xSTAE: Explaining Classifier Decisions through EEG Signal Style Transfer Autoencoding

Natalia Koliou[1,*,†], Panagiotis Zazos[2,†], Christoforos Romesis[1,†], Cristian Bosch[3], Stasinos Konstantopoulos[1] and Panagiotis Trakadas[2]

[1]*Institute of Informatics and Telecommunications, NCSR 'Demokritos', Ag. Paraskevi, Greece*
[2]*Four Dot Infinity, Athens, Greece*
[3]*CeADAR, University College Dublin, Dublin, Irelard*

Abstract

Style transfer methods are a powerful visualization tool that can be used to generate counterfactual explanations, plausible alternatives to the original input that leads to a different classification. In this paper we present xSTAE, a system that restyles a misclassified example into the correct class, in order to help the expert understand what patterns the classifier was looking for to assign the correct class, and failed to see in the instance. The system is based on an Autoencoder trained on a loss function that balances between identity loss (similarity with the original instance) and a classification loss derived from a pre-trained classifier, allowing xSTAE to remain completely agnostic with respect to the internals of the classifier it interprets. We present promising experimental results on sleep-stage classification decisions over EEG data, which validate the core of the idea and show future research directions.

Keywords

EEG, Style Transfer, Autoencoders, Sleep Stage Classification, Generative AI, Explainable AI

1. Introduction

Explainable AI (XAI) has gained significant attention in the last few years, mainly due to the wide application of deep learning models in high-stake domains, prominently including healthcare where understanding model decisions can directly impact patient needs, treatment and overall well-being. In practice, XAI helps users not only determine whether to trust individual predictions, but also compare different models and identify areas for improvement when a model performs poorly [1].

While most XAI research has focused on image and tabular data, time-series remain relatively under-explored. One possible reason regarding this gap is that, unlike images, the semantics of time-series cannot be easily visualized [2] and their interpretation requires combining domain expertise with an understanding of temporal dependencies. This gap has considerable impact in the healthcare domain, where bio-signals are widely used in clinical practice. *Electroencephalography (EEG)* signals for instance—which are the focus of this paper—are widely used in clinical practice to monitor brain activity and support the diagnosis of neurological conditions [3], sleep and mental disorders [4, 5], and cognitive impairments [6].

EXPLIMED 2025 - Second Workshop on Explainable Artificial Intelligence for the Medical Domain - 25-30 October 2025, Bologna, Italy

*Corresponding author.
†These authors contributed equally.
nataliakoliou@iit.demokritos.gr (N. Koliou); pzazos@fourdotinfinity.com (P. Zazos); chris.romesis@iit.demokritos.gr (C. Romesis); cristian.boschserrano@ucd.ie (C. Bosch); konstant@iit.demokritos.gr (S. Konstantopoulos); ptrak@fourdotinfinity.com (P. Trakadas)
https://www.linkedin.com/in/natalia-koliou-b37b01197/ (N. Koliou); https://www.linkedin.com/in/panagiotis-zazos-ba1a02188/ (P. Zazos); https://www.linkedin.com/in/cristian-bosch/ (C. Bosch)
0009-0004-3920-9992 (N. Koliou); 0009-0004-2127-9714 (P. Zazos); 0009-0001-6485-5548 (C. Romesis); 0000-0002-2962-4226 (C. Bosch); 0000-0002-2586-1726 (S. Konstantopoulos); 0000-0002-5146-5954 (P. Trakadas)

In this paper we propose a method for generating instance-based interpretations of sleep-stage classification decisions over EEG data. Our method is based on training a generative AI model to interpret a pre-trained classifier by giving counterfactual examples of what a misclassified instance *should* have looked like to be correctly classified. Observing such examples can help the operator observe the patterns that lead to misclassifications and create a basis for a more focused collection and annotation of further training data.

In the remainder of this paper we first discuss related work (Section 2) and formally state the problem (Section 3). We then present our method (Section 4) and proceed to provide and discuss experimental results (Section 5) and conclude (Section 6).

2. Related Work

While deep learning is used for a variety of tasks, classification has been a primary focus within the XAI research community due to its broad applicability and conceptual simplicity. Relevant research has primarily focused on image and tabular data, where model interpretability is more intuitive and visual explanations are easier to generate [7, 8, 9].

As attention has only recently turned to the, more challenging, problem of explaining time-series classifiers, the literature on this topic remains relatively limited [10, 2]. In many cases, the need to interpret decisions drives the development of the classifier itself, with LAXCAT and PatchX being characteristic examples. LAXCAT [11] is a deep neural network architecture that simultaneously identifies the time intervals and the variables of a multi-variate time-series that contribute to classification decisions. LAXCAT uses convolutional layers to extract features from the time series, and two attention modules to identify which variables and time intervals are most important for the classification. PatchX [12] divides time series into smaller patches and performs fine-grained classification on each patch using deep neural networks. These patch-level results are then combined with a traditional classifier to generate the final prediction, and explanations are provided by interpreting the importance of each patch in the overall decision.

A different line of research couples the training of the classifier with the training of its interpreter. Gee et al. [13] propose a prototype-based approach for explaining deep time-series classifiers by learning diverse and representative latent prototypes that highlight class-discriminative patterns across multiple modalities, including ECG, respiration, and audio waveforms. They achieve this by integrating a prototype learning mechanism directly into the training process, encouraging the model to associate input instances with learned prototypes that reflect meaningful and diverse latent representations for each class. XTF-CNN [14] is a dual-channel convolutional neural network that learns representations of microseismic waveforms from both the time and frequency domains to improve classification of rock fracturing events. XTF-CNN is integrated with EUG-CAM that generates fine-grained, gradient-based activation maps over input waveforms to illustrate which parts of the signal influence the model's decisions. Finally, DeepVix [15] is a visual analytics system designed to explain Long Short-Term Memory (LSTM) networks applied to high-dimensional multivariate time series data. DeepVix provides interactive visualizations of the LSTM architecture, including node activations and gate weights across layers and time steps, enabling users to investigate intermediate computations and trace how different input variables contribute to the model's predictions.

Naturally, the lines of research above require a tight coupling between the classifier and its interpretation module, restricting the range of possible networks that can be used to those for which an appropriate interpretation module has been devised. On the other hand, timeXplain [16] is a post-hoc, model-agnostic explanation framework for time series classifiers based on SHAP. timeXplain uses domain-specific *perturbation strategies* organized by time, frequency, and statistical mappings. By observing the effect of perturbing the inputs to the results, timeXplain evaluates feature importance. timeXplain demonstrates that certain mappings (e.g., time slices with noise replacements) can produce explanations that are more faithful than some

model-specific methods.

Focusing in the EEG domain, Apicella et al. [17] evaluated several established XAI methods to explain machine learning models trained on EEG data for emotion recognition, aiming to address the dataset shift problem, a common issue in Brain-Computer Interfaces where EEG signal characteristics vary across recording sessions causing models trained on one session to perform poorly on others. They applied these XAI techniques to identify which EEG signal components the models rely on and tested how consistently these important features appear within and across different recording sessions of the same subjects. Their results showed that many relevant features detected by XAI remain stable across sessions, indicating that these explanations can be used to improve the generalization and robustness of EEG-based classification systems.

Zanola et al. [18] developed xEEGNet, a compact and fully interpretable neural network for EEG-based dementia classification that transforms a traditional "black box" model (ShallowNet) into a "white box" by progressively modifying its architecture to highlight explainable components. They achieve interpretability by designing the network to learn EEG band-specific filters and spatial topographies, which correspond to meaningful brain signal features clinicians can relate to dementia pathology. This approach not only reduces the number of parameters by over 200 times—helping to resist overfitting—but also allows direct inspection of the learned kernels and weights to provide clear, medically relevant explanations for the model's decisions.

Hussain et al. [19] developed machine learning models to classify human activities—resting, motor, and cognitive—using EEG spectral features collected from healthy individuals. They applied the model-agnostic explainability technique LIME to interpret which EEG features most influenced the classification decisions, providing clinically relevant insights into brain activity during these tasks. Their results showed strong classification performance and meaningful explanations that could support improved patient monitoring and rehabilitation.

3. Problem Statement

According to the taxonomy introduced by Theissler et al. [2], *counterfactual explanations* are considered a promising instance-based approach for interpreting time-series classifiers. A counterfactual is a plausible alternative to the original input that leads to a different classification, while remaining as similar as possible to the original. By comparing the original time series with its counterfactual, one can infer which parts of the signal had the greatest impact on the model's decision.

Let $D = \{x^{(i)}\}_{i=1}^{N}$ be an EEG dataset of N sequences, where each sequence $x^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^d$ is represented as a d-dimensional vector, and $C : \mathcal{X} \to \mathcal{Y}$ a trained classifier that maps each input $x \in \mathcal{X}$ to one of n discrete class labels in $\mathcal{Y} = \{1, 2, \ldots, n\}$. Given an input x, the classifier outputs a label $y = C(x)$. Our goal is to provide an explanation for why the classifier assigned that specific label to the input. This becomes particularly insightful in cases of misclassification, where understanding what made the classifier predict an incorrect label can reveal class-specific patterns that were strong enough to override the correct label. Identifying the distinguishing characteristics that the model associates with the incorrect class provides a means to explain its decision-making process.

