Development of a foundation model for EEG time series

Clement Wang

CentraleSupélec × ENS Paris-Saclay
Beacon Biosignals

May 2024 - Oct 2024

University supervisors:
Laurent Oudre
laurent.oudre@ens-paris-saclay.fr
Bich-Liên Doan
bich-lien.doan@centralesupelec.fr
Arthur Tenenhaus
arthur.tenenhaus@centralesupelec.fr

Company supervisor:
Antoine Guillot
antoine.guillot@beacon.bio

Abstract

Electroencephalogram (EEG) is a critical tool for sleep staging and drug discovery, enabling insights into brain activity across different sleep stages. However, the cost and scarcity of annotated data, especially in the context of EEG recordings from privately owned devices like the Dreem headband, pose significant challenges. This project, inspired by recent advancements in computer vision and natural language processing, focuses on the development of a large EEG foundation model pretrained on extensive datasets of EEG recordings. The aim is to adapt this model for various downstream tasks, such as sleep stage classification, leveraging learned features to accelerate model development and enhance performance in settings with limited data, particularly with Dreem-acquired data.

The project investigates two pretext tasks, contrastive learning and DINO learning, which rely solely on EEG data for pretraining. For PSG sleep staging, the model demonstrates a 3% drop in accuracy compared to fully supervised models, despite being trained on only 5% of the annotated dataset. When applied to Dreem data, the model achieves 77% accuracy, slightly lower than the 85% accuracy of existing production models. Contrastive and DINO models yield similar downstream performances. Considering that the pretraining dataset contains less than 1% Dreem-specific data, there is considerable room for enhancement, showcasing the model's potential for further fine-tuning and optimization with more Dreem data.

Acknowledgement

I would like to express my deepest gratitude to Antoine Guillot, manager of the Device Algorithm Team (DAT) and my mentor, for his invaluable guidance throughout this journey. His insightful discussions constantly challenged my ideas and helped me grow both technically and professionally. His support, willingness to read through papers, and approachable nature made my experience immensely rewarding.

I also want to thank Jason Manley, Machine Learning Engineer on our team, who joined two months before the end of my internship. Despite the short overlap, Jason and I worked closely on the same foundation models project, and his technical insights and fresh perspectives significantly contributed to the project's success. His enthusiasm and collaborative spirit made working together both productive and enjoyable.

A heartfelt thanks to the rest of the DAT team — Franz, Eric, and Kendal — for their friendly one-on-one meetings and daily interactions, which were always insightful and often filled with fun and laughter.

I am also very grateful to Alex Chan, VP of Analytics and Machine Learning, for his monthly reviews and our engaging discussions. He provided me the invaluable opportunity to present my work to the broader company, which was a truly enriching experience.

Special thanks to Ahmet Cakir, my former manager, for onboarding me at the beginning of my internship and helping me navigate the company in those early days. His guidance during that period was crucial in helping me find my footing.

I would also like to thank Mkrtich Vatinyan from the Quality Tools team for his tremendous help during the onboarding process, particularly in understanding the codebase and getting my work environment set up.

Lastly, to the French office team — Aude, Kevin, Lahib, Lara, Mehdi, and Silvia — thank you for the fun breakfasts, lunches, and team-building events. You made every day at the office an enjoyable experience.

Contents

1 Introduction

Electroencephalography (EEG) is a non-invasive method used to record the brain's electrical activity. By placing electrodes on the scalp, EEG captures the brain's electrical signals. EEG is widely used in both clinical and research settings for various purposes, including diagnosing neurological conditions (like epilepsy), studying brain function, and developing brain-computer interfaces (BCIs). In recent years, there has been a growing interest in applying machine learning techniques to analyze EEG data. The goal of such models is to interpret the complex patterns in EEG signals for various applications such as identifying patterns that indicate neurological disorders. Indeed, EEG data is a crucial tool in measuring and analyzing sleep quality, as it provides direct insight into the brain's electrical activity during different stages of sleep.

In the long term, Beacon Biosignals plans to leverage EEG technology to enhance pharmacokinetics and pharmacodynamics studies. Beacon Biosignals aims to gain detailed insights for drug discovery through the analysis of EEG. Given that individuals spend about a third of their time sleeping, analyzing EEG during sleep provides valuable information on a drug's impact on the brain over extended periods. This approach not only improves our understanding of drug effects but also supports more precise dosing strategies and better-informed clinical trial designs, ultimately leading to better-targeted treatments and enhanced patient outcomes.

Currently, annotating an entire night of EEG data, which can last about 8 hours, requires 30 minutes to 1 hour of manual work from clinicians. This process is not only time-consuming but also costly, limiting its scalability. To address these challenges, my project focuses on developing a foundation model for EEG data. Inspired by recent advancements in machine learning, such as large language models (LLMs), the project seeks to leverage large, unannotated EEG datasets to train a robust foundation model. By doing so, the model will be adaptable to new tasks and datasets, requiring less labeled data for fine-tuning. This approach promises to make EEG analysis more scalable, cost-effective, and accessible, potentially transforming research and clinical practices by enabling more efficient and comprehensive brain activity monitoring.

2 Context and objectives

2.1 Polysomnography and Dreem Headband

Traditionally, EEG data is captured through polysomnography (PSG), a complex process that requires the patient to be in a controlled clinical environment under the supervision of a clinician. During PSG, multiple electrodes are placed across the scalp, as well as around the eyes and on the body, to monitor brain activity, eye movements, muscle tone, and other physiological signals. Each electrode records the electrical potential at a specific location on the scalp. An EEG channel represents the difference in potential between two electrodes. However, there is no universally accepted standard for PSG montages, leading to significant variability in channels across different PSG datasets. A widely used montage is the 10-20 system (see Figure 2).

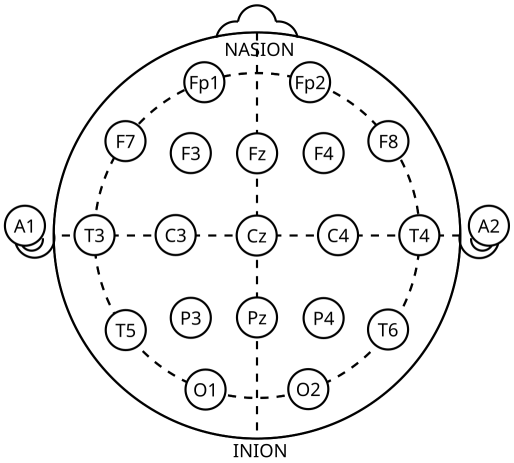


Figure 2: 10-20 system (Wikipedia)

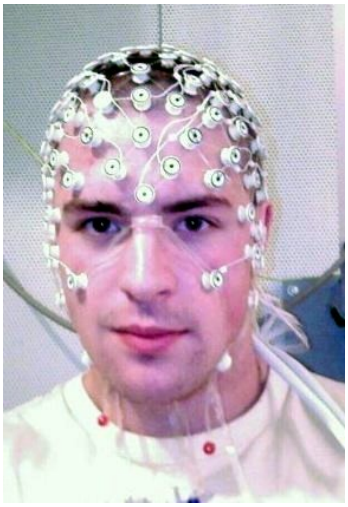


Figure 3: PSG montage on a patient (Wikipedia)

Beacon Biosignals has developed the Dreem Headband (Figure 4), a device designed to simplify EEG data collection for at-home use, eliminating the need for extensive equipment and clinician supervision. The Dreem Headband streamlines the process by using fewer sensors. These include EEG sensors (A, B, C), an audio sensor (D), and an accelerometer (E). Several studies have compared the EEG data collected by the Dreem Headband with that of traditional PSG, demonstrating its effectiveness in various applications [1].

Figure 4: Dreem 3 Headband

Figure 5: Dreem comparison to PSG

2.2 From EEG signals to effective biomarkers for drug discovery

The concept behind using EEG in drug discovery is to identify biomarkers that are strongly correlated with specific diseases. Monitoring these biomarkers enables the tracking of patients' health status over time.

2.2.1 Sleep staging

A crucial step in deriving biomarkers from EEG recordings is annotating sleep stages throughout the night. Typically, an overnight recording is segmented into non-overlapping 30-second intervals, each referred to as an "epoch." Every epoch is then classified into one of five sleep stages according to American Academy of Sleep Medicine (AASM) scoring rules [2]:

- **NREM Sleep (Non-Rapid Eye Movement Sleep):**
  - **Stage 1 (N1):** The initial transition from wakefulness to sleep, characterized by a decrease in alpha waves (8-13 Hz) and an increase in theta waves (4-8 Hz).
  - **Stage 2 (N2):** A light sleep stage, identifiable by the presence of sleep spindles (12-16 Hz bursts) and K-complexes (sudden sharp waves) in the EEG.

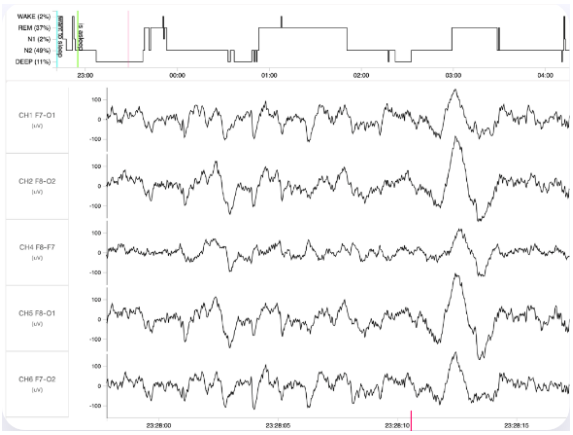


Figure 6: Dreem data and hypnogram visualization

  - **Stage 3 (N3):** Also known as deep sleep or slow-wave sleep (SWS), this stage is marked by a dominance of delta waves (0.5-4 Hz) and is essential for physical restoration and memory consolidation.

- **REM Sleep (Rapid Eye Movement Sleep):** During REM sleep, the EEG pattern resembles wakefulness with mixed frequency and low-amplitude waves. This stage is associated with vivid dreaming and is crucial for emotional regulation and memory processing.

Traditionally, these sleep stages are annotated by clinicians. However, human annotators are prone to inconsistencies, which can impact the accuracy of sleep stage classification. To reduce these errors, datasets often use multiple annotators for each recording, with the final label for each epoch determined by consensus. In cases of poor signal quality, some data may be left unlabeled to avoid inaccurate classification due to the degraded signal.

2.2.2 Examples of biomarkers

A hypnogram, which represents the sequence of sleep stages for each epoch throughout the night, is then used to compute various biomarkers. Some examples include:

- **Sleep Latency:** The time taken to transition from wakefulness to sleep. Longer latency can be an indicator of insomnia [3].

- **Latency to persistent sleep:** The total time from "lights off" to the first sleep epoch that lasts for at least 10 consecutive minutes, which provides insight into how quickly a patient settles into sustained sleep [3].

- **Sleep Fragmentation Index:** A measure calculated by dividing the number of awakenings and transitions to Stage 1 sleep (from Stages 2, 3, or REM) by

the total sleep time in hours. High sleep fragmentation index correlate well with diseases such as Sleep apnea [4].

- **Presence of Sleep-Onset REM Periods (SOREMP):** The occurrence of REM sleep shortly after falling asleep is often related to narcolepsy [5].

