ETH zürich

medical data science

Self-Supervised Contrastive Learning for EEG-Based Sleep Stage Classification

Signal Augmentations and Downstream Performance

Author: Shaswat Gupta

Medical Data Science Group,
D-INFK, ETH Zurich

Contact: shagupta@ethz.ch

Supervisor: Prof. Dr. Julia Vogt
Advisors: Alice Bizeul, Ami Beuret

Semester Project Report

January 21, 2025

Contents

Abstract

Purpose: Reliable sleep stage classification (SSC) is crucial for understanding human sleep physiology and diagnosing sleep disorders. This study explores a *self-supervised learning (SSL)* framework rooted in *contrastive representation learning (CRL)*, unlocking learning utility from the abundant unlabeled EEG data while minimizing the reliance on manual annotations.

Methods: We systematically evaluate thirteen data augmentations categorized into amplitude, frequency, masking-cropping, noise-filtering, and temporal domains. Incremental experiments—ranging from single augmentations and intra-category combinations to inter-category synergies—identify which transformations yield the highest downstream classification accuracy and Macro-F1. We further analyze augmentation "severity," showing that applying 3–4 well-chosen augmentations strikes an optimal balance between under- and over-distortion.

Results: Masking-Cropping (RandomZeroMasking, CutoutResize), Frequency-Based (TailoredMixup), and select Temporal (TimeWarping, Permutation) transformations emerge as top performers. Fine-tuning an *extremely lightweight CNN* ($\approx$ 200k parameters) yields over *80% accuracy* and *70% Macro-F1* without class imbalance handling, temporal modeling, or large-scale architectures, underscoring the effectiveness of our contrastive pretraining.

Contributions :A systematic evaluation of EEG augmentations for SSC within SSL paradigms. Evidence that contrastive pretraining with carefully selected augmentations significantly boost downstream classification over baseline Gaussian Noise.

Conclusion: These findings highlight the value of targeted data transformations for EEG-based SSL and offer practical insights for future work aiming to further SSC with limited labeled data.

SimpleSleepNet Repository: https://gitlab.ethz.ch/shagupta/simplesleepnet

1 Introduction

1.1 Motivation

Sleep, a physiological cornerstone of health, occupies nearly one-third of human life and governs critical neurophysiological and restorative processes. Accurate sleep stage classification (SSC) is essential for understanding sleep physiology and diagnosing sleep disorders [1]. Traditionally, SSC relies on manual annotation of overnight Electroencephalogram (EEG) recordings—subset of a collection of signals called a Polysomnograph (PSG) recorded in a sleep lab —by experts. This manual process is labor-intensive, time-consuming, and prone to human error and inter-rater variability (Cohen's $\kappa \approx 0.76$), especially in challenging sleep stages like N1 ($\kappa \approx 0.24$) [2].

While supervised Deep Learning (DL) approaches achieve human-comparable accuracy in SSC, they suffer from significant limitations. An over-reliance on labeled datasets yields only marginal performance gains with architectural refinements [3]. The overwhelming majority of EEG signals remain unlabeled due to the prohibitive costs of manual annotation [4], presenting a compelling opportunity for Self-Supervised Learning (SSL)—a paradigm that has redefined state-of-the-art approaches in natural language processing and computer vision [5]. Despite its transformative potential, the application of SSL to SSC is still in its nascent stages, leaving a critical gap in the field.

Contrastive Representation Learning (CRL), a subset of SSL, uses data augmentations to learn

meaningful representations by bringing similar (augmented) data points closer in the latent space while pushing dissimilar ones apart. In the context of EEG signals, CRL involves applying various transformations to unlabeled EEG signals to create positive pairs—different augmented versions of the same signal—and negative pairs—augmented versions from different signals. By optimizing the encoder to maximize the agreement between positive pairs and minimize it for negative pairs, CRL effectively captures the underlying characteristics of EEG signals. [6]

1.2 Background and Gaps

Recent works in this domain include NeuroNet [7], which utilizes a hybrid approach combining two fundamental SSL paradigms: Contrastive Learning (CL) and Masked Prediction (MP). SleePyCo [8] focuses solely on CL, while EEG2REP [9] utilizes Masked Augmentation (MA) in the latent space and compares it to MA applied directly on raw EEG data, as explored in MAEEG [10]. While these studies show promise to improve accuracy metrics to achieve state-of-the-art (SOTA) performance and generalization against inter-patient and recording variability; however, there is a gap in understanding how data augmentations lead SSL paradigms to learn feature representations. There is yet no systematic research on how different data augmentations in the design space of SSL frameworks correlate with the quality of representations learned by pre-trained encoders and how they influence downstream task performance, generalizability, and robustness.

1.3 Research Questions and Contributions

While *SSL-CRL* holds promise for EEG-based sleep stage classification, the specific influence of various data augmentations on latent representations remains underexplored. This thesis examines thirteen EEG augmentations across amplitude, frequency, masking-cropping, noise-filtering, and temporal domains, aiming to quantify their individual and combined impact on single-epoch classification.

Research Questions

1. *Single Augmentation Effectiveness*
   Which individual augmentations most significantly enrich contrastively learned representations for single-epoch EEG classification?

2. *Combinational Synergy or Redundancy*
   Do multiple transformations within the same domain reinforce or overlap each other, and do cross-domain mixes offer complementary or antagonistic effects?

3. *Optimal "Goldilocks" Severity*
   What is the ideal number (or intensity) of active augmentations that yield maximum downstream performance without excessively distorting EEG signals?

Contributions

1. *Comprehensive Augmentation Taxonomy*
   A thorough evaluation of thirteen data augmentations, ranking their standalone and combined effectiveness in an SSL context.

2. *Iterative Experimental Framework*
   A staged series of experiments—covering single, intra-category, and inter-category strategies—to isolate high-impact augmentation sets and assess their synergy.

3. *Feasibility in Lightweight Settings*
   Demonstration that a compact CNN (~0.26M parameters) can achieve >80% accuracy

and ~70% Macro-F1 without temporal modeling, class imbalance handling, or large-scale architectures, validating our approach while highlighting its efficacy for resource-constrained scenarios.

2 Self-Supervised Contrastive Representational Learning

2.1 Mathematical Framework

From [11], Self-Supervised Contrastive Representation Learning (SSL-CRL) uses a discriminative framework to learn meaningful representations by maximizing the similarity between augmented versions of the same data sample (positive pairs) while minimizing it for different samples (negative pairs). The key components of SSL-CRL include data augmentations, feature extraction, and a contrastive loss function.

2.1.1 Similarity Metric

Given feature vectors $\mathbf{u}$ and $\mathbf{v}$, their similarity $s(\mathbf{u}, \mathbf{v})$ is defined as the cosine similarity:

$$s(\mathbf{u}, \mathbf{v}) = \cos(\theta_{\mathbf{u},\mathbf{v}}) = \frac{\mathbf{u}^T \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}, \quad s(\mathbf{u}, \mathbf{v}) \in [0, 1]$$

A value of $s(\mathbf{u}, \mathbf{v})$ close to 1 indicates high similarity (e.g., positive pairs), while a value near 0 indicates dissimilarity (e.g., negative pairs).

2.1.2 Contrastive Loss

The training objective is to maximize the similarity for positive pairs $(\mathbf{z}_i, \mathbf{z}_j)$ and minimize it for negative pairs $(\mathbf{z}_i, \mathbf{z}_k)$. The loss for a single positive pair is given by the normalized temperature-scaled cross-entropy (NT-Xent) loss:

$$\ell(i, j) = -\log \frac{\exp(s(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

where τ is a temperature scaling parameter and N is the batch size.

The overall loss is computed as:

$$L = \frac{1}{2N} \sum_{i=1}^{N} \big(\ell(2i-1, 2i) + \ell(2i, 2i-1)\big)$$

2.1.3 Optimization and Training

The encoder network f maps input signals x to a latent representation $\mathbf{z}$. During training:

1. Two augmentations T_1 and T_2 are applied to each sample x, producing $T_1(x)$ and $T_2(x)$.
2. The encoder extracts features $\mathbf{z}_i = f(T_1(x_i))$ and $\mathbf{z}_j = f(T_2(x_i))$.
3. The contrastive loss is computed over positive and negative pairs within a batch.

2.2 Algorithm

Input: Batch of signals $x_1, x_2, \ldots, x_N$, transformations $T_1(\cdot), T_2(\cdot)$, encoder f, batch size N.

Output: Loss L.

1. *Augmentation*: For each x_i, generate two augmented versions: $T_1(x_i)$ and $T_2(x_i)$.
2. *Feature Extraction*: Encode the augmented samples: $\mathbf{z}_{2i-1} = f(T_1(x_i)), \quad \mathbf{z}_{2i} = f(T_2(x_i)), \quad i \in \{1, \ldots, N\}$.
3. *Similarity Computation*: Compute cosine similarities between all pairs $(\mathbf{z}_i, \mathbf{z}_j)$.
4. *Loss Calculation*: For each positive pair $(\mathbf{z}_{2i-1}, \mathbf{z}_{2i})$, calculate the NT-Xent loss $\ell(2i-1, 2i)$.
5. *Aggregate Loss*: Compute the average loss $L = \frac{1}{2N} \sum_{i=1}^{N} \big(\ell(2i-1, 2i) + \ell(2i, 2i-1)\big)$.
6. *Update*: Backpropagate L to update the encoder f.

2.3 Visualization

The learning framework is visualized in Figure 1, where augmented pairs $T_1(x_r)$ and $T_2(x_r)$ are embedded into a latent space. Positive pairs are drawn closer together, while negative pairs are pushed apart. [4]

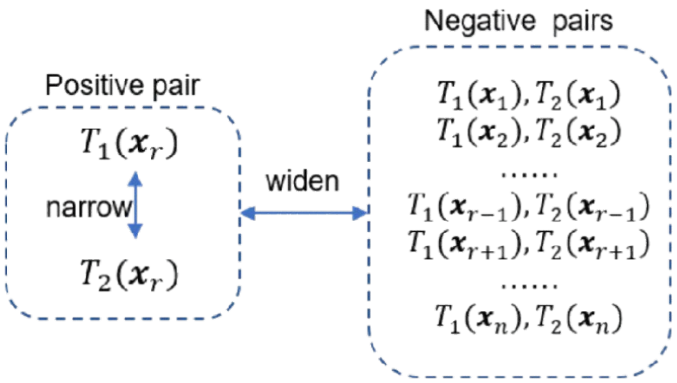


Figure 1: The learning framework visualizes how augmented pairs are embedded into a latent space.

3 Data Augmentations

In this section, we detail the data augmentations used in our study, including a concise description, definition and mathematical formulation. Original and augmented signals are visualized in the Appendix 12.4.

3.1 Amplitude-Based

These techniques modify the amplitude characteristics of the EEG signal. By scaling the signal's amplitude (e.g., RandomAmplitudeScale), adding a constant offset (RandomDCShift), or flipping its polarity (SignFlip), these augmentations simulate variations in signal strength and baseline shifts, improving model robustness to amplitude-related variations. [8][12]

Category	Augmentation	Description
Amplitude-Based	Random Amplitude Scaling	Scales the amplitude of the signal to simulate variations in signal strength.
Amplitude-Based	Random DC Shift	Shifts the signal vertically by adding a constant offset to the baseline.
Amplitude-Based	Sign Flip	Flips the polarity of the signal to introduce variability.
Frequency-Based	Random Band-Stop Filtering	Removes a random frequency band to mimic selective frequency attenuation.
Frequency-Based	Tailored Mixup	Combines two signals in the frequency domain using weighted averages of their magnitude and phase spectra.
Masking-Cropping	Random Zero Masking	Masks a random segment of the signal with zeros to simulate missing data.
Masking-Cropping	Cutout and Resize	Removes a random segment and resizes the remaining parts to the original length.
Noise-and-Filtering	Random Additive Gaussian Noise	Adds Gaussian noise to simulate environmental and instrumentation noise.
Noise-and-Filtering	Average Filtering	Applies a moving average filter to smooth the signal.
Temporal	Random Time Shift	Shifts the signal temporally by a random offset.
Temporal	Time Warping	Stretches or compresses segments of the signal non-uniformly along the time axis.
Temporal	Time Reversal	Reverses the temporal order of the signal to ensure robustness to time directionality.
Temporal	Permutation	Divides the signal into segments and randomly reorders them to disrupt temporal consistency.

Table 1: Overview of EEG Signal Augmentations

3.1.1 Random Amplitude Scaling

Description: Randomly scales the amplitude of the EEG signal to simulate variations in signal strength.

Definition: For a signal $X(t)$, a scaling factor α is sampled uniformly from the interval $\mathcal{U}(r_1, r_2)$ and applied as follows:

$$\text{ScaledSignal}[X](t) = \alpha \cdot X(t)$$

where r_1 and r_2 define the range of scaling factors.

3.1.2 Random DC Shift

Description: Introduces variability in the baseline voltage by adding a constant offset to the EEG signal.

Definition: For a signal $X(t)$, a shift value β is sampled uniformly from the interval $\mathcal{U}(d_1, d_2)$ and applied as follows:

$$\text{ShiftedSignal}[X](t) = X(t) + \beta$$

where d_1 and d_2 define the range of shift values.

3.1.3 Sign Flip

Description: Flips the polarity of the EEG signal to introduce variability in signal orientation.