Formally, we define the problem as follows: Given that C is an imperfect classifier, we focus on the subset of inputs $x_m \in \mathcal{X}$ for which the predicted label $C(x_m)$ differs from the true label $y^*(x_m)$. For each such misclassified instance, we aim to identify the minimal modification x'_m of the input such that the classifier's prediction aligns with the ground truth, i.e., $C(x'_m) = y^*(x_m)$, while ensuring that the modification is as small as possible according to a chosen distance metric $d(\cdot, \cdot)$. This can be expressed as the optimization problem:

$$x'_m = \arg\min_{z \in \mathcal{X}} \; d(x_m, z) \quad \text{so that} \quad C(z) = y^*(x_m) \tag{1}$$

Here, x'_m serves as a *counterfactual* that reveals how the original input x_m would need to change to be correctly classified. By analyzing the difference $\Delta x_m = x'_m - x_m$, we can identify dominant patterns associated with class $C(x_m)$ and gain insight into the model's decision boundaries.

4. Proposed Methodology

To generate meaningful counterfactuals, we propose a generative framework based on a set of class-conditional autoencoders, where each autoencoder is trained to reconstruct inputs from any class, while restyling them toward a specific target class $y_{\text{tgt}} \in \mathcal{Y}$.

For every target class, we train a separate autoencoder $E_{\text{tgt}} : \mathcal{X} \to \mathcal{X}_{\text{tgt}}$. Given an input x and a target label $y_{\text{tgt}} \neq C(x)$, the corresponding autoencoder E_{tgt} generates a counterfactual $x' = E_{\text{tgt}}(x)$ that closely resembles x, but is modified just enough for C to classify it as y_{tgt}, i.e., $C(x') = y_{\text{tgt}}$. By comparing the original input x with its counterfactual x', one can identify the patterns in x that were responsible for the classifier's original decision, and thus detect the characteristics of class $C(x)$ that were most prominent.

Training each of these autoencoders requires incorporating two forms of feedback:

1. The first evaluates how well the generated output approximates the original input. This can be quantified using a similarity function $d(x, x')$, where $x \in \mathcal{X}$ is the original input and x' is the reconstructed output. The goal is to keep $d(x, x')$ sufficiently small to ensure the generated output remains close to the original.
2. The second evaluates how well the output aligns with the desired target class $y_{\text{tgt}} \in \mathcal{Y}$. This feedback is provided by the classifier C that we aim to explain. During training, the generated output x' is passed through C, and its prediction $C(x')$ is compared against the target label y_{tgt}.

This dual feedback guides the autoencoders towards producing counterfactuals that are both similar to the input and representative of the target class.

4.1. EEG Data

Our methodology is specifically designed for time-series EEG data. Let $e \in \mathbb{R}^{n_s \times n_c}$ denote an EEG segment in the time domain (or epoch), where n_s is the number of time samples and n_c is the number of recording channels. Each epoch corresponds to a fixed-duration window of w seconds of EEG signal recording.

To reduce the complexity of the input data, we transform raw EEG time-series from the time domain to the frequency domain. Given an input signal $e \in \mathbb{R}^{n_s \times n_c}$, we first apply the Fast Fourier Transform (FFT) to obtain its spectral representation. To retain only domain-relevant information, we filter out frequencies outside a predefined range of interest (e.g., those not associated with meaningful brain activity). The remaining frequency band is then divided into $n_{s'}$ non-overlapping segments, each represented by three features: the midpoint frequency, the phase at that frequency, and the average amplitude across the segment. This results in a matrix $e^f \in \mathbb{R}^{n_{s'} \times n_f}$, where each row represents a frequency-region waveform using n_f features.

4.2. Classifier

The classifier is a two-stage convolutional neural network designed to process EEG data in the time domain. The architecture is based on the one proposed by Youness [20] and Esparza-Iaizzo et al. [21]. This architecture captures short-term temporal dependencies by analyzing sequences of consecutive epochs. To predict the label for some epoch $e_t^f \sim e_t$ (for simplicity), the model considers a sequence of k epochs, including the current epoch and the previous $k-1$ epochs:

Table 1
Summary of the classifier architecture.

Component	Description
Base Feature Extractor	Processes each epoch independently: 2 × Conv1D(16, k=5) + ReLU + MaxPool(s=2) + Dropout 2 × Conv1D(64, k=5) + ReLU + MaxPool + Dropout 2 × Conv1D(128, k=3) + ReLU AdaptiveMaxPool + Flatten + Dense(128) + ReLU + Dropout *Output:* 128-dim feature vector
Temporal Sequence Model	Takes k stacked epoch features (sequence length = k): Conv1D(64, k=3) + ReLU + Dropout Conv1D(64, k=3) + ReLU Conv1D(n, k=3) + AdaptiveMaxPool *Output:* n-length logits vector

$$X_t^f = [e_{t-k+1}, e_{t-k+2}, \ldots, e_{t-1}, e_t] \in \mathbb{R}^{k \times n_{s'} \times n_f} \quad \rightarrow \quad \hat{y}_t \in \{1, 2, \ldots, n\} \tag{2}$$

The architecture consists of two primary components as illustrated in Table 1.

4.3. Autoencoder

The autoencoder is a hybrid architecture combining self-attention mechanisms and convolutional operations. The input to E is a sequence of $n_{s'}$ vectors, each of dimensionality $n_f + 1$:

$$X_t^f = e_t \oplus pos \in \mathbb{R}^{n_{s'} \times (n_f+1)} \quad \rightarrow \quad X_{t,tgt}^f \tag{3}$$

The first n_f dimensions correspond to the extracted features, while the last dimension is a positional encoding added to inject a sense of temporal order. Since Transformers inherently treat their input as a set of unordered tokens, this positional encoding, implemented as a simple increasing counter (e.g., 1, 2, 3, ..., $n_{s'}$), allows the model to capture temporal dependencies across the samples in the epoch. A detailed overview of the autoencoder architecture is provided in Table 2.

5. Experiments

We apply our methodology to sleep stage classification using EEG data from the *Bitbrain Open Access Sleep (BOAS)* dataset [22]. This dataset consists of 128 full-night recordings collected from healthy volunteers wearing a two-channel EEG headband ($n_c = 2$). The EEG signals were sampled at 256 Hz and segmented into non-overlapping 30-second epochs, each containing 7,680 samples per channel ($n_s = 7680$). Every epoch was independently scored by three certified sleep experts following the *American Academy of Sleep Medicine (AASM)* guidelines [23]. These guidelines define five standard sleep stages: Wake (W), N1 (light sleep), N2 (intermediate sleep), N3 (deep sleep), and REM (rapid eye movement sleep). Since our focus is on sleep stage classification, we exclude the Wake class and restrict our task to the four sleep-related stages: N1, N2, N3, and REM ($n = 4$). To address typical inter-scorer variability ($\sim 85\%$ agreement [24][25]), a fourth expert reviewed the annotations and produced a consensus label for each epoch. These consensus sleep-stage labels were then aligned with the EEG segments to provide a reliable ground truth for each 30-second window.

5.1. Setup

According to our methodology, the first step involves transforming the Bitbrain EEG data from the time domain to the frequency domain. To achieve this, we pre-process each epoch

Table 2
Summary of the autoencoder architecture.

Component	Description
Attention Mechanism	Learns meaningful temporal dependencies 3 × MultiHeadAttention(num_heads = 1) *Output:* Refined input vector $x_A \in \mathbb{R}^{n_s \times n_f}$
Encoder	Downsamples refined input into a latent representation ConvINReLU(16, k=7, s=1, p=3) ConvINReLU(32, k=4, s=2, p=1) ConvINReLU(64, k=4, s=2, p=1) ConvINReLU(128, k=3, s=2, p=1) ConvINReLU(256, k=3, s=1, p=1) *Output:* Latent vector $z \in \mathbb{R}^{n_z \times 256}$
Bottleneck	Applies deep transformations in latent space N × ConvINReLU(256, 256, k=3, s=1, p=1) *Output:* Transformed latent vector $z \in \mathbb{R}^{n_z \times 256}$
Decoder	Upsamples the latent representation to reconstruct the original input DeconvINReLU(256, 128, k=3, s=1, p=1) DeconvINReLU(128, 64, k=3, s=2, p=1) DeconvINReLU(64, 32, k=4, s=2, p=1) DeconvINReLU(32, 16, k=4, s=2, p=1) ConvTranspose1D(16, F, k=7, s=1, p=3) *Output:* Reconstructed vector $\hat{x} \in \mathbb{R}^{n_s \times n_f}$

(per channel) using the pipeline shown in Figure 1. The raw time-domain signal $e \in \mathbb{R}^{n_s \times n_c}$ (Figure 1a) is first converted to the frequency domain using a Fast Fourier Transform (FFT) (Figure 1b). We then filter out all frequency components outside the 0.4-30 Hz range (Figure 1c), which includes the primary EEG waveforms relevant to sleep staging: Delta (0.5-4 Hz), Theta (4-8 Hz), Alpha (8-13 Hz), and Beta (13-30 Hz) [26].

Next, we split the retained frequency range into $n_{s'}$ equal, non-overlapping segments. From each segment, we extract a representative waveform defined by its midpoint frequency and phase, along with the average amplitude across the segment (Figure 1d). This produces a compact and structured representation of the original time-series in the frequency domain: $e^f \in \mathbb{R}^{n_{s'} \times n_f}$, where $n_f = 6$, corresponding to three features (frequency, phase, amplitude) for each of the two EEG channels. In our experiments, we set $n_{s'} = 300$ to balance expressiveness with computational efficiency.

