Self-Supervised Learning with Attention-based Latent Signal Augmentation for Sleep Staging with Limited Labeled Data

Harim Lee* , **Eunseon Seong*** and **Dong-Kyu Chae†**
Department of Artificial Intelligence, Hanyang University, Seoul, South Korea
{hrimlee, emilyseong, dongkyu}@hanyang.ac.kr

Abstract

Sleep staging is an important task that enables sleep quality assessment and disorder diagnosis. Due to dependency on manually labeled data, many researches have turned from supervised approaches to self-supervised learning (SSL) for sleep staging. While existing SSL methods have made significant progress in terms of its comparable performance to supervised methods, there are still some limitations. Contrastive learning could potentially lead to false negative pair assignments in sleep signal data. Moreover, existing data augmentation techniques directly modify the original signal data, making it likely to lose important information. To mitigate these issues, we propose **S**elf-**S**upervised **L**earning with **A**ttention-aided **P**ositive **P**airs (SSLAPP). Instead of the contrastive learning, SSLAPP carefully draws high-quality positive pairs and exploits them in representation learning. Here, we propose attention-based latent signal augmentation, which plays a key role by capturing important features without losing valuable signal information. Experimental results show that our proposed method achieves state-of-the-art performance in sleep stage classification with limited labeled data. The code is available at: https://github.com/DILAB-HYU/SSLAPP

1 Introduction

Sleep staging is essential for sleep-related healthcare such as sleep quality assessment [Perez-Pozuelo *et al*., 2020] and disorder diagnosis [Sateia, 2014]. Traditionally, sleep staging was performed manually by experts using polysomnography (PSG) records [Hedner *et al*., 2011]. Each 30 second epochs is assigned to a sleep stage (e.g., WAKE, REM, N1, N2, N3) based on sleep standards [Wolpert, 1969; Berry *et al*., 2012]. Although these standards provide specific transition rules between stages, manual sleep staging is time consuming, laborious, and potentially prone to bias. Thus, many studies have attempted to automate this process employing machine learning models [Supratak *et al*., 2017; Malafeev *et al.*, 2018; Phan *et al*., 2019; Jiang *et al.*, 2019; Jia *et al*., 2021]. Among them, recent researches employ Deep Neural Networks that require little to no feature engineering have been achieving state-of-the-art performance.

However, most studies have formulated this task as a fully supervised learning problem, where we assume that all of the training examples in the dataset are labeled. Unfortunately, it is not a realistic assumption in many real-world situations. For example, annotating PSG records, which is in the form of time-series signal data and collected continuously on a daily basis from wearable devices, is extremely costly and laborious [Eldele *et al*., 2021b], thus leading to scarcely labeled data. Supervised learning would fail in this kind of situation because its performance heavily depends on the amount and quality of labels. In addition, the latent representation of data learned by a supervised learning model is usually limited to the training data, which is not generalizable to other applications, thereby leading to a good performance only within the specific training dataset. Therefore, assuming scarcity of labeled data is more in line with reality and more reasonable.

In this context, there have been many new methods aiming towards sleep staging with limited labeled data, where notable examples include CoSleep [Ye *et al*., 2021] and SleepDPC [Xiao *et al*., 2021]. These methods adopt *self-supervised learning* (SSL) that learns latent representations of data in an unsupervised manner and then solves its main problem (e.g., sleep staging) in the downstream task. In the representation learning step, *contrastive learning* which assigns positive and negative pairs and forces positive pairs to have similar latent representation and negative pairs to have dissimilar representation is employed. Empirically, these methods show a classification accuracy comparable to that of a fully supervised model (trained with $80 \sim 90\%$ of labeled data) using only a small portion ($10 \sim 20\%$) of the dataset.

Despite the huge accomplishments of the SSL-based methods, there are still some limitations that need to be addressed. First, the contrastive learning employed in their unsupervised learning has problem in its positive and negative pair assignment. Since there is no labeled information, negative pair is created by an example's augmentation and anything that is different from the example. However, the probability that they could be false negative is much higher than in the image do-

*Both authors contributed equally to this research.
†Corresponding author.

main where contrastive learning is widely used. This is because since the number of classes for sleep staging is usually around 5-7, which is much smaller than the number of classes in the image domain, the negative pairs sampled without label information are highly likely to belong to the same class[1]. Second, the data augmentation techniques employed in their methods to create positive pairs, such as adding noise, flipping, and scaling, are not tailored to the signal domain; instead, these are simple variations of augmentation techniques used in the image domain. These augmentation methods take modification directly on the original signal data, which eventually leads to loss of significant information.

In order to mitigate the aforementioned limitations, we propose **S**elf-**S**upervised **L**earning with **A**ttention-aided **P**ositive **P**airs (SSLAPP) for the sleep staging task where labeled data is not readily available. SSLAPP is composed of fully unsupervised model for latent representation learning, and a fine-tuning model with limited labels for sleep stage classification. For the unsupervised representation learning, we develop an adversarial training framework together with an auxiliary network to learn latent representation. In order to extract class identity representation, we carefully draw positive pairs and let our model push them to be projected closely. Here, we propose an attention-based augmentation technique that we believe is able to capture important features within the signal while not losing important information, thus providing better transformation compared to other simple augmentation techniques. Unlike existing augmentation methods that directly modify raw signal, our augmentation is performed in the latent space. Our experimental results show that the proposed method achieves state-of-the-art performance in sleep stage classification with limited labeled data used. Our work especially exhibits an outstanding performance in its transferable generalization to other datasets owing to the well pre-trained representations. The ablation study confirms the effectiveness of the strategies taken by SSLAPP.

2 Related Work

This section briefly summarizes the recent studies on sleep staging based on the deep learning from raw signals.

