with 1D convolutional layers applied along the temporal dimension.
- **CNN-based Feature Extractor:** A stack of 10 Conv1D layers, each followed by ELU activation and Batch Normalization. Some layers use a stride greater than 1 to progressively reduce the temporal resolution. This block extracts local temporal patterns and increases the representational capacity up to a dimensionality of  $d_{\text{model}}$ .
- Adaptive Pooling: An AdaptiveAvgPool1D layer reduces the temporal length to a fixed number of steps (S), independent of the input sequence length. This step ensures a consistent temporal resolution before the Transformer.
- **Positional Encoding:** Learnable positional embeddings of shape  $(1, S, d_{model})$  are added to the feature representations to preserve temporal ordering before passing through the Transformer encoder.
- **Transformer Encoder:** A standard Transformer encoder composed of *L* layers, each consisting of multi-head self-attention and a feedforward sublayer. This module models global temporal dependencies across the *S* steps.
- Classification Head: After transposing the data to shape  $(B, d_{model}, S)$ , a final 1D convolution with a kernel size of 1 projects the output to  $n_{classes}$ , yielding predictions for each epoch segment.

The model is trained end-to-end using standard optimization techniques. The use of adaptive pooling and self-attention enables it to generalize across variable-length inputs while maintaining temporal resolution. A full summary of the architecture is provided in Table 4.

#### A.4 Equation for BatchNorm and InstanceNorm

**BatchNorm** The BatchNorm layer [6] normalizes features maps in a neural network to have zero mean and unit variance. At train time, given a batch  $\mathcal{B} = {\mathbf{G}^{(1)}, \dots, \mathbf{G}^{(N)}} \subset \mathbb{R}^{c \times \ell}$  of N pre-BatchNorm feature maps and for all  $j, m, l \in [1, N] \times [1, \ell] \times [1, \ell]$ , the BatchNorm layer is computed as

$$\widetilde{G}_{m,l}^{(j)} = \gamma_m \frac{G_{m,l}^{(j)} - \widehat{\mu}_m}{\sqrt{\widehat{\sigma}_m^2 + \varepsilon}} + \beta_m , \qquad (14)$$

where  $\gamma, \beta \in \mathbb{R}^c$  are learnable parameters. The mean and standard deviation  $\hat{\mu} \in \mathbb{R}^c$  and  $\hat{\sigma} \in \mathbb{R}^c$  are computed across the time and the batch,

$$\widehat{\mu}_{m} \triangleq \frac{1}{N\ell} \sum_{j=1}^{N} \sum_{l=1}^{\ell} G_{m,l}^{(j)},$$

$$\widehat{\sigma}_{m}^{2} \triangleq \frac{1}{N\ell} \sum_{j=1}^{N} \sum_{l=1}^{\ell} \left( G_{m,l}^{(j)} - \widehat{\mu}_{m} \right)^{2}.$$
(15)

Stage	Operation	Details	Output Shape
Input	Raw signal	Multichannel EEG signal with $S$ segments and $T$	(B, S, C, T)
		time samples per segment	
Reshape	Permute & flatten	Rearranged as $(B, C, S \cdot T)$ to process with 1D	$(B, C, S \cdot T)$
		convolutions	
Feature Extractor	1D CNN stack	10-layer sequence of Conv1D $\rightarrow$ ELU $\rightarrow$ Batch-	$(B, d_{\text{model}}, T')$
		Norm; includes temporal downsampling via stride	
Temporal Pooling	AdaptiveAvgPool1D	Downsamples to fixed temporal resolution defined	$(B, d_{\text{model}}, S)$
		by $S$	
Positional Encoding	Learnable embeddings	Added to temporal dimension to encode temporal	$(B, d_{\text{model}}, S)$
		order before transformer layers	
Transformer Encoder	Multi-head attention	2 Transformer layers with $d_{\text{model}}$ embedding di-	$(B, d_{\text{model}}, S)$
		mension, $n_{\text{head}}$ heads, and feedforward sublayers	
Classifier	Linear projection	Projects feature vectors to class logits at each epoch	$(B, n_{\text{classes}}, S)$
		time step	

Table 4: Architecture overview of the CNNTransformer model. In pratice,  $d_{\text{model}}$  is set to 768,  $n_{\text{head}}$  to 8, and S is 35.