We split the dataset subject-wise into training, validation, and test sets to ensure that all epochs from a given participant remained within a single split. This way, we prevent subject leakage and preserve the validity of evaluation results. Approximately 80% of the nights (80 participants) were assigned to the training set, 10% (12 participants) to the validation set, and 20% (25 participants) to the test set. To standardize the data, we apply z-score normalization using the mean and standard deviation computed from the training set. These statistics are then used to normalize the validation and test sets accordingly.

$$\tilde{x} = \frac{x - \mu_{\text{train}}}{\sigma_{\text{train}}} \tag{4}$$

5.1.1. Classifier

The sleep stage classifier described in Section 4.2 is the model we aim to explain. It takes as input a sequence of $k = 5$ consecutive EEG epochs in the frequency domain and produces a prediction for the final epoch e_5 (Equation 5).

$$X_t^f = [e_1, e_2, e_3, e_4, e_5] \in \mathbb{R}^{5 \times 300 \times 6} \quad \rightarrow \quad \hat{y}_t \in \{1, 2, 3, 4\} \tag{5}$$

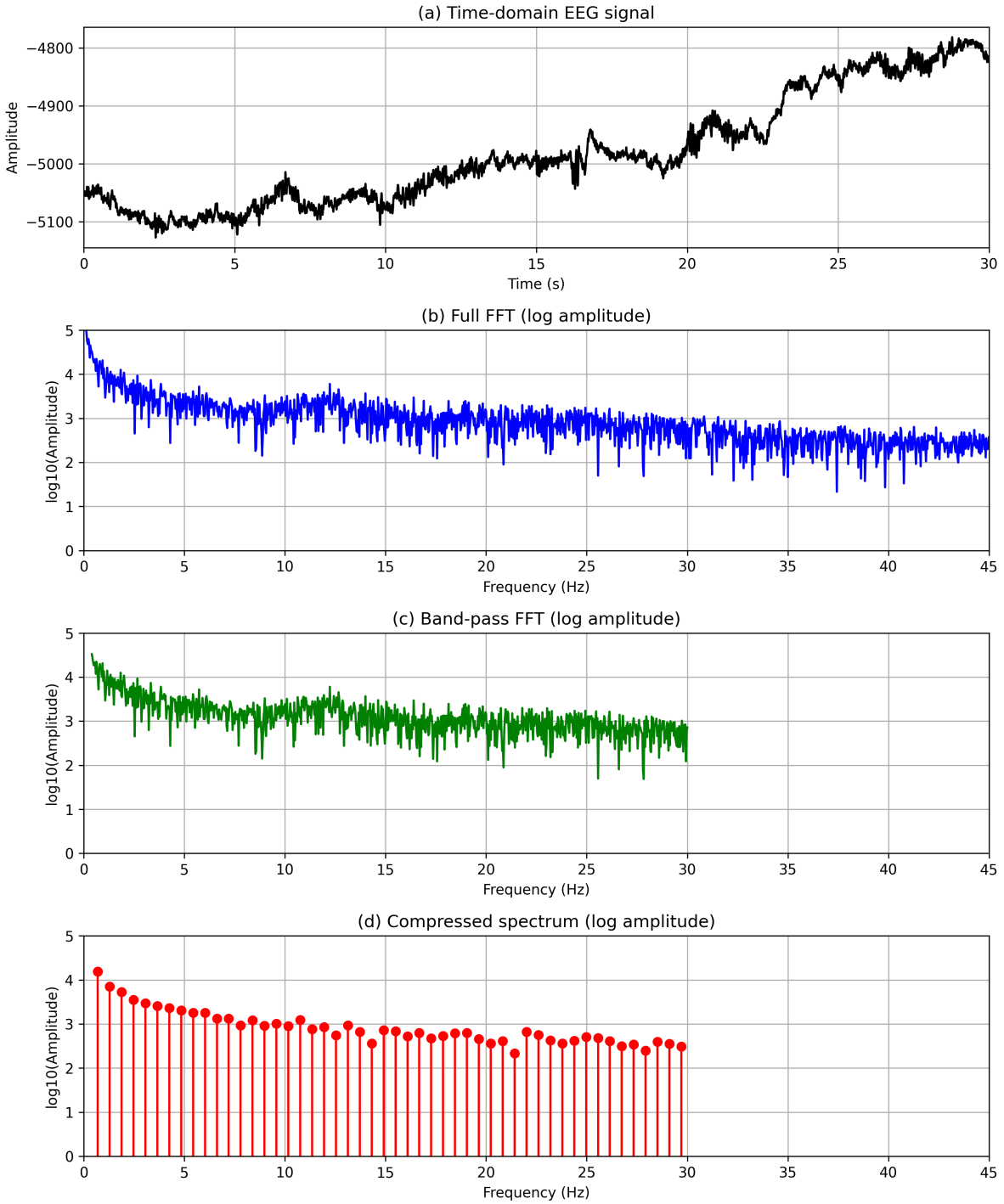


Figure 1: EEG signal preprocessing: **(a)** Time-domain EEG signal e, **(b)** Full-spectrum FFT, **(c)** Band-limited FFT (0.4-30 Hz), **(d)** Compressed frequency-domain spectrum e^f.

A critical part of training this model is choosing a loss function that matches the data distribution and learning objectives. The Bitbrain dataset has a highly imbalanced class distribution: N2 accounts for around 50% of the epochs, while N1 and N3 are less common (about 5% and 20%, respectively), and REM makes up the remaining 25%. This skewed class distribution can bias the model toward over-predicting the majority class, resulting in poor performance on minority classes. To address this pronounced class imbalance, we adopt the sparse categorical focal loss, which modifies the standard cross-entropy loss to emphasize learning

Table 3
Classifier's Hyperparameter Search Space and Optimal Values.

Parameter	Search Space	Optimal Value
base_filters	{16, 32, 64}	64
filter_1	Equal to base_filters	64
filter_2	base_filters × 4	256
filter_3	base_filters × 8	512
seq_mult	{2, 4, 6}	6
seq_filter	base_filters × seq_mult	384
kernel_1	{3, 5, 7}	3
kernel_2	{3, 5, 7}	7
kernel_3	{3, 5, 7}	5
seq_kernel	{3, 5}	5
dropout_conv	(0.1, 0.5)	0.134
dropout_seq	(0.1, 0.5)	0.205
dense_units	(64, 256) with steps of 32	128
optimizer	{Adam, AdamW}	AdamW
learning rate (lr)	5×10^{-5} to 5×10^{-4}	5.92×10^{-5}
focal loss alpha (α)	(0.0, 1.0)	0.50
focal loss gamma (γ)	(0.0, 5.0)	2.95
batch_size	{16, 32, 64}	16

from hard-to-classify examples. The focal loss is defined as:

$$\mathcal{L}_{\text{focal}} = -\sum_{c=1}^{n} \alpha_c (1 - p_c)^{\gamma} \log(p_c), \tag{6}$$

where p_c is the predicted probability for the true class c, α_c is a weighting factor that balances the importance of each class, and $\gamma \geq 0$ is a focusing parameter that reduces the contribution of well-classified examples. By tuning α_c and γ, the focal loss places greater importance on difficult or misclassified instances, enhancing performance on underrepresented classes. Although the loss is defined as a sum over all classes, in practice only the term corresponding to the true class contributes to the loss for each training sample. This is due to the one-hot encoding of the target labels: the correct class is represented with a value of 1, while all other classes are 0. As a result, all terms involving incorrect classes are multiplied by zero and do not affect the outcome.

To identify the optimal configuration for the classifier, we conducted an automated hyperparameter search using the Optuna framework. The goal was to maximize the macro-F1 score on the validation set, which better reflects performance across all sleep stages. We performed 50 trials, with each trial running for up to 20 training epochs. Early stopping was applied if the validation macro-F1 did not improve for 5 consecutive epochs. The search space included a variety of architectural and training parameters, presented in the middle column of Table 3. These ranged from convolutional filter sizes and kernel widths to dropout rates, dense layer units, learning rate, and optimizer choice. The focal loss parameters α and γ were also included to fine-tune the loss function's sensitivity to class imbalance. The optimal hyperparameters identified by Optuna's best trial are presented in the rightmost column of the same table.

Figure 2 reveals that the most influential hyperparameters were `base_filters` and `dense_units`. The focal loss parameters α and γ also had great impact.

During evaluation, our classifier achieved an overall accuracy of 0.87. Table 4 provides a detailed per-class performance breakdown, including precision, recall, F1-score, and support.

To benchmark our approach, we compare against the work of Esparza-Iaizzo et al. [21], who performed non-causal, single-channel sleep stage decoding directly on raw EEG signals, including the Wake stage. Comparing the confusion matrices in Figure 3, we note that Light Sleep (N1) remains the most challenging stage to classify, with our model achieving a recall of only 27%.

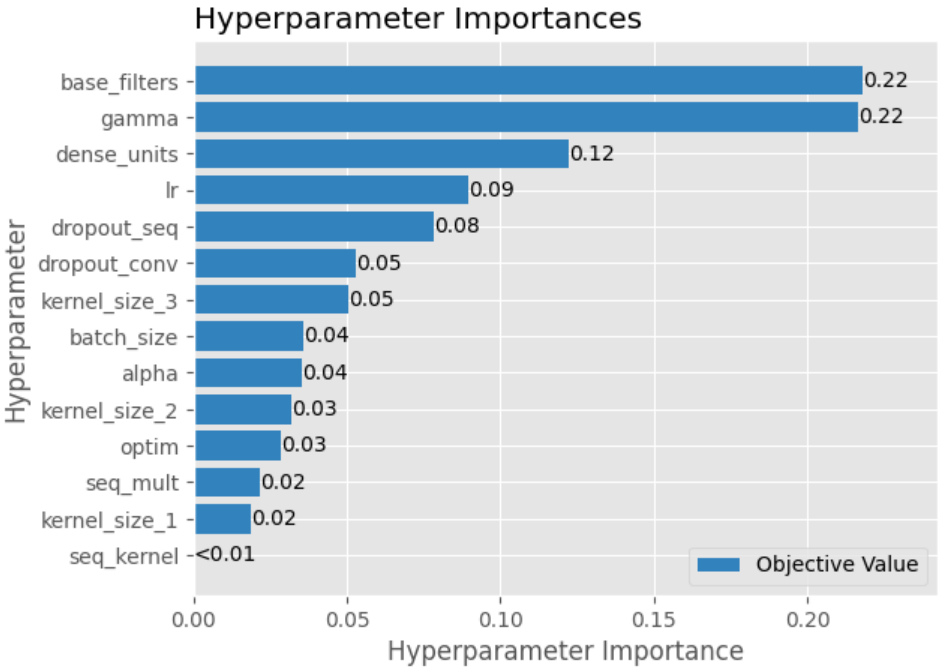


Figure 2: Optuna's Hyperparameter Importance Chart.