- **Fully supervised learning:** SleepEEGNET [Mousavi *et al.*, 2019] employs CNNs and bidirectional RNN to extract time-invariant features from raw signal. DeepSleepNET [Supratak *et al.*, 2017] utilizes CNN with two different sizes of filters and bidirectional LSTM to extract features and learn the transition rules between sleep stages. AttnSleep [Eldele *et al.*, 2021a] adopts multi-resolution CNN to extract low and high-frequency features and the adaptive feature re-calibration strategy to improve the quality of the extracted features. It also uses an encoder network based on the multi-head attention to capture temporal dependencies among features.
- **Self-supervised learning:** Recently, self-supervised contrastive learning has been introduced in the area of sleep staging, which learns representations through an unsupervised manner as a pretext task and conducts fine-tuning on a downstream task. Here, one critical aspect of representation learning is choosing how to construct the positive pairs. From this perspective, the authors of [Jiang *et al.*, 2021] generates positive pairs via various augmentation techniques such as time warping, adding Gaussian noise, flipping, and permutation. TS-TCC [Eldele *et al.*, 2021b] produces two views as positive pairs using strong and weak augmentations such as permutation, jitter and scale. However, these augmentations directly modify the raw signal which can lead to information loss ruining the continuous characteristics of these signals. Instead of augmentation, SleepDPC [Xiao *et al.*, 2021] applies an autoregressive model with length-based contrastive coding to learn better representations from signals. CoSleep [Ye *et al.*, 2021] exploits complementary information from time view and frequency view of signals to mine more positive samples.

Of the two categories of research mentioned above, our study belongs to the second group. The studies in the second category have focused on contrastive learning, which requires a massive number of negative pairs to find one semantically similar representation, and some of them use handcrafted augmentations which are not suitable for signal data. Our study tries to ameliorate the above limitations.

3 Proposed Method

The overall architecture of SSLAPP is illustrated in Figure 1. Our work is based on the following three ideas: 1) Exploiting both electroencephalography (EEG) and electrooculography (EOG) signal [Lajnef *et al.*, 2015; Huang *et al.*, 2014] so as to capture the multimodal semantic information that each bio-signal is comprised of; 2) Taking the adversarial training strategy incorporating an auxiliary network for effective representation learning; 3) Augmenting an attention-aided positive pairs on the latent space, instead of on the input space.

3.1 Adversarial Representation Learning

Our model is based on the SSL frameworks with generative model that have achieved striking success in learning representation of images: Elastic-InfoGAN [Ojha *et al.*, 2019] and SimSiam [Chen and He, 2021], and we carefully re-design several components of the framework to make it suitable for learning signal data. As shown in Figure 1, our SSLAPP is composed of the following four neural networks: the Encoder $\mathcal{F}$ for extracting features from the signal, the Generator $\mathcal{G}$ and Discriminator $\mathcal{D}$ for adversarial training, and the latent predictor $\mathcal{K}$ to realize class identity representation.

First (see the left side of Figure 1(a)), the generator receives a concatenation of a latent code $\mathbf{c}$ that represents class identity and random noise $\mathbf{z}$. In specific, latent code $\mathbf{c}$ targets the semantic feature of prior distribution which represents the class distribution of sleep stages in our task. To discover the latent factors, generation is processed to maximize the mutual information (MI) between the latent code and the generated signal, $I(\mathbf{c}; \mathcal{G}(\mathbf{c}, \mathbf{z}))$, as:

$$\min_{\mathcal{G}} \max_{\mathcal{D},\mathcal{F}} V_{MI}(\mathcal{D}, \mathcal{F}, \mathcal{G}) = V(\mathcal{D}, \mathcal{F}, \mathcal{G}) - \lambda I(c; \mathcal{G}(\mathbf{c}, \mathbf{z})) \quad (1)$$

where $V(\mathcal{D}, \mathcal{F}, \mathcal{G})$ is the general objective of the generative adversarial network ;$V(\mathcal{D}, \mathcal{F}, \mathcal{G}) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log \mathcal{D}(\mathcal{F}(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim \text{noise}}[\log(1 - \mathcal{D}(\mathcal{F}(\mathcal{G}(\mathbf{c}, \mathbf{z}))))]$.

[1] Appendix A provides more detailed description of this problem.

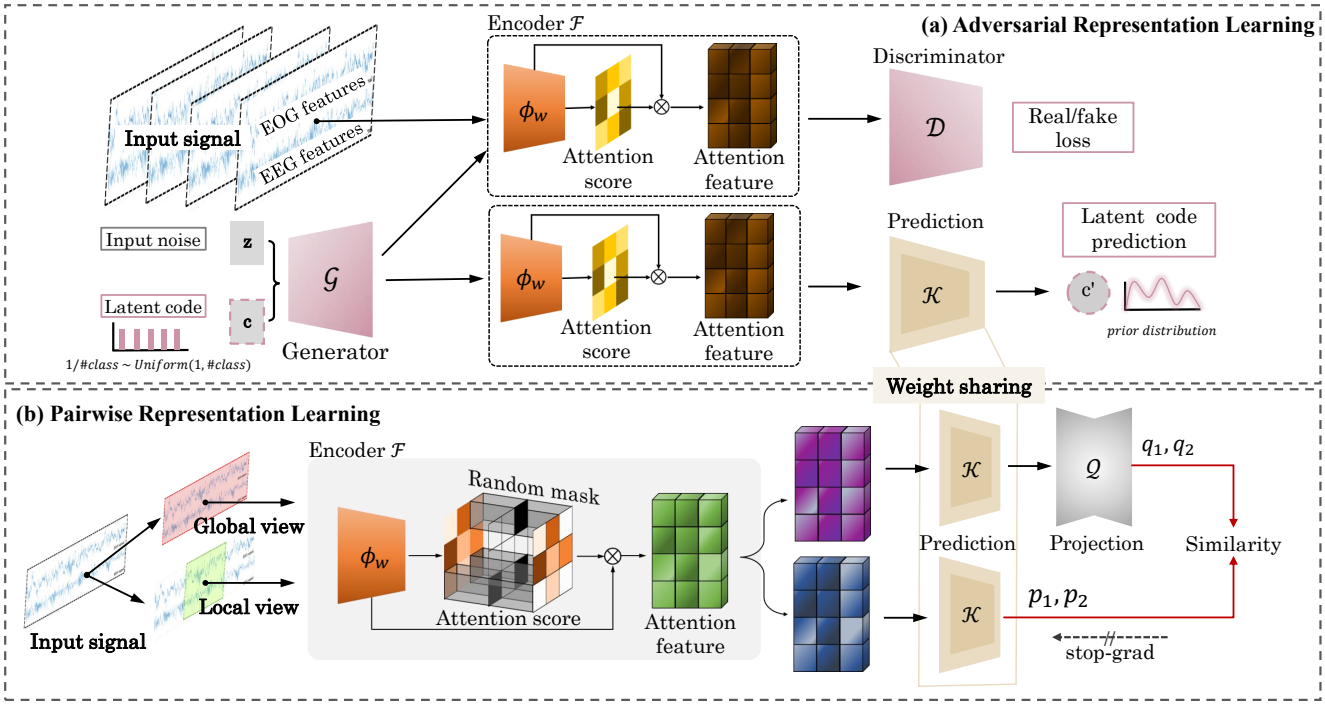


Figure 1: The overall architecture of the proposed SSLAPP composed of 2 components. (a) Adversarial representation learning: learns disentangled representation with an auxiliary network added to generative model. (b) Pairwise representation learning with only positive pairs: forces the latent representation of positive pairs to be projected closely, helping the training procedure of auxiliary network. Note that the training mechanisms of (a) and (b) are proceeded simultaneously and all the parameters are shared.