At test time, the mean and variance  $\hat{\mu}$  and  $\hat{\sigma}$  are replaced by their running mean and variance, also called exponential moving average, estimated during training.

**InstanceNorm** Another popular normalization is the InstanceNorm layer [8]. During training, InstanceNorm operates similarly to (14), but the mean and variance are computed per sample instead of across the batch dimension, *i.e.*,  $\hat{\mu}_m^{(j)}$  and  $\hat{\sigma}_m^{(j)}$  are computed for each sample *j*,

$$\widehat{\mu}_{m}^{(j)} \triangleq \frac{1}{\ell} \sum_{l=1}^{\ell} G_{m,l}^{(j)} ,$$

$$(\widehat{\sigma}_{m}^{(j)})^{2} \triangleq \frac{1}{\ell} \sum_{l=1}^{\ell} \left( G_{m,l}^{(j)} - \widehat{\mu}_{m}^{(j)} \right)^{2} .$$
(16)

Hence, each sensor of each sample is normalized independently of the others. At test time, InstanceNorm behaves identically to its training phase and therefore does not rely on running statistics contrary to the BatchNorm.

#### A.5 F1 score vs. Balanced Accuracy

In the main paper, we report Balanced Accuracy scores, which account for class imbalance in sleep stage classification. Prior work, such as the U-Time paper [45], uses the F1 score to evaluate performance. In Table 5, we report F1 scores on the left-out datasets. These scores are slightly higher than the Balanced Accuracy scores and are comparable to those reported in the U-Time paper.

Our main findings remain consistent: BatchNorm and InstanceNorm are the strongest baselines and achieve the best performance on 3 out of 10 datasets. PSDNorm outperforms all other methods on 7 out of 10 datasets. The same trend holds for the balanced@400 setup, where PSDNorm again outperforms all baselines on 7 datasets, while InstanceNorm is never the top performer.

These results confirm that our implementation achieves state-of-the-art performance in sleep stage classification. Moreover, PSDNorm maintains its advantage even in data-limited settings

#### A.6 Impact of Whitening and Target Covariance

As explained in the main paper, InstanceNorm is a special case of PSDNorm with F = 1 and an identity target covariance matrix (i.e., whitening). PSDNorm extends this by (i) using temporal context with F > 1, and (ii) mapping the PSD to a target covariance matrix, such as a barycenter (i.e., colorization).

In this section, we evaluate the impact of whitening on the performance of PSDNorm, to assess the benefit of using a barycenter as the target covariance matrix. Table 6 reports results on 10 datasets (balanced@400), with and without whitening.

Whitening improves performance on only one dataset (CCSHS), while projecting to the barycenter yields the best results on 6 datasets.

Table 5: **F1 scores of different methods on the left-out datasets.** The lower section displays results for training over datasets balanced @400 *i.e.*, **small-scale dataset**, while the upper section presents results for training over all subjects *i.e.*, **large-scale dataset**. The best scores are highlighted in **bold**. The reported standard deviations indicate performance variability across 3 seeds.

