Self-Supervised Learning for Electroencephalograms

Théo Moutakanni
École CentraleSupélec
theo.moutakanni@supelec.fr

Clément Bonnet
École CentraleSupélec
clement.bonnet16@gmail.com

Abstract

Supervised learning leads to increased performance partly thanks to deep learning. The main problem with supervision is the necessity of having large labeled datasets to achieve high performance. However, in some domains such as health, data is often subject to a lack of labels. It requires time and experienced staff to obtain labels of high quality. New methods with little or no supervision have then emerged, especially in computer vision. In this work, we focus on so-called self-supervised learning methods, specifically contrastive methods, in the context of electroencephalography. We analyze the learned embeddings for sleep stage detection and the impact of pre-training set size.

1 Introduction

Given enough data and labels, supervised learning can solve a given task well. However, good performance may require huge amounts of labels which are often collected manually, resulting in a costly and rather not scalable process. The internet gathers way more unlabeled data than any human-labeled dataset. Therefore, one may find it wasteful not to use these unlabeled data when training algorithms. Unsupervised learning, whose goal is to learn about the data generation process while only using unlabeled data, is rather difficult and not as good as supervised learning on a given final task.

Self-supervised learning (SSL) enables the supervision to come from the data itself. This way, one can make use of the much larger amount of unlabeled data that is generated all the time. The self-supervised task, sometimes called the pretext task, sets the learning objective to be a supervised loss function. This invented task, whose final performance is rather ignored, is expected to learn an intermediate representation that carries the structural meaning of the data generation process, which in turn, is used to learn downstream tasks from few examples. This way, one uses unlabeled data in a supervised way to learn a representation to be provided to the downstream task as an inductive bias. Therefore, self-supervised learning cares about producing good features for other tasks.

It is often not clear how much unlabeled data one needs to pre-train a model using self-supervised learning. As long as domain adaptation issues are tackled, one could use large amounts of unlabeled data coming from a different dataset to train a model on another low-data-regime labeled dataset. In this work, we aim to study the impact of the size of the unlabeled dataset and its impact on learned embeddings in the case of self-supervised learning of electroencephalograms.

Research project at École CentraleSupélec.

2 Background

2.1 Electroencephalograms and Sleep Staging

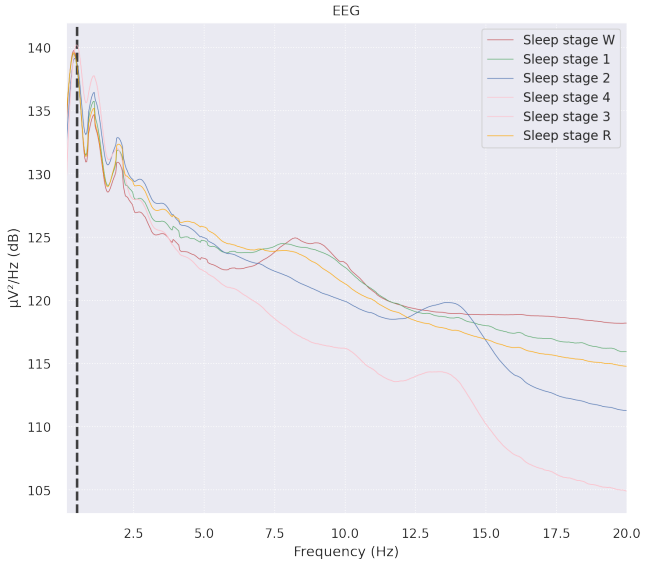


Figure 1: Analysis of the mean power spectral density by sleep stage for our dataset.

Nights are often divided into 5 sleep stages, namely "Wake", stages 1, 2, 3 & 4 (merged), and "Rapid Eye Movement" (REM). Each stage is characterized by its neuronal activity whose frequency is distributed as follows:

- Wake: beta activity, high-frequency $(15 - 60Hz)$, low-amplitude activity $(\approx 30\mu V)$.
- Stage I: theta waves $(4 - 8Hz)$ and increasing amplitude $(50 - 100\mu V)$.
- Stage II: spindles $(10 - 15Hz)$ and oscillations $(50 - 150\mu V)$ during a few seconds.
- Stage III-IV: delta waves 0.5–4 Hz $(100 - 150\mu V)$.
- REM: similar to the EEG activity of individuals who are awake.

Electroencephalography (EEG) is an electrophysiological monitoring method used to record electrical activity in the brain. The main device sets electrodes on the scalp in pre-defined positions to record specific regions of neuronal activity. A set of positions is called a montage, and each experiment may have a different one.

Diagnostic applications generally focus either on event-related potentials or on the spectral content of EEG. The former investigates potential fluctuations time locked to an event, such as 'stimulus onset' or 'button press'. The latter analyses the type of neural oscillations (popularly called "brain waves") that can be observed in EEG signals in the frequency domain.

2.2 Self-Supervised Learning

Self-supervised learning enables models to learn representations of the data and their intrinsic biases by leveraging different methods. One may see at least three ways of doing so:

- Learning a contrastive task,
- Clustering,
- Other methods such as Bootstrap Your Own Latent [1].

The incentive behind self-supervised learning is quite straightforward. Building a large dataset with clean labels is expensive, yet, unlabeled data may be generated for cheap. To make use of this much

larger amount of unlabeled data, one way is to set the learning objectives to get supervision from the data itself.

Self-supervised learning aims at building and learning to excel at a pretext task. Although its final performance is rather ignored, by learning the task, the model has to learn the intrinsic properties of the data. Therefore, training a model on a pretext task may pre-train it to perform at the final task, called downstream task. Hence, the main assumption of self-supervised learning is that the pretext task will make the model learn about the structure of the data generation process that will eventually be useful to learn the downstream task in a more efficient way and with few labels.

In this work, we will focus on a specific method that consists of learning a contrastive task. This means building a dataset of positive and negative pairs and then making the model predict their binary categories. The model starts by guessing randomly (with 50% accuracy) the pair category and then it has to learn about the structure of the data to improve on the prediction. The advantage of such a method is that building the dataset out of unlabeled data is rather cheap. The model can then be used for the downstream task by removing the last layer to reuse the learned embedding.

The positive and negative pairs are formed using a heuristic on the data structure. For example, on images, we know that there is a spatial correlation between objects and features. On EEG, the correlation is temporal, and the further away you are from an EEG window in time, the lesser the correlation.

2.3 Application to Sleep Staging

Our goal is to apply self-supervised learning to sleep staging. Namely, we will pre-train an encoder network using a contrastive task such that it will learn to recognize sleep stages without any labels. To do so, we will need a pretext task that must enforce the network to detect the specific frequencies and waves happening in the brain signal when we are in the different sleep stages.

The hypnogram, or distribution of the sleep stages along time during a night, has only a few discontinuities: we have multiple consecutive 30-s windows of the same sleep stage. Each sleep stage lasts approximately 30 min. Thus, the temporal correlation between 30-s windows is a good proxy to learn to detect sleep stages without labels.

As a positive pair, we could sample two 30-s windows which are close in time, and as a negative, two 30-s windows which are far away in time. The network will have to learn to map correlated windows to the same embedding vector, and uncorrelated windows to different embedding vectors tobe able to detect if each pair is positive or negative.

We describe precisely how we did it in 4.3.

3 Related Work

3.1 Self-Supervised Learning

Many ideas have been proposed so far for self-supervised representation learning of images, videos, and time series. Thanks to useful resources including Lilian Weng's blog [2], we provide below an overview of these methods in each of these machine learning fields.

3.1.1 Images

As we mentioned above, the internet is overflowing with unlabeled images. A common workflow for self-supervised representation learning of images is to train a model on one or several pretext tasks, and then use an intermediate layer of the model as a basis for a supervised learning architecture eventually trained on ImageNet [3] or any labeled dataset.

Some inductive biases used in SSL for images are that the spatial context of objects should be understood by a model. One way to achieve this is to frame a pretext task to recover the relative positions of random patches extracted from the same image. In Doersch et al. [4], the authors described the following SSL procedure. A first patch is randomly selected from an image. Then, a second patch is randomly chosen among its 8 neighbors in a 3x3 grid. The model is trained to predict

at which of the 8 neighboring areas the second patch was located. This way, the model is forced to understand contextual information of objects in images.

3.1.2 Videos

When dealing with videos as sequences of frames, a common workflow for self-supervised learning of images is to train a model on a pretext task with unlabeled videos and use one of its intermediate feature layers to train a simple model on a final task such as object tracking or video classification. Here, the idea is to introduce inductive biases such as to take physical principles into account. For instance, nearby frames must be correlated in a way that does not allow objects to move too fast, or that gravity tends to make things fall.

One way to use SSL to learn a good representation of video frames is to expect a model to be able to correctly order frames of the same video. Naturally, albeit being unlabeled, video frames are positioned in chronological order. Therefore, one may learn a visual representation from the raw spatiotemporal signals in videos. In Misra et al. [5], the authors formulate the representation learning task as an unsupervised sequential verification task in which a model has to determine whether a sequence of frames from the same video is in the correct temporal order. This way, the model is forced to reason and track small motions of objects across frames to succeed in such a pretext task.

3.1.3 Time Series

Self-supervised learning may be used in the context of multivariate times series. A successful approach used in Banville et al. [6] is to predict whether time windows are sampled from the same temporal context or not. From electroencephaloghy signals, the authors learn a representation using a temporal contrastive task that they describe as follows. From the multivariate times series, they produce labels by sampling pairs of time windows from the same long time series. The main assumption used by the authors is that an appropriate representation of the data should evolve slowly over time suggesting that time windows close in time should share the same label. They come upon two self-supervised tasks based on this assumption.