2.3 High level goals of the project

This project focuses on the development of a foundation model for EEG data and its thorough evaluation. A foundation model is trained on a large and diverse dataset to learn general data features. At Beacon, foundation models serve two primary purposes:

1. To speed up the development of new models for specific tasks by offering a pretrained model that can be fine-tuned on smaller datasets. We also expect these models to improve the performance of existing models by providing a better initialization point.

2. To offer meaningful data representations that can be used for exploration and visualization.

The foundation model paradigm is gaining prominence in the machine learning community. As computing power increases, models tend to grow in size, leading to improved performance. However, this also results in greater data requirements. The foundation model approach mitigates this by starting from a pretrained model that can be fine-tuned for specific downstream tasks. Additionally, foundation models can generate useful data representations, providing valuable insights into subjects or pathologies.

A pretrained foundation model is expected to exhibit the following properties:

1. Generalization: The model should generalize to unseen data due to its training on a large, diverse dataset.

2. Transferability: The model should be adaptable to a variety of downstream tasks with minimal fine-tuning.

3 Literature review

Self-supervised learning is a machine learning approach where models are trained on large amounts of unlabeled data by leveraging a pretext task. In this method, the model learns to predict certain aspects of the data itself, such as missing parts or context, from the data's inherent structure. The goal is to generate useful representations or features from the data without requiring explicit labels. Once these representations are learned, they can be adapted or fine-tuned for specific downstream tasks using smaller labeled datasets. Models trained through this approach are referred to as foundation models. This technique has been effectively used in fields such as computer vision and natural language processing.

3.1 Sleep staging

Before diving into self-supervised learning, we first go through the literature on sleep staging to get an idea of the architectures and the data. Indeed, supervised classification is always the starting point of all more complex pipelines and serves as a baseline.

Sleep staging approaches can broadly be categorized into two groups: expert models and deep learning models. Expert models rely on handcrafted features, which are then fed into traditional machine learning algorithms such as linear regression models [6], decision trees [7], or ensemble methods [8]. These approaches have been foundational but are increasingly outperformed by deep learning models, thanks to advancements in computational power and the availability of larger datasets.

Convolutional neural networks (CNNs), widely successful in image analysis, have also demonstrated their efficiency in sleep staging [9]. Architectures such as DeepSleepNet [10] employ multi-scale convolutions to capture information at various resolutions. This model introduces the combination of convolution layers for feature extraction from raw EEG signals, which are then processed by recurrent neural networks (RNNs). Building on this idea, SeqSleepNet [11] includes a preprocessing step with Fast Fourier transform (FFT) to improve feature representation.

Inspired by the U-Net architecture [12] from computer vision, U-Sleep [13] adopts an hourglass-shaped network that enforces multi-scale feature extraction across different stages. Notably, most previous models only use a single EEG channel as input, potentially limiting classification performance. EEG data can be recorded from various montages, and ignoring this diversity might lead to suboptimal results. RobustSleepNet [14] addresses this by applying attention mechanisms across channels, allowing the network to handle varying channel configurations.

Guillot et al. [15] highlights important considerations regarding the data, particularly the effect of classifying multiple consecutive 30-second epochs, which has been shown to improve accuracy by a few percentage points. Unlike image datasets, where labels are generally more consistent, sleep stages are annotated by clinicians based on standardized guidelines [2]. However, inter-annotator variability can bias

the metrics [14]. To address this, many recent datasets are annotated by multiple clinicians, with models being trained using a majority voting approach to ensure better consistency. State-of-the-art models in sleep staging typically achieve accuracy ranging from around 85% to 90% on various datasets.

3.2 Time series foundation models and self-supervised learning

There are very few works that have applied self-supervised learning to time series data and even less to EEG data. Moreover, most published works were not in a top-tier conference or journal which hints us not to rely too much on these works.

Banville et al. [16] present self-supervised learning pretext tasks for EEG data, focusing on the relative position of EEG segments, demonstrating how self-supervised learning can yield meaningful representations for downstream tasks. A straightforward pretext task involves sampling two 30-second windows and predicting whether the time between them is greater or less than 5 minutes. They also conclude that a typical EEG data length of 5 minutes is optimal for self-supervised tasks. Mohsenvand et al. [17] explore contrastive learning on single channels of EEG data, revealing the value of augmentations and channel-specific feature learning. Contrastive learning consists of training a model to bringing similar data points closer and pushing dissimilar ones apart in the feature space. They demonstrate that extracting features from individual channels first, and then combining them for classification, can be advantageous. Thapa et al. [18] build on this concept by training encoders using contrastive learning across multiple modalities, including EEG, ECG, and respiratory signals.

Eldele, Emadeldeen, et al. [19] and Das, Abhimanyu, et al. [20] take a different approach, scaling up the architectures and applying transformer models to time series data. These works demonstrate the power of transformers with masked inputs and autoregressive training. Additionally, the Neuro-GPT model [21] follows a decoder-only transformer architecture for EEG. Foumani et al. [22] and Chaoqi et al. [23] argue that generating time series is ineffective due to EEG's low signal-to-noise ratio, and therefore recommend using representation-based loss functions like contrastive learning for training.

3.3 Literature on image foundation models and self-supervised learning

Deeper insights into contrastive learning and self-supervised learning can be gained from the computer vision domain. Indeed, time series are similar to text in the sense that they are sequences of data. However, they are also similar to images in the sense that they can be represented as 2D matrices while images can be represented as 3D matrices. The computer vision domain has seen a lot of progress in

the last few years with the introduction of self-supervised learning tasks and the transformer architecture. Moreover, self-supervised learning tasks in NLP are predominantly focused on masked language modeling and next token prediction. In contrast, the computer vision domain has explored a wider range of approaches due to the limitations of contrastive learning.

A significant portion of recent advancements in computer vision can be attributed to contrastive learning in multi-modal settings. One of the most prominent examples is CLIP [24], which is trained using contrastive learning on image-text pairs. This enables the model to learn a unified representation of both modalities. As the first model trained without explicit labels, CLIP achieves remarkable robustness and captures highly meaningful feature representations.

SimCLR [25] sets a standard self-supervised pipeline which performs contrastive learning with images alone. Further details will be provided in subsequent sections. The performance of contrastive learning is highly dependent on batch size, as it determines the number of negative samples used in the loss calculation. PIRL [26] and MoCo [27] improve contrastive learning by utilizing queues of negative samples, showing that this approach enhances performance by approximating the use of the entire dataset as negative pairs. MoCo also introduces student-teacher training, where the student learns to match the teacher's predictions, with the teacher being a moving average of the student. BYOL [28] eliminates negative pairs entirely, demonstrating empirically that it avoids collapse. DINO [29] and its successor, DINOv2 [30], build on these methods to achieve even higher performance.

3.4 On the evaluation of self-supervised learning

One major challenge in self-supervised learning is the absence of reliable metrics to evaluate the quality of learned representations. The primary objective of self-supervised learning is to capture meaningful data representations. Typically, the effectiveness of these representations is assessed by using them as features in downstream tasks.

Typically, learned representations are evaluated through downstream tasks. Most studies assess performance using both linear probing and k-NN classification. In linear probing, a linear classifier is trained on top of the learned representations, while k-NN classification uses these representations as features for a k-nearest neighbors classifier. Both metrics are advantageous as they require minimal computation and do not depend on large labeled datasets. Papers often also evaluate transfer learning performance by fine-tuning on a small subset of the labeled dataset. For example, many studies assess fine-tuning on 1% or 10% of ImageNet. This approach is more computationally demanding and requires access to labeled data. Additionally, fine-tuning must be done carefully to avoid degrading the quality of the learned representations.

These derived metrics are task-specific, highlighting the need for more general

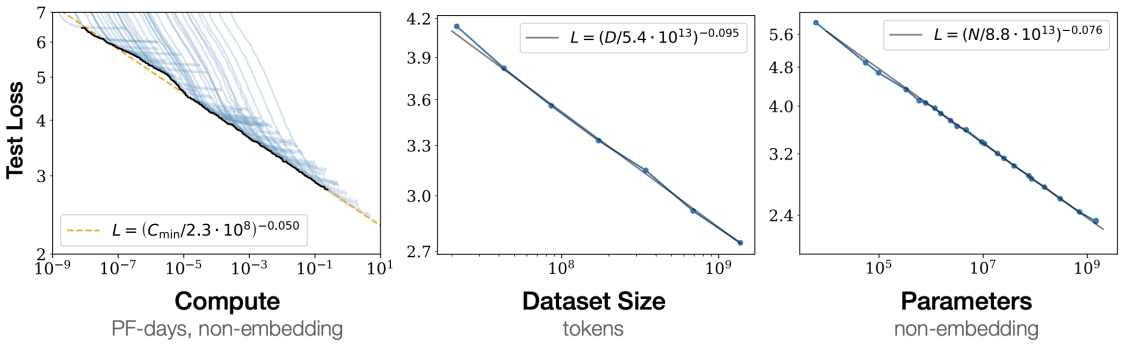


Figure 7: Power laws in NLP with respect to compute time, dataset size and the number of parameters [36]

evaluation methods. To address this, one approach is to examine the contrastive loss, which aims to maximize the similarity between positive pairs and minimize it between negative pairs. However, this alone does not ensure a well-structured distribution of the learned representations. For instance, in a 256-dimensional embedding space, the model may collapse into a smaller subspace. To address this, uniformity metrics have been introduced [31, 32] to assess how well the embeddings are spread across the space. Fang et al. [32] further demonstrated that their uniformity metric could even be used as a regularizer, leading to improved representation learning, though at the cost of slightly reduced model fitting. Unfortunately, these studies reveal limitations, as uniformity does not necessarily indicate high-quality representations. Park et al. [33] propose a metric based on the neighborhood of each sample to assess the reliability of learned representations. They demonstrate that this metric can effectively evaluate representation quality, rather than relying solely on the loss function. RankMe [34] attempts to approximate the rank of the embedding projection, with a higher rank indicating less collapse in the representation. LiDAR [35] focuses on estimating the linear probing performance of the learned representations. More details on RankMe and LiDAR will be provided in subsequent sections.

3.5 Scaling up architectures

Recent trends in machine learning focus on training larger models on increasingly vast datasets. However, as training costs continue to rise, it becomes crucial to optimize the efficiency of the training process. Various studies have explored the scaling laws of models, investigating how to properly increase model size, dataset size, and the number of training epochs. Whether applied to text [36], images [37], or text-image contrastive models [38], these scaling laws follow similar patterns.

Kaplan et al. [36] laid the foundation for scaling laws with an in-depth study on the scaling of large language models (LLMs).

The main conclusions are:

- The test loss follows a power law in relation to model parameters, dataset size, and compute power, as long as none of these factors is held constant.
- Only the total number of parameters matters; the width-to-depth ratio does not significantly affect test loss for a fixed parameter count.
- Larger models are more sample-efficient.
- Increasing model size is more beneficial than training to full convergence.
- Model size has a greater impact on performance than dataset size.

4 Datasets

This section outlines the various datasets used throughout the project. The pretraining dataset for the foundation model combines both PSG and Dreem EEG data, with the aim of developing a generalizable foundation model that performs effectively across both domains. For the downstream tasks, we utilize a PSG sleep staging dataset, along with three dedicated Dreem datasets tailored for specific analyses.