Definition: For a signal $X(t)$, the polarity is flipped as follows:

$$\text{FlippedSignal}[X](t) = -X(t)$$

3.2 Frequency-Based

These methods alter the frequency components of the signal. RandomBandStopFilter removes specific frequency bands to mimic selective attenuation, while TailoredMixup combines frequency-domain features of two signals to create augmented samples, enriching the signal diversity in the frequency space.[8][12]

3.2.1 Random Band-Stop Filtering

Description: Removes a random frequency band to mimic selective frequency attenuation in EEG signals.

Definition: A band-stop filter with center frequency f_c and bandwidth b is applied to the signal $X(t)$ as follows:

$$\text{FilteredSignal}[X](t) = \mathcal{F}^{-1}\left[\mathcal{F}[X](f) \cdot H(f)\right]$$

where $\mathcal{F}$ denotes the Fourier transform, and $H(f)$ is the filter response defined by:

$$H(f) = \begin{cases} 0 & |f - f_c| \leq \frac{b}{2} \\ \frac{1}{2}\left[1 - \cos\left(\pi \frac{|f-f_c|-\frac{b}{2}}{\Delta f}\right)\right] & \frac{b}{2} < |f - f_c| \leq \frac{b}{2} + \Delta f \\ 1 & |f - f_c| > \frac{b}{2} + \Delta f \end{cases}$$

to ensure smooth transitions and maintain Hermitian symmetry:

$$H(-f) = H(f)$$

where Δf defines the width of the transition band.

3.2.2 Tailored Mixup

Description: Combines two EEG signals in the frequency domain by applying weighted averages to their magnitude spectra and tailored adjustments to their phase spectra, creating augmented samples that preserve key signal characteristics while introducing variability.[13]

Definition: For two signals $X_1(t)$ and $X_2(t)$ with Fourier transforms $\mathcal{F}(X_1)$ and $\mathcal{F}(X_2)$, the mixed signal in the frequency domain is defined as:

$$\mathcal{F}(\text{MixedSignal}) = A_{\text{mix}} \cdot e^{iP_{\text{mix}}}$$

where:

- $A_{\text{mix}} = \lambda_A |\mathcal{F}(X_1)| + (1 - \lambda_A)|\mathcal{F}(X_2)|$ is the weighted magnitude spectrum.
- $P_{\text{mix}} = P_1 - \Delta\Theta \cdot (1 - \lambda_P)$ is the tailored phase spectrum, calculated as:

  $$\Delta\Theta = (P_1 - P_2 + \pi) \bmod (2\pi) - \pi$$

  where P_1 and P_2 are the phases of $\mathcal{F}(X_1)$ and $\mathcal{F}(X_2)$, and $\lambda_P \in [0, 1]$ is sampled from a uniform distribution $\mathcal{U}(\beta, 1.0)$.

- $|\mathcal{F}(X)|$ denotes the magnitude spectrum, and $P(X) = \arg(\mathcal{F}(X))$ denotes the phase spectrum.

The mixed signal is reconstructed by applying the inverse Fourier transform:

$$\text{MixedSignal}(t) = \mathcal{F}^{-1}\left[A_{\text{mix}} \cdot e^{iP_{\text{mix}}}\right]$$

Key Features: Magnitude and phase are mixed independently to preserve amplitude and phase relationships. Phase differences are wrapped into the range $[-\pi, \pi]$ to avoid artifacts from destructive interference. The mixup coefficients λ_A and λ_P are controlled to balance augmentation and prevent excessive distortion.

3.3 Masking and Cropping

Focused on modifying the signal's structure, these augmentations introduce gaps or remove portions of the signal. RandomZeroMasking simulates missing data by masking random segments, and CutoutResize removes sections of the signal and resizes the remaining parts to maintain its original length. [8][12]

3.3.1 Cutout and Resize

Description: Randomly removes a segment of the EEG signal and resizes the remaining portions to restore the original signal length.

Definition: Given a signal $X(t)$ segmented into n equal parts $\{X_1(t), X_2(t), \ldots, X_n(t)\}$, a randomly selected segment $X_r(t)$ is removed. The remaining segments are concatenated and resized to match the original signal length T:

$$\text{CutoutSignal}[X](t) = \text{Resize}\left(\text{Concat}\left(X_1(t), \ldots, X_{r-1}(t), X_{r+1}(t), \ldots, X_n(t)\right)\right)$$

where:

- $X_r(t)$ is the segment removed from the signal.
- Resize$(\cdot)$ Adjusts the concatenated signal back to the original length T.

3.3.2 Random Zero Masking

Description: Masks a random segment of the EEG signal by setting its values to zero, introducing variability in the signal's baseline.

Definition: For a signal $X(t)$, a mask of length M starting at time t_m is applied as follows:

$$\text{MaskedSignal}[X](t) = \begin{cases} 0 & \text{if } t \in [t_m, t_m + M] \\ X(t) & \text{otherwise} \end{cases}$$

where:

- M is the length of the mask.
- t_m is the starting time of the mask, randomly selected such that $t_m + M \leq T$, where T is the total duration of the signal.

3.4 Noise and Filtering

These techniques simulate real-world signal distortions. RandomAdditiveGaussianNoise introduces random noise to mimic environmental or instrumental artifacts, while AverageFilter smooths the signal using a moving average, reducing sharp variations and noise. [8][12]

3.4.1 Average Filtering

Description: Applies a moving average filter to smooth the EEG signal, reducing short-term fluctuations and highlighting longer-term trends.

Definition: For a kernel k of size N, the filtered signal FilteredSignal$[X](t)$ is computed as:

$$\text{FilteredSignal}[X](t) = \frac{1}{N} \sum_{i=0}^{N-1} X(t-i)$$

where:

- $X(t)$ is the original EEG signal.
- N is the size of the moving average kernel.

3.4.2 Random Additive Gaussian Noise

Description: Adds Gaussian noise to the EEG signal to simulate environmental and instrumentation noise, serving as a baseline augmentation in our experimental design.

Definition: Gaussian noise $N(t)$ with mean 0 and variance σ^2 is added to the signal $X(t)$ as follows:

$$\text{NoisySignal}[X](t) = X(t) + N(t)$$

where:

- $N(t) \sim \mathcal{N}(0, \sigma^2)$ represents the Gaussian noise.
- σ^2 controls the intensity of the noise added.

3.5 Temporal

These augmentations modify the time axis of the signal to disrupt or alter temporal consistency. RandomTimeShift shifts the signal temporally, TimeWarping stretches or compresses segments non-uniformly, TimeReverse reverses the signal's temporal order, and Permutation randomly reorders segments, encouraging the model to learn robust temporal representations. [8][12]

3.5.1 Time Reversal

Description: Reverses the temporal order of the EEG signal to ensure robustness to time directionality.

Definition: For a discrete signal $X[n]$ of length L, the reversed signal ReversedSignal$[X]$ is defined as:

$$\text{ReversedSignal}[X][n] = X[L-n-1]$$

where $n = 0, 1, 2, \ldots, L-1$.

3.5.2 Time Warping

Description: Non-uniformly stretches or compresses segments of the EEG signal along the time axis to simulate temporal variations.

Definition: For a segment $X_s[n]$ and a scale factor ω, the warped segment WarpedSegment$[X_s]$ is defined as:

$$\text{WarpedSegment}[X_s][n] = X_s[\omega n]$$

After warping, the segments are concatenated and resampled using an interpolation method (e.g., linear interpolation) to match the original signal length L:

$$\text{WarpedSignal}[X](n) = \text{Resample}\left(\text{Concat}\left(\text{WarpedSegment}_1, \ldots, \text{WarpedSegment}_k\right), L\right)$$

where:

- ω controls the degree of stretching ($\omega > 1$) or compression ($\omega < 1$).
- Resample$(\cdot)$ adjusts the concatenated signal to the original length L.

3.5.3 Permutation

Description: Divides the EEG signal into segments and randomly reorders them to disrupt temporal consistency.

Definition: For a signal $X[n]$ segmented into n equal-length segments $\{X_1[n], X_2[n], \ldots, X_n[n]\}$, a random permutation π is applied as follows:

$$\text{PermutedSignal}[X] = \text{Concat}\left(X_{\pi(1)}, X_{\pi(2)}, \ldots, X_{\pi(n)}\right)$$

where π is a bijective permutation function such that $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$.

3.5.4 Random Time Shift

Description: Temporally shifts the EEG signal by a random offset to simulate variations in signal timing.

Definition: For a discrete signal $X[n]$ and a shift value t_s, the shifted signal ShiftedSignal$[X]$ is defined as:

$$\text{ShiftedSignal}[X][n] = \begin{cases} X[n + t_s] & \text{if } 0 \leq n + t_s < L \\ 0 & \text{otherwise} \end{cases}$$

where:

- t_s is the shift amount, randomly selected within a predefined range.
- L is the length of the signal.

4 Dataset

4.1 Description

The Sleep-EDF Expanded dataset [14](2018 version) consists of 197 whole-night PSG recordings, including EEG, EOG, chin EMG, and event markers. The dataset includes two study types:

- *SC (Sleep Cassette):* 79 recordings from healthy individuals aged 25–101 years, free of sleep disorders.
- *ST (Study Temazepam):* 22 recordings from subjects in a study on the effects of Temazepam on sleep.

For this study, SC recordings were used, following prior work [15], classifying each 30-second epoch into one of eight categories: WAKE, REM, N1, N2, N3, N4, MOVEMENT, UNKNOWN.

To address class imbalance, only 30 minutes of WAKE epochs before and after the sleep period was retained.

4.2 Preparation

Step	Description
Epoch Duration	Standardized to 30 seconds (E = 30).
Downsampling	Signals downsampled to 100 Hz (F = 100).
Channel Selection	Retained only Fpz-Cz channel.
Epoch Segmentation	Segmented into consistent 30-second epochs; non-standard durations split accordingly.
Annotations	Parsed hypnogram annotations for onset, duration, and stage labels; aligned with EEG epochs and corrected mismatches.
Exclusion of Non-Sleep Periods	Removed WAKE epochs at sleep boundaries, retaining 30 minutes before and after core sleep.
Removal of Irrelevant Stages	Excluded epochs labeled as "MOVEMENT" or "UNKNOWN".
Class Consolidation	Merged sleep stages N3 and N4 into a single class (N3).

Table 2: Overview of Sleep-EDF-2018 Pre-processing Steps

We standardized EEG epochs to 30 seconds and downsampled the signals to 100 Hz. Only the Fpz-Cz channel was retained for analysis. The signals were segmented into consistent 30-second epochs, with non-standard durations appropriately split. Hypnogram annotations were parsed to extract onset, duration, and stage labels, ensuring alignment with EEG epochs and resolving any annotation mismatches. Non-sleep periods and epochs labeled as "MOVEMENT" or "UNKNOWN" were excluded to focus on relevant sleep data. Additionally, sleep stages N3 and N4 were consolidated into a single class to streamline the classification process.

4.3 Analysis

The dataset consists of *~200,000 (0.2M) sleep epochs* derived from *152 PSG recordings* (~10 hours each). Each epoch corresponds to 30 seconds of sleep data, with sleep stages labeled as WAKE, N1, N2, N3, and REM. The class distribution highlights a significant imbalance across sleep stages.

Label	Stage	# Epochs	Percentage
0	WAKE	69,532	35%
1	N1	21,391	11%
2	N2	68,651	35%
3	N3	13,000	7%
4	REM	25,715	13%
Total	**-**	**198,289**	**100%**

Table 3: Distribution of Sleep Stages by Epochs

The dataset comprises 152 NPZ files, each containing an average of 1,305 epochs and approximately 10.87 hours of EEG data. Analysis reveals that the WAKE and N2 stages are predominant, each accounting for 35% of the epochs and together constituting 70% of the dataset. In contrast, REM

and N1 stages are underrepresented, comprising 13% and 11% of the epochs respectively, while the N3 stage contributes only 7%. This class imbalance necessitates the use of imbalance-informed evaluation metrics such as macro-F1.

5 Evaluation Metrics and Strategy

5.1 Downstream Task

The downstream task involves *single-epoch EEG classification*, where each 30-second EEG epoch is independently classified into one of five sleep stages: Wake (W), N1, N2, N3, or REM. This approach is chosen to *isolate the quality of feature representations* learned by the encoder by eliminating temporal dependencies between consecutive epochs. By focusing on individual epochs, we can accurately assess the encoder's ability to extract meaningful and discriminative features from EEG signals without the influence of temporal context.

Formally, let $\mathbf{x} \in \mathbb{R}^d$ denote an EEG epoch, where d is the dimensionality of the input signal. The encoder $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ maps the input to a latent representation $\mathbf{z} = f_\theta(\mathbf{x}) \in \mathbb{R}^k$. The classifier $\mathbb{R}^k \to \mathbb{R}^C$ then predicts the sleep stage label $y \in \{0, 1, 2, 3, 4\}$, where $C = 5$ represents the number of classes.

5.2 Performance Metrics

We assessed the classification performance using a comprehensive set of metrics, including overall measures such as accuracy and macro-F1 score, as well as class-specific metrics like precision, recall, and F1 score. Additionally, confusion matrix visualizations were computed to evaluate the model's performance across different sleep stages. These metrics provide accurate proxies to quantify the effectiveness of pre-training. For detailed mathematical definitions and formulations, refer to Appendix 12.2.

6 Training Framework

The training framework is structured around a *two-stage linear evaluation protocol* comprising self-supervised contrastive pretraining followed by supervised linear evaluation.

6.1 Linear Evaluation Protocol

The training framework adopts a *Two-Stage Training Strategy*, integrating Self-Supervised Learning (SSL) with Contrastive Representation Learning (CRL) and a subsequent supervised linear evaluation.