The first task used in [6] and [7] is called Relative Positioning. The model is trained to determine whether two sampled time windows are close or not. The second idea of the authors is to train the model with another task called Temporal Shuffling. This time, the task is to predict if three sampled windows are in the correct chronological order or if they have been shuffled. This way, the model learns a good representation of the multivariate time series.

3.2 Sleep Stage Detection

Sleep stages are usually determined by a sleep expert by analyzing polysomnography records (PSG). The expert analyzes 30-seconds windows and classifies them using a normalized protocol to get the sleep stages. Each window gets a label between N1, N2, N3, and REM, the 4 sleep stages. This process is time expensive but is necessary as a first step to diagnose sleep disorders.

There exist a lot of published processes to automatize this procedure. Newer ones perform really well on the datasets they are trained with [8] [9] [10] [11]. They all use Deep Learning, which needs a lot of training labeled data. Furthermore, as shown by Guillot et al. [11], they are not robust enough to unseen PSG montage or new cohort of patients. There is a multitude of PSG devices available on the market and each hospital may use several ones. Methods trained on datasets with unseen diseases also have bad results on patients with multiple comorbidities.

3.3 Domain Adaptation

The issue automatic sleep staging algorithms have to overcome to be able to be used in real life is called Domain Adaptation. The goal is to reduce the domain bias between datasets such that an algorithm trained on one fixed dataset can be applied to new ones from hospitals. There are three major domain biases that can be found. Different EEG montage or devices (hardware), different patient cohorts (with or without comorbidities, age, sex, job category, etc.), and patient-specific differences (position, brain connectivity, sleep patterns, etc.). While some forms can be important (patient-specific bias), domain bias should be reduced and sleep staging algorithms should be more robust.

Several supervised algorithm exists, some specific to sleep staging [12], and others more general to Deep Learning [13].

A more straightforward method is to first train the network on a big fixed dataset and then fine-tune it on a labeled subset of the new unseen dataset which is an order of magnitude smaller than the number of necessary samples if the network was trained from scratch. This method is effective because the first layers of the network act as feature extractors and can be shared among tasks and datasets [14] [15]. By doing so, we can increase the sleep staging results on newer datasets without access to a lot of labeled data.

According to Guillot et al. [11], direct transfer (no fine-tuning) already leads to good results with the RobustSleepNet architecture (90% to more than 100% of the original F1 if the network was trained on the unseen dataset directly). With only 10 to 40 labeled records, the fine-tuned pre-trained network surpasses a network trained directly on the datasets used for evaluation.

4 Experiments

We tried to reproduce the work of Banville et al., 2020 [7] in which we analyzed the SSL-learned embeddings and we studied the impact of the pre-training size. In this section, we first expose the model architecture, then we present the Sleep Physionet dataset. We later describe the pretext task used for the pre-training phase, and performances on the downstream task. We finally analyze the learned embeddings and the influence of pre-training size.

4.1 Model Architecture

We used 2 models, one for the pre-training task and one for the downstream task. Both of them use the same, shared encoder network as a feature extractor.

The goal of the pre-training task is to train the encoder. Then the downstream task uses the same frozen encoder to fine-tune a classifier on top of it on sleep staging.

The Deep Learning architecture for the encoder is based on the StagerNet from Banville et al. [7], which itself is an adaptation of the architecture from Chambon et al. [16].

The StagerNet is a 3-layer convolutional neural network, with ReLU activations and Batchnorm. The final embedding dimension is D=128.

For the contrastive task, the encoder extract features from the two 30-s windows of the pair. Then we plugged one dense layer on the L1 difference between the two windows' features to classify if the pair is negative or positive.

Then, for the downstream task we used a 1 hidden layer MLP plugged on the frozen features from the encoder. We added dropout directly on the features with p=20% to reduce over-fitting.

We trained during 40 epochs with an early stopping (patience=5 epochs) using the Adam optimizer [17] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a learning rate of 5×10^4, as in [7].

4.2 Dataset

We did experiments using the Sleep Physionet Dataset [18].

The database contains 197 whole-night Polysomnographic sleep recordings, containing EEG (from Fpz-Cz and Pz-Oz electrode locations), EOG, chin EMG, and event markers. The true hypnograms (sleep stages) were manually scored by well-trained technicians according to the Rechtschaffen and Kales manual.

For each patient, 2 nights were recorded, whose first 30 minutes of wake stage were removed. Each recording is divided into 30-s windows of 2 EEG channels.

We omitted 14 patients out of the 83 because we could not access both of the 2 nights recordings.

For the training of our different tasks, we split the dataset by patients to avoid data leakage when analyzing the trained embedding (each % is in the number of patients, not in the number of 30-s windows). We have 3 main sets: $80\% - 10\% - 10\%$.

Here is how we used the first 80% of patients in an unsupervised way:

- 70% for the available patients for the training of the pretext task (pre-training).
- 10% for the pre-training task validation.

Then all of the patients from the 80% split are used to train the sleep stage classifier (fine-tuning, or downstream task) in a supervised manner. The two other 10% splits are used for the validation and the final test of the downstream task. Details can be seen in fig. 2.

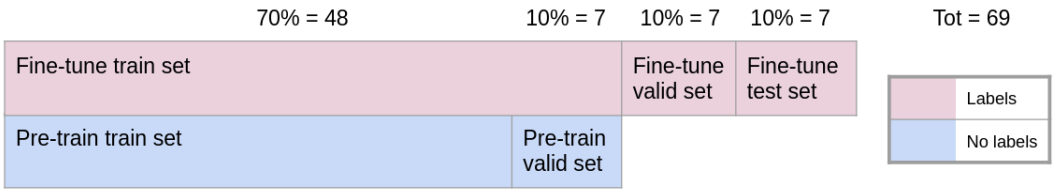
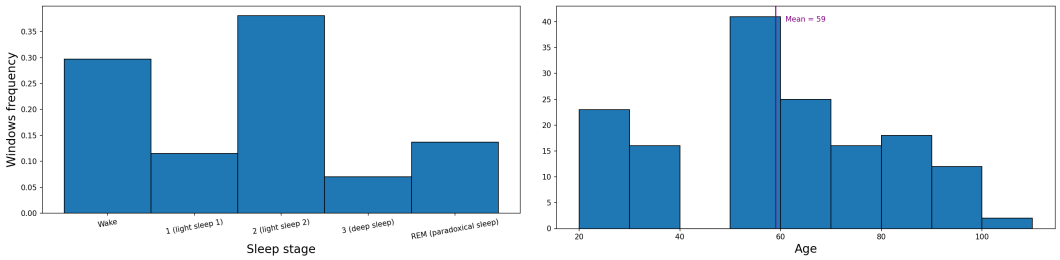


Figure 2: The splitting of the dataset, colors are set according to how we used labels.

The distribution of sleep stages and patient ages can be seen in fig. 3.



(a) Distribution of sleep stages.

(b) Age distribution of patients.

Figure 3: Statistics of the Sleep Physionet dataset.

We only kept the 2 EEG channels, discarding other signals. The EEG signals are sampled at 100 Hz.

As a pre-processing, we transformed the signal in micro-volt and used a 30 HZ low-pass FIR filter. The signal is then divided into windows of size 30s, corresponding to the hypnogram windows, and a Z-score normalization is applied per window on each channel.

4.3 Pretext Task

As for the design of the pretext task to pre-train the model, we chose to reproduce the relative positioning contrastive task from Banville et al., 2020 [7]. The way this task works is described in figure 4. For each window provided to the model, another window is sampled either close or far from the first window. The goal of the task is to classify whether a sampled pair is negative, meaning the windows are far from one another, or positive when they are close.

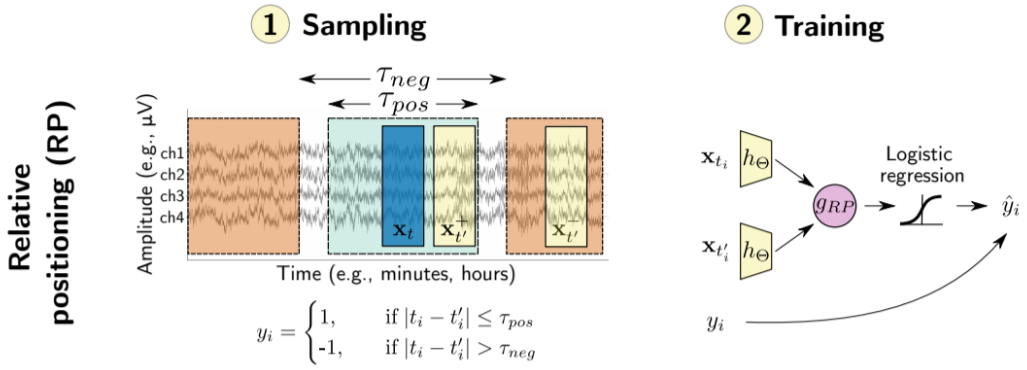


Figure 4: Contrastive task used for pre-training. Source: Banville et al., 2020 [7].