Since $I(\mathbf{c}; \mathcal{G}(\mathbf{c}, \mathbf{z}))$ requires posterior distribution $P(\mathbf{c}|\mathbf{x})$ which is computationally intractable, we use the lower bound of it which needs the auxiliary network $\mathcal{K}$ to approximate $P(\mathbf{c}|\mathbf{x})$:

$$I(\mathbf{c}, \mathcal{G}(\mathbf{c}, \mathbf{z})) = \mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}), \mathbf{x}\sim\mathcal{G}(\mathbf{c},\mathbf{z})}[\log \mathcal{K}(\mathbf{c}|\mathbf{x})] + H(\mathbf{c}) \quad (2)$$

where $H(\mathbf{c})$ is the entropy of the latent code distribution. This leads the auxiliary network to learn the latent code representation.

Then, our Encoder $\mathcal{F}(\phi_w(\mathbf{x}_k)) \in \mathbb{R}^{N\times m\times d}$ proceeds 1D convolution $\phi_w(\mathbf{x}_k)$ on input $\mathbf{x}$ to obtain its embedding, where $\mathbf{x} \in \mathbb{R}^{N\times L\times C}$ be 30 second 100Hz two channel input signal where N, L and C represents batch size, number of sampling points and channel used. Discriminator $\mathcal{D}(\mathcal{F}(\phi_w(\mathbf{x}_k)))$ outputs 0 (if fake) or 1 (if real). Generator $\mathcal{G}(\mathbf{z}, \mathbf{c}) = \mathbf{x}' \in \mathbb{R}^{N\times L\times C}$ generates a fake signal data containing the information of the latent code $\mathbf{c}$. Latent predictor $\mathcal{K}(\mathcal{F}(\phi_w(\mathbf{x}_k)) = \mathbf{p} \in \mathbb{R}^{N\times t}$ or $\mathbf{c}' \in \mathbb{R}^{N\times 5}$ returns t-dimensional latent representation and latent code output from the extracted feature. Therefore, mutual information is reformulated with the latent predictor $\mathcal{K}$ as:

$$I(\mathbf{c}, \mathcal{G}(\mathbf{c}, \mathbf{z})) = I(\mathbf{c}, \mathcal{K}(\mathcal{F}(\phi_w(\mathcal{G}(\mathbf{c}, \mathbf{z}))))) = I(\mathbf{c}, \mathbf{c}')). \quad (3)$$

Finally, the training objective of SSLAPP becomes:

$$\min_{\mathcal{G},\mathcal{K}} \max_{\mathcal{D},\mathcal{F}} \mathcal{L}_{\text{SSLAPP}} = \underbrace{V_{MI}(\mathcal{D}, \mathcal{F}, \mathcal{G}, \mathcal{K})}_{\text{Adversarial loss}} + \lambda_{\text{pos}} \underbrace{\mathcal{L}_{\text{pos}}(\mathcal{K}, \mathcal{Q})}_{\text{Pairwise loss}} \quad (4)$$

where λ_{pos} controls the contribution of the pairwise loss $\mathcal{L}_{\text{pos}}$ which will be described in 3.2 to the objective of SSLAPP.

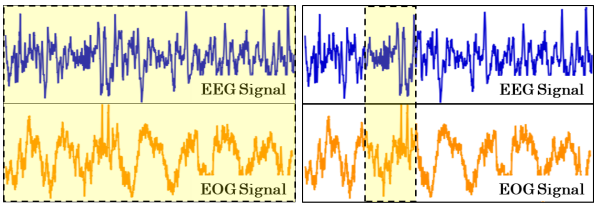
3.2 Pairwise Representation Learning

In order to enable the latent predictor to learn effective class identity representation, we employ a self-supervised learning objective $\mathcal{L}_{\text{pos}}$ illustrated in Figure 1(b). Here, contrastive learning has been the most popular choice which assigns an example's own augmentation as positive pair and all the others as negative pair. However, this leads to a false negative problem by assigning the same sleep stage as negative pair and eventually leading to a contradictory objective with what the positive pair learns. Instead of contrastive learning, we design a different learning framework that is tailored to the sleep staging problem.

Let $\mathbf{z}_1, \mathbf{z}_2$ be the positive pair, two augmented views from input $\mathbf{x}$. How the positive pair is constructed is shown in the next subsection. A projection network $\mathcal{Q}(\cdot)$ is added to transform one output view to a different view. Outputs of the latent predictor $\{\mathbf{p}_1, \mathbf{p}_2\}$ and the projector $\{\mathbf{q}_1, \mathbf{q}_2\}$ are used to measure the similarity of the positive pairs, where $\mathbf{p}_1 = \mathcal{K}(\mathbf{z}_1)$, $\mathbf{p}_2 = \mathcal{K}(\mathbf{z}_2)$ and $\mathbf{q}_1 = \mathcal{Q}(\mathcal{K}(\mathbf{z}_1))$, $\mathbf{q}_2 = \mathcal{Q}(\mathcal{K}(\mathbf{z}_2))$. To maximize the similarity of the positive pairs, negative cosine similarity is used as the loss function to minimize:

$$d(\mathbf{q}_1, sg(\mathbf{p}_2)) = -\frac{\langle \mathcal{Q}(\mathbf{p}_1), sg(\mathbf{p}_2)\rangle}{\|\mathcal{Q}(\mathbf{p}_1)\| \|sg(\mathbf{p}_2)\|} \quad (5)$$

where sg denotes the '*stop-gradient*' operator that blocks gradient flow so as to keep the model parameters from being updated [Chen and He, 2021]. As a result, one projected view is used for updating the model parameters while the other view is utilized for obtaining distance between the two views.



(a) Global loss (b) Local loss

Figure 2: Comparison of each view in global loss and local loss for effective representation learning within an epoch.