6.1.1 Stage 1 : Self-Supervised Contrastive Pretraining

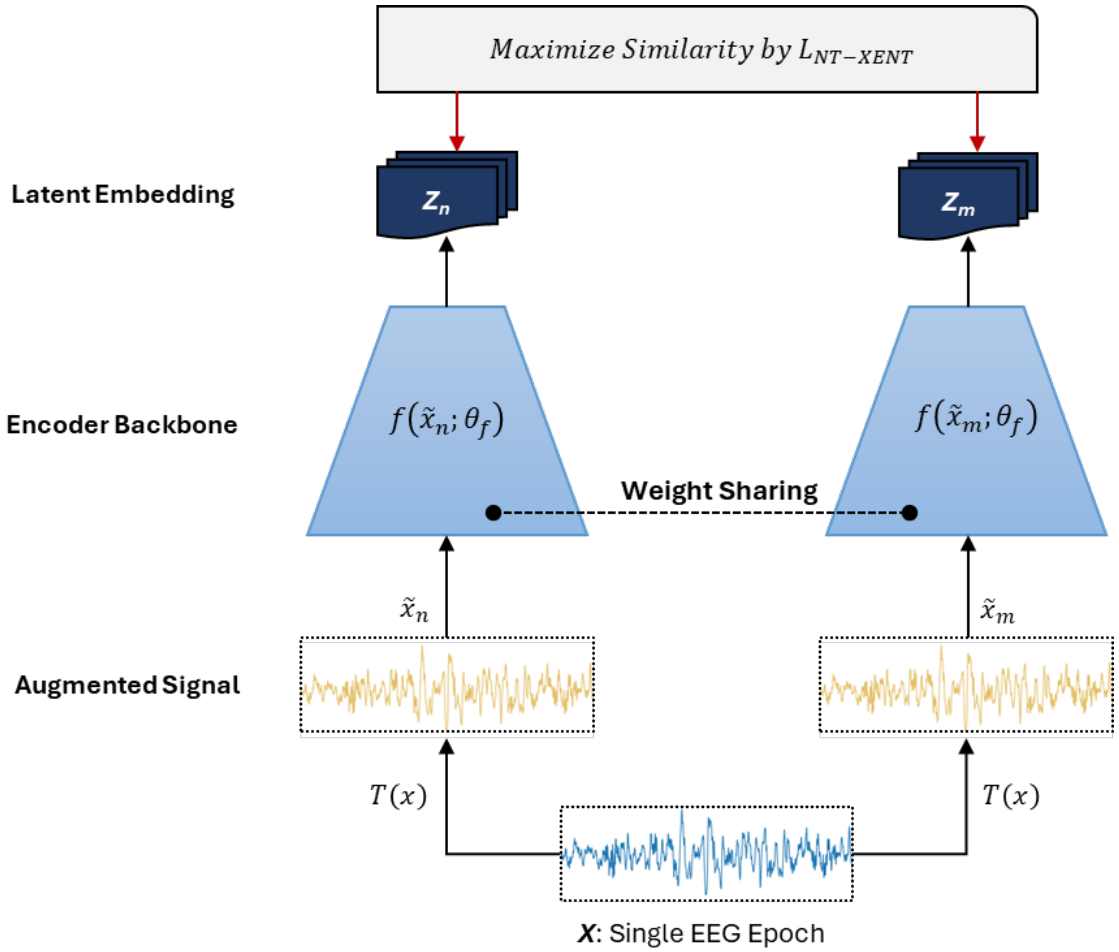
In the first stage, the encoder model undergoes self-supervised pretraining using CRL. The objective is to learn rich and invariant feature representations from unlabeled EEG data through contrastive learning mechanisms. Specifically, the encoder f_θ maps each EEG epoch $\mathbf{x} \in \mathbb{R}^d$ to a latent representation $\mathbf{z} = f_\theta(\mathbf{x}) \in \mathbb{R}^k$. The formulation requires the signal transformations $T(\dot{})$ to generate transformed signal pairs that enable the encoder to learn disentangled semantic representations.

The contrastive loss function used here is the *Normalized Temperature-scaled Cross Entropy Loss (NT-Xent)*, defined as:

$$\mathcal{L}_{\text{NT-Xent}}(\mathbf{z}_i, \mathbf{z}_j) = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

where:

- $\text{sim}(\cdot, \cdot)$ denotes the cosine similarity between two vectors.
- τ is the temperature parameter, controlling the concentration level of the distribution.
- N is the number of positive pairs in a mini-batch.



The encoder is optimized using the Adam optimizer with a default learning rate of 1×10^{-3}. Training proceeds for a predefined number of maximum epochs, with early stopping criteria based on validation loss improvements to prevent overfitting. Upon completion, the best-performing encoder model, determined by the lowest validation loss, is saved for subsequent evaluation.

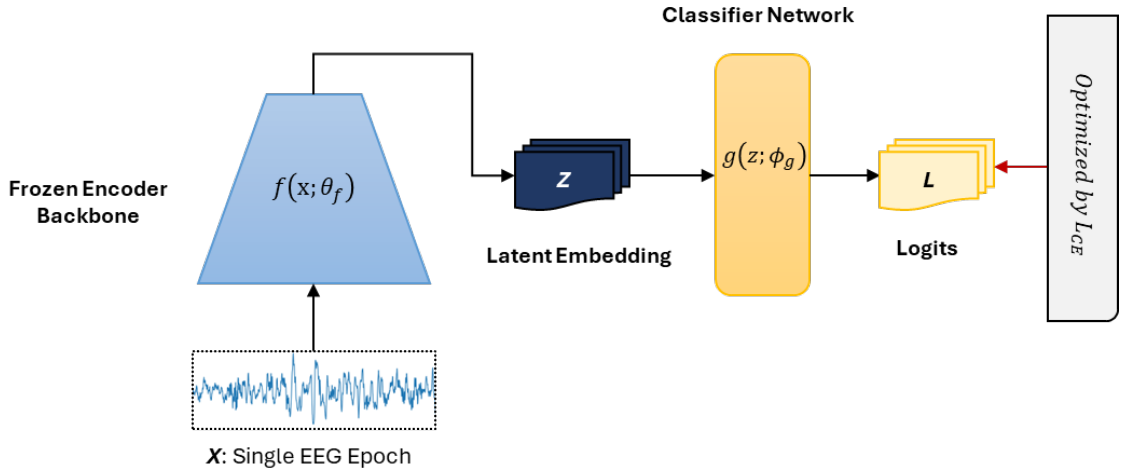
6.1.2 Stage 2 : Supervised Linear Evaluation

In the second stage, the pre-trained encoder f_θ is *frozen*, meaning its weights remain unaltered during this phase. It serves solely as a feature extractor. A simple *Multi-Layer Perceptron (MLP) classifier* g_ϕ is appended to the encoder to facilitate supervised classification of sleep stages. The classifier maps the latent representations $\mathbf{z}$ to class probabilities $\hat{\mathbf{y}} = g_\phi(\mathbf{z}) \in \mathbb{R}^C$, where $C = 5$ corresponds to the five sleep stages.

The classifier is trained using the *Cross-Entropy Loss*:

$$\mathcal{L}_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{C} y_c \log p_\phi(y = c \mid \mathbf{z})$$

where $\mathbf{y}$ is the one-hot encoded true label vector.



The Adam optimizer, with a default learning rate of 1×10^{-3}, updates only the classifier's parameters ϕ. Training incorporates early stopping based on validation loss to improve generalization. The best classifier model, exhibiting the lowest validation loss, is preserved for final evaluation.

This linear evaluation protocol facilitates a clear separation between representation learning and classification, enabling an unbiased assessment of the encoder's ability to extract meaningful features from EEG data.

6.2 Train-Test Split Strategy

We used an 85%-15% train-test split by partitioning the data at the patient level so that data from the same individual do not appear in both sets. This approach prevents data leakage and evaluates the model's ability to generalize across distinct individuals, accounting for inter-subject variability and reflecting real-world scenarios where models encounter new patients. Specifically, the training set comprises 66 subjects used for pretraining the encoder through Contrastive Representation Learning (CRL) and training the classifier during the linear evaluation stage, while the testing set includes 12 subjects reserved exclusively for validating model performance and generating evaluation metrics such as confusion matrices.

7 Model Architecture

The proposed framework comprises two main components: the encoder, *SimpleSleepNet*, and the classifier, *SleepStageClassifier*. Detailed architectural implementations and parameter configurations are provided in Appendix 12.3.

7.1 Encoder: SimpleSleepNet

SimpleSleepNet is a lightweight 1D convolutional neural network tailored for single-channel EEG-based sleep stage classification. It consists of three convolutional layers with decreasing kernel sizes (64, 32, 16), decreasing strides (8, 4, 2), and increasing dilations (1, 2, 4), enabling the extraction of multi-scale temporal features by capturing long-, medium-, and short-range dependencies. Each convolutional layer is followed by batch normalization and a dropout layer with a rate of 0.2 to

stabilize training and prevent overfitting. The network utilizes the Mish activation function for its smooth gradients. After the convolutional layers, a global average pooling layer condenses the feature maps into a fixed-size representation, which is then projected into a 128-dimensional latent space via a fully connected layer, producing normalized and discriminative embeddings for downstream classification tasks.

Design Highlights The architecture emphasizes both computational efficiency and generalizable feature extraction. *SimpleSleepNet* is engineered to capture temporal dependencies across multiple scales with approximately 200,000 trainable parameters, making it $\approx 100\times$ more lightweight than comparable models such as DeepSleepNet [16] (21 million parameters) and $\approx 10\times$ SleepEEGNet [17] (2.6 million parameters) [18]. This minimalistic design not only accelerates training and inference processes but also enables extensive experimentation.

7.2 Classifier: SleepStageClassifier

The *SleepStageClassifier* is a fully connected neural network designed to classify sleep stages based on the latent embeddings generated by the encoder. It comprises three linear layers with hidden dimensions of 256 and 128, culminating in an output layer corresponding to the number of sleep stages (default: 5). Similar to the encoder, each linear layer is succeeded by batch normalization and a dropout layer with a rate of 0.5 to mitigate overfitting. The classifier also uses the Mish activation function.

8 Experiment Design and Results

8.1 Single Augmentation

Category	Augmentation	Accuracy (%)	Macro F1 (%)
Temporal	Permutation	68.84	61.15
Temporal	TimeReverse	64.68	54.35
Temporal	TimeWarping	71.58	63.25
Temporal	RandomTimeShift	65.52	57.42
Noise-and-Filtering	AverageFilter	60.44	47.98
Noise-and-Filtering	RandomAdditiveGaussianNoise	65.33	55.53
Masking-Cropping	CutoutResize	71.13	63.37
Masking-Cropping	RandomZeroMasking	72.19	65.00
Frequency-Based	TailoredMixup	72.93	65.16
Frequency-Based	RandomBandStopFilter	66.31	55.69
Amplitude-Based	SignFlip	60.91	49.61
Amplitude-Based	RandomDCShift	66.36	56.49
Amplitude-Based	RandomAmplitudeScale	62.72	49.43

Table 4: Performance of Single Augmentations on Downstream Classification

RandomAdditiveGaussianNoise was selected as the baseline augmentation due to its widespread use in EEG and time-series analysis, achieving an Accuracy of 65.33% and a Macro-F1 score of

55.53%. This serves as a reference point for evaluating other augmentation strategies. Among the top-performing augmentations, RandomZeroMasking (Masking-Cropping) and TailoredMixup (Frequency-Based) stood out, achieving Accuracy scores exceeding 72% and Macro-F1 scores above 65%. CutoutResize, another Masking-Cropping strategy, also performed strongly.

Temporal augmentations exhibited mixed results. TimeWarping emerged as a significant contributor with an Accuracy of 71.58% and a Macro-F1 of 63.25%, showcasing the benefits of non-uniform temporal distortions. However, augmentations like TimeReverse and RandomTimeShift underperformed, indicating that certain temporal manipulations may disrupt critical signal dependencies. Similarly, moderate improvements were observed for RandomBandStopFilter (Frequency-Based) and RandomDCShift (Amplitude-Based), suggesting that targeted frequency and amplitude adjustments provide some benefit but lack consistency across all metrics.

On the other hand, AverageFilter (Noise-and-Filtering) recorded the lowest Macro-F1 score (47.98%), highlighting that excessive smoothing may obscure essential discriminative features in the signal. Amplitude-based augmentations, such as SignFlip and RandomAmplitudeScale, also underperformed, possibly due to distortions introduced to the signal's key characteristics.

At a category level, Masking-Cropping and Frequency-Based augmentations emerged as the most effective strategies, underscoring the advantages of selectively masking signal segments or manipulating spectral components. Temporal augmentations delivered mixed outcomes, with TimeWarping demonstrating significant promise while others, like TimeReverse, fell short. Overall, the results indicate that strategies involving masking and frequency manipulations promote better feature representations, whereas amplitude transformations and excessive smoothing can degrade the model's performance. TailoredMixup, RandomZeroMasking, and CutoutResize emerged as the most effective augmentations, significantly outperforming the baseline.