		BatchNorm	LayerNorm	InstanceNorm	PSDNorm(F=5)	PSDNorm(F=9)	PSDNorm(F=17)
	ABC	$\textbf{79.80} \pm \textbf{0.34}$	$77.86 \pm 0.80$	$78.36 \pm 1.20$	$78.08 \pm 0.78$	$76.93 \pm 1.29$	$77.44 \pm 0.58$
	CCSHS	$88.32\pm0.49$	$87.22\pm0.51$	$88.73 \pm 0.52$	$88.79 \pm 0.99$	$87.86 \pm 1.08$	$\textbf{88.91} \pm \textbf{0.28}$
	CFS	$87.01 \pm 0.18$	$85.61\pm0.16$	$\textbf{87.62} \pm \textbf{0.27}$	$87.06 \pm 0.77$	$87.26 \pm 0.42$	$86.52\pm0.99$
cts	CHAT	$66.56 \pm 1.42$	$61.32 \pm 2.25$	$64.19 \pm 4.63$	$\textbf{71.86} \pm \textbf{0.95}$	$68.61 \pm 5.36$	$71.12 \pm 1.09$
je	HOMEPAP	$76.20 \pm 1.25$	$76.15\pm1.13$	$77.66 \pm 0.58$	$\textbf{77.85} \pm \textbf{1.29}$	$77.57\pm0.68$	$77.18 \pm 1.25$
sut	MASS	$76.06 \pm 1.69$	$73.95\pm5.80$	$76.94 \pm 1.12$	$77.16 \pm 1.73$	$\textbf{78.50} \pm \textbf{1.82}$	$76.86 \pm 4.06$
Ę	MROS	$83.69 \pm 0.39$	$82.22 \pm 1.27$	$83.95\pm0.53$	$83.51\pm0.84$	$\textbf{84.63} \pm \textbf{1.02}$	$83.96 \pm 0.87$
<'	PhysioNet	$\textbf{76.26} \pm \textbf{1.27}$	$70.40\pm0.14$	$73.84 \pm 0.93$	$73.51\pm3.05$	$73.80 \pm 1.24$	$74.76\pm0.50$
	SHHS	$76.98 \pm 0.70$	$75.98 \pm 0.22$	$79.12\pm0.96$	$\textbf{79.26} \pm \textbf{1.35}$	$78.65 \pm 1.04$	$77.69 \pm 0.88$
	SOF	$85.49\pm0.58$	$84.23 \pm 1.30$	$85.50\pm0.86$	$84.14 \pm 1.05$	$\textbf{85.54} \pm \textbf{0.16}$	$85.33\pm0.16$
	Mean	$79.64 \pm 0.41$	$77.57\pm0.73$	$79.59\pm0.25$	$\textbf{80.12} \pm \textbf{0.57}$	$79.94 \pm 0.80$	$79.98\pm0.47$
	ABC	$81.00\pm0.11$	$79.50\pm0.49$	$80.56 \pm 0.39$	$\textbf{81.12} \pm \textbf{0.37}$	$80.80\pm0.22$	$80.90\pm0.31$
	CCSHS	$\textbf{89.83} \pm \textbf{0.19}$	$89.01\pm0.43$	$89.39\pm0.16$	$89.13\pm0.17$	$89.22\pm0.17$	$89.45\pm0.57$
	CFS	$88.30 \pm 0.52$	$87.39\pm0.06$	$88.45\pm0.17$	$\textbf{88.52} \pm \textbf{0.15}$	$88.32\pm0.34$	$88.23 \pm 0.47$
9	CHAT	$65.77 \pm 4.06$	$65.25\pm3.96$	$71.35\pm2.75$	$72.16\pm2.21$	$71.93\pm3.01$	$\textbf{72.99} \pm \textbf{4.01}$
@ 7	HOMEPAP	$77.06 \pm 0.14$	$76.62 \pm 1.06$	$77.50\pm0.46$	$77.30\pm0.24$	$77.78\pm0.62$	$\textbf{77.98} \pm \textbf{0.90}$
ede	MASS	$\textbf{77.27} \pm \textbf{1.42}$	$74.21 \pm 2.05$	$75.12\pm2.08$	$76.00\pm3.00$	$74.16 \pm 4.21$	$74.10\pm0.47$
nc	MROS	$\textbf{85.53} \pm \textbf{0.48}$	$84.02\pm0.95$	$85.22\pm0.19$	$85.02\pm0.42$	$85.37\pm0.30$	$84.93\pm0.53$
ala	PhysioNet	$74.98 \pm 1.84$	$74.29 \pm 1.50$	$75.07 \pm 1.05$	$75.29 \pm 1.21$	$\textbf{75.41} \pm \textbf{0.68}$	$74.57 \pm 1.16$
В	SHHS	$78.95\pm0.92$	$78.04 \pm 1.21$	$80.30 \pm 1.29$	$\textbf{80.32} \pm \textbf{0.91}$	$79.22 \pm 1.27$	$80.06 \pm 1.30$
	SOF	$86.30 \pm 0.40$	$85.82\pm0.22$	$86.57\pm0.60$	$\textbf{86.99} \pm \textbf{0.33}$	$86.75\pm0.54$	$86.76\pm0.23$
	Mean	$80.50 \pm 0.51$	$79.41 \pm 0.73$	$80.95\pm0.36$	$\textbf{81.19} \pm \textbf{0.11}$	$80.89\pm0.00$	$81.00\pm0.28$