In addition, we design two types of loss terms: *global loss* $\mathcal{L}_{global}$ and *local loss* $\mathcal{L}_{local}$. This is motivated by the observations from prior work [Berry *et al.*, 2012] that, according to the American Academy of Sleep Medicine (AASM) manual, sleep stages are shown to have different salient waves within an epoch. Hence capturing both the global feature and the local feature will be promising.

Figure 2 represents what each loss takes in the view. Formally, global loss $\mathcal{L}_{global}$ is subject to full 30 seconds input $\mathbf{x} \in \mathbb{R}^{N \times L \times C}$ which is:

$$\mathcal{L}_{global} = d(\mathbf{q}_1^g,\ sg(\mathbf{p}_2^g)) + d(\mathbf{q}_2^g,\ sg(\mathbf{p}_1^g)) \tag{6}$$

where $\mathbf{p}_1^g, \mathbf{p}_2^g, \mathbf{q}_1^g, \mathbf{q}_2^g$ is a global feature extracted using global view $\mathbf{x}$. $\mathcal{L}_{local}$ is applied to capture the local feature representation for each α second, $\mathbf{x}_l \in \mathbb{R}^{N \times \alpha \times C}$ within a whole 30 second epoch. For j segments sectioned by α seconds, the summated view of j segments is equal to the global view. Then the $\mathcal{L}_{local}$ is defined as follows:

$$\mathcal{L}_{local} = \frac{1}{j} \sum_{i=1}^{j} \Big\{ d(\mathbf{q}_1^i,\ sg(\mathbf{p}_2^i)) + d(\mathbf{q}_2^i,\ sg(\mathbf{p}_1^i)) \Big\}. \tag{7}$$

We minimize the final objective loss $\mathcal{L}_{\text{pos}}$:

$$\mathcal{L}_{\text{pos}} = \lambda_g \cdot \mathcal{L}_{global} + \lambda_l \cdot \mathcal{L}_{local}, \tag{8}$$

where λ_g and λ_l controls the importance of each loss term.

3.3 Attention-based Latent Signal Augmentation

This subsection elaborates on how we construct a positive pair based on the proposed Attention-based augmentation on latent space during training. The Attention mechanism originally aims to capture the important regions of the input space by applying the attention weight vector imposed on each feature. Our idea is to construct augmented pairs extending the objective of attention to detect the saliency of the *embedded* input signal. To this end, we combine the *encoding* and augmentation phase to generate semantically similar positive pairs on the *latent space*, which we expect solves the problems caused by direct manipulation of the data on the input space.

We introduce an attention structure to our encoder network $\mathcal{F}(\phi_w(\mathbf{x}_k))$ that includes the convolution layer $\phi_w(\mathbf{x}_k)$ extracting features from the input. The attention score is computed by:

$$\mathbf{s} = \frac{exp(score(\phi_w(\mathbf{x}_k)))}{\sum_{k=1}^{d} exp(score(\phi_w(\mathbf{x}_k)))} \tag{9}$$

Algorithm 1 SSLAPP's pairwise representation learning.

Input: EEG and EOG concatenated signals $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, total number of batch size N, convolution layers ϕ_w, latent predictor $\mathcal{K}$, projector $\mathcal{Q}$, Random mask M.

for sampled mini batch $\{\mathbf{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    # first augmentation
    $\mathbf{s}_1 = score(\phi_w(\mathbf{x}_k))$
    $\mathbf{z}_1 = \phi_w(\mathbf{x}_k) \cdot (\mathbf{s}_1 \odot M)$
    # second augmentation
    $\mathbf{s}_2 = score(\phi_w(\mathbf{x}_k))$
    $\mathbf{z}_2 = \phi_w(\mathbf{x}_k) \cdot (\mathbf{s}_2 \odot M)$
  end for
  $\mathbf{p}_1 = \mathcal{K}(\mathbf{z}_1)$ # latent prediction
  $\mathbf{p}_2 = \mathcal{K}(\mathbf{z}_2)$ # latent prediction
  $\mathbf{q}_1 = \mathcal{Q}(\mathbf{p}_1)$ # latent projection
  $\mathbf{q}_2 = \mathcal{Q}(\mathbf{p}_2)$ # latent projection
end for
define the negative similarity loss (5) with $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{q}_1$, $\mathbf{q}_2$.

where $score(\cdot)$ denotes inner product computation: $\phi_w(\mathbf{x}_k)^T \phi_w(\mathbf{x}_k)$. After then, we adapt a random mask M [Gao *et al.*, 2021] that enables data augmentation during encoding. The augmented data $\mathbf{z}$ is generated by masking out the attention score as:

$$\mathbf{z} = \phi_w(\mathbf{x}_k) \cdot (\mathbf{s} \odot M), \tag{10}$$

where $M \in \{0, 1\}^{m \times d}$ is a random mask obtained with a mask ratio r, and $\odot$ stands for element-wise multiplication.

In this way, we generate positive pairs $\{\mathbf{z}_1, \mathbf{z}_2\}$ by putting the input instance $\mathbf{x}_k$ to the encoder $\mathcal{F}(\phi_w(\mathbf{x}_k))$ twice. Consequently, our proposed augmentation technique do not lose and rather implies valuable information including the label content by taking modification on the latent space.

4 Experimental Settings

4.1 Dataset

We use PhysioNet Sleep-EDF Expanded Database (SleepEDF) and ISRUC [Khalighi *et al.*, 2016] to evaluate the performance of our model. SleepEDF contains 197 whole-night PSG recordings provided by Physionet [Goldberger *et al.*, 2000]. 153 subjects are obtained in a study of age effects on healthy Caucasians (SC), and 44 subjects are obtained in a study of temazepam effects (ST). We use the date from healthy 153 subjects. Each 30 second epoch is manually scored by sleep experts to eight classes (WAKE, REM, 1, 2, 3, 4, MOVEMENT, UNKNOWN) based on the R&K manual. ISRUC is comprised of three subgroups and we select 10 healthy subjects among them [Khalighi *et al.*, 2016]. ISRUC follows the AASM standard which consists of five sleep stages (WAKE, N1, N2, N3, REM). We integrate two datasets to the same AASM manual, so we merge N3 and N4 to N3 and remove the MOVEMENT and UNKNOWN classes for SleepEDF. Without any feature extraction such as converting it into time-frequency images, we put the raw Fpz-Cz EEG and horizontal EOG signals to our models.