8.2 Intra-Category Combinational Augmentations

Category	Augmentation	Accuracy (%)	Macro F1 (%)
Temporal	RandomTimeShift + TimeReverse + Permutation	72.37	64.84
Temporal	RandomTimeShift + TimeWarping + Permutation	72.23	63.42
Temporal	RandomTimeShift + TimeWarping + TimeReverse	70.78	61.54
Temporal	TimeReverse + Permutation	67.83	60.30
Temporal	TimeWarping + TimeReverse	69.16	60.25
Temporal	RandomTimeShift + Permutation	72.98	65.82
Temporal	RandomTimeShift + TimeReverse	65.39	57.28
Temporal	RandomTimeShift + TimeWarping	71.45	62.52
Noise-and-Filtering	RandomAdditiveGaussianNoise + AverageFilter	59.71	47.79
Masking-Cropping	RandomZeroMasking + CutoutResize	72.94	64.83
Frequency-Based	RandomBandStopFilter + TailoredMixup	70.09	59.72
Amplitude-Based	RandomAmplitudeScale + RandomDCShift + SignFlip	60.16	46.36
Amplitude-Based	RandomDCShift + SignFlip	64.97	53.49
Amplitude-Based	RandomAmplitudeScale + SignFlip	57.81	45.09
Amplitude-Based	RandomAmplitudeScale + RandomDCShift	65.19	52.90

Table 5: Performance of Intra-Category Augmentation Combinations on Downstream Classification

Temporal Category: Several top-performing augmentation combinations, such as RandomTimeShift paired with Permutation or TimeWarping, achieved Accuracy scores around 72% and Macro-F1 scores between 64% and 66%, indicating complementary benefits. However, combinations like TimeReverse with Permutation or TimeWarping showed lower improvements, suggesting potential redundancy.

Masking-Cropping: The combination of RandomZeroMasking and CutoutResize achieved an Accuracy of 72.94% and a Macro-F1 of 64.83%, reinforcing the effectiveness of targeted masking and cropping. This performance remains competitive with the top temporal augmentation strategies.

Frequency Category: The pairing of RandomBandStopFilter with TailoredMixup yielded modest gains, with an Accuracy of 70.09% and a Macro-F1 of 59.72%. While TailoredMixup was effective individually, adding another frequency-based method resulted in partial synergy without reaching top-tier performance levels.

Noise-and-Filtering: The combination of RandomAdditiveGaussianNoise and AverageFilter underperformed, recording an Accuracy of 59.71% and a Macro-F1 of 47.79%. The smoothing effect of AverageFilter appears to degrade the signal further when combined with noise.

Amplitude-Based: All amplitude-based augmentation combinations underperformed, with Accuracy scores of 65% or below and Macro-F1 scores of 53% or lower. This consistent underperformance aligns with single-augmentation findings, suggesting that combining amplitude transformations often leads to excessive signal distortion, thereby degrading the discriminative information necessary for accurate classification.

In summary, selective temporal transformations and masking-cropping strategies yield complementary effects, highlighting the importance of spatiotemporal manipulations and targeted signal masking. These augmentations significantly outperform amplitude and noise-based augmentations.

8.3 Inter-Category Combinational Augmentations

Overall Consistency: Performance variations generally lie within a 1–3% range for both Accuracy and Macro-F1 across different random seeds. Despite these variations, the ranking of augmentation sets remains stable, confirming consistent relative performance.

Synergistic Combinations reveal that Frequency + Masking-Cropping augmentations, such as TailoredMixup combined with RandomZeroMasking and CutoutResize, achieve high Accuracy (approximately 76%) and competitive Macro-F1 scores (around 67%), highlighting the synergy between spectral manipulation and strategic signal masking/cropping. Similarly, Temporal + Masking-Cropping combinations like TimeWarping paired with Permutation and RandomZeroMasking yield Accuracy scores around 75–76%, indicating complementary benefits of temporal distortions with spatial manipulations.

Frequency + Temporal augmentations, exemplified by TailoredMixup with TimeWarping and Permutation, rank among the better performers with Accuracy between 76.5% and 77.5%, amalgamating the strengths of frequency diversity and temporal structure manipulation to for better feature representations.

Conversely, the **Absence of Amplitude-Based Augmentations** is evident as amplitude-based methods such as SignFlip and RandomAmplitudeScale were excluded due to their consistently subpar individual and combined results. Their limited utility remains apparent regardless of inter-category mixing, as these combinations often lead to overly distorted signals that degrade discriminative information.

Categories	Augmentation	Avg. MF1 (%)	Avg. Accuracy (%)
Frequency + Masking-Cropping	TailoredMixup + RandomZeroMasking + CutoutResize	67.39	76.39
	TailoredMixup + CutoutResize	68.02	76.80
Frequency + Masking-Cropping	TailoredMixup + RandomZeroMasking	65.28	74.92
Temporal + Masking-Cropping	TimeWarping + Permutation + RandomZeroMasking + CutoutResize	66.61	75.85
	TimeWarping + Permutation + CutoutResize	66.34	75.62
	TimeWarping + Permutation + RandomZeroMasking	65.76	75.42
Temporal + Masking-Cropping	Permutation + RandomZeroMasking + CutoutResize	66.20	75.23
	TimeWarping + RandomZeroMasking + CutoutResize	66.63	75.82
Temporal + Masking-Cropping	Permutation + CutoutResize	65.81	75.19
	Permutation + RandomZeroMasking	66.69	75.81
Temporal + Masking-Cropping	TimeWarping + CutoutResize	66.87	75.85
	TimeWarping + RandomZeroMasking	63.06	73.64
Frequency + Temporal	TailoredMixup + TimeWarping + Permutation	67.63	76.87
	TailoredMixup + Permutation	66.36	75.67
Frequency + Temporal	TailoredMixup + TimeWarping	66.09	76.15

Table 6: Performance of Inter-Category Augmentation Combinations on Downstream Classification

In summary, selective temporal transformations and masking-cropping strategies yield complementary effects, underscoring the importance of spatiotemporal manipulations and targeted signal masking. These methods surpass the utility of amplitude and noise-based augmentations in increasing feature representation quality, significantly improving classification performance.

8.4 Best of All Categories - Augmentation Severity

We now understand individual performances from section 8.1, intra-category reinforcing vs. redundancy effects from 8.2 and, inter-category synergy effects from 8.3.

Clearly, the winning compound augmentation can be engineered by blending the reinforcing top performers from categories that most synergize with each other.

Winning Combination: *TailoredMixup + TimeWarping + Permutation + RandomZeroMasking + CutoutResize*

However, excessive use of augmentations can degrade signal quality and diminish benefits, pointing to the need for a balanced or "Goldilocks" level of severity. To explore this, we vary the **application**

Experiment Design (Severity)	Avg. MF1 (%)	Avg. Accuracy (%)
Weighted p sum to 3	68.62	77.55
Weighted p sum to 4	68.62	77.54
Equal p sum to 3	68.84	77.70
Equal p sum to 4	68.49	77.52
Equal p sum to 5	68.25	77.25

Table 7: Performance of Severity Variation on Downstream Classification

probability of each transform, interpreting the sum of these probabilities as the effective number of active augmentations per signal.

Parameterizing Severity. In the *Equal* setting, each augmentation has the same probability (e.g., 0.6 for a sum of 3, 0.8 for a sum of 4, 1.0 for a sum of 5), while the *Weighted* setting assigns probability 1 to TailoredMixup and CutoutResize—both proven highly beneficial—and divides the remainder equally among TimeWarping, Permutation, and RandomZeroMasking. Table 8.4 shows that sums of 3 or 4 yield the best trade-off, with average Macro-F1 scores around 68–69% and Accuracies of 77–78%. Pushing severity to a sum of 5 slightly reduces performance, emphasizing that **over-saturation** with all augmentations at probability 1 can obscure essential signal features.

Goldilocks Zone. This moderate severity consistently outperforms single-augmentation or simpler multi-augmentation strategies, underlining the advantage of carefully orchestrated distortion. Assigning probability 1 to TailoredMixup and TimeWarping while moderately applying the others preserves salient spectral and temporal distortions while preventing excessive masking or cropping. This approach is robust across random seeds, with performance fluctuations confined to a 1–3% range, aligning with prior findings. The confusion matrix (Fig. 8.4) for the "Equal p sum to 3" setting further illustrates strong class-wise recognition.

In conclusion, these results solidify the concept of a **"Goldilocks zone"** for augmentation severity, where **3–4 active augmentations** per signal achieve an ideal balance between under- and over-perturbation. By distributing probabilities strategically, we capitalize on the synergy among diverse augmentations—especially frequency, temporal, and masking transformations—while avoiding the destructive effects of excessive distortions. This balanced methodology sets a new benchmark for enriching EEG feature representations in contrastive learning frameworks.

8.5 Full Fine-Tuning and Final Performance

We moved beyond linear evaluation to *fully fine-tune* our lightweight CNN-based encoder, allowing it to adapt more closely to the labeled data. Despite having only $\sim$ 200,000 parameters, this tiny architecture achieved remarkable performance. As a benchmark, we also compared the final performance of augmentations based on RandomAdditiveGaussianNoise under full fine-tuning conditions, providing a reference for how baseline noise augmentation fares against our more sophisticated augmentation strategy.

Key Results and Significance: The fine-tuned model achieves *80.55% accuracy* and *71.68% Macro-F1*, as compared to *79.9% accuracy* and *70.63% Macro-F1* beating the RandomGaussian-Noise baseline. These gains arise without using class imbalance mitigation (no weighted losses or oversampling), temporal modeling (no LSTMs or Transformers), or advanced deep learning

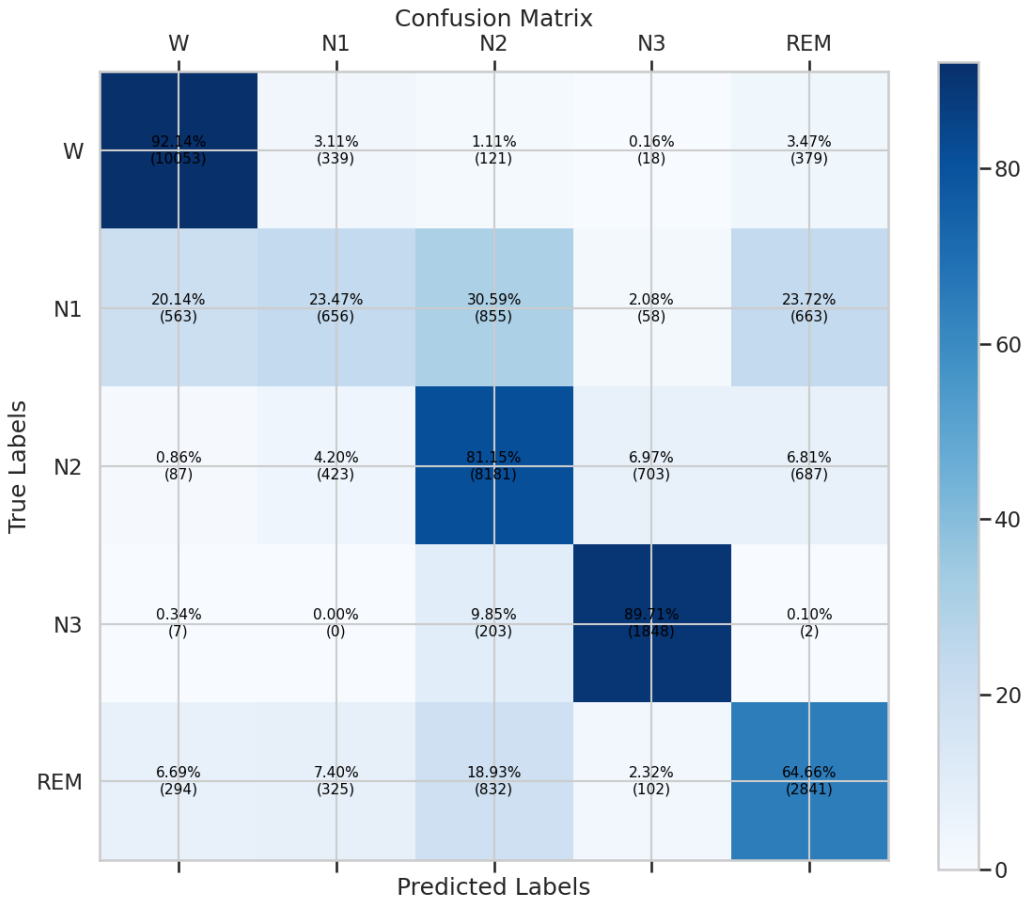


Figure 2: Confusion Matrix for **Equal p sum to 3**

components such as multi-head attention. The results validate our SSL pretraining pipeline and confirm that carefully tuned data augmentations can yield impressive outcomes even within a tiny model footprint. Surpassing 80% accuracy with a model of this scale validates our approach.

Future Enhancements: Although our model demonstrates strong performance, several avenues exist for further refinement. Addressing *class imbalance* through weighted losses or oversampling could improve minority class recognition. Incorporating *temporal modules* such as LSTMs or Transformers might capture inter-epoch dependencies more effectively while exploring sophisticated architectures could push performance toward even higher accuracy and macro-F1 scores.

Conclusion: By fully fine-tuning a compact CNN, we achieved *80.55% accuracy* and *71.68% Macro-F1* on single-epoch EEG classification. This outcome highlights the effectiveness of our SSL-CRL pipeline, demonstrating that a modest but carefully structured model can attain decent results without architectural complexity.

9 Discussion

This chapter integrates the findings from our augmentation experiments (Sections 8.1–8.5) with broader considerations related to model design, real-world deployment, and future research. We aim to contextualize the results of our self-supervised learning (SSL) framework for single-epoch EEG classification, highlighting both its effectiveness and inherent constraints.