This suggests that, while whitening may help when F = 1, it is less effective when F > 1. Using a barycenter leads to a more robust and stable target covariance matrix.

Table 6: Impact of the whitening on the performance of PSDNorm on the 10 datasets balanced @ 400.

	BatchNorm	InstanceNorm	PSDNorm(F=5)		PSDNorm(F=9)		PSDNorm(F=17)	
		Whitening	Barycenter	Whitening	Barycenter	Whitening	Barycenter	Whitening
ABC	$78.26 \pm 1.33$	$78.73 \pm 0.42$	$78.18\pm0.68$	$77.86 \pm 1.33$	$78.18\pm0.91$	$78.16\pm0.93$	$77.76 \pm 1.00$	$78.34\pm0.33$
CCSHS	$87.42 \pm 0.16$	$87.62\pm0.42$	$87.58\pm0.30$	$87.80\pm0.23$	$87.35 \pm 0.52$	$87.53 \pm 0.21$	$87.62 \pm 0.48$	$\textbf{87.90} \pm \textbf{0.57}$
CFS	$84.32\pm0.57$	$\textbf{84.72} \pm \textbf{0.33}$	$84.29\pm0.36$	$84.01\pm0.60$	$84.06 \pm 0.10$	$84.20\pm0.23$	$84.46 \pm 0.06$	$84.04\pm0.91$
CHAT	$66.55\pm0.88$	$64.43 \pm 4.41$	$\textbf{70.28} \pm \textbf{1.70}$	$69.07\pm3.73$	$68.11 \pm 3.94$	$69.31 \pm 2.50$	$69.88 \pm 0.46$	$69.43 \pm 1.80$
HOMEPAP	$75.25\pm0.50$	$76.47\pm0.63$	$\textbf{76.83} \pm \textbf{0.61}$	$76.13\pm0.93$	$76.61 \pm 0.74$	$76.45\pm0.29$	$76.49 \pm 0.45$	$76.81\pm0.22$
MASS	$70.00 \pm 1.91$	$71.52 \pm 1.13$	$72.77 \pm 1.09$	$69.11 \pm 1.51$	$\textbf{73.07} \pm \textbf{1.30}$	$69.01\pm0.78$	$72.23 \pm 2.40$	$70.41 \pm 1.50$
MROS	$80.37\pm0.20$	$80.28 \pm 0.21$	$80.26\pm0.11$	$80.50\pm0.75$	$80.32 \pm 0.22$	$80.52\pm0.15$	$\textbf{80.70} \pm \textbf{0.42}$	$79.96 \pm 0.53$
PhysioNet	$\textbf{75.81} \pm \textbf{0.13}$	$74.68 \pm 0.55$	$74.82 \pm 2.11$	$74.58 \pm 1.57$	$73.77 \pm 1.73$	$75.08 \pm 2.18$	$75.09 \pm 0.97$	$75.61\pm0.71$
SHHS	$76.44 \pm 0.92$	$78.68 \pm 0.37$	$\textbf{78.88} \pm \textbf{0.68}$	$78.77\pm0.67$	$77.28 \pm 0.91$	$78.85\pm0.16$	$78.41 \pm 0.49$	$77.77 \pm 1.35$
SOF	$81.08 \pm 1.14$	$80.68 \pm 1.38$	$79.49\pm0.41$	$80.10\pm0.62$	$\textbf{81.44} \pm \textbf{0.97}$	$80.53\pm0.05$	$81.07 \pm 0.66$	$79.63\pm0.89$
Mean	$77.55 \pm 0.34$	$77.78\pm0.46$	$78.34 \pm 0.42$	$77.79\pm0.30$	$78.02\pm0.67$	$77.96 \pm 0.12$	$\textbf{78.37} \pm \textbf{0.38}$	$77.99 \pm 0.35$

## A.7 Generalization of PSDNorm in CNNTransformer

The CNNTransformer architecture is a hybrid model that combines convolutional and transformer layers for time series classification.