4.2 Evaluation Protocols

For our representation learning performed in a fully unsupervised manner, we used 80% of the 153 patients data from SleepEDF where the data labels were completely masked. After representation learning, we transfer the knowledge to the task of sleep staging via fine-tuning the latent predictor $\mathcal{K}(\cdot)$ using the remaining 20% patient data from SleepEDF as well as ISRUC. For classification, we consider two protocols: a limited label setting and a full label setting. For the first setting, we fine-tune the latent predictor where only some labels are used, i.e., only 10% or 20% of the training data are used, and the remaining 80% or 90% are used as test sets for measuring accuracy. For the second setting, a sufficient amount of labeled data is used: 90% is used as training set and 10% as test set. We compute the classification accuracy and F1 score for all the experiments.

4.3 Baselines

We compare our **SSLAPP** with four methods based on fully supervised learning under the full label setting and nine SSL-based methods under the limited label setting.

- The supervised methods include **DeepSleepNet** [Supratak *et al*., 2017] and **AttnSleep** [Eldele *et al*., 2021a] that employ deep neural networks and Random Forest (**RF**) and Support Vector Machine (**SVM**) [Hearst *et al*., 1998] models which are traditional machine learning models.
- The SSL-based methods include **SleepDPC** [Xiao *et al*., 2021], **CoSleep** [Ye *et al*., 2021], Time-Series representation learning via Temporal and Contextual Contrasting (**TS-TCC**) [Eldele *et al*., 2021b], Relative Positioning (**RP**) [Banville *et al*., 2021], Temporal Shuffling (**TS**) [Banville *et al*., 2021], Momentum Contrast (**MoCo**) [He *et al*., 2020], Contrastive Predictive Coding (**CPC**) [Oord *et al.*, 2018], Dense Predictive Coding (**DPC**) [Han *et al.*, 2019] and Triplet Loss [Franceschi *et al.*, 2019].

4.4 Implementation Details

Representation learning was applied in batch size of 128 and 100 epochs. The Adam optimizer is used with learning rate of 0.0002, $\beta_1 = 0.5$, $\beta_2 = 0.999$. The fine-tuning model also adopts the Adam optimizer, where the learning rate is 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.98$ with batch size of 64, and 10 epochs for training. We report the total training time of each representation and finetuning stage in Appendix C.

5 Results and Analysis

This section reports the experimental results and discusses them in detail. In the first experiment, we compare the sleep staging accuracy of our proposed SSLAPP and other existing methods in a limited label setting and in a full label setting. The experimental results are summarized in Table 1 and Table 2 (see Appendix B for per-class results in various label settings). Next, to confirm the superiority of the proposed Attention-based latent signal augmentation, the results were compared with the results of several heuristic augmentation methods used in existing papers. An ablation study was then conducted to confirm the contribution of each component of SSLAPP. Finally, we analyzed influence of the key hyperparameters on performance. The following subsections report each result in detail.

Method	Label Ratio	SleepEDF		ISRUC	
		F1-Score	Accuracy	F1-Score	Accuracy
RP	10%	39.3±2.3	62.4±6.0	17.1±5.7	21.7±7.5
TS	10%	38.4±1.9	62.5±5.2	12.7±2.1	21.2±5.4
MoCo	10%	24.8±4.1	40.9±6.3	14.7±3.9	27.1±17.2
CPC	10%	46.9±6.2	65.6±5.4	38.6±3.4	51.8±4.7
DPC	10%	40.1±10.6	59.8±11.1	32.1±3.2	43.9±3.3
Triplet Loss	10%	37.9±3.4	59.4±4.5	39.8±3.5	49.6±3.1
SleepDPC	10%	64.0±1.5	70.1±0.8	48.9±1.8	53.6±1.5
CoSleep	10%	55.8±3.0	71.6±4.3	50.1±5.6	57.9±5.1
SSLAPP	10%	**72.0±0.4**	**77.2±0.3**	**72.5±0.3**	**74.4±0.2**
TS-TCC	20%	73.5±0.7	**83.0±0.7**	-	-
SSLAPP	20%	**73.9±0.6**	78.9±0.2	**74.1±0.3**	**76.1±0.1**

Table 1: Sleep staging performance of our method and others on SleepEDF and ISRUC datasets with limited labeled data. $'-'$ means that the corresponding value is not provided in the paper of the corresponding baseline.

Method	SleepEDF		ISRUC	
	F1-Score	Accuracy	F1-Score	Accuracy
SVM	57.8	71.2	72.1	73.3
RF	62.4	72.7	70.8	72.9
DeepSleepNet	75.3	78.5	-	-
AttnSleep	75.1	81.3	-	-
SSLAPP (Full label)	**78.0**	**82.0**	**77.0**	**79.9**

Table 2: Comparison between our method and other supervised methods on SleepEDF and ISRUC.

5.1 Comparison Under the Limited Label Setting

Table 1 compares the sleep staging accuracy under the limited label setting with that of the nine SSL-based methods. We report the mean and standard deviation of accuracy and F1 score obtained from the experiments repeated for 5 times with random seed. Most of the baselines reported their results obtained from the experiments using a 10% label in their papers, and only TS-TTC reported the results with 20% label. Therefore, we reported the results using 20% to compare with TS-TTC, and reported the result using 10% label to compare with the other baselines.

We observe that SSLAPP exhibits substantially improved accuracy than the baselines. Especially, on 10% labels setting, SSLAPP greatly outperforms the state-of-the-art baselines, especially CoSleep[Ye *et al*., 2021] on both datasets which has been the state-of-the-art in sleep staging under the limited label setting. Although TS-TCC performs higher accuracy on SleepEDF, we believe that the F1-score is a more important evaluation metric for sleep staging since the data is highly imbalanced in nature. Furthermore, the performance improvement of SSLAPP is more noticeable in the ISRUC dataset, which was not used in the unsupervised representation learning phase. This result shows the superiority of our generalization performance which is not restricted to the trained data: SSLAPP has an ability to achieve reasonable performance on