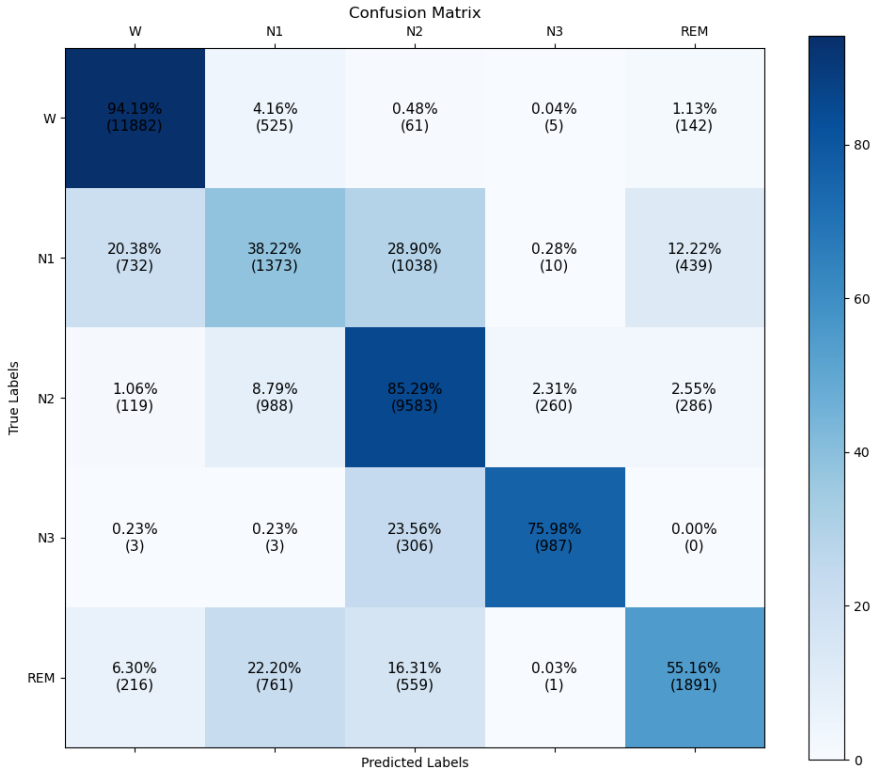


Figure 3: Confusion Matrix for RandomAdditiveGaussianNoise

9.1 Interpretation of Key Findings

Augmentation Effectiveness: Our experiments show that masking/cropping methods (RandomZeroMasking, CutoutResize) and frequency-based transformations (TailoredMixup) consistently outperform amplitude-based and noise/filtering augmentations, suggesting that targeted spatial and spectral distortions provide stronger contrastive signals. Among temporal augmentations, TimeWarping excels at promoting invariance without excessively distorting crucial temporal relationships.

Intra-Category and Inter-Category Synergy: Combining top augmentations within the same domain (e.g., RandomTimeShift + Permutation) often leads to incremental improvements, but inter-category synergies (temporal, frequency, and masking/cropping) produce the most pronounced gains by enriching both spectro-temporal and spatial features. Our final experiments confirm an optimal "Goldilocks" severity of *3–4 active augmentations*, as applying all five at probability 1 slightly diminishes accuracy.

Significance of a Small Model Achieving $\sim$ 80% Accuracy: Surpassing 80% accuracy and 70% Macro-F1 with a $\sim$ 200,000-parameter CNN—without temporal modeling, imbalance handling, complex architectures, or large parameter budgets—underscores the efficacy of carefully orchestrated augmentations and contrastive pretraining. These results demonstrate that targeted transformations can substantially close the performance gap often addressed by deeper or more

sophisticated models.

9.2 Implications for Real-World Applications

These findings suggest that compact CNN-based models might feasibly perform near-real-time EEG classification in resource-limited settings—such as wearable electronics or embedded clinical monitors—by reducing the need for extensive annotations and computational infrastructure. Leveraging unlabeled data through SSL could further accelerate deployment in clinical contexts, where manual labeling proves costly and time-intensive. While surpassing 80% accuracy in single-epoch classification is encouraging, it may speak more to the potential for on-site EEG processing and automated sleep monitoring than guaranteeing broad clinical adoption or replacing more comprehensive diagnostic frameworks.

9.3 Limitations

Absence of Temporal Modeling: While focusing on single-epoch classification isolates encoder effectiveness, it disregards inter-epoch correlations. Future work should incorporate recurrent or attention-based modules to exploit these temporal dependencies.

Restricted Architectures: Our use of a single CNN backbone with $\sim 200k$ parameters limits conclusions about scalability. Evaluating larger or more sophisticated models (e.g., transformer-based EEG pipelines) may reveal whether current augmentation strategies generalize.

Limited Dataset: All experiments rely on the Sleep-EDF database. Testing additional EEG datasets, such as SHHS or MASS, would confirm the model's generalization under varied recording conditions.

No Class Imbalance Handling: Our approach omits weighted losses or oversampling for underrepresented sleep stages, potentially leaving classification gains on the table. Investigating these methods could further improve performance easily.

9.4 Future Work

Exploring Augmentation Impacts: Detailed spectro-temporal analyses of how each augmentation reshapes EEG signals could illuminate which distortions most effectively enrich feature extraction and downstream classification accuracy.

Temporal Modeling: Incorporating bi-LSTMs, transformers, or other attention-based architectures may capture inter-epoch dynamics, particularly relevant for long-term EEG monitoring.

Advanced SSL Paradigms: Exploring supervised contrastive losses or masked auto-encoder-like frameworks could enrich learned features, extending beyond our current contrastive design.

Imbalance Handling: Weighted cross-entropy, focal loss, or SMOTE-like oversampling merit exploration to address underrepresented stages (e.g., N1), potentially improving overall performance.

Generalization and Benchmarking: Scaling to larger backbones ($\geq$ 1M parameters) and evaluating on multiple EEG databases would establish stronger baselines and validate generalization across diverse sleep populations.

Real-Time Deployments: On-device optimizations—quantization, pruning, or hardware-specific accelerations—can facilitate ultra-low-latency EEG classification in embedded or wearable devices.

10 Conclusion

This thesis has demonstrated that carefully curated data augmentations, integrated within a self-supervised contrastive learning (SSL-CRL) framework, can substantially improve sleep stage classification from single-epoch EEG signals. Through a methodical exploration—spanning single augmentations, intra-category synergies, inter-category combinations, and calibrated augmentation severity—we established a "Goldilocks" level of data transformation that maximizes downstream performance without excessively distorting the signal. Central to this success are masking/cropping (RandomZeroMasking, CutoutResize), frequency-based (TailoredMixup), and specific temporal (TimeWarping, Permutation) strategies, which consistently generated stable latent representations conducive to high-accuracy classification.

By engineering an extremely lightweight CNN backbone ($\approx$ 200k parameters), we surpassed *80% accuracy* and *70% Macro-F1*, underlining the feasibility of real-time inference in resource-constrained environments. This achievement did not require advanced architectures or specialized imbalance remedies, underscoring the power of augmentations and contrastive objectives to close performance gaps typically filled by more complex models. Furthermore, our findings contribute to the broader landscape of SSL research in EEG-based sleep staging by illustrating how targeted transformations, in conjunction with contrastive pretraining, unlock significant performance gains. Extending these insights to deeper models, additional EEG datasets, and refined augmentation policies would further validate and expand the applicability of this pipeline, ultimately accelerating the development of accurate, efficient, and deployable EEG-based solutions for sleep staging and beyond.

11 References

[1] K Šušmáková. Human sleep and sleep eeg. *MEASUREMENT SCIENCE REVIEW*, 4, 01 2004.

[2] Yun Ji Lee, Jae Yong Lee, Jae Hoon Cho, and Ji Ho Choi. Interrater reliability of sleep stage scoring: a meta-analysis. *Journal of Clinical Sleep Medicine*, 18(1):193–202, January 2022.

[3] Maksym Gaiduk, Ángel Serrano Alarcón, Ralf Seepold, and Natividad Martínez Madrid. Current status and prospects of automatic sleep stages scoring: Review. *Biomedical Engineering Letters*, 13(3):247–272, August 2023.

[4] Xue Jiang, Jianhui Zhao, Bo Du, and Zhiyong Yuan. Self-supervised contrastive learning for eeg-based sleep staging. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[5] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *CoRR*, abs/1902.06162, 2019.

[6] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876, 2023.

[7] Cheol-Hui Lee, Hakseung Kim, Hyun jee Han, Min-Kyung Jung, Byung C. Yoon, and Dong-Joo Kim. Neuronet: A novel hybrid self-supervised learning framework for sleep stage classification using single-channel eeg, 2024.

[8] Seongju Lee, Yeonguk Yu, Seunghyeok Back, Hogeon Seo, and Kyoobin Lee. Sleepyco: Automatic sleep scoring with feature pyramid and contrastive learning. *Expert Systems with Applications*, 240:122551, 2024.

[9] Navid Mohammadi Foumani, Geoffrey Mackellar, Soheila Ghane, Saad Irtza, Nam Nguyen, and Mahsa Salehi. Eeg2rep: Enhancing self-supervised eeg representation through informative masked inputs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 5544–5555, New York, NY, USA, 2024. Association for Computing Machinery.

[10] Hsiang-Yun Sherry Chien, Hanlin Goh, Christopher M. Sandino, and Joseph Y. Cheng. Maeeg: Masked auto-encoder for eeg representation learning, 2022.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

[12] Cédric Rommel, Joseph Paillard, Thomas Moreau, and Alexandre Gramfort. Data augmentation for learning predictive models on eeg: a systematic comparison. *Journal of Neural Engineering*, 19(6):066020, November 2022.

[13] Berken Utku Demirel and Christian Holz. Finding order in chaos: A novel data augmentation method for time series in contrastive learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 30750–30783. Curran Associates, Inc., 2023.

[14] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. Physiobank, physiotoolkit, and physionet. *Circulation*, 101(23):e215–e220, 2000.

[15] Huy Phan, Kaare Mikkelsen, Oliver Y. Chén, Philipp Koch, Alfred Mertins, and Maarten De Vos. Sleeptransformer: Automatic sleep staging with interpretability and uncertainty quantification. *IEEE Transactions on Biomedical Engineering*, 69(8):2456–2467, 2022.

[16] Akara Supratak, Hao Dong, Chao Wu, and Yike Guo. Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(11):1998–2008, November 2017.

[17] Sajad Mousavi, Fatemeh Afghah, and U. Rajendra Acharya. Sleepeegnet: Automated sleep stage scoring with sequence to sequence deep learning approach. *PLOS ONE*, 14(5):e0216456, May 2019.

[18] Akara Supratak and Yike Guo. Tinysleepnet: An efficient deep learning model for sleep stage scoring based on raw single-channel eeg. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 641–644, 2020.

12 Appendices

12.1 Implementation Details and Hyperparameters

12.1.1 SSL-CRL Pretraining Parameters

Early stopping occurs if the validation loss does not improve by at least `0.005` after a validation check, preventing overfitting.

Parameter	Default Value	Description
Batch Size	1024	Number of contrastive pairs processed per batch.
Temperature	0.1	Controls the NT-Xent loss concentration.
Dropout Rate	0.2	Probability of dropout in the encoder network.
Latent Dim	128	Dimensionality of the latent space for the encoder output.
Learning Rate	0.001	Initial learning rate for the Adam optimizer.
Max Epochs	1500	Maximum number of epochs for the contrastive pretraining stage.
Check Interval	50	Epoch frequency for performing validation checks and possible early stopping.
Min Improvement	0.005	Minimum drop in validation loss required to reset the early-stopping counter.

Table 8: Default Parameters for Contrastive Pretraining

12.1.2 Supervised Training Parameters

Parameter	Default Value	Description
Learning Rate	0.001	Initial learning rate for the classifier's Adam optimizer.
Max Epochs	500	Maximum number of supervised training epochs.
Dropout Rate	0.5	Probability of Dropout in the classifier's fully connected layers.
Check Interval	25	Epoch frequency for validation checks during classification training.
Min Improvement	0.005	Minimum drop in validation loss required to avert early stopping.

Table 9: Default Parameters for Classifier Training

Similar to pretraining, **early stopping** and **periodic validation checks** for generalization.

12.1.3 Random Seeds

All single augmentation experiments (section 8.1), intra-category augmentation experiments (section 8.2) were performed with random seed = 42. For inter-category augmentations (section 8.3) and best of all categories-severity (section 8.4) were performed with 3 random seeds = {0,1,42}.

12.2 Performance Metrics

Evaluating the classification performance requires a comprehensive set of metrics that capture both overall accuracy and class-specific performance. These metrics provide insights into the effectiveness of Contrastive Representation Learning (CRL) in extracting generalizable features from EEG data. Along with confusion matrix visualizations, this evaluation framework provides a holistic evaluation of the single-epoch EEG classification task.

12.2.1 Overall Metrics

1. **Accuracy:** Measures the proportion of correctly predicted instances out of the total number of instances. It provides a general measure of the model's correctness and has to be analyzed within the context of other metrics in cases of class imbalance.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

TP : True Positives
TN : True Negatives
FP : False Positives
FN : False Negatives

2. **Macro-F1 Score:** The unweighted mean of F1 scores calculated independently for each class. It accounts for both precision and recall, providing a balanced measure for evaluating performance on imbalanced datasets.

$$\text{Macro-F1} = \frac{1}{C} \sum_{c=1}^{C} F1_c$$

C: Number of classes
$F1_c$: F1 score for class c

12.2.2 Class-Wise Metrics

1. **Precision**: The ratio of true positive predictions to the total predicted positives for class c indicating the accuracy of positive predictions for each class, reflecting the model's ability to avoid false positives.

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}$$

2. **Recall**: The ratio of true positive predictions to the actual positives for class c, measuring the model's ability to capture all relevant instances of a class, highlighting its sensitivity.