Table 4
Classification report including precision, recall, F1-score, and support per class on the test set.

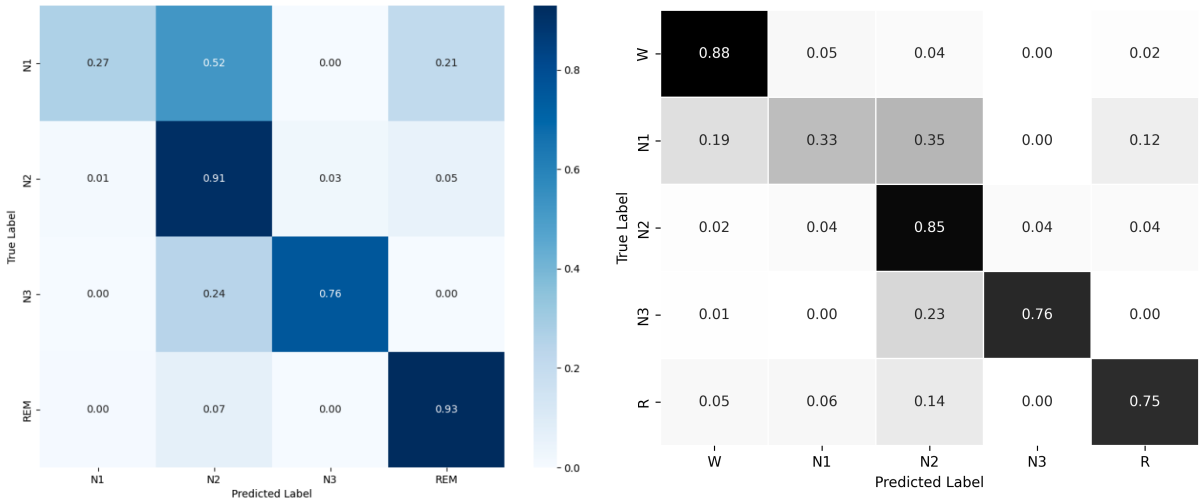
Class	Precision	Recall	F1-score	Support
N1	0.65	0.27	0.38	1267
N2	0.94	0.91	0.92	17357
N3	0.56	0.76	0.64	725
REM	0.75	0.93	0.83	3870
Accuracy			**0.87**	**23219**
Macro avg	0.72	0.72	0.69	23219
Weighted avg	0.88	0.88	0.87	23219

However, by excluding Wake, our pipeline avoids masking N1 errors within dominant Wake bins, as seen in previous works. Instead, misclassified N1 epochs are primarily confused with N2 (52%) and REM (21%), reflecting the transitional nature of light sleep. Mid-stage non-REM sleep (N2) detection shows considerable improvement, with recall increasing from 85% in the baseline to 91% in our model. Performance on deep slow-wave sleep (N3) remains comparable, with both methods effectively capturing delta-band features. The largest boost is observed in REM sleep detection, where recall rises from 75% to 93%. This suggests that spectral representation combined with sequential modeling better isolates the characteristic EEG patterns of REM.

Together, these results validate that converting raw headband signals into compact, two-channel spectral slices and modeling their temporal evolution over consecutive epochs yields a more discriminative representation for the predominant sleep stages (N2 and REM) without sacrificing accuracy in deep sleep. Light sleep (N1) remains inherently difficult due to its low prevalence ($\sim 5\%$) and high intra-stage variability in healthy adults [27].

5.1.2. Autoencoders

To perform style transfer across the $n = 4$ sleep stages, we train a separate autoencoder E_{tgt} for each target class $y_{\text{tgt}} \in \{1, 2, 3, 4\}$. Each model learns to restyle input epochs from any class to



(a) Our sleep-stage classifier (b) Esparza-Iaizzo et al.'s classifier

Figure 3: Comparison of confusion matrices between our sleep-stage classifier and the baseline one in [21].

match the target class y_{tgt}. Unlike the classifier, which takes a sequence of epochs as input, the autoencoder processes a single 30-second EEG epoch in the frequency domain. Instead of using all frequency-domain features (frequency, phase, and amplitude), we choose to transform only the amplitude values, as we observed that learning is more effective when focusing on energy patterns alone. This reduces the input dimensionality to $n_f = 3$, corresponding to two amplitude values (one per channel) along with the positional encoding.

$$X_t^f(A) = e_t \oplus \text{pos} \in \mathbb{R}^{300\times 3} \quad \rightarrow \quad X_{t,\text{tgt}}^f(A) \tag{7}$$

After decoding, we reconstruct the full frequency-domain representation by adding back the original frequency and phase features:

$$X_t^f = X_{t,\text{tgt}}^f(A) \oplus X_t^f(F,P) = X_{t,\text{tgt}}^f \in \mathbb{R}^{300\times 7} \tag{8}$$

To train the autoencoders, we combine two loss functions:

$$\mathcal{L}_{\text{ae}} = \lambda_{\text{id}} \cdot \mathcal{L}_{\text{id}} + \lambda_{\text{clf}} \cdot \mathcal{L}_{\text{clf}}, \tag{9}$$

where $\mathcal{L}_{\text{id}}$ is the reconstruction loss computed as the mean squared error (MSE) between the input and its reconstruction, $\mathcal{L}_{\text{clf}}$ is the focal loss defined in Equation 6, and λ_{id} and λ_{clf} are their respective weighting coefficients.

Identity Loss To ensure that the counterfactual remains close to the original input, we compute the identity loss in the time domain. The predicted amplitude-phase-frequency triplets $X_{t,\text{tgt}}^f$ are denormalized and decoded back into the four canonical EEG bands (delta, theta, alpha, and beta). We denote the resulting restyled signal in the time domain as $X_{t,\text{tgt}}$. The identity loss is then computed as the mean squared error between the reconstructed signal and the original time-domain signal:

$$\mathcal{L}_{\text{id}} = MSE(X_t, X_{t,\text{tgt}}) \tag{10}$$

Classification Loss To encourage the output to resemble the target class, we guide the autoencoders' training using the pre-trained classifier. The classifier operates on sequences of $k = 5$ consecutive epochs. For each input epoch e_t, we retrieve four preceding epochs from a donor sequence that belongs to the target class y_{tgt}. We then concatenate the restyled output $X_{t,\text{tgt}}^f$ to

Table 5
Autoencoders' Hyperparameter Search Space and Optimal Values.

Parameter	Search Space	Optimal Value
num_heads	{1, 3, 6}	1
num_attn_layers	{1, 3, 6}	3
num_bot_layers	{1, 3, 6}	6
learning_rate	{1e-4, 5e-5, 1e-5}	5e-5
optimizer	{Adam, AdamW}	AdamW
lambda_clf	{0.2, 0.5, 1.0, 2.0}	1.0
lambda_id	{0.1, 0.5, 0.8, 1.0}	0.8
batch_size	{128, 256, 512}	512

these donor epochs (excluding positional encodings), forming a complete input sequence for the classifier.

$$X_{\text{seq}} = [e^{\text{d}}_{t-4}, e^{\text{d}}_{t-3}, e^{\text{d}}_{t-2}, e^{\text{d}}_{t-1}, X^{f}_{t,\text{tgt}}] \in \mathbb{R}^{5\times 300\times 6} \tag{11}$$

The predicted label $\hat{y}_t$ for the last epoch in the sequence is then compared to the target class y_{tgt} using the focal loss:

$$\mathcal{L}_{\text{clf}} = \mathcal{L}_{\text{focal}}(\hat{y}_t, y_{\text{tgt}}) \tag{12}$$

The autoencoders were trained for a maximum of 1000 epochs using early stopping with a patience of 50. We manually tuned the architecture and training setup based on performance on the validation set. Our experiments included varying model depth, attention configurations, learning rates, and the weighting of loss terms. Table 5 summarizes the hyperparameters explored and the final configuration that yielded the best results.

To better understand the training dynamics, we plotted the training and validation loss curves for each of the four autoencoders. Each figure displays three loss components per model: the total loss, the identity loss, and the classification loss. The identity and classification losses are unweighted (i.e., before applying λ_{id} and λ_{clf}) to provide a clearer view of how each component evolves through the epochs. Figure 4 shows the training and validation loss curves for each autoencoder. Each row corresponds to one autoencoder, with training loss on the left and validation loss on the right.