The main paper presents a critical difference diagram for the CNNTransformer evaluated on datasets balanced@400. It shows that PSDNorm with F = 5 is the best-performing normalization layer.

In Table 7, we report the results of different normalization layers used in the CNNTransformer architecture on datasets balanced@400.

First, we observe that CNNTransformer performs slightly below U-Sleep. Second, BatchNorm and InstanceNorm are the best performers on one and two datasets respectively, while PSDNorm achieves the best performance on 7 out of 10 datasets.

PSDNorm with F = 5 outperforms BatchNorm by a margin of 0.9 and InstanceNorm by 0.54 in average score.

	BatchNorm	InstanceNorm	PSDNorm(F=5)	PSDNorm(F=9)	PSDNorm(F=17)
ABC	$\textbf{76.99} \pm \textbf{0.53}$	$75.40\pm0.36$	$76.31 \pm 0.46$	$75.55\pm0.82$	$75.12 \pm 0.31$
CCSHS	$86.75\pm0.48$	$87.00\pm0.34$	$86.92\pm0.32$	$86.92\pm0.32$	$\textbf{87.27} \pm \textbf{0.40}$
CFS	$83.32\pm0.35$	$\textbf{83.77} \pm \textbf{0.34}$	$83.71\pm0.29$	$83.47\pm0.61$	$83.68 \pm 0.34$
CHAT	$66.44 \pm 0.49$	$66.40\pm2.55$	$\textbf{70.04} \pm \textbf{0.37}$	$69.70 \pm 2.49$	$67.60 \pm 3.84$
HOMEPAP	$74.81 \pm 1.36$	$\textbf{75.92} \pm \textbf{0.44}$	$75.26\pm0.55$	$75.55\pm0.81$	$75.14 \pm 1.14$
MASS	$71.51\pm0.47$	$71.70 \pm 1.17$	$72.55\pm0.81$	$72.86\pm0.11$	$\textbf{73.26} \pm \textbf{0.26}$
MROS	$79.77\pm0.31$	$79.74 \pm 0.55$	$79.77\pm0.30$	$79.52\pm0.33$	$\textbf{79.83} \pm \textbf{0.52}$
PhysioNet	$72.54\pm0.34$	$74.36\pm0.84$	$74.95\pm0.41$	$\textbf{75.32} \pm \textbf{1.08}$	$75.19 \pm 1.25$
SHHS	$75.34\pm0.34$	$76.55\pm0.92$	$\textbf{77.26} \pm \textbf{0.57}$	$76.44 \pm 0.99$	$76.64 \pm 0.80$
SOF	$80.63\pm0.60$	$80.78\pm0.54$	$80.31\pm0.90$	$\textbf{80.84} \pm \textbf{0.30}$	$80.65 \pm 1.00$
Mean	$76.81\pm0.23$	$77.16\pm0.24$	$\textbf{77.71} \pm \textbf{0.22}$	$77.62\pm0.37$	$77.44 \pm 0.53$

Table 7: Different normalization layers used in the CNNTransformer architecture for datasets balanced@400.

These results highlight that PSDNorm is a plug-and-play normalization layer that can be seamlessly integrated into various architectures to reduce feature space variability.

## A.8 Critical Difference Diagram for U-Sleep on all subjects

The main paper presents the critical difference diagram for U-Sleep on the dataset balanced@400. Figure 6 extends this analysis to all subjects across datasets. The conclusion remains consistent: PSDNorm with F = 5 is the best-performing normalization layer, while BatchNorm performs the worst. Interestingly, PSDNorm with F = 17 ranks second to last, suggesting that overly strong adaptation can hurt performance when the dataset is large.



Figure 6: Critical difference diagram for U-Sleep on all subjects.