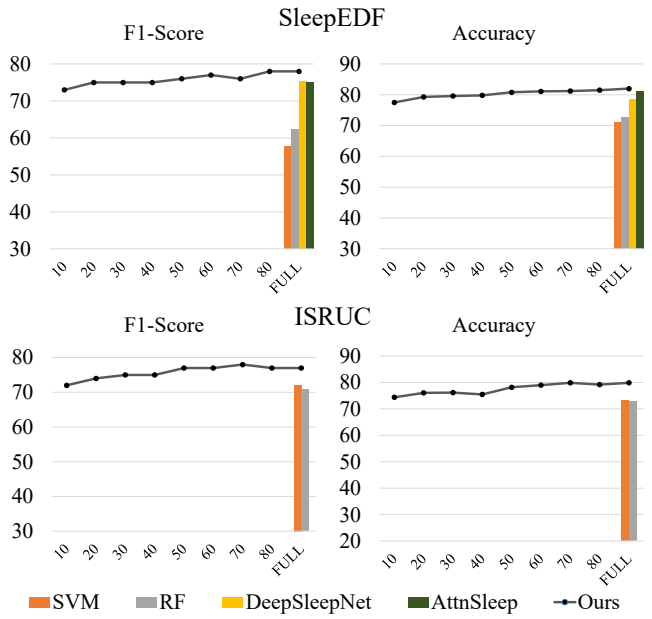


Figure 3: Performance of SSLAPP w.r.t. the different label ratio (line graph), compared with the results of the supervised models (bar).

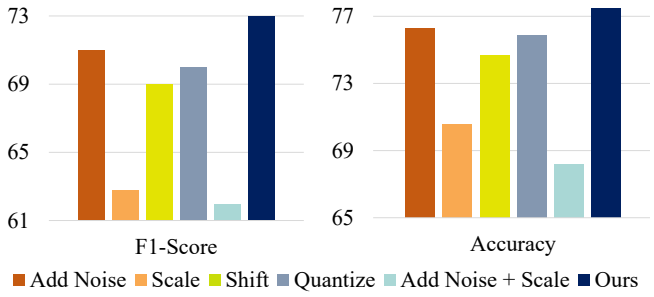


Figure 4: Results accomplished by each augmentation technique.

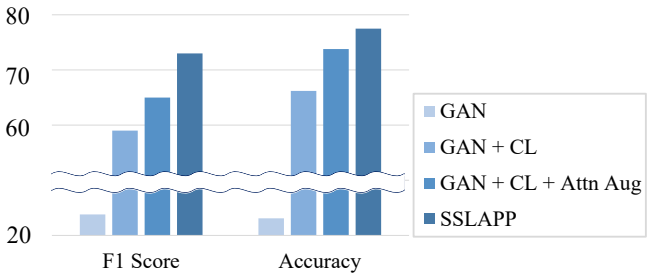


Figure 5: Contribution of each component on the accuracy.

different datasets where labeled data is sparse.

We believe the reasons for the superiority of our method are as follows: SSLAPP focuses on representation learning with high-quality positive pairs so that it employs the attention mechanism to augment such positive pairs, rather than depending on negative pairs. In contrast, existing SSL-based methods rely on the handcrafted positive pairs that modify the input signals directly, which might degrade the performance. It also would cause even more performance degradation on transferring knowledge to models that will be trained other datasets not used for representation training.

5.2 Comparison Under the Full Label Setting

Table 2 compares the results of the supervised baselines and our model in the full label situation. SSLAPP shows on par or better performance than the supervised methods. SSLAPP surpasses overall F1-score from 75.1 to 78.0 on SleepEDF. Figure 3 represents the results of our SSLAPP along increasing labeled data from 10% to 90%. Our model shows competitive results with the supervised methods even in 10%, and the performance consequently outperforms the baselines when the label is fully used.

5.3 Effectiveness of Attention-based Augmentation

In this section, we analyze the behavior of SSLAPP depending on its augmentation technique (i.e., how the positive pair is created). Appropriate transformation without losing important signal information and preserving label consistency is essential. In the image domain, various augmentations such as jittering, scaling, rotating, warping have been widely used. These methods are suitable in image recognition since those minor changes do not alter the labels. However, it is likely that these transformations preserve the label in the signal domain [Um *et al.*, 2017], which motivates us to develop a novel augmentation method for sleep staging. We evaluate the effectiveness of our augmentation method via comparing with other techniques that can be used for sleep staging as follows:

- **Adding noise**: It adds Gaussian random noise.
- **Scaling**: Scaling is applied to the raw signal.
- **Adding noise + Scaling**: It applies the two methods above.
- **Quantization**: It maps continuous values to a set of discrete values of stated level sets.
- **Reflection**: It reflect the front 15 seconds or back 15 seconds within an epoch.

Figure 4 shows the results depending on the augmentation applied while all the other aspects are the same. The fluctuating performance shows the necessity of adequate positive pair for effective representation, and the result proves the compatibility of our proposed idea of latent signal augmentation using the attention mechanism.

5.4 Ablation Study

In order to show the effectiveness of the strategies taken by SSLAPP, we design variant models for comparison.

- **GAN**: GAN with an auxiliary network to learn disentangled latent representation without the contrastive learning term.
- **GAN + Contrastive learning**: Contrastive learning term is applied to learn class identity. It also employs a basic augmentation, '*add noise*', to create the positive pairs.
- **GAN + Contrastive learning + Attention-based augmentation**: Attention-based augmentation is additionally used instead of simply adding noise.

- **GAN + Attention-based augmentation + Adversarial and pairwise representation learning (Ours)**: It stands for our proposed SSLAPP.

Figure 5 shows the results. We observe that adding the contrastive learning term forces the model to learn latent representation of each class identity, which enhances the accuracy. The augmentation with attention leads to better performance owing to the well-augmented pairs for sleep signal. Finally, instead of contrastive learning, our proposed learning framework composed of adversarial and pairwise representation learning turns out to perform the best by eliminating the false negative problem.

5.5 Impact of Hyper-Parameters

Finally, this subsection investigates the impact of the following hyper-parameters: mask ratio r used for our attention, the pairwise loss weight λ_{pos}, and the two weights on the global and local loss terms, λ_g and λ_l, respectively. All the tables shown in this section are the experimental results on SleepEDF.

Mask Ratio

Table 3 shows the results of different mask ratios (0.05, 0.1, 0.2, and 'Random') applied on augmentation. 'Random' is a random number sampled from a uniform distribution between 0.05 and 0.2. (i.e. $r \sim \mathbf{U}(0.05, 0.2)$). The best ratio is observed to be 0.1, which provides a good balance between transformation and preserving information. The positive pair constructed with mask ratio 0.05 is too similar, which is not sufficient to capture the distinctive representations from sleep signals. This setting leads to a quick convergence while not trying to push the positive pairs to be close to each other since they may already be quite similar to each other. High masked ratio 0.2 results seems to be slightly large for constructing a positive pair, which leads to some missing representations.

r	0.05	0.1	0.2	Random
10% label	76.0	**77.5**	75.2	65.7
20% label	77.6	**79.3**	78.0	67.7

Table 3: Sensitivity test on the masking ratio.