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c}$$

3. **F1 Score**: The harmonic mean of precision and recall for class c, provides a single metric that balances precision and recall, offering a comprehensive view of the model's performance for each class.

$$F1_c = 2 \times \frac{\text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

12.2.3 Visualization: Confusion Matrix Heatmap

A **confusion matrix** is utilized to visualize the model's performance across different classes, providing a detailed breakdown of correct and incorrect predictions.

$$\begin{pmatrix} \blacksquare & CM_{1,2} & \dots & CM_{1,C} \\ CM_{2,1} & CM_{2,2} & \dots & CM_{2,C} \\ \vdots & \vdots & \ddots & \vdots \\ CM_{C,1} & CM_{C,2} & \dots & CM_{C,C} \end{pmatrix}$$

where $CM_{i,j}$ represents the number of instances with true label i predicted as label j.

For interpretability, the confusion matrix is normalized to reflect percentages:

$$CM_{\text{percentage}} = \frac{CM_{i,j}}{\sum_{j=1}^{C} CM_{i,j}} \times 100\%$$

A heatmap of the normalized confusion matrix is generated, with each cell annotated with both the absolute count and the corresponding percentage. This visualization highlights areas of misclassification and the model's strengths across different sleep stages.

	W	N1	N2	N3	REM
W	$CM_{1,1}$	$CM_{1,2}$	$CM_{1,3}$	$CM_{1,4}$	$CM_{1,5}$
N1	$CM_{2,1}$	$CM_{2,2}$	$CM_{2,3}$	$CM_{2,4}$	$CM_{2,5}$
N2	$CM_{3,1}$	$CM_{3,2}$	$CM_{3,3}$	$CM_{3,4}$	$CM_{3,5}$
N3	$CM_{4,1}$	$CM_{4,2}$	$CM_{4,3}$	$CM_{4,4}$	$CM_{4,5}$
REM	$CM_{5,1}$	$CM_{5,2}$	$CM_{5,3}$	$CM_{5,4}$	$CM_{5,5}$

This heatmap facilitates the identification of specific misclassifications, enabling targeted improvements in model training and data augmentation strategies.

12.3 Model Architecture Implementation

12.3.1 Encoder

```
class Mish(nn.Module):
    def forward(self, x):
        return x * torch.tanh(F.softplus(x))

class SimpleSleepNet(nn.Module):

    def __init__(self, latent_dim=128, dropout=0.2):
        super(SimpleSleepNet, self).__init__()

        self.latent_dim = latent_dim
        self.dropout = nn.Dropout(p=dropout)

        self.conv_path = nn.Sequential(

            # First Convolutional Block:
            nn.Conv1d(in_channels=1, out_channels=32, kernel_size=64, stride=8, padding=63, dilation=1, bias=False),
            nn.BatchNorm1d(32),
            Mish(),
            self.dropout,

            # Second Convolutional Block:
            nn.Conv1d(in_channels=32, out_channels=64, kernel_size=32, stride=4, padding=62, dilation=2, bias=False),
            nn.BatchNorm1d(64),
            Mish(),
            self.dropout,
```

```
            # Third Convolutional Block:
            nn.Conv1d(in_channels=64, out_channels=128, kernel_size=16, stride=2, padding=60, dilation=4, bias=False),
            nn.BatchNorm1d(128),
            Mish(),
            self.dropout,
        )

        # Fully Connected Layer for Embedding:
        self.fc = nn.Sequential(
            nn.Linear(128, self.latent_dim),
            nn.BatchNorm1d(self.latent_dim),
            Mish(),
            self.dropout
        )


    def forward(self, x):
        x = self.conv_path(x)                          # Apply convolutional path (Batch, 128, L')
        x = F.adaptive_avg_pool1d(x, 1)                # Global average pooling (Batch, 128, 1)
        x = x.view(x.size(0), -1)                      # Flatten the tensor (Batch, 128)
        x = self.fc(x)                                 # Apply fully connected layer (Batch, latent_dim)
        x = F.normalize(x, p=2, dim=1)                 # Normalize embeddings to unit length
        return x
```

12.3.2 Classifier

```
class Mish(nn.Module):
    def forward(self, x):
        return x * torch.tanh(F.softplus(x))

class SleepStageClassifier(nn.Module):

    def __init__(self, input_dim: int = 128, num_classes: int = 5, dropout_probs: float = 0.5):
        super(SleepStageClassifier, self).__init__()

        self.classifier = nn.Sequential(
            nn.Linear(input_dim, 256),     # First linear layer
            nn.BatchNorm1d(256),           # Batch normalization
            Mish(),                        # Mish activation
            nn.Dropout(p=dropout_probs),   # First dropout layer
            nn.Linear(256, 128),           # Second linear layer
            nn.BatchNorm1d(128),           # Batch normalization
            Mish(),                        # Mish activation
            nn.Dropout(p=dropout_probs),   # Second dropout layer
            nn.Linear(128, num_classes)    # Output layer
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.classifier(x)
        return x
```

12.4 Visualizing Data Augmentations

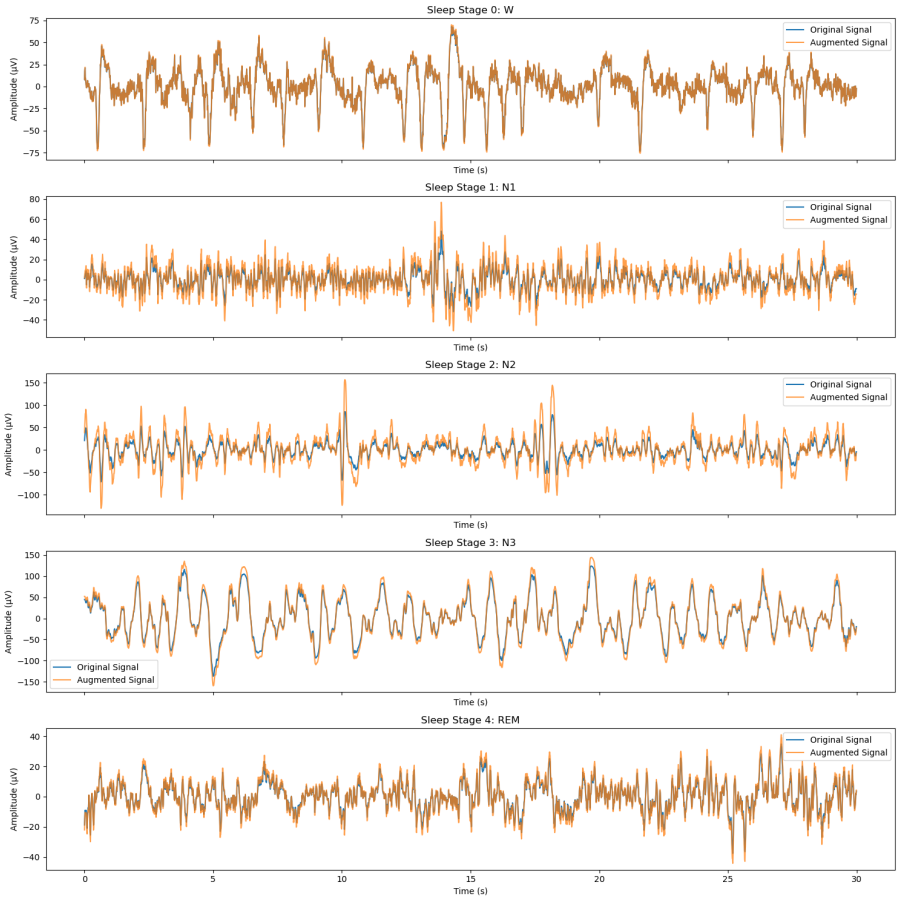


Figure 4: Effect of Random Amplitude Scaling on an EEG signal.

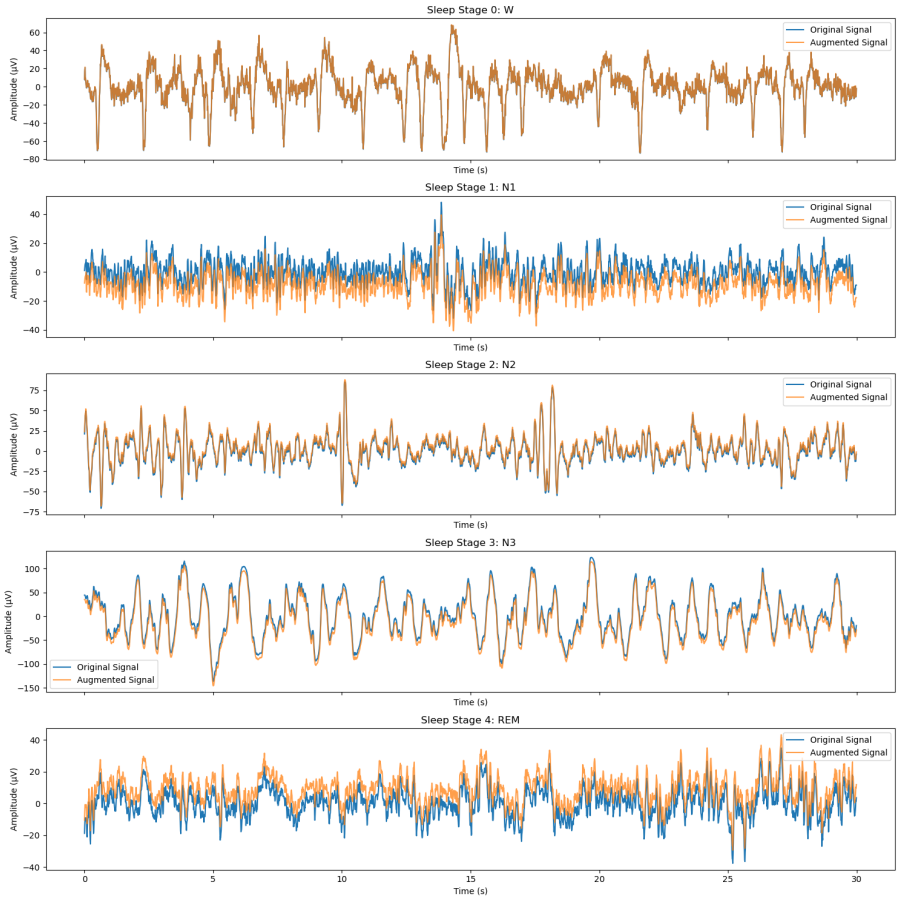


Figure 5: Effect of Random DC Shift on an EEG signal.

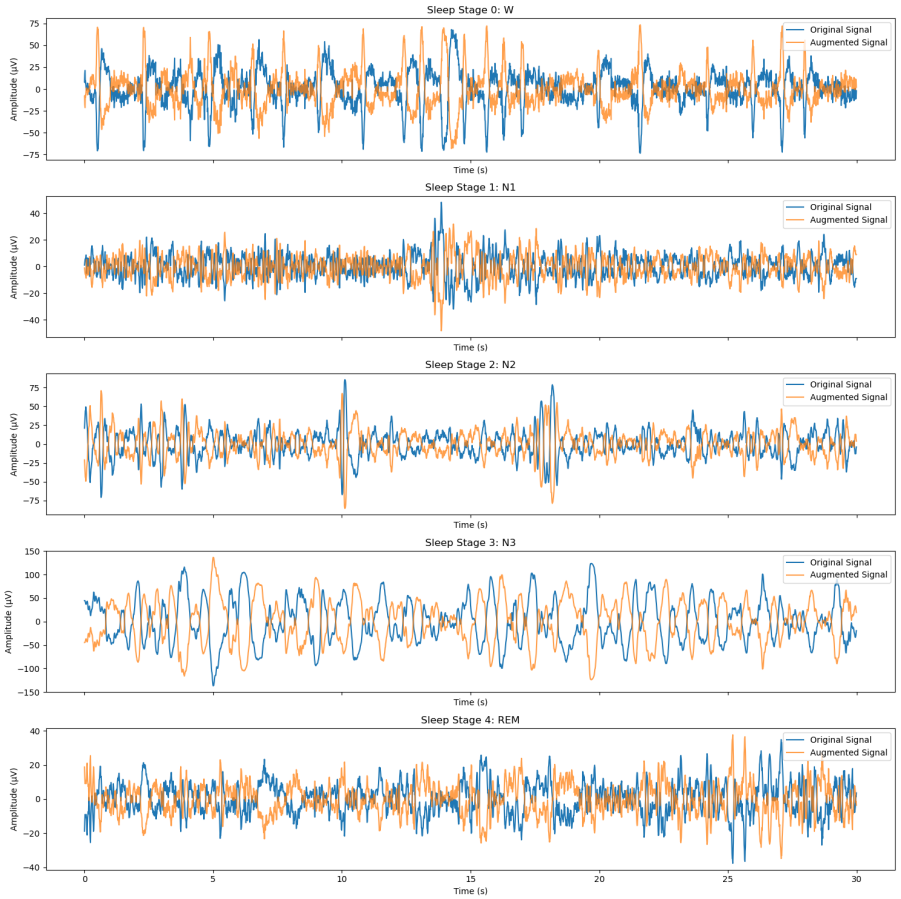


Figure 6: Effect of Sign Flip on an EEG signal.