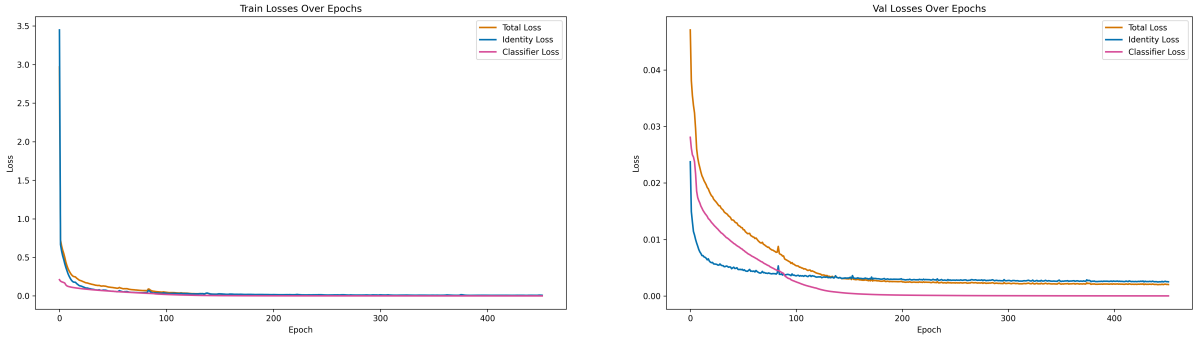
We refer to xSTAE as the complete architecture that combines the autoencoder with the classifier guiding its training. Figure 5 illustrates the entire pipeline, from the input EEG spectrum to the style-transferred output. The top section of the figure depicts the classifier architecture, while the bottom section shows the autoencoder.

5.2. Quantitative Results

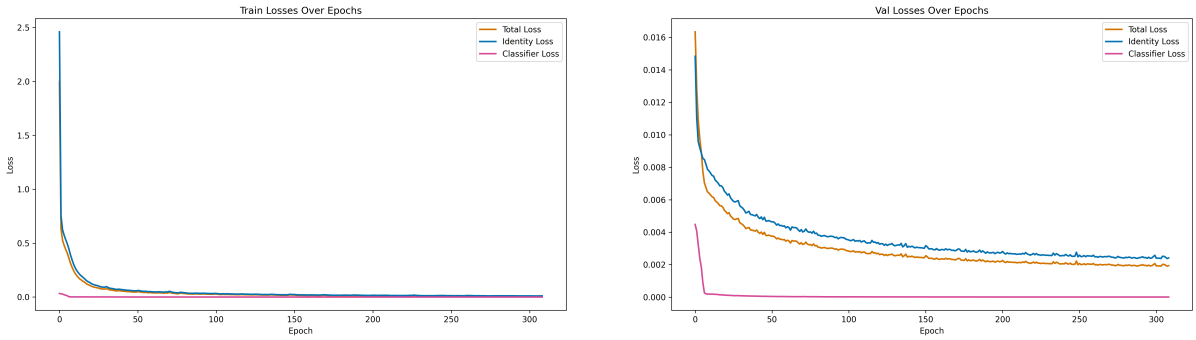
To quantitatively evaluate how well each autoencoder learns the style of its corresponding sleep stage, we pass the entire test dataset through each class-specific autoencoder and then classify the restyled outputs using the original sleep stage classifier. Table 6 shows the classification accuracy for each sleep stage when the test signals are restyled into that stage. The fact that the restyles signals dramatically reduce the classification error, demonstrates that the autoencoders have indeed captured the patterns that the classifier uses in order to recognize sleep stages. That is to say, that xSTAE learns to interpret the answers of *that specific* classifier.

5.3. Qualitative Discussion

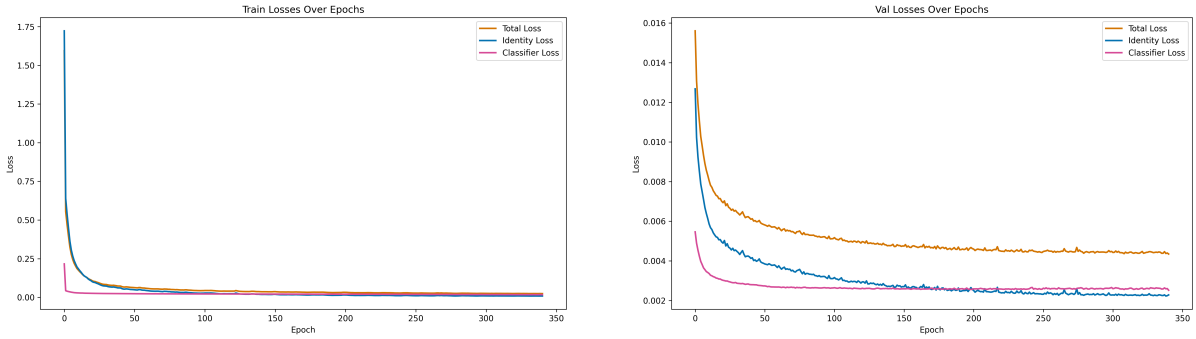
We present three examples of misclassified EEG segments and use xSTAE to explore why the classifier may have assigned the wrong label. By comparing the original signals with their



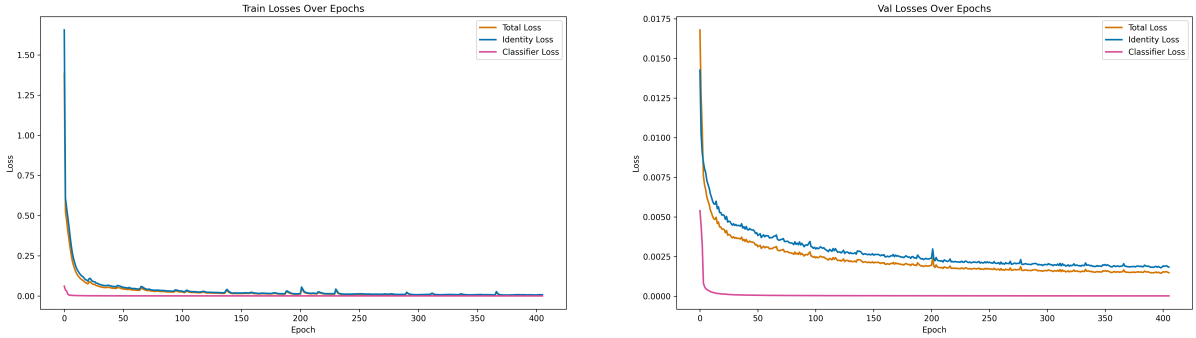
(a) E_{N1} training and validation loss curves.



(b) E_{N2} training and validation loss curves.



(c) E_{N3} training and validation loss curves.



(d) E_{REM} training and validation loss curves.

Figure 4: Training and validation loss curves for each autoencoder. Each row shows the training loss (left) and validation loss (right) for a single autoencoder. The plots include total loss, identity loss, and classification loss.

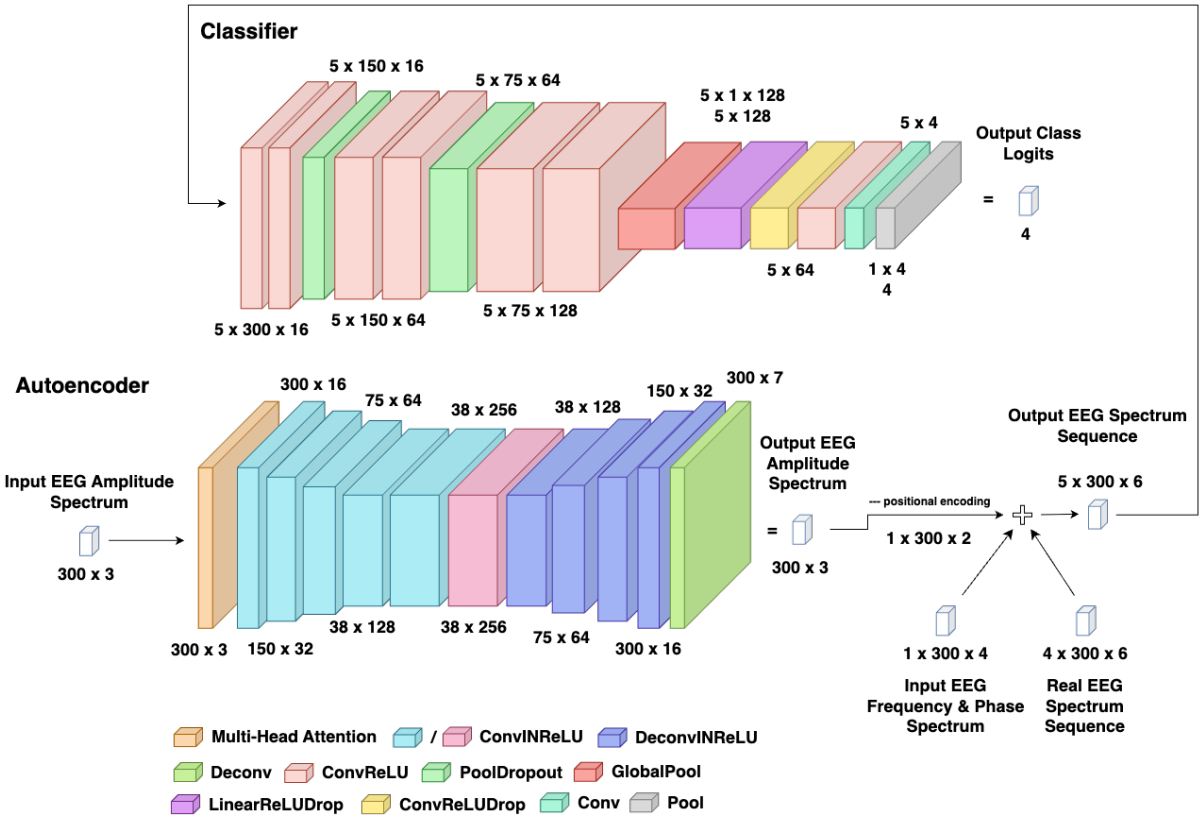


Figure 5: Overview of the xSTAE architecture. The diagram shows the full pipeline, from the input EEG spectrum to the transformed, style-transferred output.