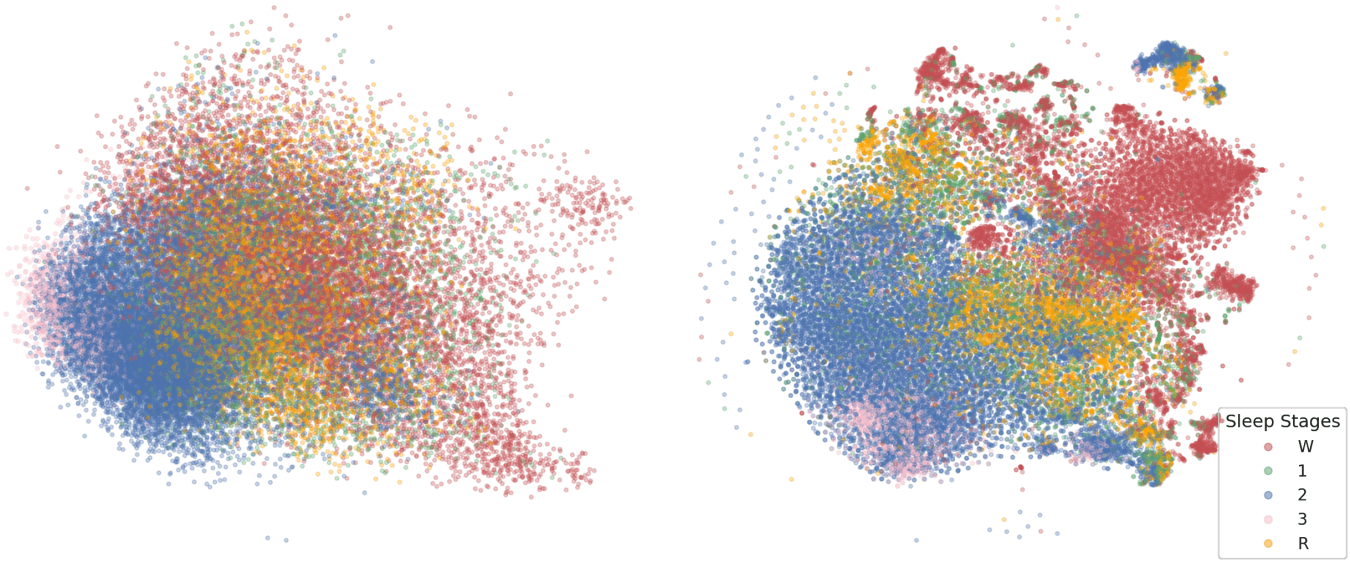
The category of the pair (positive or negative) is sampled uniformly (probability 0.5) and the pair windows is then sampled uniformly considering the positive or negative limits within the same night of the same patient. In our experiments we chose $\tau_{neg} = 100 * 30s$ and $\tau_{pos} = 10 * 30s$. This means positive pairs contain windows that are away from at most 10 windows, i.e. 5 minutes, while negative pairs are away from at least 100 windows, i.e. 50 minutes.

4.4 Proof Of Concept

We pre-train our model using the previously described contrastive task and we compare the embeddings of the pre-trained model with those of the baseline.

4.4.1 Visualization of the Embeddings

We now visualize the embeddings that are then used for sleep stage classification. We decide to plot both Principal Component Analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) to have a more complete qualitative understanding of the different clusters that can appear in the data.

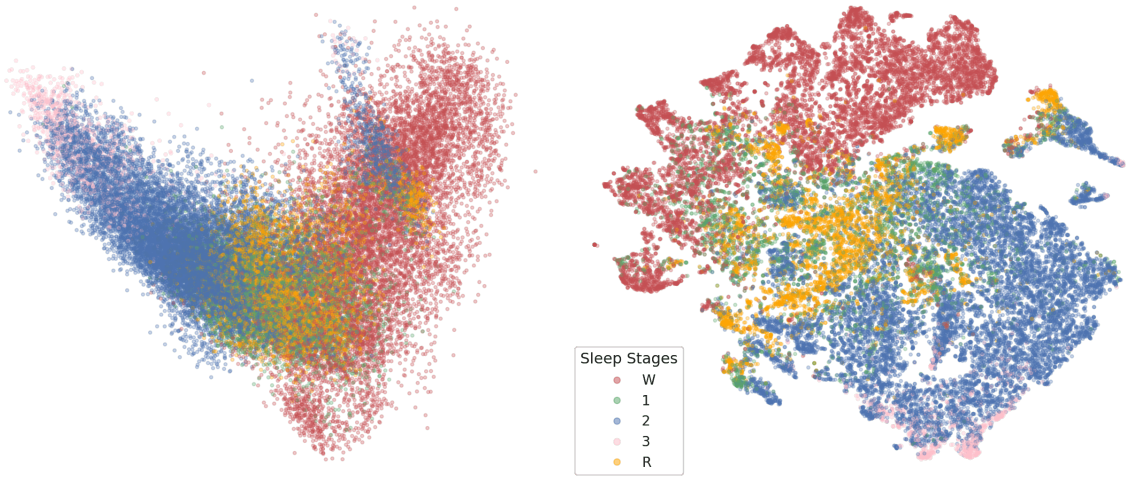


(a) PCA

(b) TSNE

Figure 5: Visualization of the PSD features according to the sleep stages.

As one can see in figure 5, power spectral density (PSD) embeddings allow some clustering among sleep stages, but a classification remains rather hard, explaining lower results of PSD in figure 7a.



(a) PCA (b) TSNE

Figure 6: Visualization of the SSL embeddings according to the sleep stages.

The embeddings learned from the contrastive task are displayed in figure 6. They demonstrate a higher clustering potential than features obtained from PSD or power bands. One observes identifiable clusters in 2D representations.

4.4.2 Benchmark with the Embeddings

In figure 7a, we compare the balanced accuracy of a pre-trained (using SSL) classifier and two other classifiers that either use PSD or power bands, which are aggregations of frequencies. All classifiers are tested using a k-nearest neighbors algorithm with $k = 10$. One may observe that the embeddings learned after SSL pre-training lead to better balanced accuracy on sleep staging than using PSD or power bands.

Another kNN experiment was done in figure 7b to assess the patient ID learned bias in the embeddings. It shows that SSL embeddings are better at learning to classify a patient, demonstrating clusters for patients. The model would thus learn to identify patients and distribute the patient windows within this space.

4.5 Fine Tuning

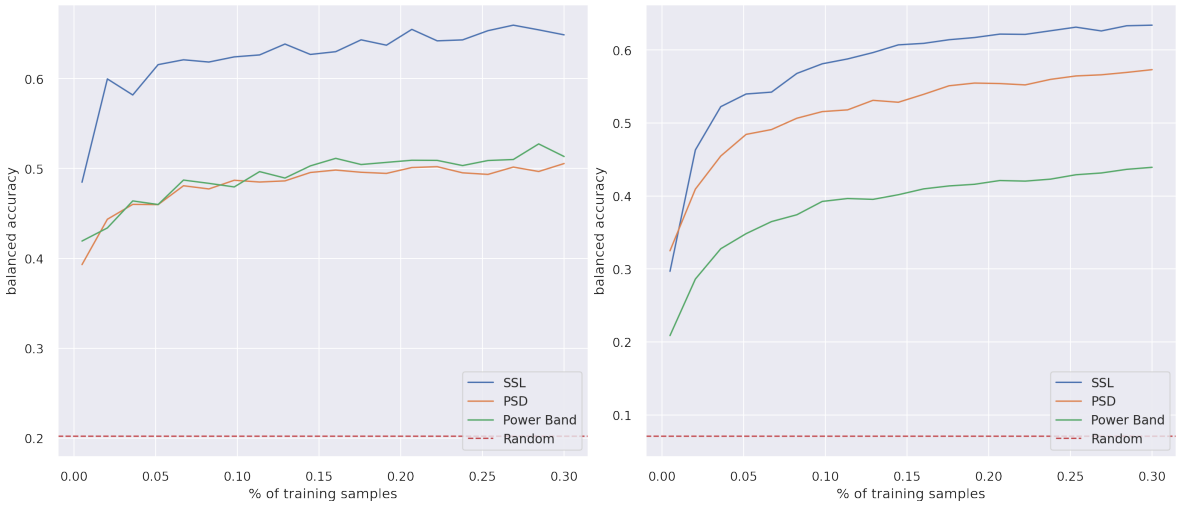
We now compare in Figure 8 performances of two models on the final downstream task: sleep staging. The pre-trained model is shown in blue while the same model without pre-training is displayed in orange. The pre-trained model had access to 70% of the dataset as unlabeled data to learn the pretext task. On the horizontal axis, one finds the number of patients the final task is trained on, meaning the further on the right, the more labels are given to the models.

We recall that the pre-trained encoder is frozen and that for the blue line, only the 2-layer MLP classifier is trained.

We identify two regimes, namely low-data and high-data regimes. With less than 30 labeled patients, the models may seem to be in a low-data regime where pre-training the model leads to a higher accuracy on the final task. This advantage over supervised training from scratch vanishes with more than 30 patients, where both models' learning curves overlay one another.

4.6 Analysis of Learned Representations

We analyze in Figure 9 the distribution of patient age in the representations learned by the embeddings. As we may later see in Figure 10, the embeddings do not seem to cluster ages but rather patients, which happens to make some clusters emerge in the age distribution too.



(a) Sleep stages (b) Patient ID

Figure 7: Benchmark of 3 unsupervised embedding methods using KNN trained on a fraction of the test data to classify sleep stages or patient ID.