Weight on the Pairwise Loss

Next, we report the performance results according to varying λ_{pos}, which controls the contribution of the pairwise loss $\mathcal{L}_{\text{pos}}$ used in the training objective of SSLAPP. Table 4 reports the results where λ_{pos} is set to 1, 5, and 10 in the 10% label setting. The experimental results were best when λ_{pos} is 5, and all the experiments in the paper is conducted with this setting.

λ_{pos}	F1-Score	Accuracy
1	72.0	77.2
5	**73.0**	**77.5**
10	70.0	76.0

Table 4: Results according to the pairwise loss weight.

Local/Global Loss Configuration

Table 5 shows the impact of global / local loss weight (λ_g / λ_l) on the accuracy and F1 score in 20% label setting. We observed that taking both the global and local loss into account helps improve the model performance. We can also see that placing less value on the local loss (i.e., $\lambda_l = 0.0$ or 0.1) results in a degraded performance. We achieved the best performance when $\lambda_l = 0.25$ and 0.5. However, as it dominates λ_g, the performance degrades.

λ_g	λ_l	F1 score	Accuracy
1.0	0.0	73.1	78.7
0.9	0.1	72.4	78.4
0.75	0.25	75.0	79.3
0.5	**0.5**	**75.0**	**79.3**
0.25	0.75	74.0	79.0

Table 5: Results according to the λ_g and λ_l weights.

The Duration of a Segment and the Number of Segments

Finally, Table 5 shows the results depending on different α (duration of a segment) and j (# of segments) values. Since $\alpha \times j = 30$, we tested three candidates for (α, j) as follows: (3, 10), (5, 6) and (10, 3). Even though we observed that their impact was not very significant, we finally chose ($\alpha = 5$, $j = 6$) since it provides the best performance while using less model parameters (note that the larger j, the more network parameters).

α (sec)	j (#)	F1 score	Accuracy
3 sec	10	75.0	79.0
5 sec	**6**	**75.0**	**79.3**
10 sec	3	74.0	79.3

Table 6: Results depending on the α and j values.

6 Conclusion

In this paper, we propose SSLAPP that overcomes the limitations of existing SSL methods for sleep staging. By using the high-quality positive pairs and leveraging an attention-based latent signal augmentation, it captures valuable representations for sleep staging unlike other existing SSL methods that could lose meaningful information due to direct modification of signals. It also successfully avoids the false negative problem. Experimental results demonstrate the superiority of our method in sleep stage classification under both the limited label and full label settings. Our method also demonstrates its strength in its transferable generalization to other datasets. The effectiveness of each idea is also confirmed through our ablation studies.

A False Negative Probability in the Sleep Staging Task

False negative problem intrinsically occurs in contrastive learning when two data points in the same class (actual positive) are assigned as the negative pair. The probability of False

Negative (F.N) pair $\mathbb{E}(F.N)$ to appear in a specific dataset is:

$$\mathbb{E}(F.N) = \frac{N \times \Sigma_{i=1}^{k} p_i^2}{N} = \sum_{i=1}^{k} p_i^2,$$

where N denotes for batch size and p_i stands for each i^{th} class's probability. Dataset consisted of few classes and with an imbalanced class distribution tends to have higher probability of false negative assigned as negative pairs.

Class	Number	Proportion
Wake	65,951	34%
N1	21,522	11%
N2	69,132	35%
N3	13,039	7%
REM	28,535	13%

Table 7: Class proportion of SleepEDF.

In the case of SleepEDF, the false negative probability is 27.2% with its class label ratio reported in Table 7. The high false negative probability is the main reason why we do not directly apply the original contrastive framework to our work.

B Per-Class Results

Table 8 and Table 9 reports per-class results of SSLAPP on SleepEDF and ISRUC. Unfortunately, previous self-supervised methods do not report their F1-scores per class, so we compare with supervised methods. The performance of DeepSleepNet and AttnSleepNet on ISRUC is not provided on their papers, thus denoted as '-'.

In the context of SleepEDF, although supervised-based models may achieve better performance in some classes such as Wake 92.0 [Eldele *et al*., 2021a], and REM 79.0 [Supratak *et al*., 2017], their performances are usually due to largely annotated datasets. Therefore, these methods show difficulty detecting features from minority classes such as N1, N3 dominated by major classes. Compared to supervised baselines, SSLAPP achieves 0.48 for N1, 0.86 for N2, and 0.87 for N3, which surpasses all the supervised baselines.

Model	Overall Result		F1-score for each class				
	F1-Macro	Accuracy	W	N1	N2	N3	REM
SVM	0.58	0.712	0.80	0.14	0.80	0.57	0.59
RF	0.62	0.727	0.82	0.23	0.81	0.66	0.61
DeepSleepNet	0.75	0.785	0.91	0.47	0.81	0.69	**0.79**
AttnSleepNet	0.75	0.813	**0.92**	0.42	0.85	0.82	0.74
SSLAPP 10%	0.73	0.775	0.89	0.43	0.83	0.78	0.70
20%	0.75	0.793	0.90	0.47	0.84	0.79	0.74
30%	0.75	0.796	0.89	0.46	0.85	0.79	0.76
40%	0.75	0.798	0.89	0.46	0.85	0.78	0.76
50%	0.76	0.808	0.91	0.47	0.85	0.80	0.77
60%	0.77	0.811	0.90	**0.50**	0.85	0.82	0.77
70%	0.76	0.812	0.91	0.47	0.85	0.84	0.78
80%	0.78	0.815	0.92	0.51	0.85	0.84	0.78
90%	**0.78**	**0.820**	0.91	0.48	**0.86**	**0.87**	0.78

Table 8: Overall and Per-class result on SleepEDF.