Table 6
Classification report with metrics (rows) across all sleep stages (columns).

Metric	N1	N2	N3	REM
Accuracy	$18371/18376 = 0.99$	$18374/18376 = 0.99$	$17018/18376 = 0.93$	$18351/18376 = 0.99$

restyled counterparts, we aim to uncover the patterns the classifier has implicitly learned to associate with each sleep stage.

In each case, the original EEG fragment is passed through the autoencoder corresponding to the correct target class. The restyled signal represents what the classifier "expects" to see for that class, highlighting which features may have been overlooked in the original. This visual comparison allows us to extract two kinds of insights: (1) why the original signal was misclassified and (2) what changes xSTAE introduced that led the classifier to correctly identify the restyled signal as belonging to the target class. Table 7 summarizes the typical EEG characteristics of each sleep stage, which serve as a reference in our interpretations.

Table 7
Summary of typical EEG characteristics for each sleep stage.

Sleep Stage	EEG Characteristics
N1	Low amplitude, moderate density, almost no spindles.
N2	Low amplitude, high density, clear presence of spindles.
N3	High amplitude, low density, no spindles.
REM	Low amplitude, low density, almost no spindles.

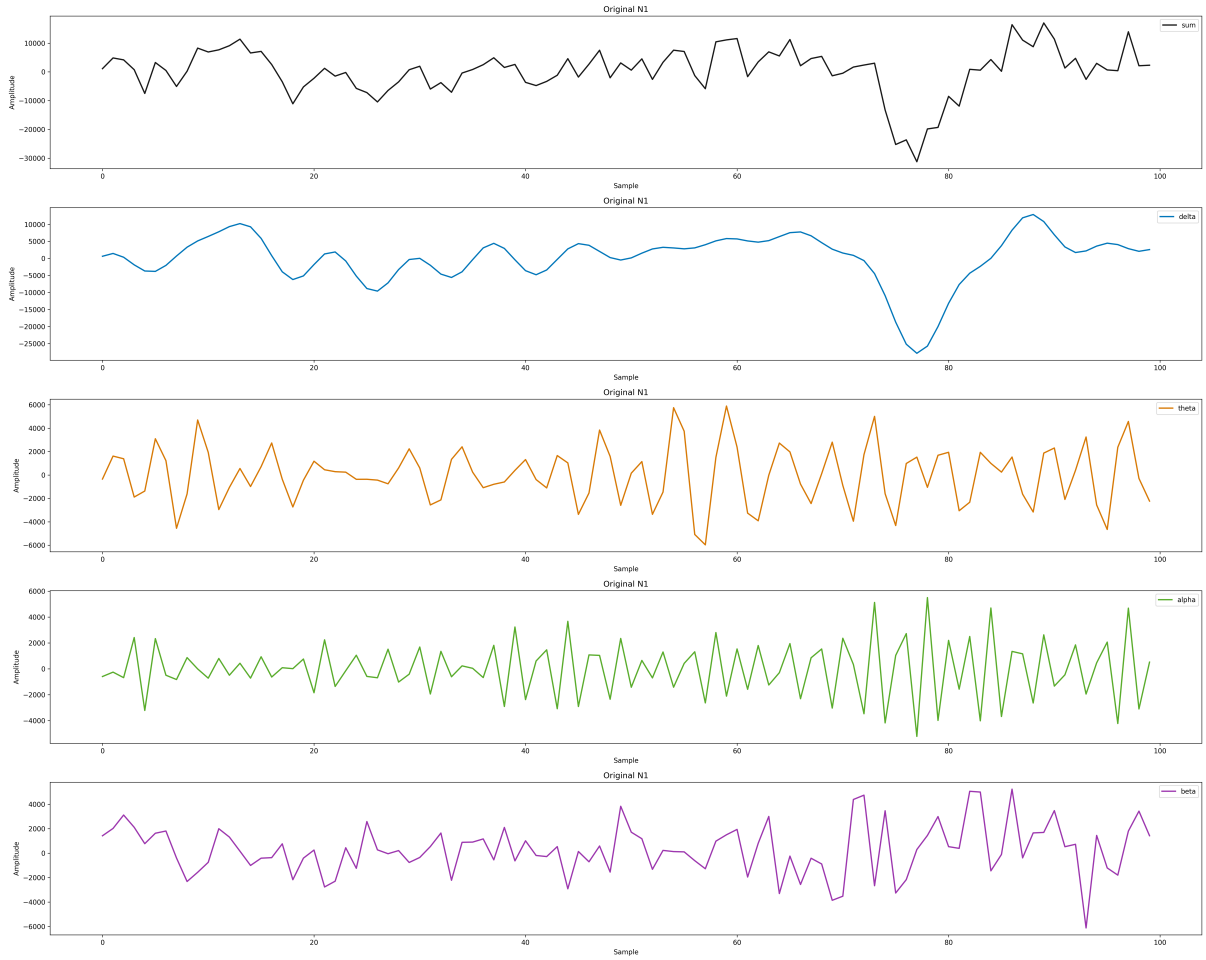


Figure 6: The original signal (top) and its decomposition into the four brainwave bands. This signal is labeled as N2 but was misclassified as N1.

N1 signal misclassified as N2 The EEG segment shown in Fig. 6 was originally labeled as N2 but was misclassified by the model as N1. To interpret this decision, we use xSTAE to restyle the signal into the correct target class (N2), as shown in Fig. 7. This allows us to visualize what the classifier was expecting to see in order to recognize the signal as N2.

Comparing the original and restyled signals, we observe that the restyled version has more pronounced features, e.g., around timesteps 55-60, 70-75, and 90-95, where existing spikes rise more prominently above the baseline. This suggests that the classifier's patterns are overall in the correct direction, but in this case, it did not assign enough importance to the existing features. The restyling reveals that the cause of the misclassification was not that the classifier is looking for the wrong features, but that it does not place enough significance on the spikes it does identify and expects a higher number of spike instances or more prominent spikes.

N3 signal misclassified as N2 The EEG segment shown in Fig. 8 was originally labeled as N3 but was misclassified by the model as N2. To interpret this decision, we again use xSTAE to restyle the signal into the correct target class (N3), as shown in Fig.9.

Comparing the original and restyled signals, we observe that the restyled version shows significantly higher amplitudes, especially around timesteps 60-80 and 80-100, where the fluctuations become more pronounced. This suggests that the classifier had already detected some high-amplitude activity but did not consider it strong enough to classify the signal as N3. The

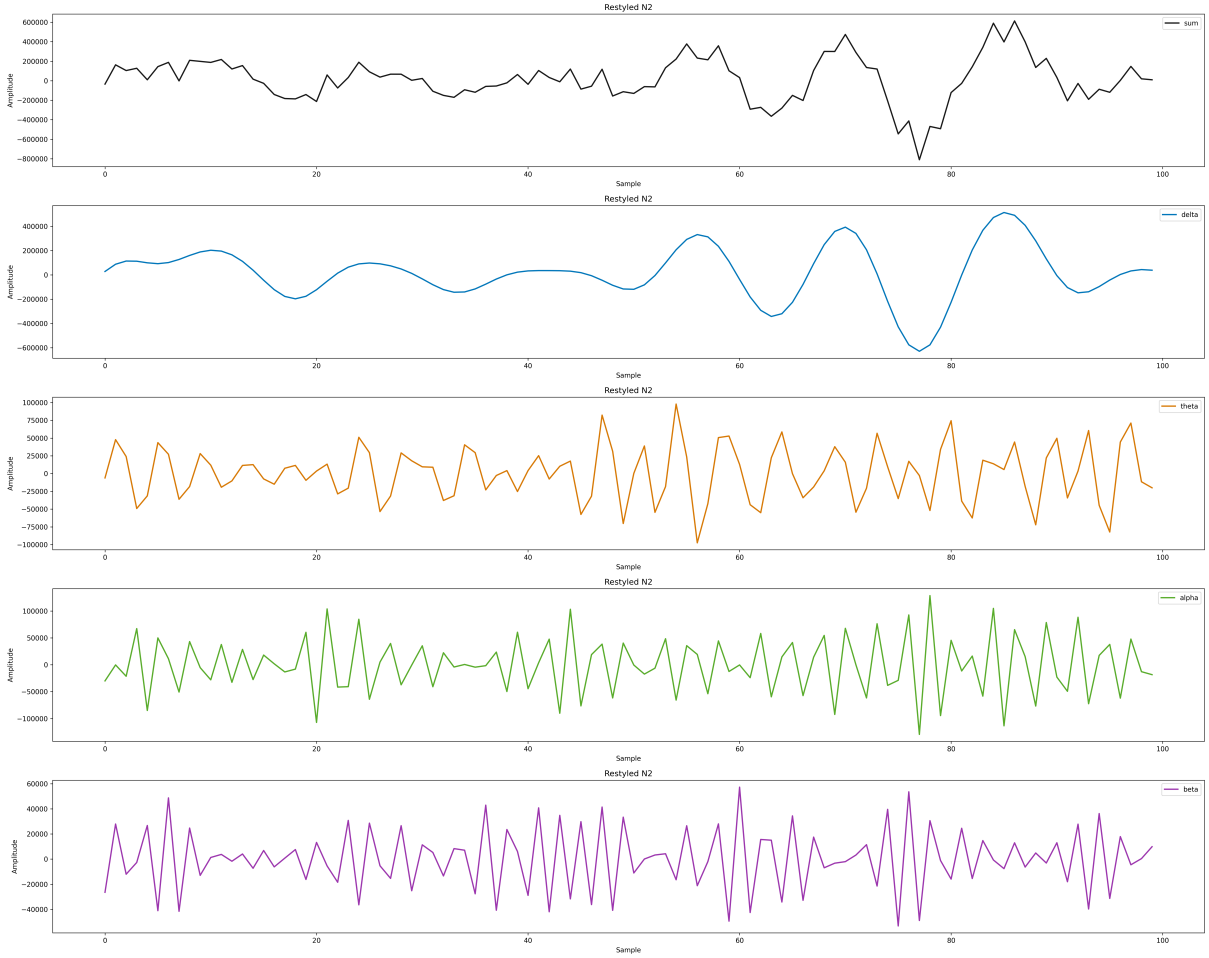


Figure 7: The signal from Fig. 6 after restyling into N2; both the full signal (top) and its decomposition into the four brainwave bands are shown.

restyling process amplifies these features, making them more consistent with the typical N3 pattern. So again, the classifier appears to be looking for the right features (namely, high-amplitude waveforms), but fails to assign sufficient weight to the ones already present.