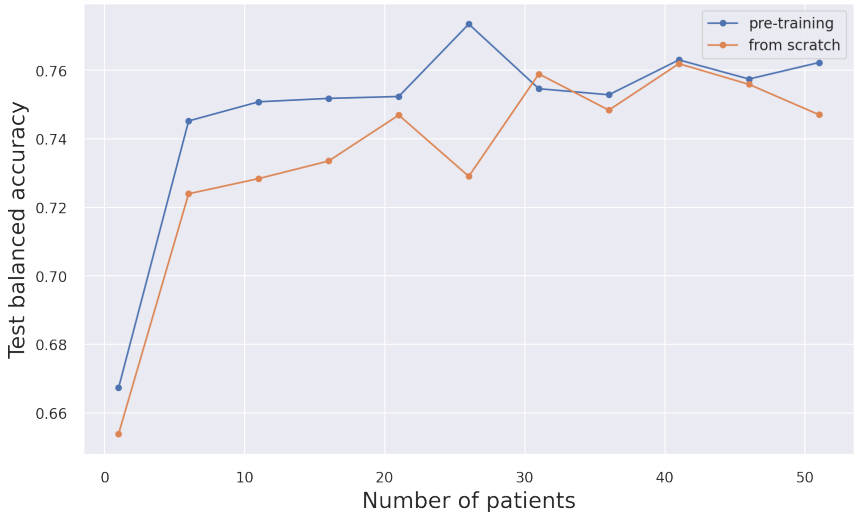


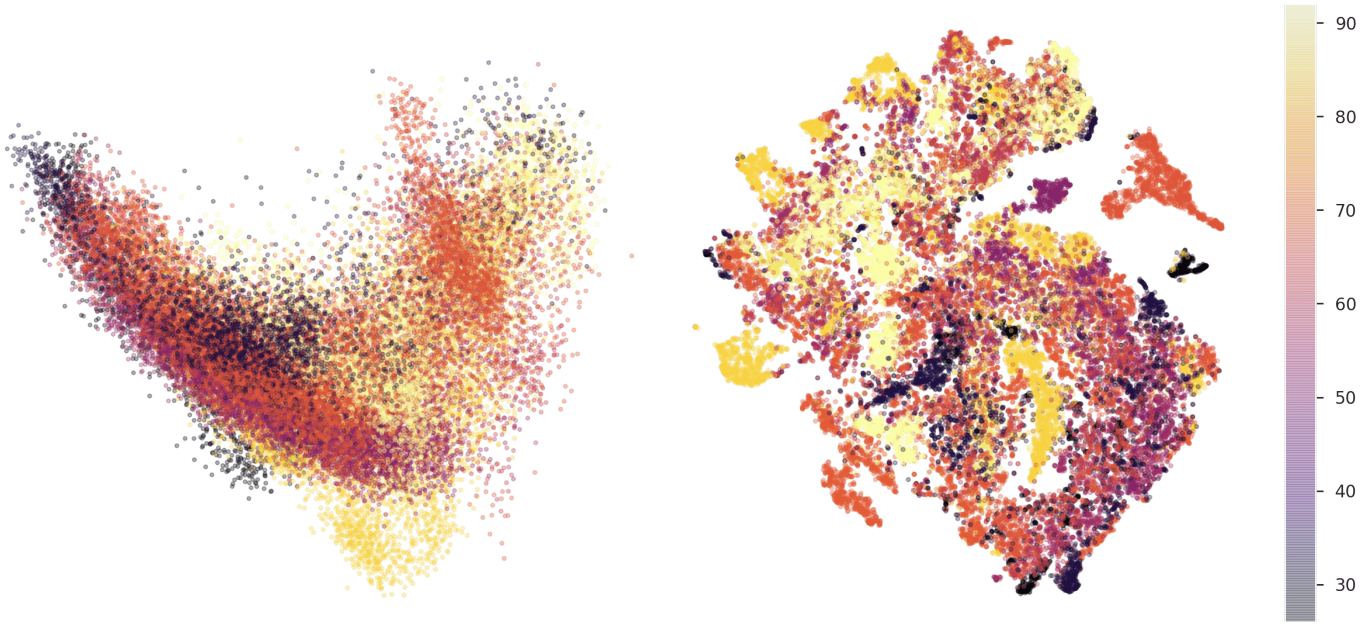
Figure 8: Fine-tuning on the SSL embeddings for sleep staging.

However, if not ages, the model seems to cluster patients. As one can see in Figure 10, the pre-training task seems to be very biased toward patient-specific features. We can see a lot of clusters of the same color made from windows coming from the same patients. This means the model learns patient-specific details/modeling that may be irrelevant to understand sleep stages.

It seems that learning from whom comes each window is a shortcut that the network easily takes instead of learning more generally, patient-agnostic features.

Adding data augmentation may be the key to reduce patient-specific learning. The main bottleneck is to find good data augmentation for EEG. While state-of-the-art self-supervised models in computer vision use strong data-augmentation to increase the complexity of classifying negative vs positive pairs, such data-augmentation functions are still not easily created for EEG.

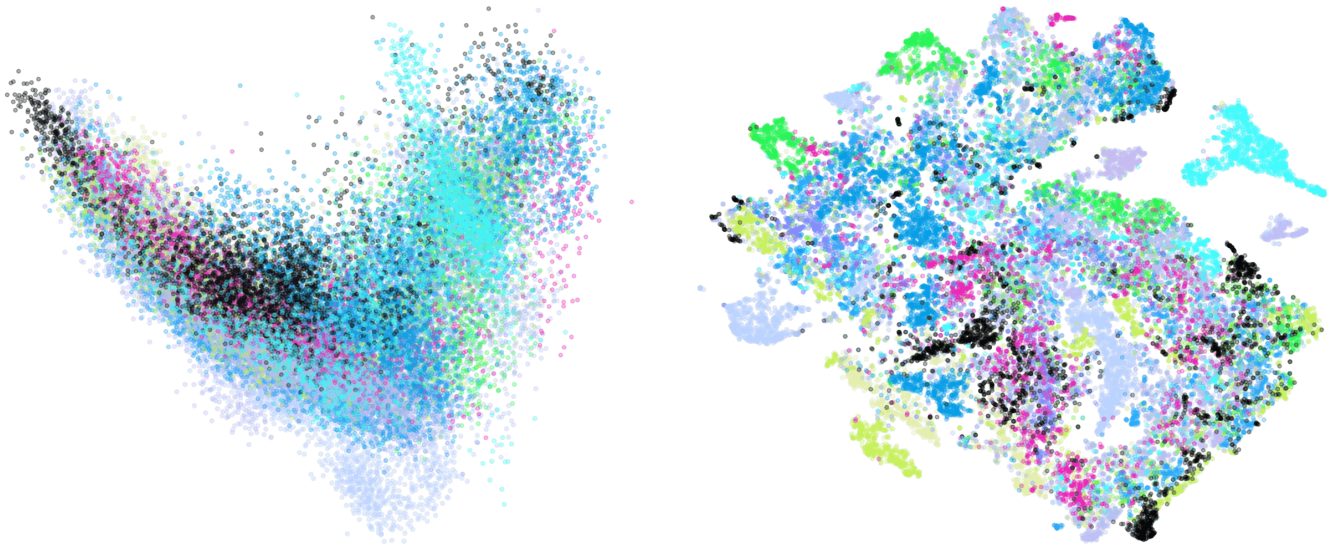
Some papers as Guillot et al. 2021 [11] use channel dropping or swapping. These kind of augmentations do not remove all patient-specific biases. They however help with montage-specific bias or when dealing with noisy electrodes.



(a) PCA (b) TSNE

Figure 9: Patient age distribution in the learned embeddings.

As explained in Tian et al. [19], to reduce patient-specific bias, these functions should change the captured signal without removing the relevant information.



(a) PCA (b) TSNE

Figure 10: Visualization of the embeddings according to the patient ID.

4.6.1 Embedding Differences Between the Two Nights

We also plot the embeddings of the two nights of cherry-picked patients in Figure 21. Other cherry-picked figures can be found in the appendix (fig. 22). We observe that, for some patients, both nights are equally spread in the embedding space, whereas for others, the model learns to cluster each night in tight pockets (which can be independent for each night or not).

When two tight clusters are formed with windows from both nights of the same patient, we can say that we have a patient-specific bias. When the clusters are not containing windows from both nights, the bias is night/experience-specific.

When the windows are evenly spread out, we can conclude that the nights contain less bias. However, the patient-specific bias is usually still there, as the spreading of the two nights is done on the same region of the embedding space. The only difference is that the sleep stages bias is more important than the patient-specific and night-specific bias.

The last thing to see is that the spreading seems to be even between the two nights:

- If night 1 is tightly clustered, then night 2 is also likely to be tightly clustered (not necessarily in the same clusters though).
- If night 1 is spread out, night 2 may also be spread out (not necessarily in the same regions).

The plots can be found in the appendix in Figure 21.