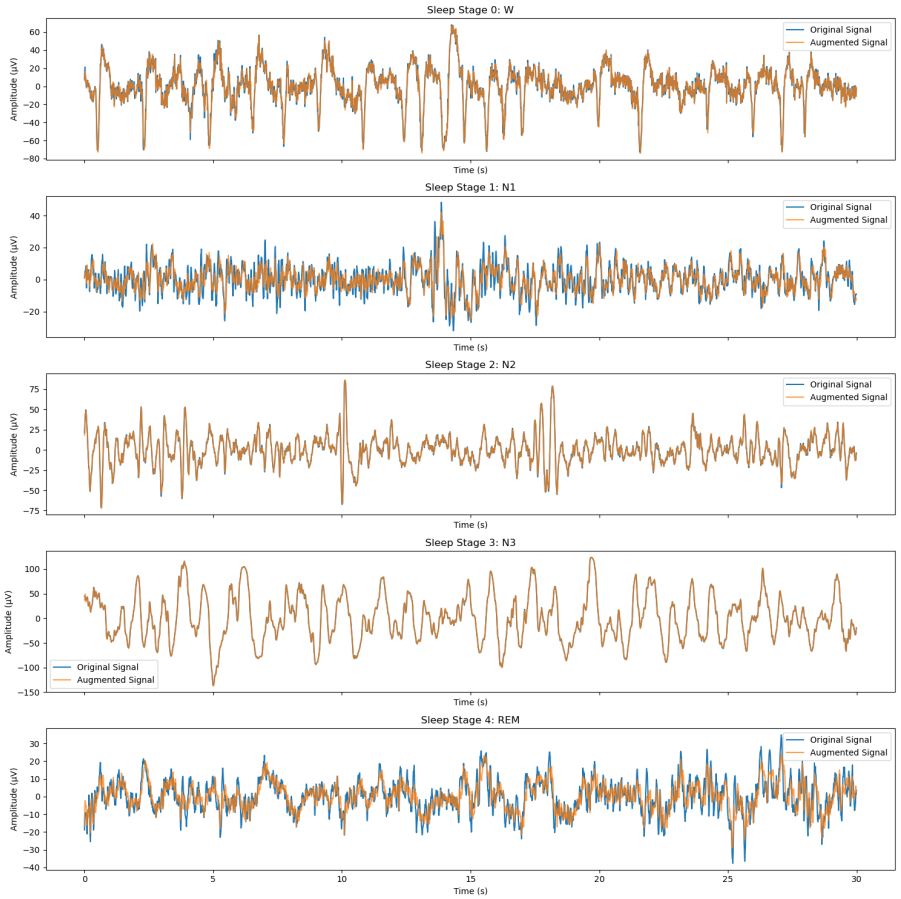


Figure 7: Effect of Random Band-Stop Filtering on an EEG signal.

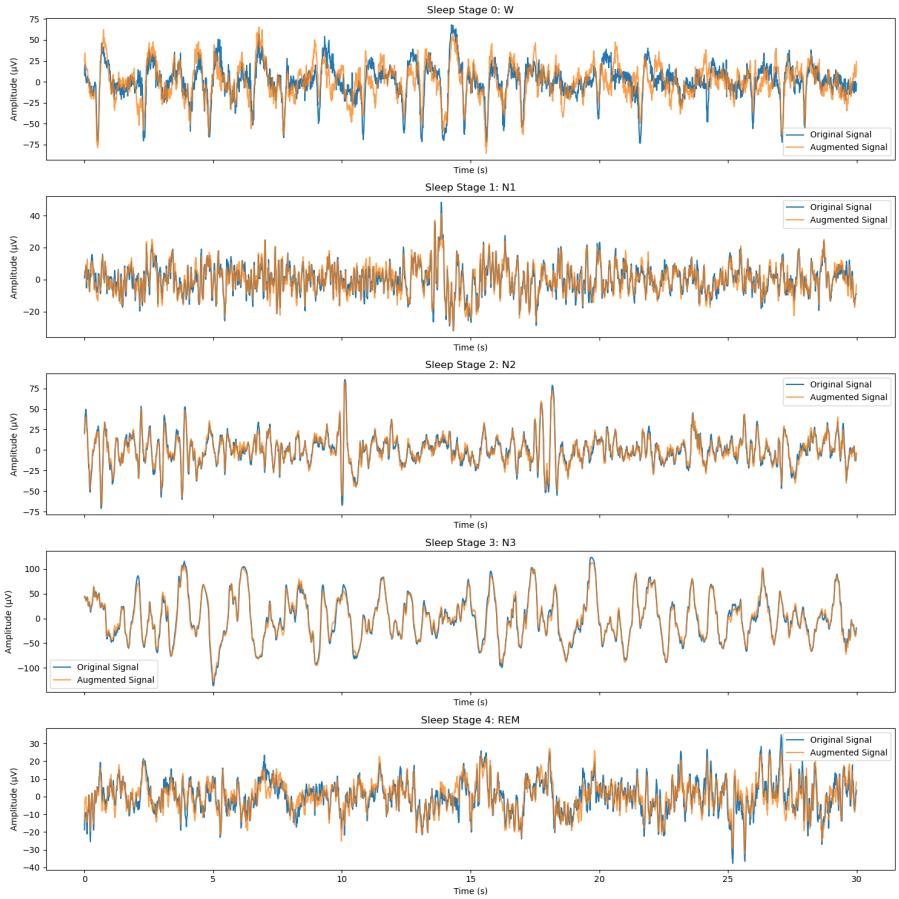


Figure 8: Effect of Tailored Mixup on two EEG signals.

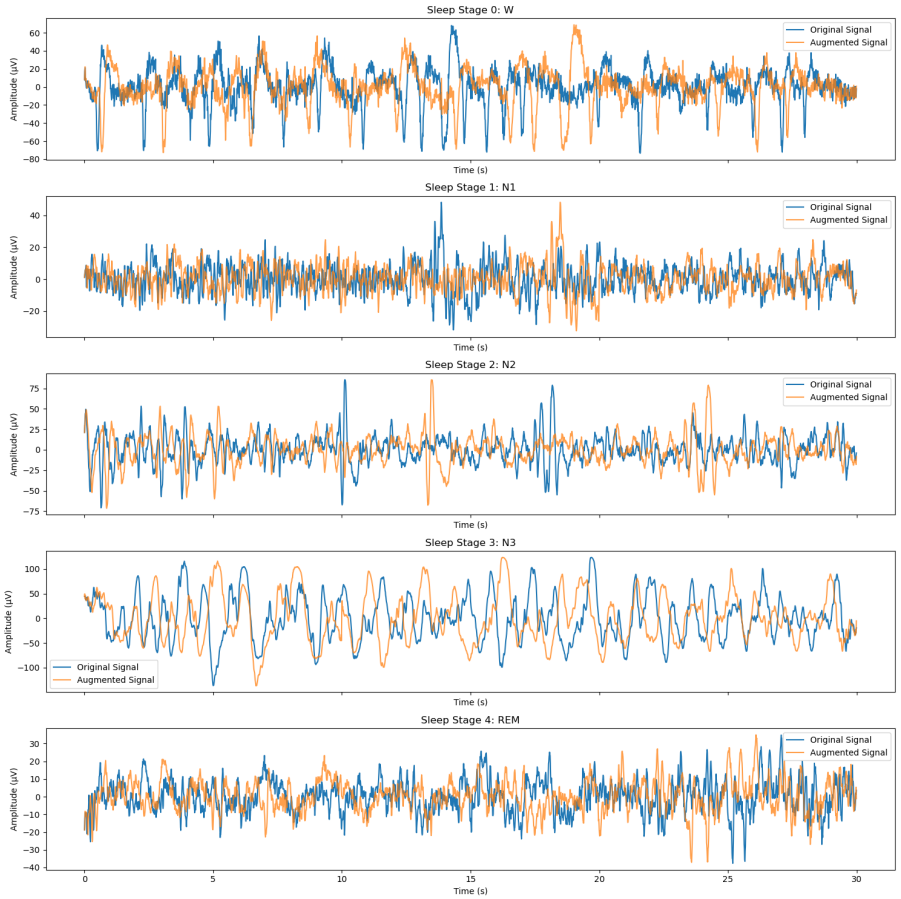


Figure 9: Effect of Cutout and Resize on an EEG signal.

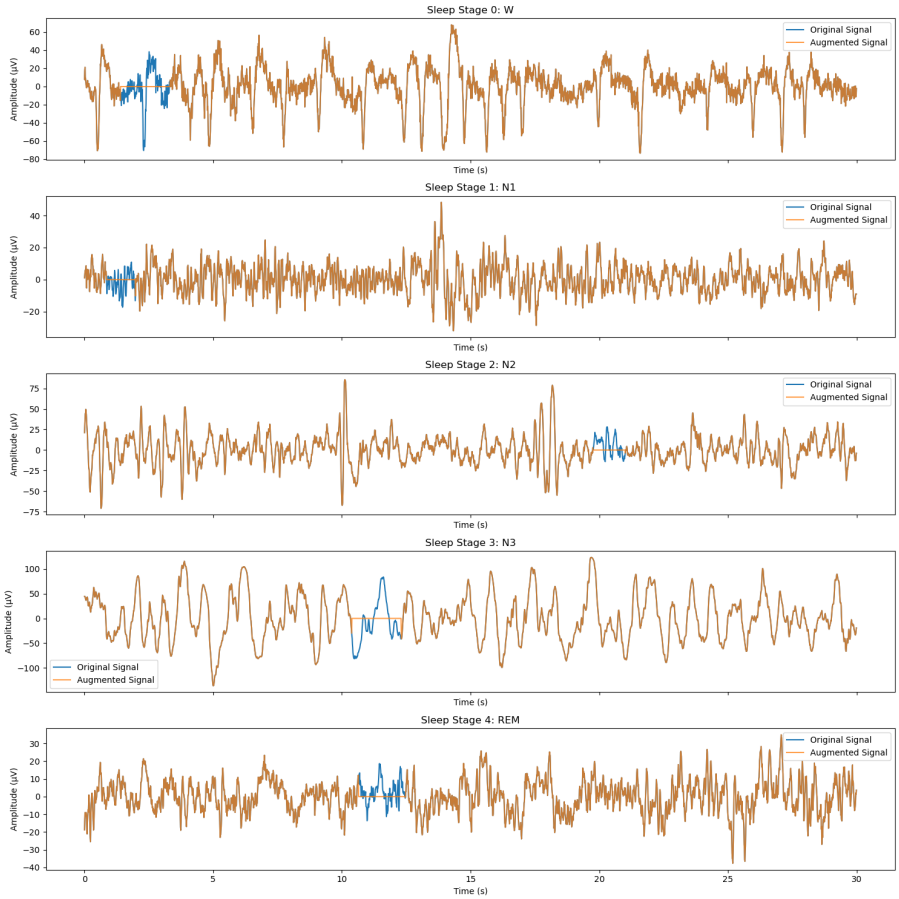


Figure 10: Effect of Random Zero Masking on an EEG signal.

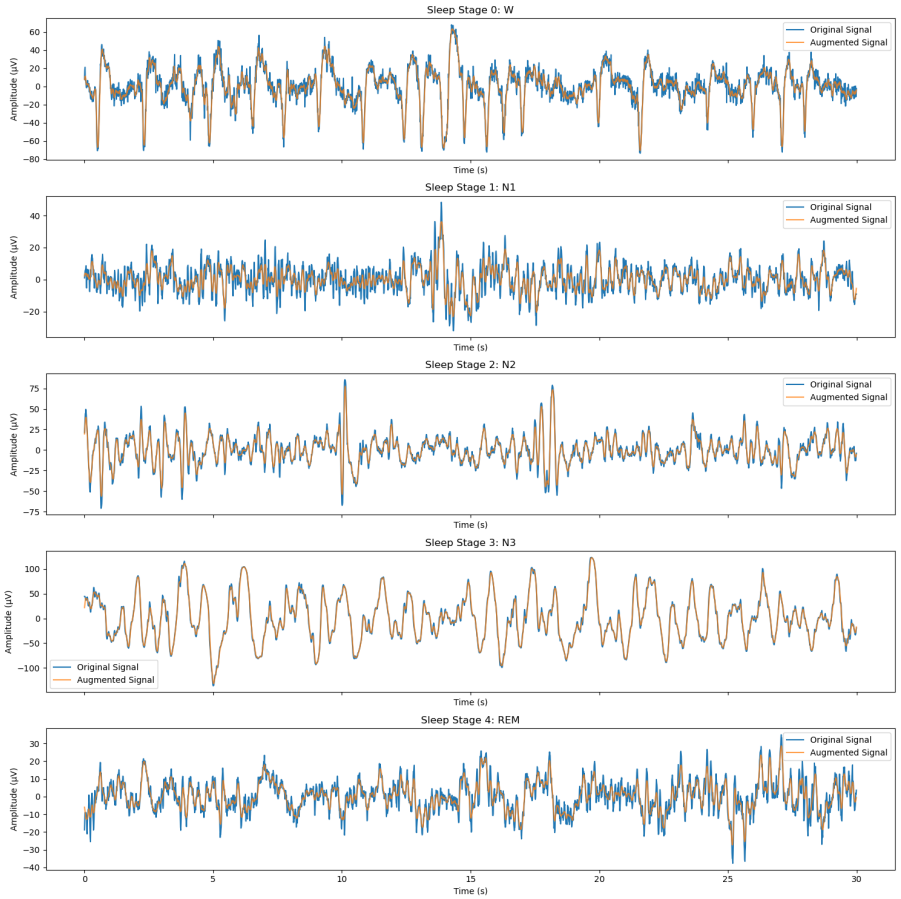


Figure 11: Effect of Average Filtering on an EEG signal.