4.1 Foundation model dataset

The foundation model dataset is an aggregation of several datasets:

- **MGH 2019 (Massachusetts General Hospital)** [39]: A large dataset containing various 10-20 EEG and PSG recordings in multiple formats/encodings. It includes data from around 32k patients across different hospital departments, covering the period from 2009 to 2018.
- **Clean-Sleep**: An aggregation of multiple studies where signals were annotated for sleep staging. See next subsection for more details.
- **Clean-Seizure**: A subset of the MGH 2019 dataset that has been cleaned and annotated for seizure detection.
- **Temple University Hospital (TUH)** [40]: An ongoing project that currently includes over 60,000 EEG recordings collected from 2002 to the present.
- **Internal PSG dataset**
- **Dreem-Pretraining**: Privately owned, unlabeled data captured using the Dreem headband.

The datasets were cleaned such that recordings overlapping from several datasets were not added. This project focuses solely on EEG data, although some studies integrate other sensor types like EOG, ECG, EMG, snore audio, and heart rate alongside EEG.

4.2 Downstream datasets

4.2.1 Sleep staging on PSG EEG: Clean-sleep dataset

The clean-sleep dataset is an aggregation of several PSG datasets that were cleaned and annotated for sleep staging. It comprises:

Source dataset	Approximate number of recordings	Proportion of Total Dataset (%)
mgh2019	34,000	14.41%
mesa	2,000	0.85%
mnc	1,000	0.42%
mros	6,000	2.55%
shhs	9,000	3.81%
temple_seizures	200	0.08%
temple_events	10	0.00%
internal PSG dataset	182,000	77.12%
dreem-pretraining	1,000	0.42%
Total	**236,000**	100%

Table 1: Summary of foundation model dataset.

- **MROS (MrOS Sleep Study)** [41, 42]: Ancillary study focused on understanding the relationship between sleep disorders and falls, fractures, mortality, and vascular disease in 2,911 men aged 65 or older. The data spans two sleep study cycles: 2003-2005 and 2009-2012.
- **SHHS (Sleep Heart Health Study)** [41, 43]: A multi-cohort study aimed at exploring sleep-disordered breathing and its cardiovascular outcomes. The dataset includes 5,804 adults aged 40 and older, with two exam cycles (1995-1998 and 2001-2003), and cardiovascular outcomes tracked until 2010.
- **MESA (Multi-Ethnic Study of Atherosclerosis)**: The MESA (Multi-Ethnic Study of Atherosclerosis) is a large medical research study involving over 6,000 participants from diverse racial and ethnic backgrounds, including African Americans, Latinos, Asians, and Caucasians. Launched in 1999, it focuses on the early stages of atherosclerosis and has since expanded its research to brain and lung health, as well as the effects of factors like neighborhoods, air pollution, sleep, stress, and nutrition on overall health.
- **MNC (Mignot Nature Communications)** [41, 44]: A study focused on large-scale neural data collection.

The Clean-Sleep dataset consists of roughly 25,000 annotated nights from about 15,000 patients, with one annotation provided for each 30-second epoch.

4.2.2 Sleep staging on Dreem EEG

As previously mentioned, the Dreem EEG data is collected using the Dreem Headband, a device designed, developed, and maintained by Beacon Biosignals. A primary objective of Beacon's device algorithm team is to maximize the performance of models using this data. Consequently, models are validated and tested on this dataset, making it critical to have a robust evaluation process. The Food and Drug

Administration (FDA) bases its regulatory decisions regarding the Dreem Headband on the results derived from this dataset.

The Dreem dataset is an aggregation of various datasets, each corresponding to a specific cohort from a particular study. For consistency, the test dataset remains fixed across all studies. Evaluation follows a leave-one-out approach, with training and validation performed on a combination of Dreem datasets with 294 recordings: 247 for training, 47 for validation. Testing is conducted on a dataset of 39 recordings.

The data format is consistent with PSG data, where each 30-second epoch receives an annotation for one of the five sleep stages. Each recording is annotated by multiple clinicians, with the final label determined by majority vote. Clinicians also have the option to mark stages as unknown or unclear when necessary.

4.2.3 On-head dataset

This EEG task was introduced at Beacon Biosignals to address the challenge of distinguishing when the Dreem headband is not being worn, as the EEG signals in such cases can resemble those from a properly worn headband. This issue arises because the headband is used by patients at home without clinical supervision. To develop and assess algorithms for detecting whether the headband is on the head, Beacon Biosignals has annotated EEG data specifically for this task. Given the scarcity of data, Beacon places significant emphasis on maintaining a fixed test set for consistent evaluation. The dataset is divided into 235 recordings for training, 39 for validation, and 150 for testing, with the task framed as a binary classification on 30-second epochs. The data also includes accelerometer time series and consists of recordings from over 50 subjects, each lasting more than six hours.

4.2.4 Signal Quality dataset

Signal quality assessment is another task unique to Beacon Biosignals. This task emerged from the observation that sleep staging algorithms perform poorly when the input EEG signal is of low quality. To address this, Beacon annotated data to develop and evaluate algorithms capable of predicting the signal quality. This is also a binary classification task, but each EEG channel is individually labeled, with labels corresponding to 2-second windows. The dataset comprises 1,200 hours of labeled data.

Task	Class	Count	Total	Proportion (%)
PSG Staging	REM	210,000	3,380,000	6.21%
	NREM1	790,000		23.37%
	NREM2	1,080,000		31.95%
	NREM3	440,000		13.02%
	Wake	750,000		22.19%
	No Stage	2,200		0.07%
Dreem Staging	REM	46,000	370,000	12.43%
	NREM1	14,000		3.78%
	NREM2	110,000		30.81%
	NREM3	44,000		11.89%
	Wake	120,000		33.24%
	No Stage	27,000		7.30%
On-Head Detection	On Head	1,100	1,300	82.09%
	Off Head	230		17.91%
Signal Quality	Interpretable	17,000	32,000	53.48%
	Uninterpretable	15,000		46.52%

Table 2: Summary of downstream dataset distributions.

5 Reference metrics and baselines

Before delving into the core of the topic, we begin with some preliminary work to better understand the subject and establish a benchmark for what constitutes an effective model. This section is divided into two parts. The first part presents a preliminary study using fully supervised transformers to evaluate the model's performance on our data. The second part introduces a baseline experiment with a self-supervised pipeline, demonstrating that pretraining a model on EEG data can be beneficial for sleep staging.

5.1 Fully supervised sleep staging

Beacon Biosignals' sleep staging models are primarily based on either RobustSleepNet [14] or U-Sleep [13], both of which are lightweight convolutional models without restrictions on input length. Since self-supervised models typically require larger architectures, we have narrowed the scope of this project to models that process 30-second epochs, similar to the approach taken in BIOT [23]. For a fair comparison, we are also training transformer models to perform sleep staging using a single epoch of data, adopting the same architecture outlined in the BIOT paper, as detailed in the next section. This serves both as a sanity check to confirm that the transformer model can effectively process EEG data and as an opportunity to fine-tune the transformer architecture for EEG analysis, key groundwork for subsequent

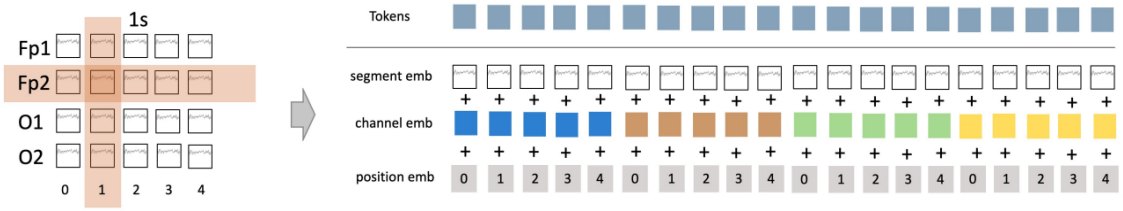


Figure 8: BIOT tokenization procedure [23]

self-supervised pretraining studies.

5.1.1 BIOT transformer architecture for EEG data

We reimplemented the BIOT transformer architecture for the classification task, following an approach similar to the Vision Transformer (ViT) paper [45], which applies Transformer models to image data.

Tokenization. We follow the tokenization procedure of BIOT (Figure 8):

- **Resampling**: The EEG signal is resampled to 100Hz, resulting in a data shape of (N_c, T), where N_c is the number of channels, and T is the time dimension.
- **Fourier Transform**: We compute the Fast Fourier Transform (FFT) with a 50% overlap, converting the signal into the frequency domain. This step transforms the data to (N_c, T', N_f), where T' represents the time bins (with each bin corresponding to 1-second intervals of data), and N_f is the number of frequency bins.
- **Fully Connected Layer**: A fully connected layer is applied along the frequency axis, mapping the frequency representation into a hidden latent space, resulting in (N_c, T', n_hid), where n_hid is the size of the hidden dimension.
- **Channel Embedding**: Channel embeddings are added to identify each channel.
- **Positional Encoding**: Positional encodings are included to retain temporal ordering within the tokenized sequence.
- **Flattening**: The final tokenized representation is flattened, preparing the input for the subsequent Transformer layers.

Architecture. The architecture follows an encoder-only transformer design. The BIOT paper proposes utilizing a linear transformer, specifically the Linformer architecture [46]. This approach is particularly useful since flattening the channel axis

into the time axis can result in a large number of tokens when dealing with multiple channels.

Let $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{l_1 \times l_2}$ represent the query, key, and value matrices, respectively. In Linformer's self-attention mechanism, we approximate the softmax attention matrix, which is of size $N \times N$, using a low-rank factorization with matrices $\mathbf{E}^\top \in \mathbb{R}^{N \times d}$ and $\mathbf{F} \in \mathbb{R}^{d \times N}$, where $d \ll N$. The output $\mathbf{H} \in \mathbb{R}^{N \times l_2}$ is then computed as:

$$\mathbf{H} = \text{Attention}(\mathbf{X}\mathbf{W}^Q, \mathbf{E}\mathbf{X}\mathbf{W}^K, \mathbf{F}\mathbf{X}\mathbf{W}^V) \tag{1}$$

$$\mathbf{H} = \text{softmax}\left(\frac{(\mathbf{X}\mathbf{W}^Q)(\mathbf{E}\mathbf{X}\mathbf{W}^K)^\top}{\sqrt{l_2}}\right) \cdot \mathbf{F}\mathbf{X}\mathbf{W}^V \tag{2}$$

This approach reduces the complexity of computing self-attention, as proposed in the Linformer architecture, by compressing the attention matrix using the low-rank approximation. However, it also reduces the expressiveness of the transformer architecture.

Embedding reduction. There are two common approaches for obtaining the final embedding:

- In BIOT, the final embedding is obtained by averaging all the tokens.
- In Vision Transformer (ViT) and BERT [47], a trainable [CLS] token is appended at the beginning of the sequence, and the output corresponding to this token is used as the final representation.

For all subsequent experiments, we default to using the [CLS] token. This approach is the standard practice in both computer vision and natural language processing tasks.

5.1.2 Experiments and results

The dataset used in this experiment is the Clean-sleep PSG staging dataset restricted to 20,000 nights of polysomnography (PSG) data. The goal is to train a transformer architecture from scratch in a fully supervised setting.