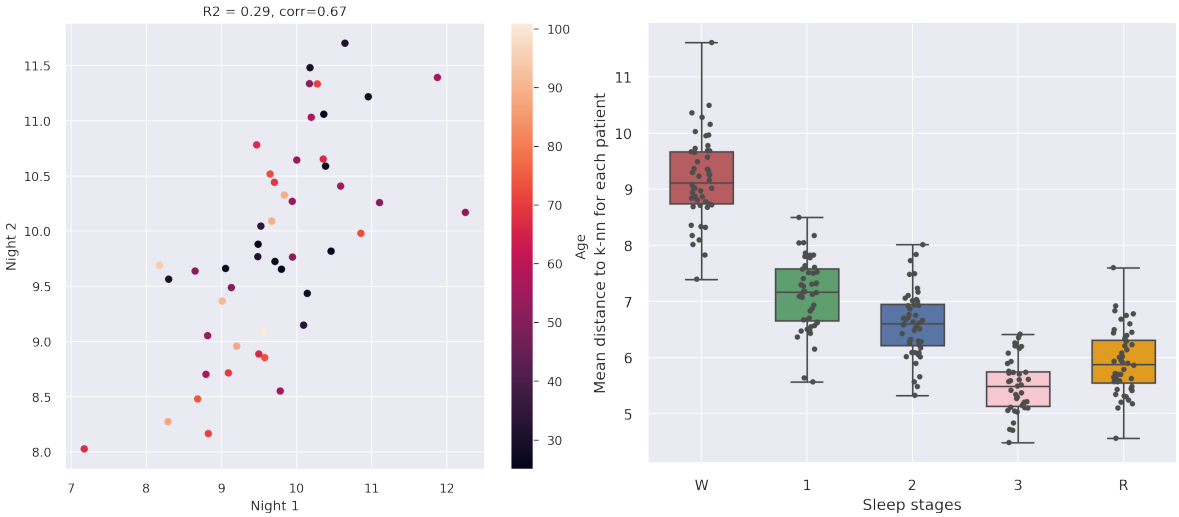
4.6.2 Quantitative Analysis of the Spreading within Embeddings

To get a more quantitative view of this intuition, we computed the mean distance to the k-nearest neighbors (k=20% of total number of valid windows per aggregation) of each window in the embedding space. Then we aggregated them per patient and per night (each window has for neighbors windows from the same patient and the same night) and plotted Night 1 vs Night 2 in Figure 11a. We can see pretty good matching between them, with a $R2 = 0.29$ and a correlation coefficient of 0.67. We also plot the age thanks to a colormap for each patient.

We added the same figure but grouped by the sleep stage in the appendix in fig. 20).

In Figure 11b, we can see the mean distance to k-nearest neighbors per patient according to each sleep stage for every patient (neighbors of windows are computed for the same patient and the same sleep stage).

We saw a correlation between the mean distance to the k-nn aggregated per patient and the age of each patient (coefficient of -0.22). This must come from the fact that older people tends to have more wake phases, and wake windows seem more spread out in the embeddings, leading to this correlation of the mean distance and the age.



(a) Aggregated for each patient and each night, comparing the two nights

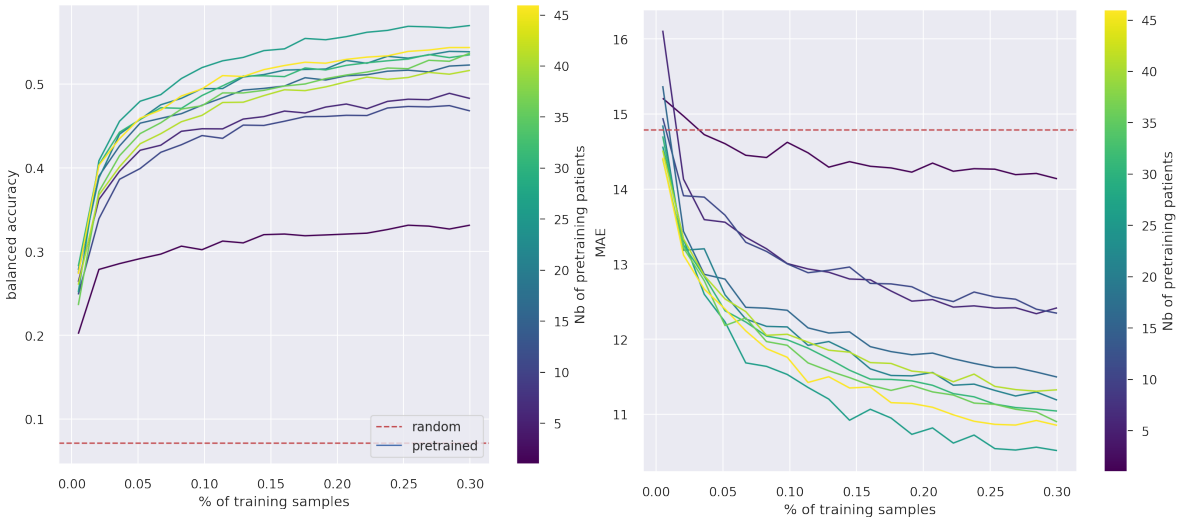
(b) Aggregated for each patient and sleep stage, each point is a patient and each column a sleep stage

Figure 11: Mean distance of each windows to their k-nearest neighbors in the embedding space. Age is displayed with colors.

4.7 Impact of Pre-Training Set Size

We display in Figure 12 k-nearest neighbors algorithms (with $k = 10$) trained to recover the patient ID or their age from the embeddings. We make the pre-training size vary to study the performance of both classifiers as a function of the quantity of unlabeled data used during self-supervised learning.

We observe in Figure 12a that the more unlabeled data, the better the accuracy when predicting to which patient a window belongs. This means the embedding gathers more patient-specific information if it has access to more patients. One may expect this result to get reversed as the quantity of unlabeled data goes to infinity since the model would be forced to learn less patient-specific features. Although it is hard to verify this assumption, one may remark that the highest performances in both estimating the patient ID and their age seem to start to decrease after a certain quantity of unlabeled data. This effect may require much more unlabeled data than what was used in the experiments.



(a) Accuracy when predicting patient ID

(b) Mean Absolute Error of age prediction.

Figure 12: Detection of some patient information using a k-nearest neighbors algorithm trained on a small subset of the test patients.

We finally display in Figure 13 the test accuracy on the final task, which is sleep staging, as a function of the quantity of unlabeled data used during the pre-training phase. After learning to perform the contrastive task, the model is trained on a fixed number of patients and we use early stopping with a patience of 5 epochs using the validation set. We finally report the accuracy on the test set and plot it as a function of the number of patients used for the pre-training phase.

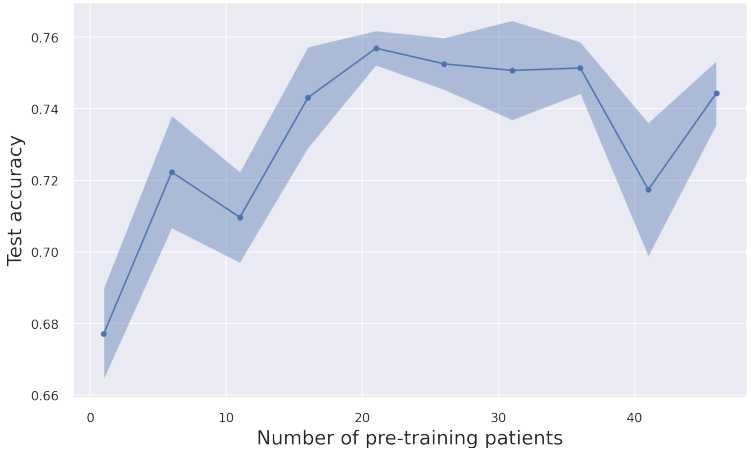


Figure 13: Fine tuning accuracy when changing the number of pre-training patients.

We observe that, trained from scratch, the model reaches a lower accuracy than when one increases the data available for learning the pretext task. However, it starts to plateau after 20 patients' data, meaning it is not much necessary to obtain more unlabeled data than this amount for sleep staging here.

5 Conclusion

In this work, we have used self-supervised learning to pre-train a model on a final task, namely sleep stage classification. We could use unlabeled windows of EEG to pre-train our model on a pretext task in which the goal was to identify whether windows were close to one another. This way, we could learn inductive biases that enabled us to better learn the final task. We then analyzed the learned embeddings and the way they could cluster patient-specific information such as the patients themselves or their age. We found that learning from such a contrastive task raises the issue that the model may cluster sleep stages within clusters of patients. We finally studied the impact of the pre-training size, i.e. the quantity of unlabeled data, on the overall performances of the classifier on the final task. We showed that there were diminishing returns to scale, leading to a plateau in terms of performance with respect to the amount of unlabeled data.

Here, the main issue faced with self-supervised learning comes from the bias learned toward patient-specific feature detection. The network gathers patients into very tight clusters. To reduce the patient-specific bias, new data-augmentation functions (specific to the EEG signal) could be imagined to make it harder for the network to recognize the patients. By doing so, information regarding sleep stages would become more dominant and no or fewer shortcuts toward patient-specific modeling would occur.

We have seen that self-supervised learning may help algorithms to perform well on EEG data in low-data regimes. Yet, it remains crucial to analyze what biases a pretext task may provide to the network during the pre-training phase. One would not want to learn patient-specific features when the final task requires generalization over several patients.

Acknowledgments

We would like to thank our supervisors Alexandre Gramfort and Nora Ouzir for their feedback and support. The code for the experiments is available at `https://github.com/TheoMoutakanni/TCC-EEG/`. In the code, we use the MNE library [20] for EEG preprocessing and loading along with the braindecode library [21] for Deep Learning based models. Classical ML models implementations are taken from scikit-learn [22].

References

[1] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[2] Lilian Weng. Self-supervised representation learning. *lilianweng.github.io/lil-log*, 2019.

[3] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[4] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction, 2016.

[5] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification, 2016.

[6] Hubert Banville, Isabela Albuquerque, Aapo Hyvärinen, Graeme Moffat, Denis-Alexander Engemann, and Alexandre Gramfort. Self-supervised representation learning from electroencephalography signals, 2019.

[7] Hubert Banville, Omar Chehab, Aapo Hyvarinen, Denis Engemann, and Alexandre Gramfort. Uncovering the structure of clinical eeg signals with self-supervised learning. *Journal of Neural Engineering*, 2020.