6. Conclusions and Future Work

We presented xSTAE, system that interprets erroneous decisions by a classifier by restyling an instance where the classifier failed into an instance that would make the classifier give the correct label. At the core of xSTAE is an autoencoder that reconstructs the signal in a way that minimizes a loss that balances between maintaining identity and pushing towards the correct label. This makes the restyled instance not just any example of what the correct class looks like, but what morphological features the classifier was looking for in *this* instance (and didn't find) in order to avoid the error.

The general strategy of providing *counterfactual explanations* is an established method for explaining AI systems, but one of the most developed ones. As demonstrated by recent—mostly visual— advances in Generative AI, this strategy is rapidly approaching within grasp. Our contribution is to (a) ground the general strategy into a specific problem statement for time-series data; (b) identify the spectral representation that work well specifically for EEG signals, where training converges and where the restyled EEG is visually convincing; and (c) validate the

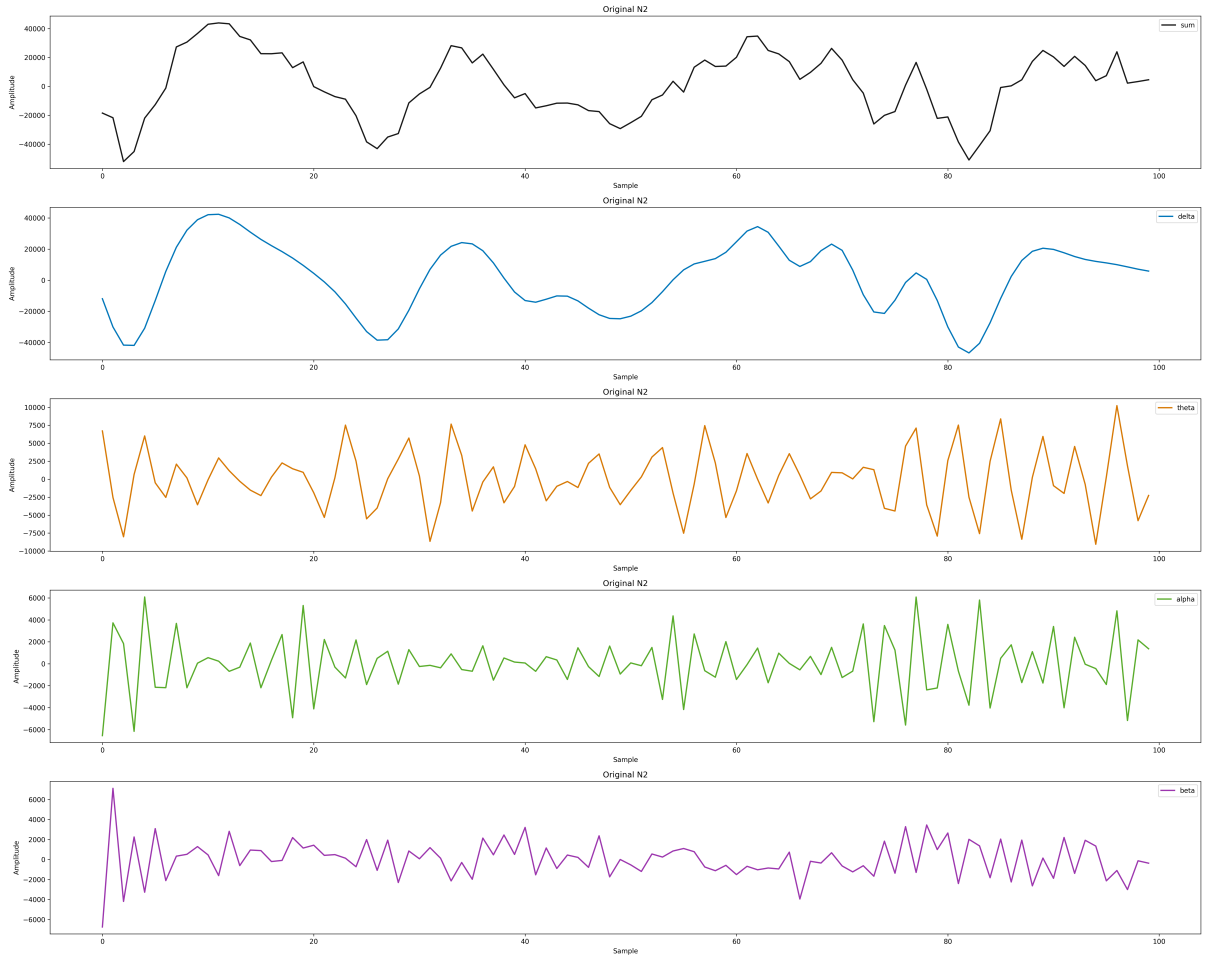


Figure 8: The original signal (top) and its decomposition into the four brainwave bands. This signal is labeled as N3 but was misclassified as N2.

approach on open data and publish the complete experimental setup as open source.[1]

Although the empirical validation was promising, there are some further steps before the system can be validated in trials with experts. Specifically, we plan to further explore different ways to define the identity loss. For one, visual observation has shown that mean square error biases the model towards making small changes everywhere, which makes it harder for the expert to identify what changes have been effected. Before trials, we need to define (and validate the convergence and low classification loss) of alternative identity loss definitions, e.g. preferring bigger local changes or preferring changes that do not affect all four brainwave bands. The trials can then be used to establish with notion of 'identity' makes it easier to spot the changes affected in order to achieve the intended re-labelling.

A further, more ambitious, step is to link the insights extracted from interpreting misclassifications to possible actions for alleviating them. Since xSTAE is specifically designed to be model-agnostic and can be matched to any pre-trained classifier, such actions also need to operate at the same level of abstraction to maintain the generality of the system. In other words, the outcome of an expert's understanding of the classifier's pain-points should operate at the level of re-balancing data or of post-hoc establishing the confidence of specific classification decisions; as opposed to recommending hyper-parameter or architectural changes or similar

[1]The data is the BOAS dataset [28] and the experimental setup is available at https://doi.org/10.5281/zenodo.17085776.