We began by setting up the transformer model using the parameters from the BIOT paper, which served as our baseline. To enhance the model's expressivity, we switched to a vanilla transformer architecture, but we had to reduce the learning rate from 0.001 to 0.0001 to prevent the model from failing to converge.

Next, we decreased the number of transformer layers from 6 to 4. This adjustment was necessary to address overfitting, which we observed with the deeper architecture, and to accelerate the training process. While switching from average pooling to using a CLS token slightly improved the model's performance, it wasn't a major breakthrough.

As the model started to overfit early in training, we introduced a cosine learning rate scheduler with a warmup phase. This helped the model generalize better, resulting in a noticeable improvement in accuracy.

To further scale up the model, we employed mixed precision training, allowing us to increase the batch size from 512 to 1024 without running into memory issues. This step contributed significantly to the accuracy boost.

Finally, we reduced the weight decay from 0.01 to 0.0001. The previous value was too high, which likely prevented better fit of the data.

Configuration Change	**Accuracy (%)**
Linear transformer, 6 layers, average pooling, batch size 64, Adam, lr 0.001, wd 0.01	63.14
Linear → Vanilla transformer, lr 0.001 → 0.0001	65.05 (+1.91)
6 layers → 4 layers, average pooling → CLS token	65.44 (+0.39)
+ cosine learning rate scheduler with warmup	67.00 (+1.56)
Batch size 512 → 1024, Adam → AdamW, lr 0.0001 → 0.001, + mixed precision (bf16)	75.65 (+8.65)
Batch size 1024 → 2048, wd 0.01 → 0.0001	77.89 (+2.24)

Table 3: Summary of configuration changes and corresponding accuracies.

5.1.3 Fully supervised with limited training data

We then report the accuracy on progressively smaller subsets of the dataset, using 10, 100, and 1,000 recordings, to compare against the full dataset results. Naturally, this reduction leads to overfitting, so we apply early stopping to mitigate it. The results are summarized in Table 4.

Number of Training Records	**Test Accuracy (%)**	**Test Cohen Kappa**
20,000	82.64	76.44
1,000	76.54	68.08
100	61.31	40.44
10	46.89	26.09

Table 4: Test results of fully supervised transformer across different training set sizes.

5.2 A simple baseline: Relative positioning

Before exploring scaled-up architectures, we first conduct a preliminary study to familiarize ourselves with the self-supervised learning pipeline and establish a baseline model for comparison. In this study, we employ a simple CNN backbone and train it using the relative positioning pretext task [16].

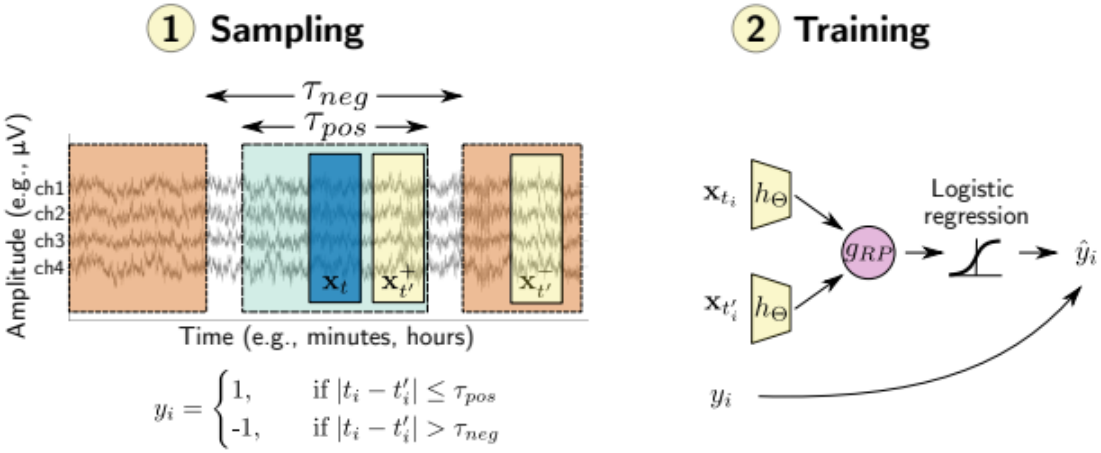


Figure 9: Relative positioning [16]

The pretext task. Relative positioning [16] is a straightforward yet effective pretext task used to train time series models to capture temporal features (Figure 9). The task is controlled by three key parameters: τ_{pos}, the positive context threshold, τ_{neg}, the negative context threshold, and T, the duration of a window. The process involves sampling two windows, (x, x'), from an EEG recording, with respective start times t_i and t_i'. If the absolute time difference between the windows satisfies $|t_i - t_i'| \leq \tau_{pos}$, the pair is labeled as positive ($y = 1$). Conversely, if $|t_i - t_i'| > \tau_{neg}$, the pair is labeled as negative ($y = -1$).

The architecture. The goal is to train an embedding neural network, h_Θ, to extract useful temporal features. A linear layer, g_{RP}, is simultaneously trained to predict whether the two windows form a positive or negative pair. The loss function used for training can be expressed as:

$$L(\Theta, w, w_0) = \sum_{(x,x',y)} \log(1 + \exp(-y[w^T|h_\Theta(x) - h_\Theta(x')| + w_0]))$$

This approach encourages the model to learn representations that capture meaningful temporal relationships in the EEG data.

Implementation details. The model architecture is a straightforward 3-layered 1D convolutional neural network (CNN). Each convolutional layer is followed by max-pooling and ReLU activation. After the final convolutional layer, the feature maps are flattened and passed through a linear projection layer. The network takes as input a raw time series of 30 seconds from a single EEG channel and outputs a latent vector of size 128. Three models are trained respectively with 10, 100 and

1000 recordings during 300 steps with Adam and a learning rate of 0.001.

Evaluation on Clean-sleep dataset. We follow the evaluation procedure of most self-supervised papers: getting downstream tasks metrics with a Knn classifier on top of the embedding model. The evaluation of knn enables us to get an idea of how well the model will perform on downstream tasks. Here, the downstream task of interest is PSG sleep staging. The results of the Knn experiment are shown in Figure 10.

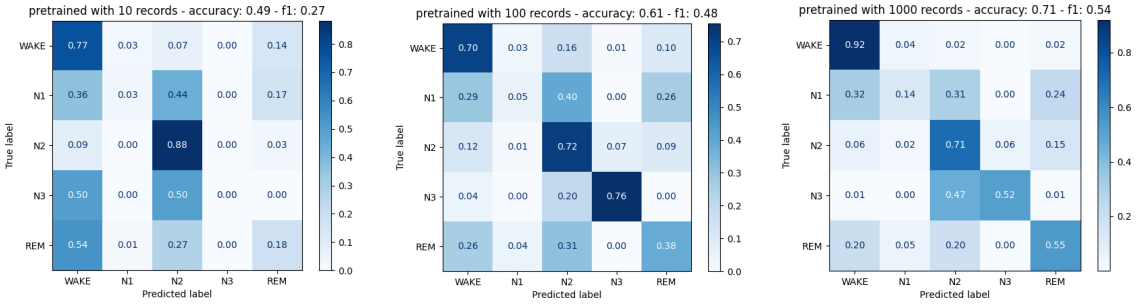
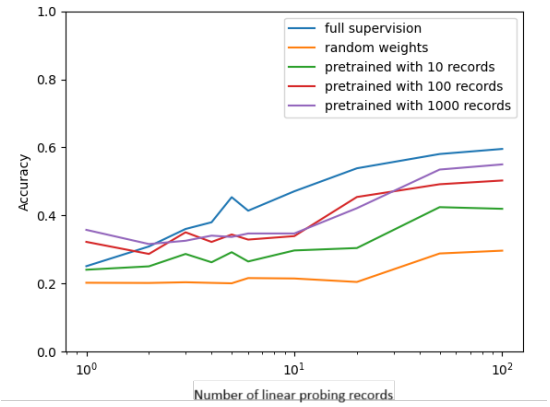
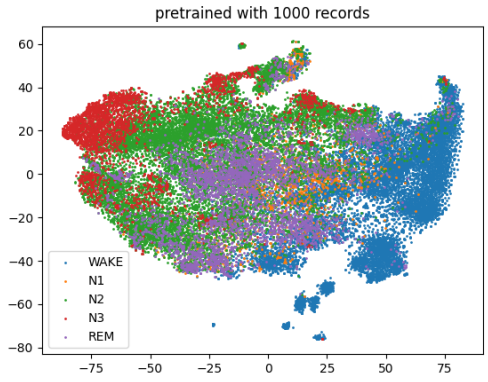


Figure 10: Relative positioning Knn results for 10, 100, and 1,000 recordings

We also evaluate the models on linear probing with different annotated data regime. Linear probing consists of freezing the pretrained encoder and train a linear layer on top for the downstream task (Figure 11a).



(a) Relative positioning linear probing results



(b) TSNE visualization of embeddings from relative positioning

Figure 11: Relative positioning results visualization.

These experiments showed that relative positioning can effectively learn useful features for sleep staging. With a simple Knn on top of the learned representation,

we perform slightly worse than the transformer trained with full supervision on 1000 recordings. Linear probing is performing worse which can be explained by the features not being linearly separable.

6 Method

6.1 Pretraining task

This section describes two pretraining pretext tasks, namely contrastive representation learning and DINO.

6.1.1 Contrastive representation learning

Contrastive representation learning is a self-supervised learning technique that aims to learn useful feature representations by contrasting positive and negative pairs of data. The primary goal is to bring similar (positive) examples closer in the embedding space while pushing dissimilar (negative) examples further apart. Positive pairs typically consist of different augmentations or views of the same data instance, while negative pairs are created from different instances. The method encourages the model to capture meaningful patterns in the data without requiring explicit labels, making it especially useful in domains with limited labeled data. We base our implementations on BIOT [23] and SimCLR [25].

Motivation for contrastive learning. Several studies [22, 23] advocate for contrastive learning over masked pretraining in EEG research. Masked pretraining, akin to its use in NLP, involves predicting missing segments of time series data, which may not effectively capture meaningful features due to high signal-to-noise ratios. Contrastive learning, on the other hand, operates similarly to masked pretraining by aiming to produce consistent representations even when parts of the input are masked. Thus, contrastive learning serves as an adaptation of masked pretraining for scenarios where direct prediction of missing segments is less suitable.

Contrastive representation learning pipeline. More specifically, the standard approach to contrastive representation learning involves defining a transformation function, $\mathcal{T}$, that introduces variations to which the learned foundation model should be invariant. From a given data point x, two distinct views, x^i and x^j, are generated by applying different transformations from $\mathcal{T}$. These views form a positive pair. The model processes these views through an encoder f followed by a projection head g, producing representations $z_i = g(f(x^i))$ and $z_j = g(f(x^j))$. The objective is to maximize the cosine similarity between these representations, encouraging the model to learn invariant features (Figure 12).