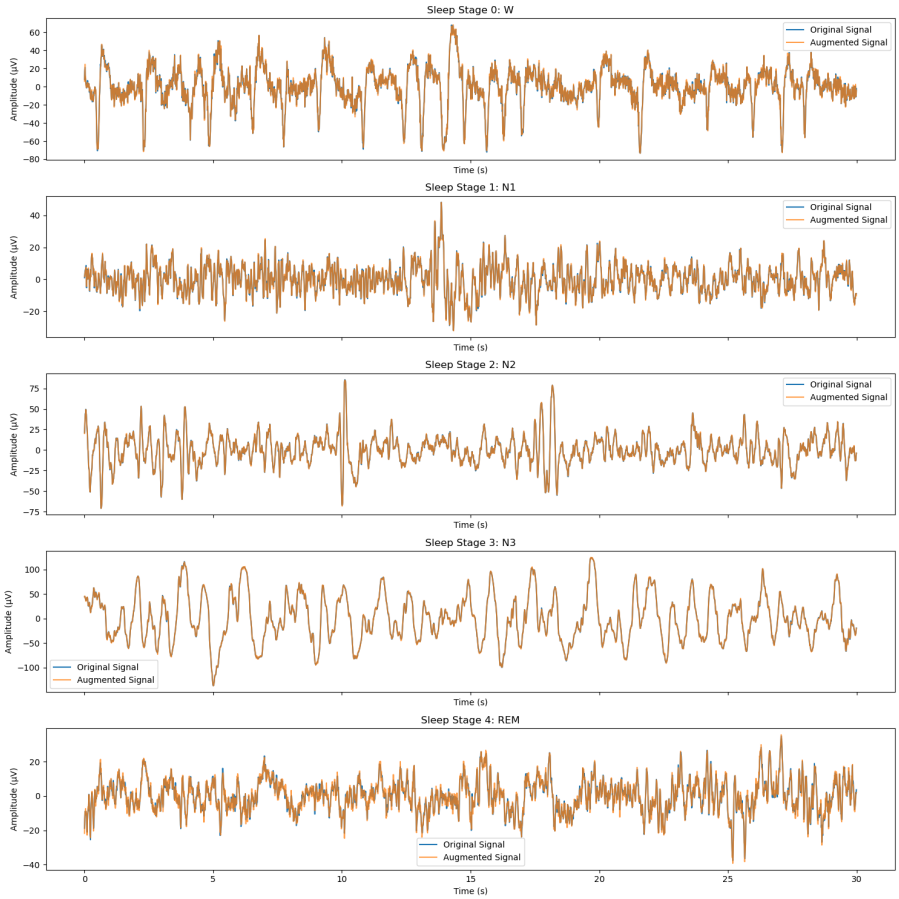


Figure 12: Effect of Random Additive Gaussian Noise on an EEG signal.

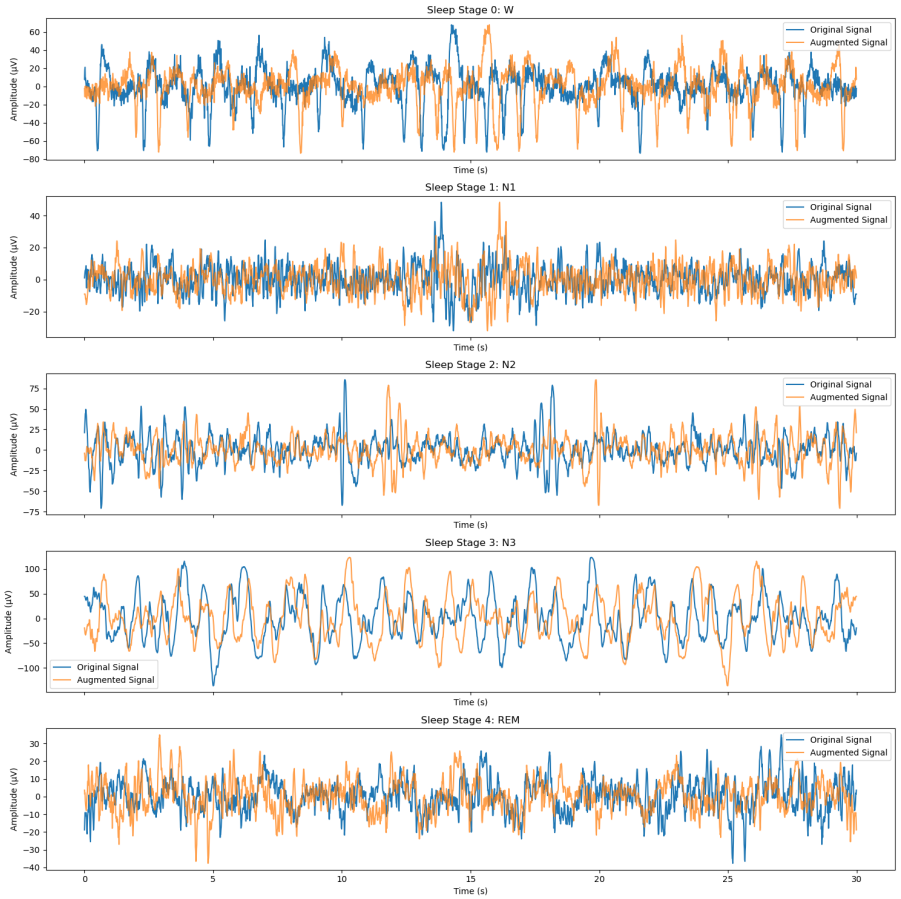


Figure 13: Effect of Time Reversal on an EEG signal.

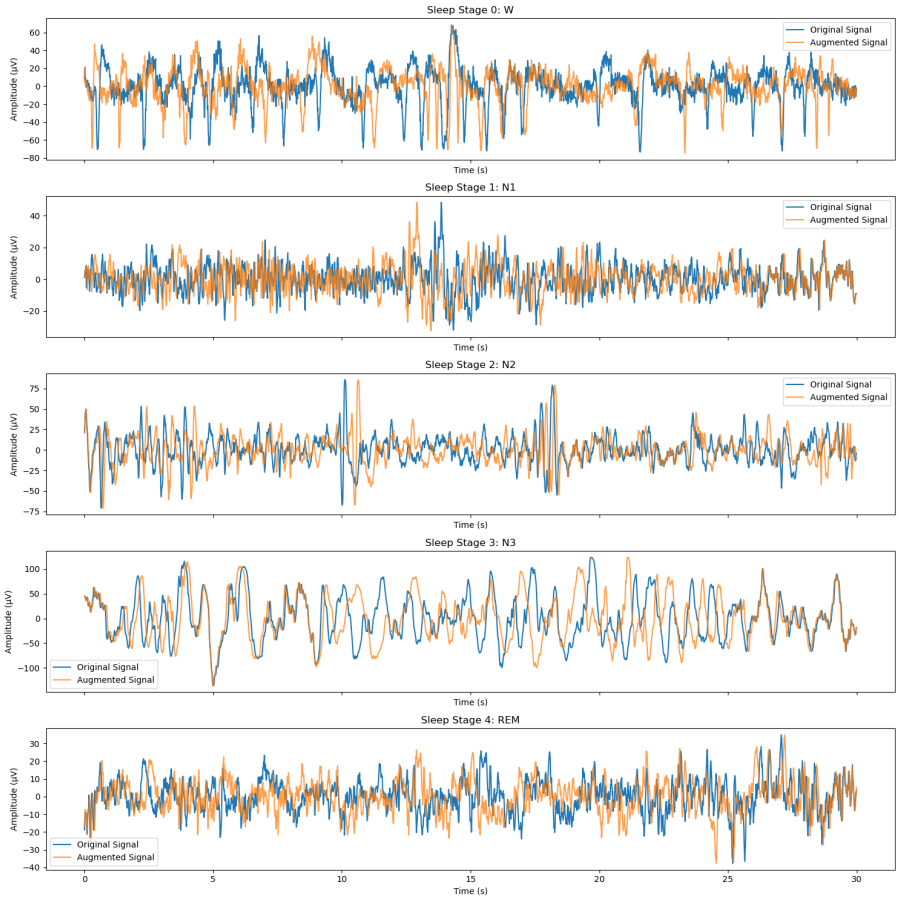


Figure 14: Effect of Time Warping on an EEG signal.

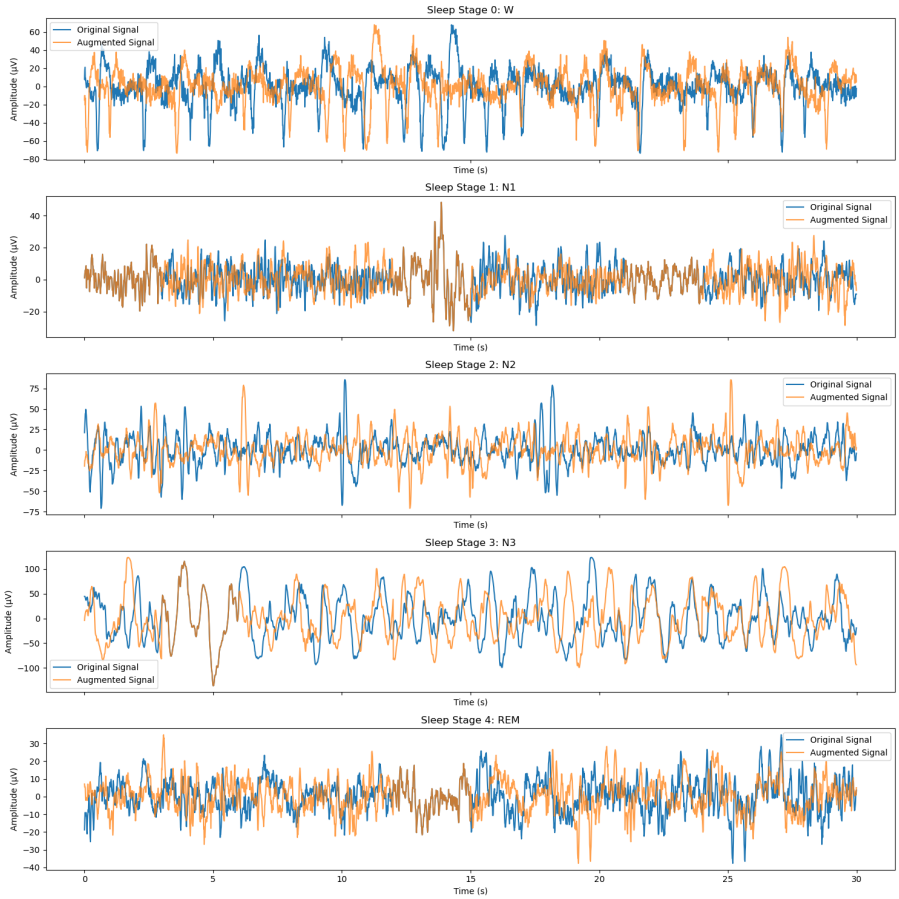


Figure 15: Effect of Permutation on an EEG signal.

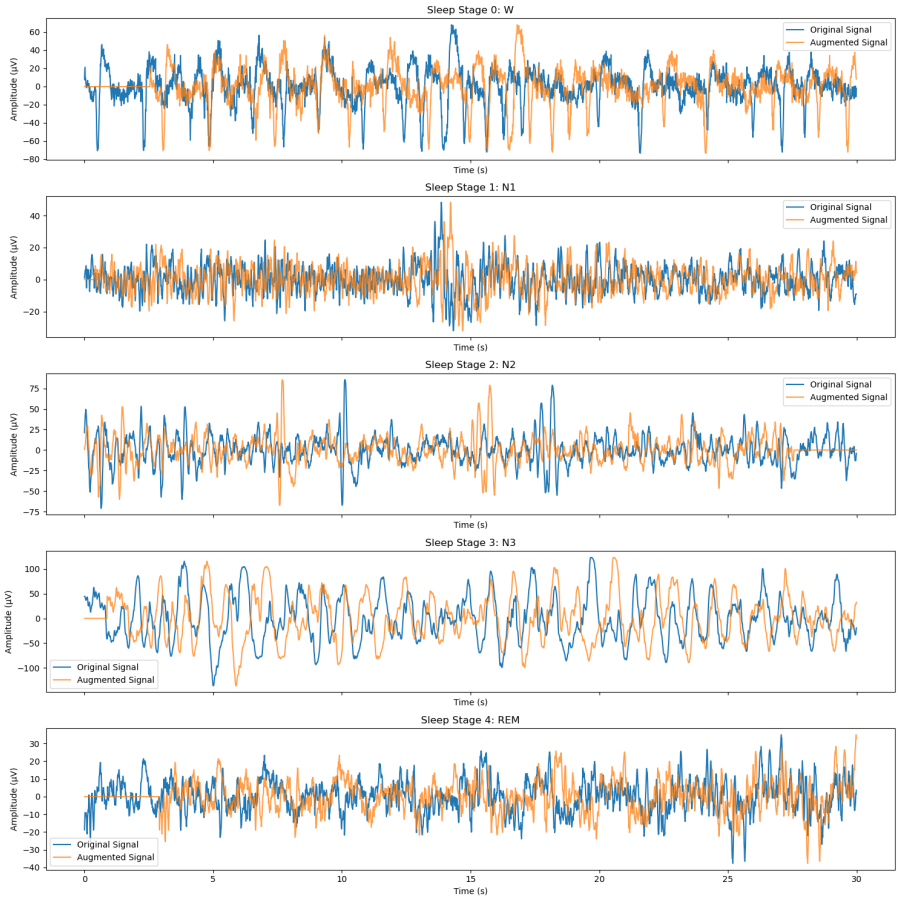


Figure 16: Effect of Random Time Shift on an EEG signal.