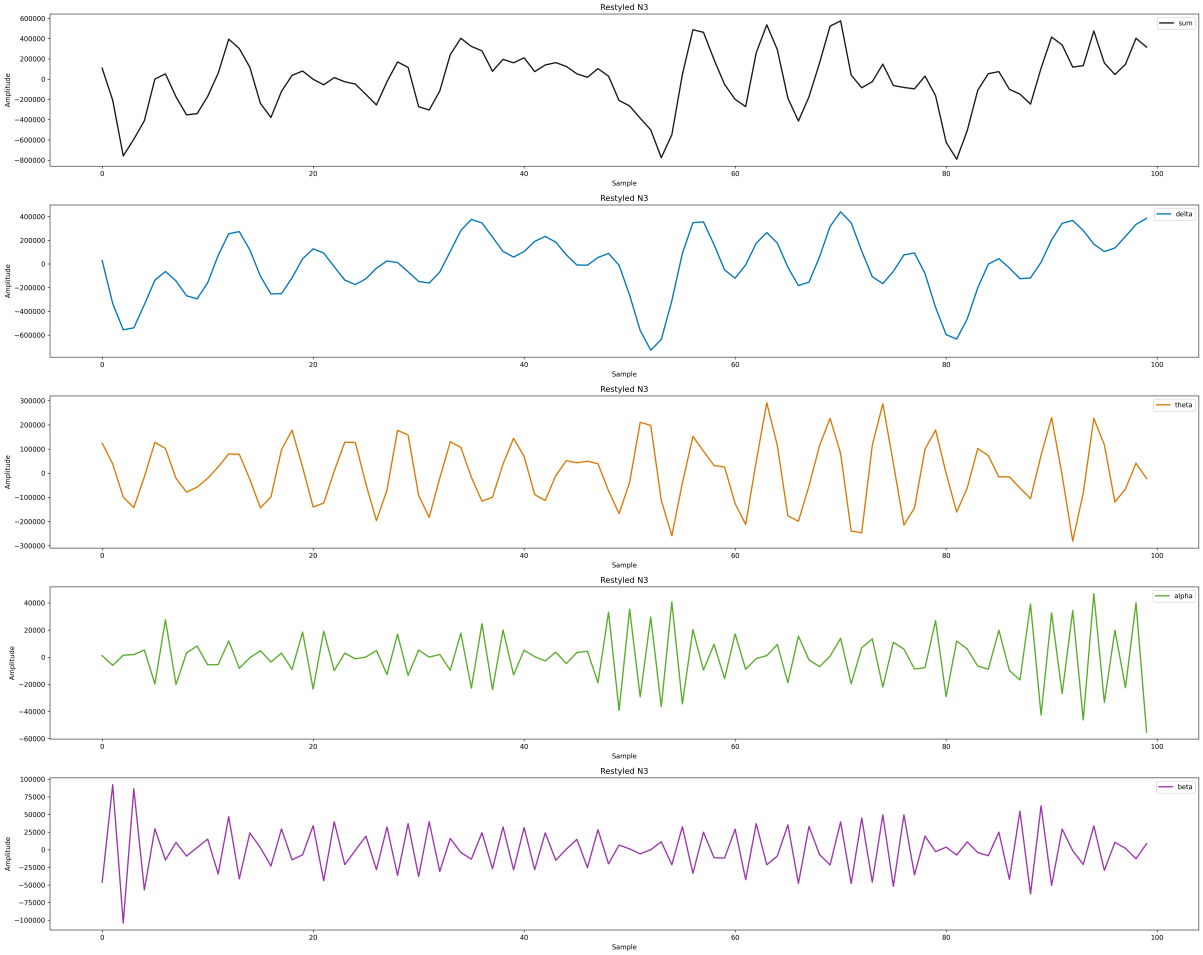


Figure 9: The signal from Fig. 8 after restyling into N3; both the full signal (top) and its decomposition into the four brainwave bands are shown.

classifier-specific actions.

Acknowledgments

This research was co-funded by the European Union under GA no. 101135782 (MANOLO project). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or CNECT. Neither the European Union nor CNECT can be held responsible for them.

Declaration on Generative AI

As the subject of this article is generative AI, the example outputs in Figures 7 and 9 are generated by AI. No generative AI was used to prepare any of the remaining content, either textual or graphical.

References

[1] M. T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier, arXiv:1602.04938 [cs.LG], 2016. URL: https://arxiv.org/abs/1602.04938.

[2] A. Theissler, F. Spinnato, U. Schlegel, R. Guidotti, Explainable AI for time series classification: A review, taxonomy and research directions, IEEE Access PP (2022) 1–1. doi:10.1109/ACCESS.2022.3207765.

[3] M. Allahbakhshi, A. Sadri, S. O. Shahdi, Diagnosis of Parkinson's disease using EEG signals and machine learning techniques: A comprehensive study, arXiv:2405.00741 [eess.SP], 2024. URL: https://arxiv.org/abs/2405.00741.

[4] H.-N. Jo, Y.-S. Kweon, S.-H. Lee, EEG spectral analysis in gray zone between healthy and insomnia, 2024. doi:10.48550/arXiv.2411.09875.

[5] M. Jafari, D. Sadeghi, A. Shoeibi, H. Alinejad-Rokny, A. Beheshti, D. L. García, Z. Chen, U. R. Acharya, J. M. Gorriz, Empowering precision medicine: AI-driven schizophrenia diagnosis via EEG signals: A comprehensive review from 2002-2023, arXiv:2309.12202 [eess.SP], 2023. URL: https://arxiv.org/abs/2309.12202.

[6] A. Ortiz, F. J. Martínez-Murcia, M. A. Formoso, J. L. Luque, A. Sánchez, Dyslexia Detection from EEG Signals Using SSA Component Correlation and Convolutional Neural Networks, Springer International Publishing, 2020, pp. 655–664. URL: http://dx.doi.org/10.1007/978-3-030-61705-9_54. doi:10.1007/978-3-030-61705-9_54.

[7] M. M. Karim, Y. Li, R. Qin, Towards explainable artificial intelligence (XAI) for early anticipation of traffic accidents, arXiv:2108.00273 [cs.CV], 2022. URL: https://arxiv.org/abs/2108.00273.

[8] E. Kadar, G. Gilboa, DXAI: Explaining classification by image decomposition, arXiv:2401.00320 [cs.CV], 2024. URL: https://arxiv.org/abs/2401.00320.

[9] T. Vermeire, D. Martens, Explainable image classification with evidence counterfactual, arXiv:2004.07511 [cs.LG], 2020. URL: https://arxiv.org/abs/2004.07511.

[10] T. Rojat, R. Puget, D. Filliat, J. D. Ser, R. Gelin, N. Díaz-Rodríguez, Explainable artificial intelligence (XAI) on timeseries data: A survey, arXiv:2104.00950 [cs.LG], 2021. URL: https://arxiv.org/abs/2104.00950.

[11] T.-Y. Hsieh, S. Wang, Y. Sun, V. Honavar, Explainable multivariate time series classification: A deep neural network which learns to attend to important variables as well as informative time intervals, arXiv, 2020. URL: https://arxiv.org/abs/2011.11631. arXiv:2011.11631.

[12] D. Mercier, A. Dengel, S. Ahmed, Patchx: Explaining deep models by intelligible pattern patches for time-series classification, in: 2021 International Joint Conference on Neural Networks (IJCNN), IEEE, 2021, pp. 1–8. URL: http://dx.doi.org/10.1109/IJCNN52387.2021.9533293. doi:10.1109/ijcnn52387.2021.9533293.

[13] A. H. Gee, D. Garcia-Olano, J. Ghosh, D. Paydarfar, Explaining deep classification of time-series data with learned prototypes, arXiv:1904.08935 [cs.LG], 2019. URL: https://arxiv.org/abs/1904.08935.

[14] X. Bi, Z. Chao, Y. He, X. Zhao, Y. Sun, Y. Ma, Explainable time-frequency convolutional neural network for microseismic waveform classification, Information Sciences 546 (2021) 883–896. doi:10.1016/j.ins.2020.08.109.

[15] T. Dang, H. Van, H. Nguyen, P. Vung, R. Hewett, DeepVix: Explaining long short-term memory network with high dimensional time series data, in: Proceedings of the 11th International Conference on Advances in Information Technology (IAIT '20), 2020, pp. 1–10. doi:10.1145/3406601.3406643.

[16] F. Mujkanovic, V. Doskoč, M. Schirneck, P. Schäfer, T. Friedrich, timexplain – a framework for explaining the predictions of time series classifiers, arXiv:2007.07606 [cs.LG], 2023. URL: https://arxiv.org/abs/2007.07606.

[17] A. Apicella, F. Isgrò, A. Pollastro, R. Prevete, Toward the application of XAI methods in EEG-based systems, arXiv:2210.06554 [cs.LG], 2024. URL: https://arxiv.org/abs/2210.06554.

[18] A. Zanola, L. F. Tshimanga, F. D. Pup, M. Baiesi, M. Atzori, xEEGNet: Towards explainable AI in EEG dementia classification, arXiv:2504.21457 [cs.LG], 2025. URL: https://arxiv.org/abs/2504.21457.

[19] I. Hussain, R. Jany, R. Boyer, A. Azad, S. A. Alyami, S. J. Park, M. M. Hasan, M. A. Hossain, An explainable EEG-based human activity recognition model using machine-learning approach and lime, Sensors 23 (2023). doi:10.3390/s23177452.

[20] M. Youness, CVxTz/EEG_classification: v1.0, Zenodo, 2020. doi:10.5281/zenodo.4060151.

[21] M. Esparza-Iaizzo, M. Sierra-Torralba, J. G. Klinzing, J. Minguez, L. Montesano, E. López-Larraz, Automatic sleep scoring for real-time monitoring and stimulation in individuals with and without sleep apnea, bioRxiv, 2024. doi:10.1101/2024.06.12.597764.

[22] E. López-Larraz, M. Sierra-Torralba, S. Clemente, G. Fierro, D. Oriol, J. Minguez, L. Montesano, J. G. Klinzing, Bitbrain open access sleep dataset, OpenNeuro, 2024. doi:10.18112/openneuro.ds005555.v1.0.0.

[23] R. B. Berry, R. Brooks, C. Gamaldo, S. M. Harding, R. M. Lloyd, S. F. Quan, M. T. Troester, B. V. Vaughn, AASM Scoring Manual Updates for 2017 (Version 2.4), Journal of Clinical Sleep Medicine 13 (2017) 665–666. doi:10.5664/jcsm.6576.

[24] H. Danker-Hopfe, D. Kunz, G. Gruber, G. Klösch, J. L. Lorenzo, S.-L. Himanen, et al., Interrater reliability between scorers from eight european sleep laboratories in subjects with different sleep disorders, Journal of Sleep Research 13 (2004) 63–69. doi:10.1046/j.1365-2869.2003.00375.x.

[25] R. S. Rosenberg, S. Van Hout, The American Academy of Sleep Medicine inter-scorer reliability program: Sleep stage scoring, Journal of Clinical Sleep Medicine 9 (2013) 81–87. doi:10.5664/jcsm.2350.

[26] L. Cao, Fourier-based spectral analysis of eeg signals for sleep stage classification, Theoretical and Natural Science 109 (2025) 130–140. doi:10.54254/2753-8818/2025.GL24090.

[27] A. Patel, V. Reddy, K. Shumway, et al., Physiology, Sleep Stages, StatPearls Publishing, Treasure Island (FL), 2024. URL: https://www.ncbi.nlm.nih.gov/books/NBK526132/, [Updated 2024 Jan 26].

[28] E. López-Larraz, M. Sierra-Torralba, S. Clemente, G. Fierro, D. Oriol, J. Minguez, L. Montesano, J. G. Klinzing, The Bitbrain open access sleep (BOAS) dataset, OpenNeuro, 2025. doi:10.18112/openneuro.ds005555.v1.1.1.