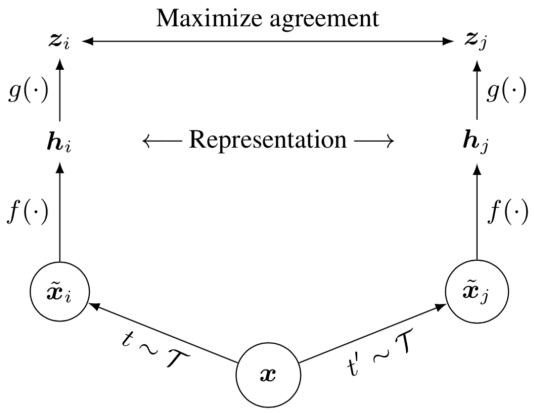


Figure 12: Contrastive representation learning pipeline [25].

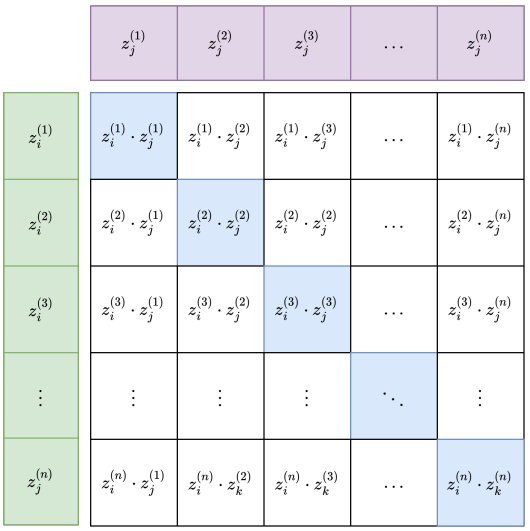


Figure 13: Similitude matrix of a batch.

InfoNCE loss. This is achieved using the Information Noise Contrastive Estimation (InfoNCE) loss. The InfoNCE loss is designed to both maximize the similarity between positive pairs and minimize the similarity with negative pairs. Let's assume we have a batch of n samples: $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$. For each sample $x^{(k)}$, we generate two views $x_i^{(k)}$ and $x_j^{(k)}$, which form a positive pair. The encoded representations are computed as $z_i^{(1)}, z_i^{(2)}, \ldots, z_i^{(n)}$ for the first view and $z_j^{(1)}, z_j^{(2)}, \ldots, z_j^{(n)}$ for the second view.

For each positive pair $z_i^{(k)}$ and $z_j^{(k)}$, the InfoNCE loss treats all other pairs in the batch as negative samples. Therefore, for each sample $z_i^{(k)}$, there is 1 positive pair and $n-1$ negative pairs. The goal is to maximize the similarity between $z_i^{(k)}$ and $z_j^{(k)}$ (the positive pair) while minimizing the similarity between $z_i^{(k)}$ and $z_j^{(l)}$ for all $l \neq k$ (the negative pairs).

The InfoNCE loss is computed as follows:

$$L = -\sum_{k=1}^{n} \log \frac{\exp(\text{sim}(z_i^{(k)}, z_j^{(k)})/\tau)}{\sum_{l=1}^{n} \exp(\text{sim}(z_i^{(k)}, z_j^{(l)})/\tau)}$$

- $\text{sim}(z_i^{(k)}, z_j^{(l)})$ is the cosine similarity between the representations of $x_i^{(k)}$ and $x_j^{(l)}$, calculated as:

$$\text{sim}(z_i^{(k)}, z_j^{(l)}) = \frac{z_i^{(k)} \cdot z_j^{(l)}}{||z_i^{(k)}|| \, ||z_j^{(l)}||}$$

- τ is a temperature hyperparameter that controls the sharpness of the distribution.

In this formulation:

- The numerator $\exp(\text{sim}(z_i^{(k)}, z_j^{(k)})/\tau)$ encourages high similarity between the positive pair $(z_i^{(k)}, z_j^{(k)})$.
- The denominator sums over all pairs, including both positive and negative pairs, and serves to push apart negative samples by minimizing their similarity:

$$\sum_{l=1}^{n} \exp(\text{sim}(z_i^{(k)}, z_j^{(l)})/\tau)$$

The symetrized loss is expressed as:

$$L = -\frac{1}{2}\sum_{k=1}^{n}\left[\log\frac{\exp(\text{sim}(z_i^{(k)}, z_j^{(k)})/\tau)}{\sum_{l=1}^{n}\exp(\text{sim}(z_i^{(k)}, z_j^{(l)})/\tau)} + \log\frac{\exp(\text{sim}(z_j^{(k)}, z_i^{(k)})/\tau)}{\sum_{l=1}^{n}\exp(\text{sim}(z_j^{(k)}, z_i^{(l)})/\tau)}\right]$$

InfoNCE loss can be understood as an n-way classification task, where each element in the batch is treated as a distinct class. The loss acts as a multi-class cross-entropy, with the similarity matrix (Figure 13) serving as the logits.

It is also important to denote that with n being the batch size, for each batch, there are n positive pairs and $n(n-1)$ negative pairs. The contrastive tasks becomes much harder with increasing batch sizes (see literature review).

6.1.2 DINO

DINO (Distillation with No Labels) [29, 30] is a self-supervised learning framework designed to learn representations without the need for labeled data. It can be seen as an extreme extension of Contrastive learning incorporating many ideas to remove the limitations of Contrastive learning. It uses a teacher-student setup, which removes the need of negative pairs and increased batch size. We implement and adapt DINO framework for time series (Figure 14).

Motivation for DINO. DINO integrates multiple innovations from self-supervised learning in computer vision, addressing limitations of contrastive learning. Unlike contrastive methods, which often require large batch sizes to increase task complexity, DINO effectively bypasses this need and allows techniques like gradient accumulation. Additionally, DINO excels at capturing both local and global information in a more explicit and structured manner, an area where contrastive learning can fall short. As a result, DINO has consistently outperformed contrastive learning approaches in computer vision tasks, making it a great choice for improving self-supervised learning.

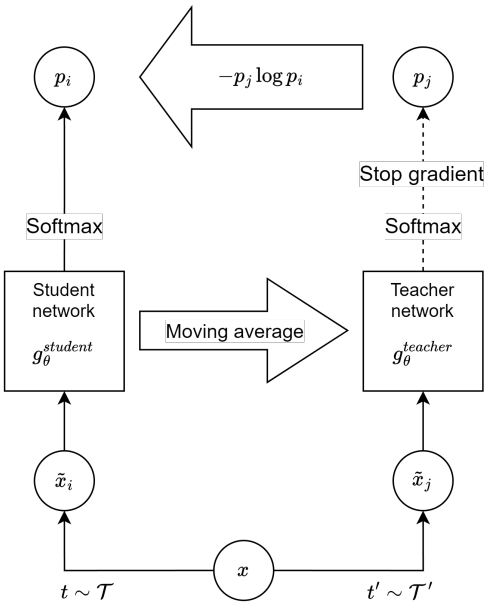


Figure 14: DINO training pipeline

DINO learning pipeline. More specifically, we define two sets of transformations: global transformations $\mathcal{T}^{\text{glob}}$ and local transformations $\mathcal{T}^{\text{local}}$, that introduce variations to which the learned foundation model should be invariant to. Local and global transformations correspond to cropping at different scales (typically, global views correspond to crops that keep more than 50% of the data while local crops keep less than 50%). At each step, given data point x, a local or a global view x^i and a global view x^j of x. The student and the teacher networks respectively output a distribution vector p_i and p_j. Before applying the softmax operation in the teacher network, we include a centering step to prevent representation collapse. This centering is achieved using Sinkhorn-Knopp (SK) batch normalization [30]. Additionally, a stop-gradient operation is applied to p_j. The student network is then trained to match the teacher's output distribution using a cross-entropy loss. The teacher's weights are updated as a moving average of the student's weights. This teacher-student framework limits gradient computation to a single branch. For further details, refer to the pseudocode (Alg. 1).

6.2 Implementation choices and motivation

Single channel encoders. We focus on single channel encoders in this study, following the approach of previous works such as [17] and [18]. Single channel encoders offer several advantages. First, they provide more flexibility for downstream tasks, as the model can easily adapt to a wide variety of input data. Second, they effectively augment the data by treating each channel as an independent view, thus

Algorithm 1 DINO PyTorch pseudocode

Input: Student and teacher networks gs, gt, teacher momentum rate l
Initialization: $gt.params = gs.params$

for x in loader **do** ▷ Load a minibatch x with n samples
$x_i^{glob}, x_i^{loc} \leftarrow t^{glob}(x), t^{loc}(x)$ ▷ Random global and local views for the student
$x_j^{glob} \leftarrow t^{glob}(x)$ ▷ Random global view for the teacher
$z_i^{glob}, z_i^{loc} \leftarrow gs(x_i^{glob}), gs(x_i^{loc})$ ▷ Student outputs
$z_j^{glob} \leftarrow SKC(gt(x_j^{glob}))$ ▷ Teacher output with SK centering
$z_j^{glob} \leftarrow z_j^{glob}.detach()$ ▷ Stop gradient for the teacher
$loss^{loc} \leftarrow \text{Crossentropy}(z_j^{glob}, z_i^{loc})$
$loss^{glob} \leftarrow \text{Crossentropy}(z_j^{glob}, z_i^{glob})$
$loss \leftarrow \frac{1}{2} \times loss^{loc} + \frac{1}{2} \times loss^{glob}$
$loss.backward()$ ▷ Back-propagate through student network
Update student and teacher
update(gs) ▷ Update student using SGD
$gt.params \leftarrow l \times gt.params + (1 - l) \times gs.params$ ▷ Momentum update for teacher
end for

increasing the available training data. This also allows for the use of different channels as distinct views of the same data, helping the model to capture richer, more diverse representations for both contrastive and DINO learnings. Then as BIOT tokenizer [23] contenates channels on time dimension, it greatly reduces the compute complexity of the architecture. Lastly, even with single channel encoders, it is possible to combine the embeddings from different channels for the downstream task, enabling the extraction of complementary information.

No channel embedding. We decided to modify BIOT tokenizer [23], excluding the channel embedding from the transformer architecture. This decision was based on preliminary findings that indicated the channel embedding had little impact on sleep staging performance. Additionally, the variability in channel structures across different datasets complicates model transferability, making it challenging to apply the model to new studies with different channel configurations. This approach aligns with the need for greater generalization across various datasets. It also makes the contrastive task more challenging, as the model may need to learn how to encode channel-specific information directly from the time series data, rather than relying on pre-defined channel embeddings.

[CLS] token. As previously mentioned, we use the [CLS] token as the aggregation function, which is a widely adopted choice in the literature. However, recent studies, such as Zhai et al. [37], suggest that mean average pooling over the output tokens

can achieve comparable performance. In future work, the aim is to replace the [CLS] token with mean average pooling, as it offers greater flexibility in the output scale. For instance, our three downstream tasks operate with one annotation per 30-second epoch, but the signal quality task requires annotations every second. This issue also arises when working with multiple epochs of 30 seconds. Given these challenges, future approaches should prioritize mean average pooling to better accommodate varying temporal resolutions and enhance generalization across tasks.

Architectures choice. Beacon Biosignals predominantly utilizes CNN architectures, such as USleep [13], in their production models. Research on self-supervised learning pipelines, such as the work by Chen et al. [25], advocates for the use of more expressive architectures. Transformers, in particular, have demonstrated superior expressivity and scalability compared to CNNs. Based on this, we perform experiments with both the USleep architecture and a Transformer architecture. In contrast to BIOT [23], we choose a Vanilla Transformer over Linformer. This choice is motivated by the significant reduction in input sequence length after switching to single-channel models, as well as the observation that the Vanilla Transformer outperforms Linformer in fully supervised settings. The USleep and Transformer architectures contain 5.1 and 5.3 million parameters, respectively. USleep consists of 11 convolutional encoder layers and 10 convolutional decoder layers with residual concatenation, while the Transformer comprises 4 transformer encoder layers with a hidden dimension of 256.