[8] H. Phan, F. Andreotti, N. Cooray, O. Y. Chén, and M. De Vos. 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.

[9] Jens Stephansen, Alexander Olesen, Mads Olsen, Aditya Ambati, Eileen Leary, Hyatt Moore, Oscar Carrillo, Ling Lin, Fang Han, Han Yan, Yun Sun, Yves Dauvilliers, Sabine Scholz, Lucie Barateau, Birgit Högl, Ambra Stefani, Seung Hong, Tae Kim, Fabio Pizza, and Emmanuel Mignot. Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy. *Nature Communications*, 9, 12 2018.

[10] Mathias Perslev, Michael Hejselbak Jensen, Sune Darkner, Poul Jørgen Jennum, and Christian Igel. U-time: A fully convolutional network for time series segmentation applied to sleep staging, 2019.

[11] Antoine Guillot and Valentin Thorey. Robustsleepnet: Transfer learning for automated sleep staging at scale, 2021.

[12] S. Chambon, M. N. Galtier, and A. Gramfort. Domain adaptation with optimal transport improves eeg sleep stage classifiers. In *2018 International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, pages 1–4, 2018.

[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks, 2016.

[14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[15] Mohsen Ghafoorian, Alireza Mehrtash, Tina Kapur, Nico Karssemeijer, Elena Marchiori, Mehran Pesteie, Charles R. G. Guttmann, Frank-Erik de Leeuw, Clare M. Tempany, Bram van Ginneken, Andriy Fedorov, Purang Abolmaesumi, Bram Platel, and William M. Wells. Transfer learning for domain adaptation in mri: Application in brain lesion segmentation. In Maxime Descoteaux, Lena Maier-Hein, Alfred Franz, Pierre Jannin, D. Louis Collins, and Simon Duchesne, editors, *Medical Image Computing and Computer Assisted Intervention MICCAI 2017*, pages 516–524, Cham, 2017. Springer International Publishing.

[16] S. Chambon, M. N. Galtier, P. J. Arnal, G. Wainrib, and A. Gramfort. A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(4):758–769, 2018.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[18] 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.

[19] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *arXiv preprint arXiv:2005.10243*, 2020.

[20] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7:267, 2013.

[21] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human Brain Mapping*, aug 2017.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