Model	Overall Result		F1-score for each class				
	F1-Macro	Accuracy	W	N1	N2	N3	REM
SVM	0.72	0.733	0.87	0.52	0.70	0.79	0.73
RF	0.71	0.729	0.86	0.47	0.70	0.81	0.70
DeepSleepNet	-	-	-	-	-	-	-
AttnSleepNet	-	-	-	-	-	-	-
SSLAPP 10%	0.72	0.744	0.85	0.42	0.74	0.85	0.75
20%	0.74	0.761	0.87	0.49	0.74	0.85	0.78
30%	0.75	0.762	0.85	0.52	0.76	0.86	0.76
40%	0.75	0.755	0.88	0.53	0.70	0.83	0.78
50%	0.77	0.782	0.87	**0.54**	0.78	0.86	0.78
60%	0.77	0.790	0.88	0.49	0.79	0.87	0.80
70%	0.78	0.799	**0.89**	0.50	0.79	0.88	0.82
80%	0.77	0.792	0.86	0.53	0.78	**0.89**	0.81
90%	**0.77**	**0.799**	0.86	0.49	**0.81**	0.88	**0.82**

Table 9: Overall and Per-class result on ISRUC.

Interestingly, SSLAPP performs better in N2 than supervised learning methods which is the most dominant class on the sleepEDF dataset. These results support our model's performance is owing to the well-learned representations from the data itself and not from data annotation.

C Training Time Analysis

We report the model training time for each step constituting our SSLAPP. All the running time is measured by using a single machine equipped with an i9 11900K Intel CPU, 128GB RAM, and NVIDIA GeForce RTX3090 GPU.

	Training time
Representation learning with SleepEDF	12.6 hrs
Finetuning on SleepEDF	64.1 sec
Finetuning on ISRUC	32.0 sec

Table 10: Training time for each step.

Acknowledgments

This work was partly supported by (1) Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2020-0-01373,Artificial Intelligence Graduate School Program(Hanyang University)) and (2) the Bio & Medical Technology Development Program of the National Research Foundation (NRF) funded by the Korean government (MSIT) (No. NRF-2021M3E5D2A01021156).

References

[Banville *et al*., 2021] Hubert Banville *et al*. Uncovering the structure of clinical eeg signals with self-supervised learning. *Journal of Neural Engineering*, 18(4):046020, 2021.

[Berry *et al*., 2012] Richard B Berry *et al*. The aasm manual for the scoring of sleep and associated events. *Rules, Terminology and Technical Specifications, Darien, Illinois, American Academy of Sleep Medicine*, 176:2012, 2012.

[Chen and He, 2021] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[Eldele *et al*., 2021a] Emadeldeen Eldele *et al*. An attention-based deep learning approach for sleep stage classification with single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:809–818, 2021.

[Eldele *et al*., 2021b] Emadeldeen Eldele *et al*. Time-series representation learning via temporal and contextual contrasting. *arXiv:2106.14112*, 2021.

[Franceschi *et al.*, 2019] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *arXiv:1901.10738*, 2019.

[Gao *et al.*, 2021] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv:2104.08821*, 2021.

[Goldberger *et al*., 2000] Ary L Goldberger *et al*. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[Han *et al.*, 2019] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[He *et al*., 2020] Kaiming He *et al*. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[Hearst *et al*., 1998] Marti A. Hearst *et al*. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[Hedner *et al*., 2011] Jan Hedner *et al*. Sleep staging based on autonomic signals: a multi-center validation study. *Journal of clinical sleep medicine*, 2011.

[Huang *et al*., 2014] Chih-Sheng Huang *et al*. Knowledge-based identification of sleep stages based on two forehead electroencephalogram channels. *Frontiers in neuroscience*, 8:263, 2014.

[Jia *et al*., 2021] Ziyu Jia *et al*. Salientsleepnet: Multimodal salient wave detection network for sleep staging. *arXiv:2105.13864*, 2021.

[Jiang *et al.*, 2019] Dihong Jiang, Ya-nan Lu, MA Yu, and WANG Yuanyuan. Robust sleep stage classification with single-channel eeg signals using multimodal decomposition and hmm-based refinement. *Expert Systems with Applications*, 121:188–203, 2019.

[Jiang *et al*., 2021] Xue Jiang *et al*. Self-supervised contrastive learning for eeg-based sleep staging. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[Khalighi *et al*., 2016] Sirvan Khalighi *et al*. Isruc-sleep: A comprehensive public dataset for sleep researchers. *Computer methods and programs in biomedicine*, 124:180–192, 2016.

[Lajnef *et al*., 2015] Tarek Lajnef *et al*. Learning machines and sleeping brains: automatic sleep stage classification using decision-tree multi-class support vector machines. *Journal of neuroscience methods*, 250:94–105, 2015.

[Malafeev *et al.*, 2018] Alexander Malafeev, Dmitry Laptev, Stefan Bauer, Ximena Omlin, Aleksandra Wierzbicka, Adam Wichniak, Wojciech Jernajczyk, Robert Riener, Joachim Buhmann, and Peter Achermann. Automatic human sleep stage scoring using deep neural networks. *Frontiers in neuroscience*, page 781, 2018.

[Mousavi *et al.*, 2019] 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, 2019.

[Ojha *et al*., 2019] Utkarsh Ojha *et al*. Elastic-infogan: Unsupervised disentangled representation learning in class-imbalanced data. *arXiv:1910.01112*, 2019.

[Oord *et al.*, 2018] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

[Perez-Pozuelo *et al*., 2020] Ignaci Perez-Pozuelo *et al*. The future of sleep health: a data-driven revolution in sleep science and medicine. *NPJ digital medicine*, 3(1):1–15, 2020.

[Phan *et al*., 2019] Huy Phan *et al*. Seqsleepnet: end-to-end hierarchical recurrent neural network for sequence-to-sequence automatic sleep staging. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(3):400–410, 2019.

[Sateia, 2014] Michael J Sateia. International classification of sleep disorders. *Chest*, 146(5):1387–1394, 2014.

[Supratak *et al*., 2017] Akara Supratak *et al*. Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(11):1998–2008, 2017.

[Um *et al*., 2017] Terry T Um *et al*. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 216–220, 2017.

[Wolpert, 1969] Edward A Wolpert. A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects. *Archives of General Psychiatry*, 20(2):246–247, 1969.

[Xiao *et al*., 2021] Qinfeng Xiao *et al*. Self-supervised learning for sleep stage classification with predictive and discriminative contrastive coding. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1290–1294. IEEE, 2021.

[Ye *et al*., 2021] Jianan Ye *et al*. Cosleep: A multi-view representation learning framework for self-supervised learning of sleep stage classification. *IEEE Signal Processing Letters*, 2021.