Preprocessing and augmentations We preprocess the EEG data following Beacon Biosignals' approach. This involves applying a bandpass filter between 0.4 and 35 Hz and using notch filters at 50, 60 and 62.5 Hz to remove electrical noise. The signals are resampled at 100 Hz and standardized to ensure consistency and improve model performance. The approach varies based on the training objective and architecture. Below is a description of the contrastive and DINO augmentation techniques used for the Transformer and USleep architectures:

- Contrastive Augmentation for Transformer:
  - Random Channel Selector: Select one random channel, applying contrastive learning on two different channels.
  - Short-Time Fourier Transform (STFT) using Hamming windows with parameters $n_{fft} = 100$ and $n_{hop} = 50$.
  - Segment Masking: For a uniformly chosen proportion p between 0% and 95%, each token has a probability p of being masked.

  For USleep, the same random channel selector is applied. Instead of using STFT, raw data is passed directly to the model. Masking is done by replacing data points with Gaussian noise. Unlike BIOT [23], we employ a standard

contrastive learning pipeline with augmentations on both branches, as a preliminary study indicated better sample efficiency.

- DINO Augmentation for Transformer:
  - 2 Global and 8 Local Views:
    - Global views: Out of 3000 data points, crops are made at 1000, 1500, or 2000 data points.
    - Local views: Crops are made at 100, 200, 300, or 500 data points.
  - Channel Selector: Randomly select one channel.
  - Same STFT as for contrastive learning.

  Similar changes are applied for USleep to maintain consistency across architectures.

Implementation details. DINO teacher updates updates uses a momentum m with a cosine scheduler from 0.9995 to 1. All architectures are trained with a similar training schedule: Same number of seen samples, learning rate at batch size 1024 is set to 0.001 for transformer and 0.01 for Usleep. AdamW optimizer is used for all experiments. All of them are trained with a Cosine scheduler with warmup: 20k warmup steps, 200k total steps. The number of warmup steps, total number of steps is adapted if the batch size has to be set lower. The learning rate is also adjusted if that's is the case.

6.3 Evaluation

During the training, we keep track of the RankMe [34] and the LiDAR [35] metrics. These two metrics are known to have great correlation with performances on downstream tasks. The RankMe metric is used to estimate the quality of learned representations by evaluating the "effective rank" of the embedding matrix. It is calculated as the Shannon entropy of normalized singular values of the embedding matrix. The RankMe reflects the distribution of information in the embedding space—lower entropy indicates more collapsed representations, while higher entropy suggests richer representations. LiDAR evaluates the "effective rank" of the Linear Discriminant Analysis matrix, which is computed from the embeddings of different views of the same data point.

Evaluation is conducted on three tasks: PSG staging, Dreem staging, and on-head analysis (see the datasets section for details). For PSG staging, we assess performance under different data regimes, specifically with 10, 100, and 1,000 training recordings. For Dreem staging, we follow a leave-one-out dataset strategy, as commonly used at Beacon Biosignals. For the on-head algorithm, we adopt a multi-scale approach, using 2, 23, and 235 training recordings. These three tasks enable us to evaluate the trained foundation models across varying data regimes, tasks,

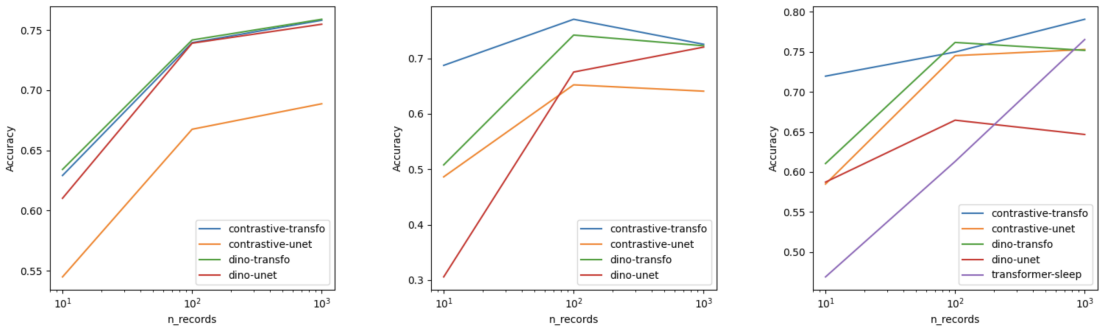
and data distributions. For each task, we assess the performance using three methods: k-nearest neighbors (k-NN) trained on the learned features, linear probing, and fine-tuning. k-NN is computationally inexpensive, linear probing requires moderate tuning and training time, while fine-tuning is more resource-intensive. For PSG and Dreem staging, we present both Accuracy and Cohen's Kappa. For the on-head algorithm, we emphasize the F1-score as the primary evaluation metric.

6.4 Experiments and results

Before scaling up the data and architectures, scaling laws (as discussed in the literature review) indicate that improvements observed at smaller scales will generalize to larger scales. Therefore, we begin by experimenting with 1,000 pretraining recordings. Next, we conduct a series of experiments using DINO. Finally, we scale up to 30,000 recordings and use larger architectures for further experimentation. Lastly, we make a brief analysis of RankMe and Lidar metrics. The experiments related to hyperparameter tuning for linear probing and fine-tuning are detailed in Appendix B.

6.4.1 Pretraining with 1,000 recordings

Results on PSG Staging. In low-data settings with 10 and 100 labeled recordings, all pretrained models outperform fully supervised models significantly (Figure 15). As expected, transfer learning helps to reduce overfitting. Among the models, the contrastive transformer delivers the best performance (Table 5). After fine-tuning, we observe only a 3% decrease in accuracy compared to full supervision, despite using just 5% of the labeled data required for fully supervised training (Section 5.1.3).



(a) KNN accuracy. (b) Linear probing accuracy. (c) Fine-tuning accuracy.

Figure 15: Performance comparison on PSG staging of models pretrained with 1,000 recordings. Transformer-sleep corresponds to the fully supervised model.

Results on Dreem staging and Onhead detection. Knn and linear probing results on Dreem staging ensure that the models learn general EEG features that are

Model Name	n_records = 10		n_records = 100		n_records = 1,000	
	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
contrastive-transfo-lp	**0.6875**	**0.5676**	**0.7706**	**0.6880**	**0.7256**	**0.6216**
contrastive-unet-lp	0.4864	0.3050	0.6524	0.5142	0.6409	0.4913
dino-transfo-lp	0.5080	0.1750	0.7422	0.6513	0.7229	0.6175
dino-unet-lp	0.3059	-0.1045	0.6755	0.5338	0.7204	0.5955
contrastive-transfo-ft	**0.7197**	**0.6169**	0.7498	0.6607	**0.7907**	**0.7123**
contrastive-unet-ft	0.5849	0.4140	0.7453	0.6537	0.7530	0.6663
dino-transfo-ft	0.6106	0.4858	**0.7618**	**0.6750**	0.7519	0.6597
dino-unet-ft	0.5873	0.4483	0.6647	0.5379	0.6468	0.5139
transformer-sleep	0.4689	0.2609	0.6131	0.4044	0.7654	0.6808

Table 5: PSG staging linear probing (lp) and fine-tuning (ft) results with 1,000 pretraining recordings.

also useful for Dreem staging (Table 6). However, our models still fall significantly short of the 85.9% accuracy achieved by the production models.

Model Name	Knn		Linear Probing		Fine-tuning	
	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
contrastive-transfo	**0.7120**	**0.5717**	**0.7293**	**0.6092**	0.7700	0.6677
contrastive-unet	0.5665	0.3255	0.5525	0.3138	0.7348	0.6116
dino-transfo	0.6991	0.5570	0.7021	0.5665	**0.7764**	**0.6781**
dino-unet	0.6992	0.5554	0.6294	0.4401	0.7068	0.5796

Table 6: Dreem staging results with 1,000 pretraining recordings.

Model Name	n_records = 2	n_records = 23	n_records = 235
contrastive-transfo	**0.8338**	0.9247	**0.9757**
contrastive-unet	0.5962	0.7835	0.9240
dino-transfo	0.8017	**0.9291**	0.9718
dino-unet	0.8172	0.4438	0.8037

Table 7: Onhead detection fine-tuning results (F1 scores).

We observe similar trends in the On-head detection task (Table 7), where the contrastive-transformer model slightly outperforms the dino-transformer model. Yet, even with the full dataset, we are far from reaching the 0.997 F1 score of the production model.

Dreem staging dataset and Onhead detection dataset have respectively 249 and 235 training recordings which should still be in the range where pertaining mitigates overfitting. One potential reason for this discrepancy may be the considerable imbalance in the Dreem data used during pretraining, where Dreem data makes up

less than 1% of the dataset, in contrast to PSG data. Additionally, for On-head detection, the production models have access to accelerometer time series data, which is not available to our EEG-only models.

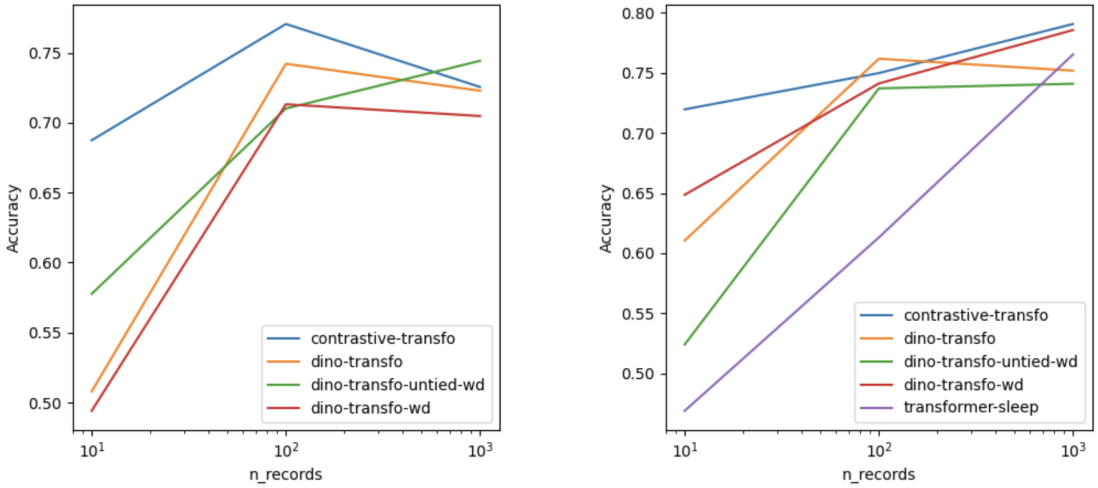
Experiment conclusion. When fine-tuning (Figure 15c), the contrastive-transformer model significantly outperforms all other models, including the fully supervised model. Unet architecture falls short compared to transformer architecture despite having equivalent number of parameters. This can be explained by the masking strategy being more advantageous for the transformer as the Unet is fed with random data while the transformer just does not see anything. The transformer architecture is also known to be more expressive because of its attention layers. We have decided to focus on transformer-based architectures for the subsequent experiments, as they demonstrate superior performance overall.