A Evolution of Learned Embeddings as a Function of Pre-Training Size

A.1 Sleep Staging

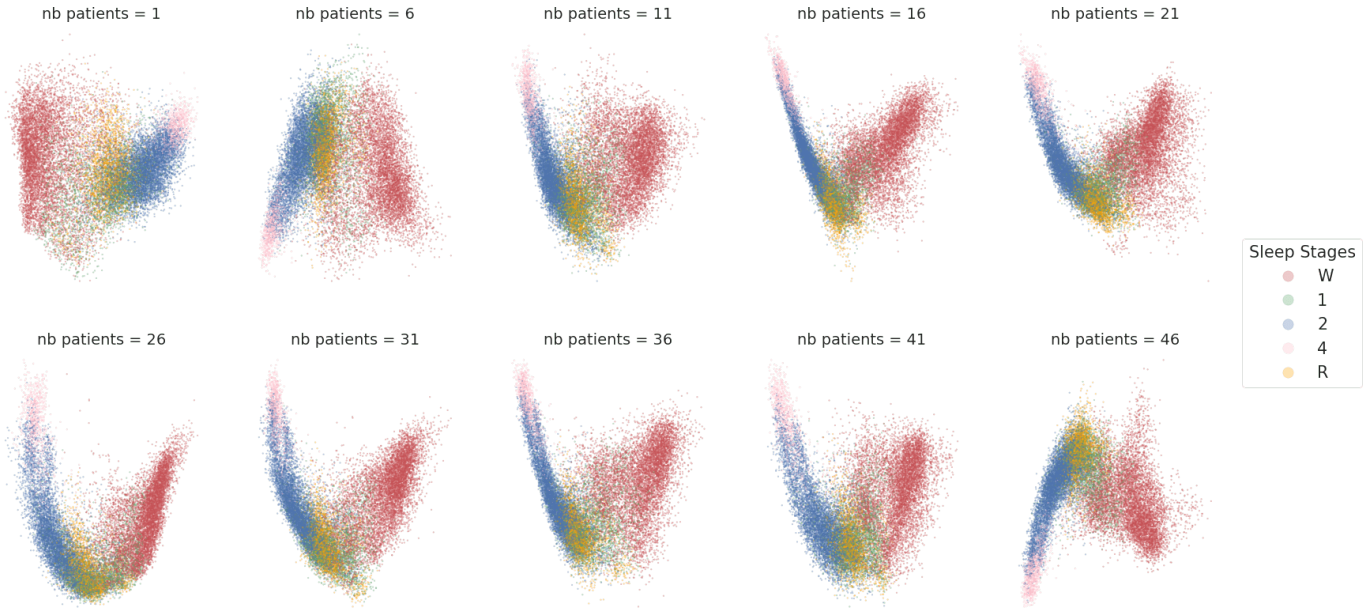


Figure 14: Evolution of PCA according to sleep stages

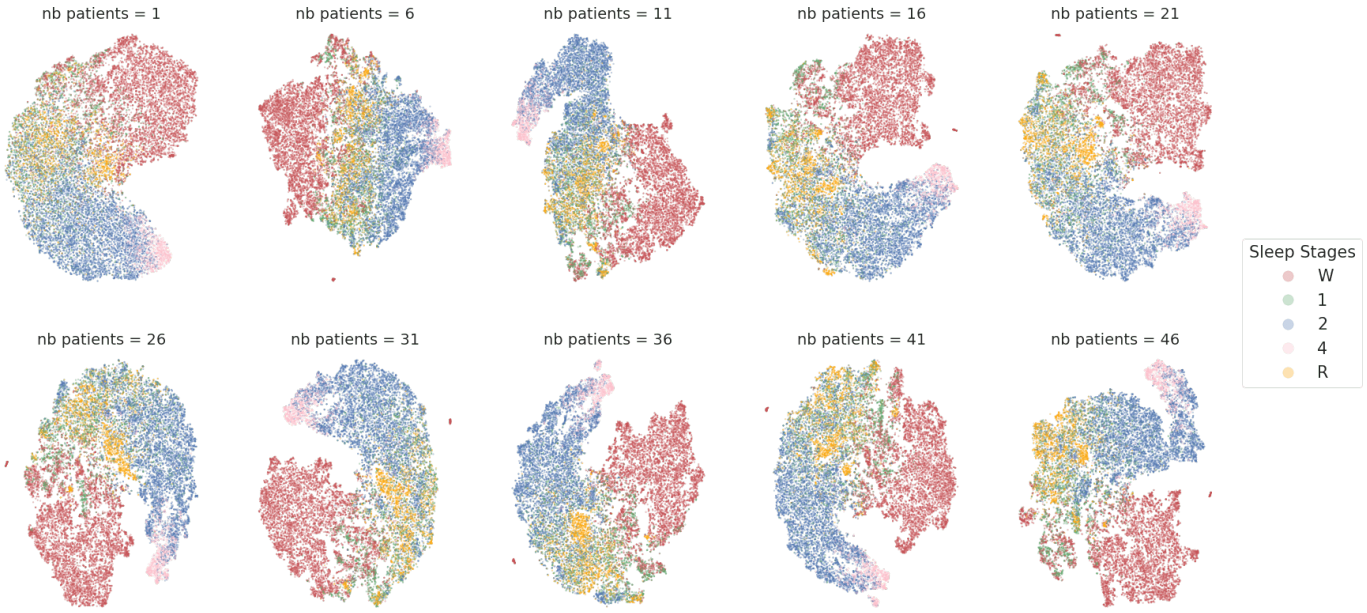


Figure 15: Evolution of T-SNE according to sleep stages

A.2 Patient Age

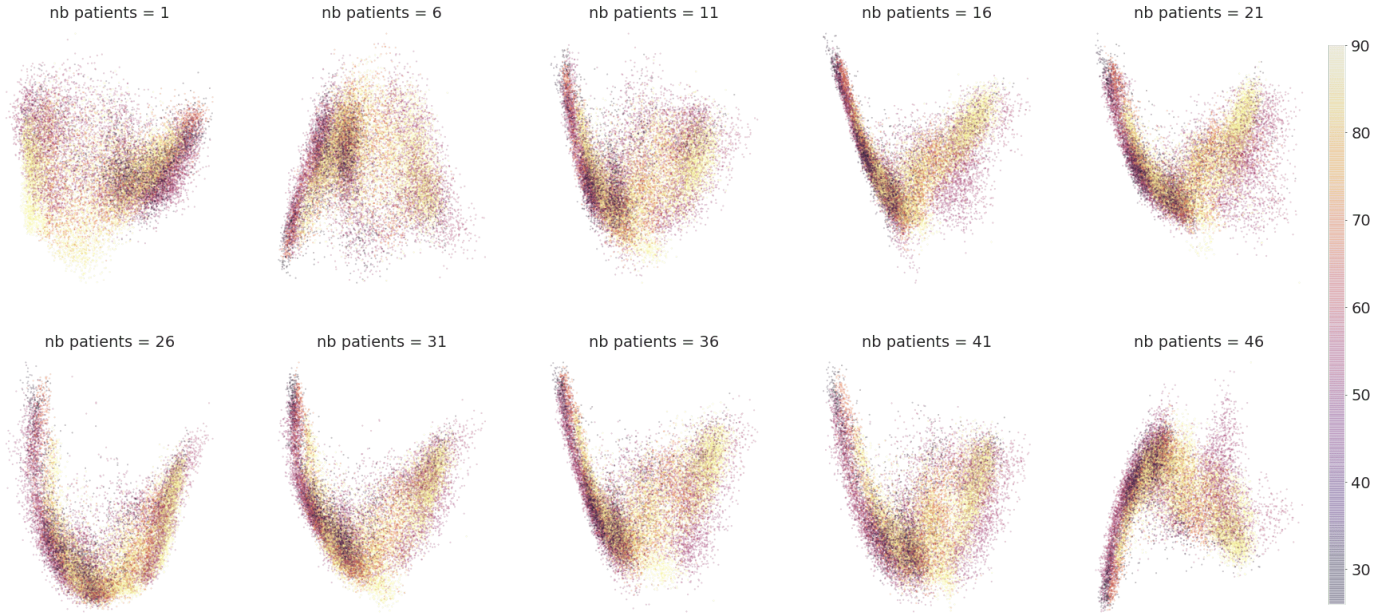


Figure 16: Evolution of PCA according to patient age

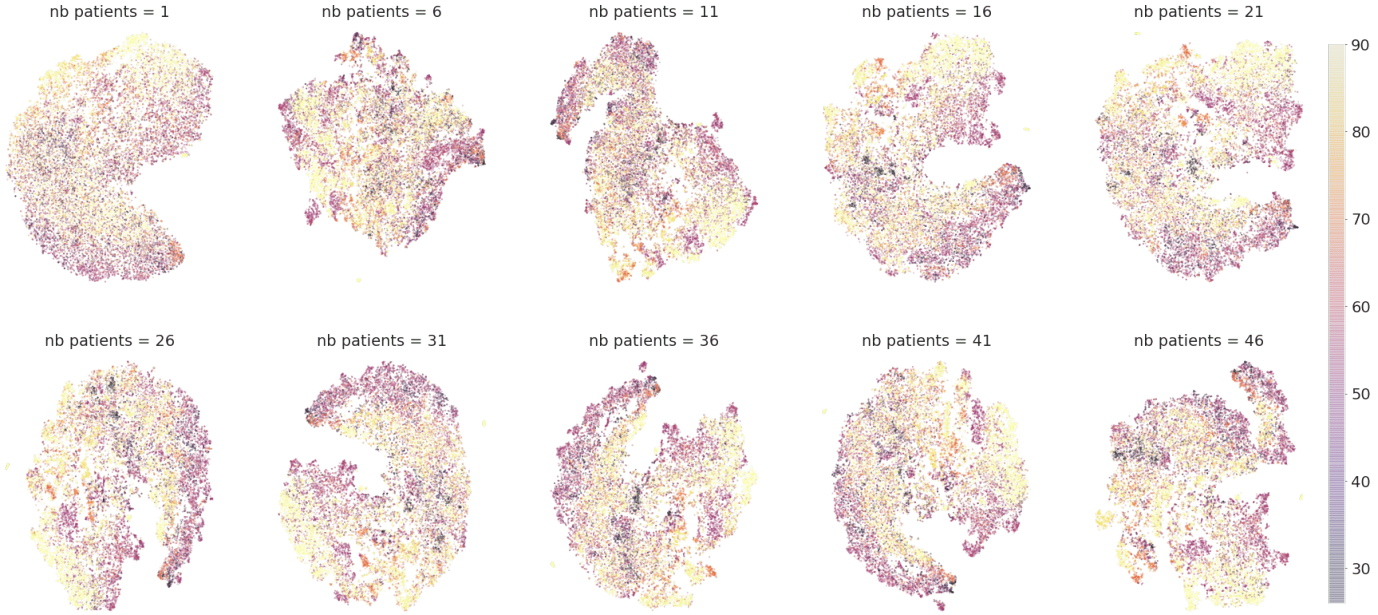


Figure 17: Evolution of T-SNE according to patient age

A.3 Patient ID

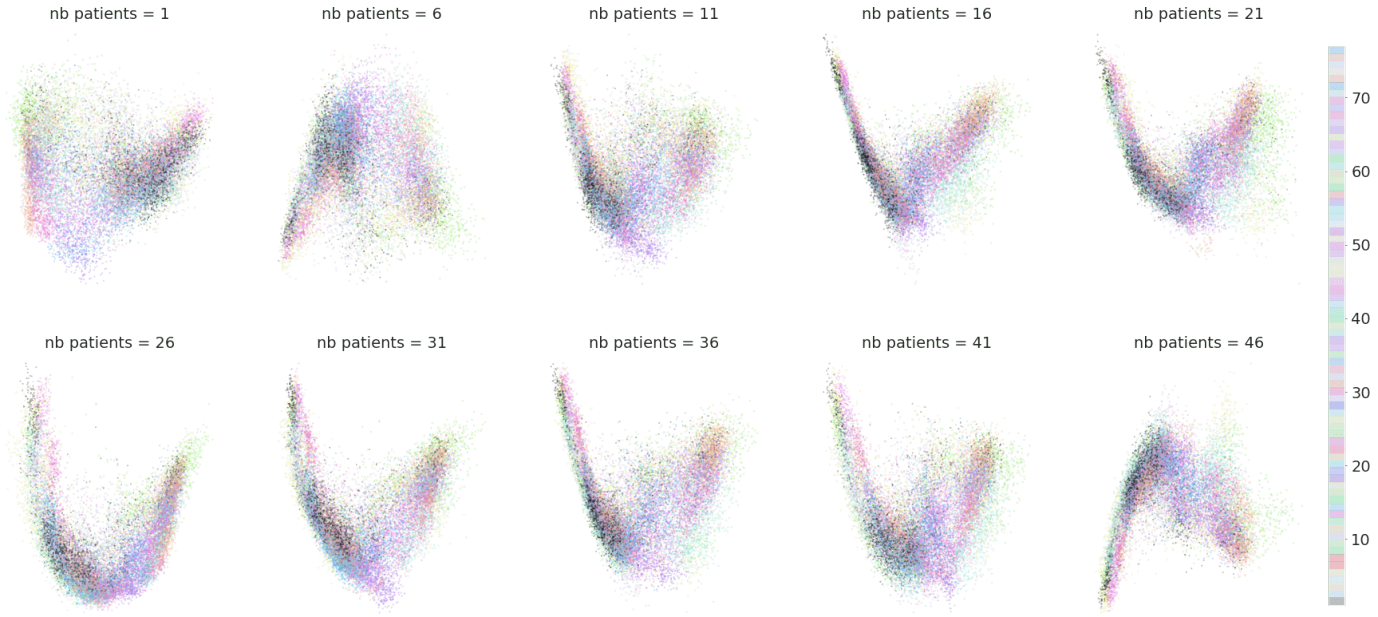


Figure 18: Evolution of PCA according to patient ID

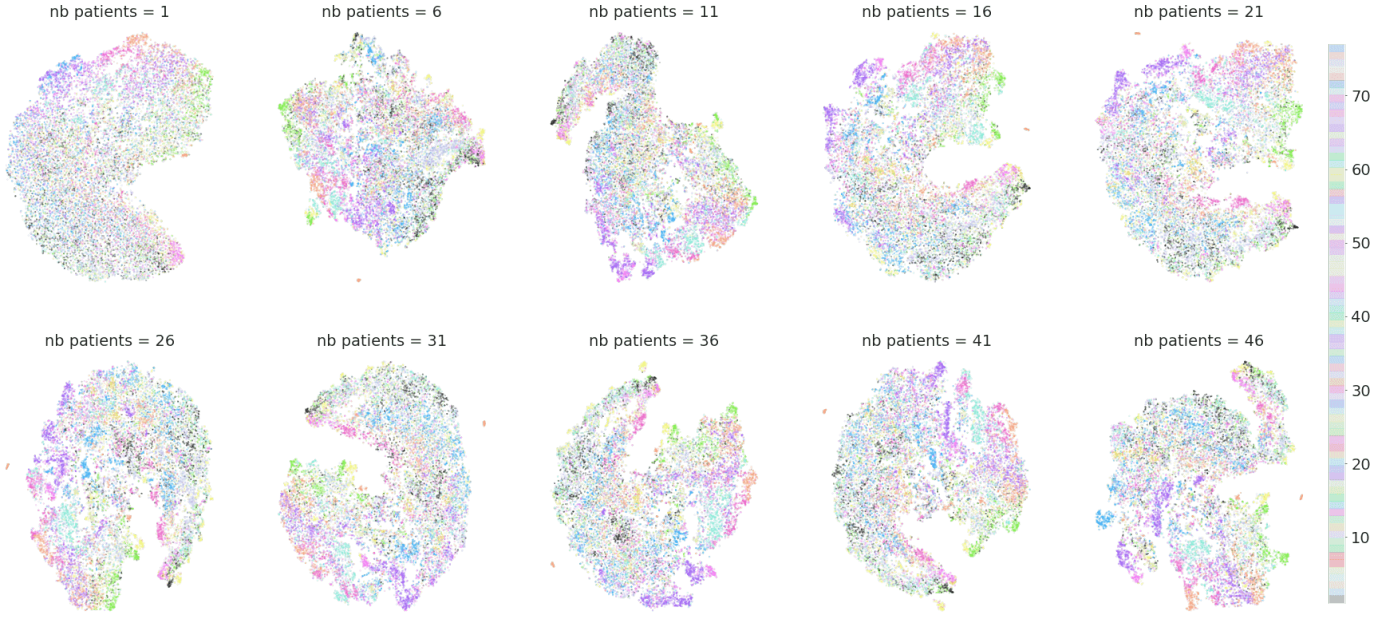


Figure 19: Evolution of T-SNE according to patient ID

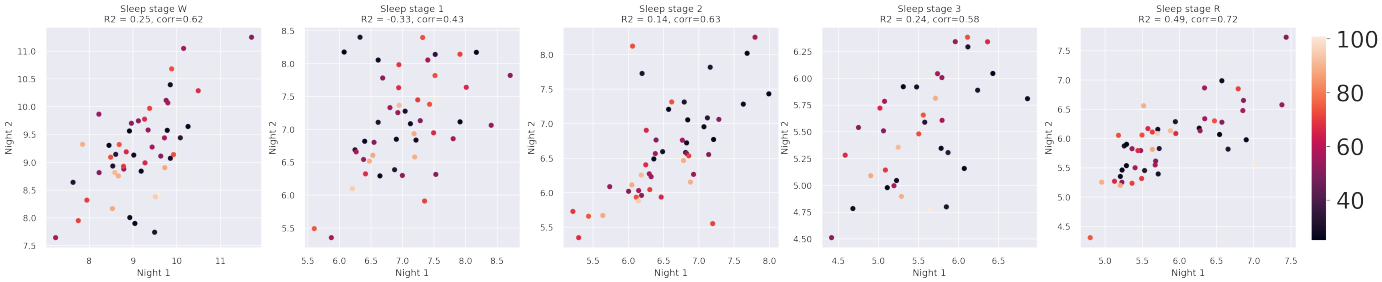


Figure 20: Mean distance of each windows to their k-nearest neighbors in the embedding space. Age is displayed with colors. Aggregated by patient, night and sleep stage.

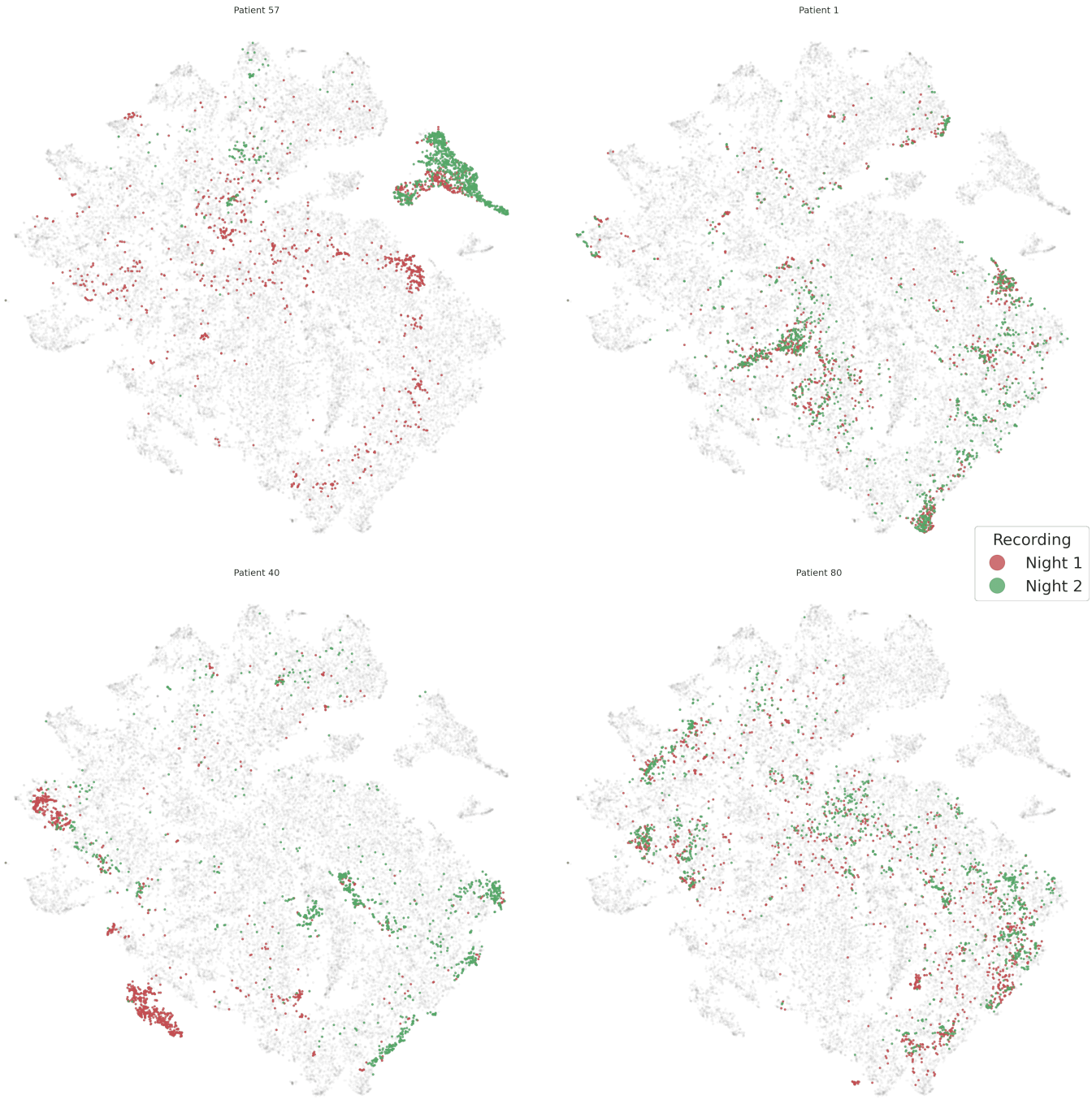


Figure 21: Analysis according to the two recordings of cherry picked patient.

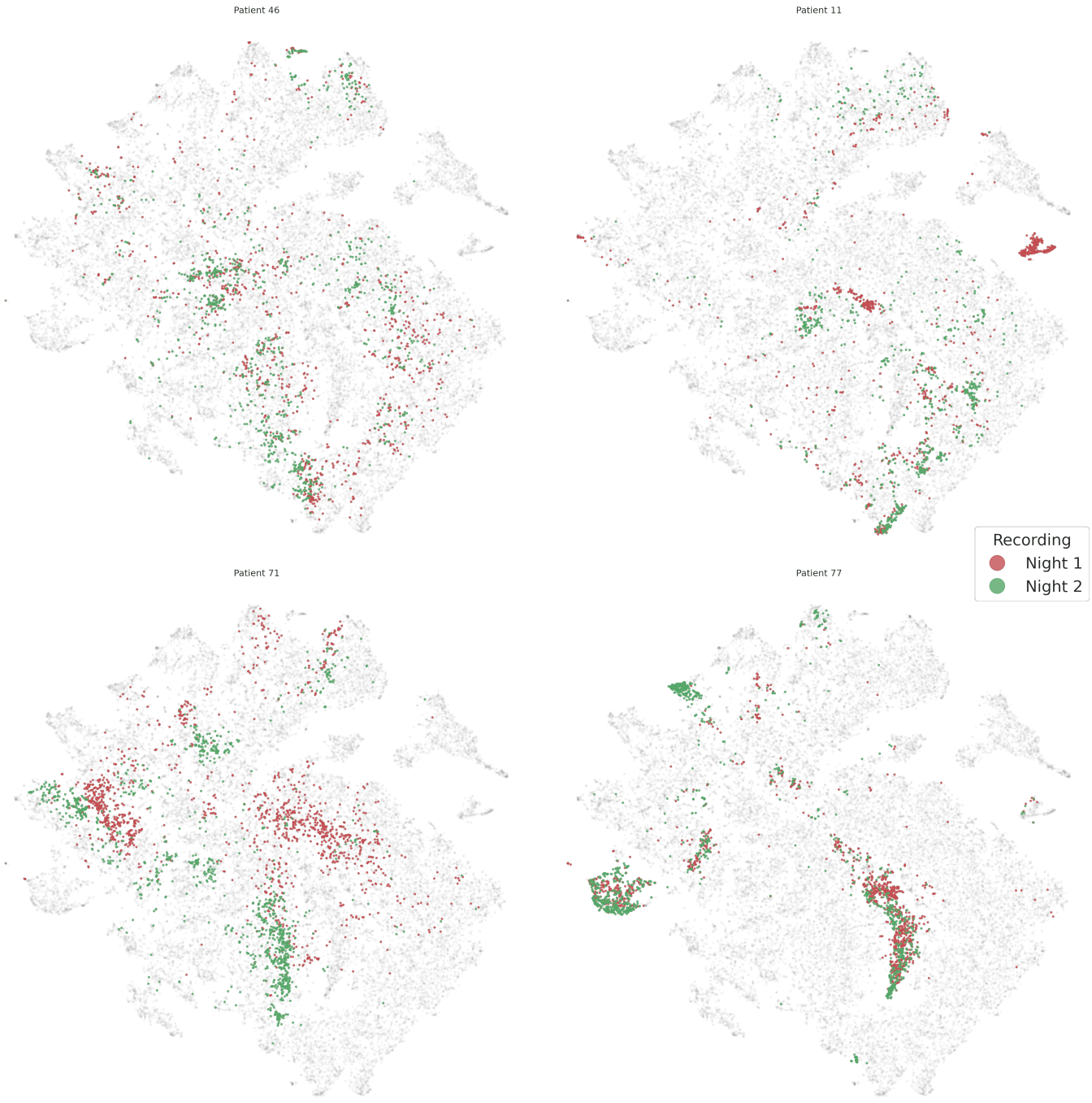


Figure 22: Analysis according to the two recordings of cherry picked patient.