A notable trend emerges in the linear probing results (Figure 15b), where accuracy decreases as we increase the number of labeled recordings from 100 to 1,000. This decline can likely be attributed to the embeddings not being linearly separable, which makes it harder for the linear probe to perform well with larger datasets.

6.4.2 Experiments on DINO and weight decay with 1,000 recordings

Even though contrastive learning shows slightly better results than DINO, we opted to conduct further experiments with DINO using 1,000 recordings. This choice is motivated by DINO's scalability compared to contrastive learning. The latter depends on large batch sizes to generate numerous negative pairs, which becomes computationally expensive. As we plan to scale up to longer sequences and larger architectures, DINO's learning approach offers an advantage by not requiring negative pairs, allowing for the use of gradient accumulation when necessary. Moreover, we believe that DINO is better at capturing local and global features as it is explicitly trained to do so.

Upon a closer review of the original DINO implementations [29, 30], we note that they employ weight decay schedulers that start with a low weight decay of 0.004, gradually increasing to 0.4 with a Cosine schedule. In contrast, the contrastive learning method [25] only utilizes a low weight decay throughout. Furthermore, Kaplan et al. [36] recommend using a low weight decay for the backbone and a higher one for the projection heads during pretraining. We test both strategies and evaluate their performance on DINO. The weight decay scheduler enhances the fine-tuning accuracy (Figure 16), bringing the performance of dino-transformer-wd in line with that of contrastive-transformer (Figure 16b). For the remainder of the study, we will refer to dino-transformer-wd simply as dino-transformer, in alignment with the original paper. The complete evaluation of all models is available in Appendix C.



(a) Linear Probing accuracy.

(b) Fine-Tuning accuracy.

Figure 16: Effect of weight decay strategies on PSG staging downstream task.

Model Name	n_records = 10		n_records = 100		n_records = 1,000	
	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
contrastive-transfo	**0.7197**	**0.6169**	0.7498	0.6607	**0.7907**	**0.7123**
dino-transfo	0.6106	0.4858	**0.7618**	**0.6750**	0.7519	0.6597
dino-transfo-untied-wd	0.5240	0.3561	0.7371	0.6412	0.7410	0.6474
dino-transfo-wd	0.6486	0.5107	0.7412	0.6522	0.7857	0.7039
transformer-sleep	0.4689	0.2609	0.6131	0.4044	0.7654	0.6808

Table 8: PSG staging with fine-tuning results for weight decay experiments.

6.4.3 Scaling up experiments

This section focuses on scaled-up experiments. Initially, we attempt to scale up the dataset without altering the architecture, maintaining the same 4-layer vanilla transformer with 8 heads and a latent space of 256, which results in a transformer with 5M parameters. However, increasing the dataset size did not lead to improved downstream performance (Table 9 and 10). Additional experiments, involving the scaling of both the architecture and the dataset, are currently in progress. While k-NN results clearly demonstrate the advantages of larger architectures, the fine-tuning outcomes are less conclusive. Since we used the same hyperparameters for fine-tuning the 25M parameter models as with the smaller ones, we believe that optimizing the hyperparameters could lead to further improvements in fine-tuning performance especially for DINO.
A complete evaluation of all pretrained models is available in Appendix C.

Model Name	n_records = 10		n_records = 100		n_records = 1,000	
	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
contrastive-transfo	0.6292	0.4718	0.7396	0.6243	0.7582	0.6518
contrastive-transfo-30k	0.5989	0.4468	0.7438	0.6353	0.7677	0.6697
contrastive-transfo-30k-25M	0.5724	0.4151	**0.7542**	**0.6568**	**0.7793**	**0.6926**
dino-transfo	0.6342	0.4727	0.7418	0.6275	0.7590	0.6538
dino-transfo-30k	0.5614	0.4003	0.7298	0.6122	0.7620	0.6607
dino-transfo-30k-25M	0.5675	0.3938	0.7430	0.6419	0.7714	0.6801
dino-transfo-wd	**0.6414**	**0.4838**	0.7342	0.6148	0.7547	0.6471

Table 9: PSG staging Knn results for scaled up experiments.

Model Name	n_records = 10		n_records = 100		n_records = 1,000	
	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
contrastive-transfo	**0.7197**	**0.6169**	0.7498	0.6607	0.7907	0.7123
contrastive-transfo-30k	0.6699	0.5463	0.7517	0.6595	0.7703	0.6901
contrastive-transfo-30k-25M	0.6172	0.4645	0.7528	0.6660	**0.8002**	**0.7249**
dino-transfo	0.6106	0.4858	**0.7618**	**0.6750**	0.7519	0.6597
dino-transfo-30k	0.6917	0.5463	0.7354	0.6427	0.7750	0.6987
dino-transfo-30k-25M	0.5590	0.3715	0.6685	0.5572	0.7040	0.5981
dino-transfo-wd	0.6486	0.5107	0.7412	0.6522	0.7857	0.7039
transformer-sleep	0.4689	0.2609	0.6131	0.4044	0.7654	0.6808

Table 10: PSG staging fine-tuning results for scaled up experiments.

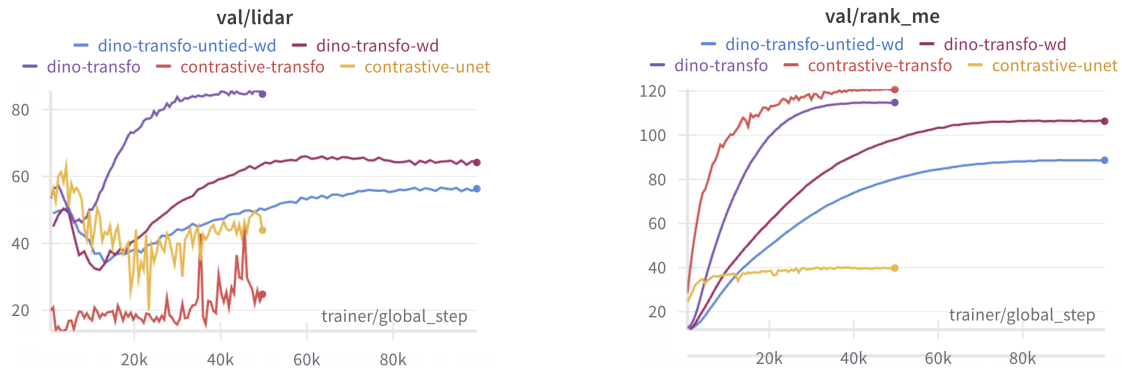
6.4.4 On the analysis of LiDAR and RankMe

We report the LiDAR [35] and RankMe [34] metrics (Figure 17), as they may play a crucial role in model selection, particularly if a metric computable during the pre-training phase becomes available. Contrastive models exhibit relatively low LiDAR scores compared to DINO models, which does not correspond well with downstream fine-tuning accuracies. The LiDAR score is highly sensitive to the augmentation method used, which likely explains the lower scores observed for contrastive models. As for RankMe, the results are inconclusive. For instance, dino-transfo-wd achieved similar results to contrastive-transfo, yet there is a significant difference in their RankMe scores.

6.4.5 Future directions

Scaling up the architecture. scaling up the architectures is crucial, as larger models tend to perform better when trained on larger datasets [36]. The current 5M-parameter transformer is on the smaller end of the scaling curve, and moving towards larger models could yield significant performance gains. Additional hyper-parameter tuning may be required to fully leverage the capabilities of scaled-up models.

More context length. Another avenue is training with longer sequences. EEG



(a) LiDAR (b) RankMe

Figure 17: LiDAR and RankMe for 1000 recordings pretraining runs

data typically spans over several minutes, but working with 30-second windows imposes limitations, likely forcing the model to focus on more local features. By increasing the sequence length, we can potentially improve the efficiency of augmentations and allow the model to capture more global patterns. Models in the literature mostly focus on more than 30 epochs at a time for sleep staging as it improves the staging accuracy [15]. Instead of pretraining with multiple epochs, a fine-tuning head could be designed to take embeddings from multiple epochs as input to predict the class labels for each epoch simultaneously. This approach could be a straightforward way to explore the potential benefits of increasing input length and may improve performance on tasks requiring more contextual information.

Cleaning of the dataset. A growing part of scaling works in both Nature language processing [48] and Computer vision [30] focus on data quality rather than data quantity. Especially in DINOv2, an ablation is done on data cleaning, showing that having good data quality is crucial for the training of DINO (Figure 18). That avenue is even more important as EEG data is likely to have low sample variation compared to images.

Extanding Dreem pretraining data. As previously observed, our foundation models underperform compared to production models, even when fine-tuned on the same datasets. This discrepancy is likely due to the imbalance in the pretraining dataset, which consists of 99% PSG data and only 1% Dreem data. Evaluating the downstream performance of models pretrained on a larger share of Dreem data would be valuable.

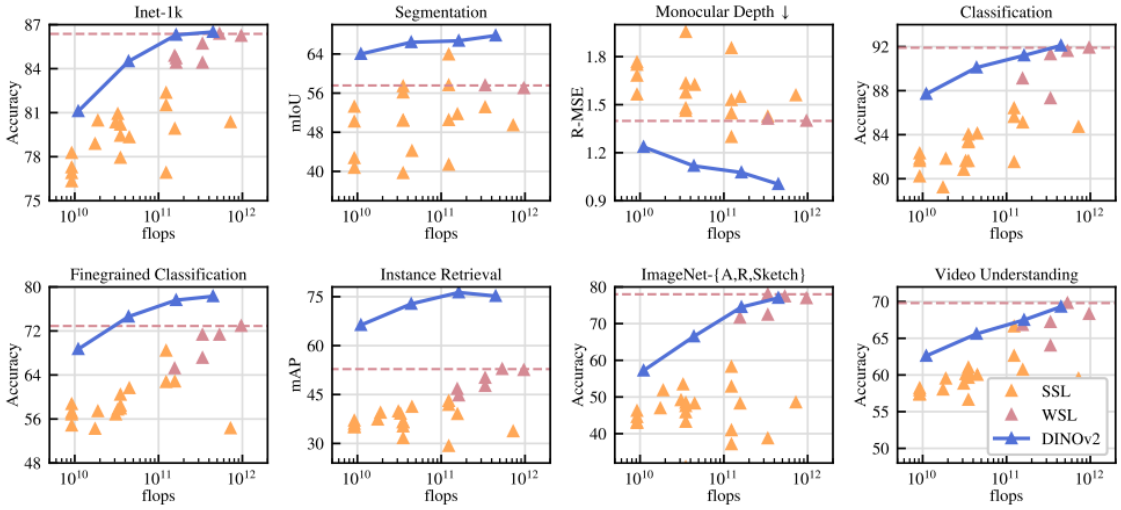


Figure 18: DINOv2 [30] data cleaning ablation. DINOv2 with cleaning (blue) is compared to without cleaning (Orange) and weakly supervised learning (Pink)

7 Conclusion

In this study, we explored the development of EEG foundation models, leveraging recent advances in self-supervised learning techniques like contrastive learning and DINO learning. Our results demonstrate the capability of these pretrained models to generalize well on commonly used PSG data, achieving competitive performance with fully supervised models while using significantly less annotated data. Specifically, the model trained with just 5% of the labeled dataset achieved only a 3% drop in accuracy compared to fully supervised methods.

However, when applied to Dreem-acquired data, the model's performance is less remarkable, reaching 81% accuracy compared to 85% from existing production models. This discrepancy can largely be attributed to the underrepresentation of Dreem data in the pretraining set, which accounted for less than 1% of the total data used.

Despite these limitations, the foundation models present promising potential. By incorporating more Dreem-specific data and refining the pretraining strategies, the models' performance could be significantly improved. Moving forward, scaling up both the architectures and training sequences will be critical to unlocking the full potential of these models in more specialized EEG tasks.

References

[1] Pierrick J Arnal, Valentin Thorey, Eden Debellemaniere, Michael E Ballard, Albert Bou Hernandez, Antoine Guillot, Hugo Jourde, Mason Harris, Mathias Guillard, Pascal Van Beers, et al. The dreem headband compared to polysomnography for electroencephalographic signal acquisition and sleep staging. *Sleep*, 43(11):zsaa097, 2020.

[2] Richard B Berry, Rohit Budhiraja, Daniel J Gottlieb, David Gozal, Conrad Iber, Vishesh K Kapur, Carole L Marcus, Reena Mehra, Sairam Parthasarathy, Stuart F Quan, et al. Rules for scoring respiratory events in sleep: update of the 2007 aasm manual for the scoring of sleep and associated events: deliberations of the sleep apnea definitions task force of the american academy of sleep medicine. *Journal of clinical sleep medicine*, 8(5):597–619, 2012.

[3] Anna K Morin, Courtney I Jarvis, and Ann M Lynch. Therapeutic options for sleep-maintenance and sleep-onset insomnia. *Pharmacotherapy: The Journal of Human Pharmacology and Drug Therapy*, 27(1):89–110, 2007.

[4] Lesley S Bennett, Beverly A Langford, John R Stradling, and Robert JO Davies. Sleep fragmentation indices as predictors of daytime sleepiness and ncpap response in obstructive sleep apnea. *American journal of respiratory and critical care medicine*, 158(3):778–786, 1998.

[5] Joel Reiter, Eliot Katz, Thomas E Scammell, and Kiran Maski. Usefulness of a nocturnal soremp for diagnosing narcolepsy with cataplexy in a pediatric population. *Sleep*, 38(6):859–865, 2015.

[6] Christian Berthomier, Xavier Drouot, Maria Herman-Stoïca, Pierre Berthomier, Jacques Prado, Djibril Bokar-Thire, Odile Benoit, Jérémie Mattout, and Marie-Pia d'Ortho. Automatic analysis of single-channel sleep eeg: validation in healthy individuals. *Sleep*, 30(11):1587–1595, 2007.

[7] Tarek Lajnef, Sahbi Chaibi, Perrine Ruby, Pierre-Emmanuel Aguera, Jean-Baptiste Eichenlaub, Mounir Samet, Abdennaceur Kachouri, and Karim Jerbi. 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.

[8] Jeroen Van Der Donckt, Jonas Van Der Donckt, Emiel Deprost, Nicolas Vandenbussche, Michael Rademaker, Gilles Vandewiele, and Sofie Van Hoecke. Do not sleep on traditional machine learning: Simple and interpretable techniques are competitive to deep learning for sleep scoring. *Biomedical Signal Processing and Control*, 81:104429, 2023.

[9] Stanislas Chambon, Mathieu N Galtier, Pierrick J Arnal, Gilles Wainrib, and Alexandre 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.

[10] Akara Supratak, Hao Dong, Chao Wu, and Yike Guo. Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg. *IEEE transactions on neural systems and rehabilitation engineering*, 25(11):1998–2008, 2017.

[11] Huy Phan, Fernando Andreotti, Navin Cooray, Oliver Y Chén, and Maarten 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.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[13] Mathias Perslev, Sune Darkner, Lykke Kempfner, Miki Nikolic, Poul Jørgen Jennum, and Christian Igel. U-sleep: resilient high-frequency sleep staging. *NPJ digital medicine*, 4(1):72, 2021.

[14] Antoine Guillot and Valentin Thorey. Robustsleepnet: Transfer learning for automated sleep staging at scale. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:1441–1451, 2021.

[15] Antoine Guillot, Fabien Sauvet, Emmanuel H During, and Valentin Thorey. Dreem open datasets: Multi-scored sleep datasets to compare human and automated sleep staging. *IEEE transactions on neural systems and rehabilitation engineering*, 28(9):1955–1965, 2020.

[16] Hubert Banville, Omar Chehab, Aapo Hyvärinen, Denis-Alexander Engemann, and Alexandre Gramfort. Uncovering the structure of clinical eeg signals with self-supervised learning. *Journal of Neural Engineering*, 18(4):046020, 2021.

[17] Mostafa Neo Mohsenvand, Mohammad Rasool Izadi, and Pattie Maes. Contrastive representation learning for electroencephalogram classification. In *Machine Learning for Health*, pages 238–253. PMLR, 2020.

[18] Rahul Thapa, Bryan He, Magnus Ruud Kjaer, Hyatt Moore, Gauri Ganjoo, Emmanuel Mignot, and James Zou. Sleepfm: Multi-modal representation learning for sleep across brain activity, ecg and respiratory signals. *arXiv preprint arXiv:2405.17766*, 2024.

[19] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, Xiaoli Li, and Cuntai Guan. Self-supervised contrastive representation learning for semi-supervised time-series classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[20] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

[21] Wenhui Cui, Woojae Jeong, Philipp Thölke, Takfarinas Medani, Karim Jerbi, Anand A Joshi, and Richard M Leahy. Neuro-gpt: developing a foundation model for eeg. *arXiv preprint arXiv:2311.03764*, 2023.

[22] Navid Mohammadi Foumani, Geoffrey Mackellar, Soheila Ghane, Saad Irtza, Nam Nguyen, and Mahsa Salehi. Eeg2rep: enhancing self-supervised eeg representation through informative masked inputs. *arXiv preprint arXiv:2402.17772*, 2024.

[23] Chaoqi Yang, M Brandon Westover, and Jimeng Sun. Biot: Cross-data biosignal learning in the wild. *arXiv preprint arXiv:2305.10351*, 2023.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[25] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[26] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6707–6717, 2020.

[27] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[28] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[29] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[30] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[31] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, pages 9929–9939. PMLR, 2020.

[32] Xianghong Fang, Jian Li, Qiang Sun, and Benyou Wang. Rethinking the uniformity metric in self-supervised learning. *arXiv preprint arXiv:2403.00642*, 2024.

[33] Young-Jin Park, Hao Wang, Shervin Ardeshir, and Navid Azizan. Quantifying representation reliability in self-supervised learning models. *arXiv preprint arXiv:2306.00206*, 2023.

[34] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. In *International conference on machine learning*, pages 10929–10974. PMLR, 2023.

[35] Vimal Thilak, Chen Huang, Omid Saremi, Laurent Dinh, Hanlin Goh, Preetum Nakkiran, Joshua M Susskind, and Etai Littwin. Lidar: Sensing linear probing performance in joint embedding ssl architectures. *arXiv preprint arXiv:2312.04000*, 2023.

[36] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[37] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.

[38] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2829, 2023.

[39] S. Zafar, T. Loddenkemper, J. W. Lee, A. Cole, D. Goldenholz, J. Peters, A. Lam, E. Amorim, C. Chu, S. Cash, V. Moura Junior, A. Gupta, M. Ghanta, M. Fernandes, H. Sun, J. Jing, and M. B. Westover. Harvard electroencephalography database (version 2.0), 2023. URL `https://doi.org/10.60508/g6m4-bf96`.

[40] Iyad Obeid and Joseph Picone. The temple university hospital eeg data corpus. *Frontiers in neuroscience*, 10:196, 2016.

[41] Guo-Qiang Zhang, Licong Cui, Remo Mueller, Shiqiang Tao, Matthew Kim, Michael Rueschman, Sara Mariani, Daniel Mobley, and Susan Redline. The national sleep research resource: towards a sleep data commons. *Journal of the American Medical Informatics Association*, 25(10):1351–1358, 2018.

[42] Terri Blackwell, Kristine Yaffe, Sonia Ancoli-Israel, Susan Redline, Kristine E Ensrud, Marcia L Stefanick, Alison Laffan, Katie L Stone, and Osteoporotic Fractures in Men Study Group. Associations between sleep architecture and sleep-disordered breathing and cognition in older community-dwelling men: the osteoporotic fractures in men sleep study. *Journal of the American Geriatrics Society*, 59(12):2217–2225, 2011.

[43] Stuart F Quan, Barbara V Howard, Conrad Iber, James P Kiley, F Javier Nieto, George T O'Connor, David M Rapoport, Susan Redline, John Robbins, Jonathan M Samet, et al. The sleep heart health study: design, rationale, and methods. *Sleep*, 20(12):1077–1085, 1997.

[44] Jens B Stephansen, Alexander N Olesen, Mads Olsen, Aditya Ambati, Eileen B Leary, Hyatt E Moore, Oscar Carrillo, Ling Lin, Fang Han, Han Yan, et al. Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy. *Nature communications*, 9(1):5229, 2018.

[45] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[46] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[47] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[48] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

A Exploring contrastive learning with BIOT

Before transitioning to larger models with more computational resources, we performed a series of experiments to inform our parameter choices. We pretrained BIOT [23], aiming to stay as close as possible to the original paper. Our investigation focused on several key factors:

- **Filtering EEG:** Applying a bandpass filter between 0.4 and 35 Hz, along with notch filters to remove electrical noise.
- **Using channel embeddings:** This serves two purposes—first, it simplifies the application of the foundation model since EEG channels are often inconsistent, and second, it adds complexity to the learning task.
- **Increasing the signal masking proportion:** The masking probability is drawn uniformly between 0 and the "masking proportion," resulting in an average of "masking proportion/2" data points being masked.
- **Increasing batch size:** This was done to increase the difficulty of the pretraining task.

Batch Size	Filtering EEG	Channel Embeddings	Masking Proportion	Knn accuracy	Knn F1 score
64	No	Yes	0.5	0.73	0.63
64	Yes	Yes	0.5	0.74	0.65
64	Yes	No	0.5	0.74	0.65
256	Yes	No	0.95	**0.75**	**0.66**

Table 11: Preliminary study on various hyperparameters.

B Fine-tuning and linear probing tuning experiments

Experiments were conducted on the DINO Transformer model, pretrained with 1,000 recordings on PSG staging.

Linear Probing. As outlined in the DINO paper [29], linear probing achieves better performance when using a cosine learning rate scheduler with warmup and appropriate augmentations. These techniques help optimize the model's use of the pretrained representations, providing better accuracy during evaluation.

Fine-Tuning. Fine-tuning requires careful parameter adjustment. As noted in the DINO paper, pretrained models are challenging to fine-tune effectively. In comparison, we found contrastive models easier to tune. Our fine-tuning process involved several steps:

- We initially conducted a learning rate sweep across values [0.0001, 0.0005, 0.001, 0.01, 0.1], but this yielded poor results.
- Experimenting with a fine-tuning scheduler, we adopted a strategy of gradually unfreezing the layers while using low learning rates to prevent overwriting the learned representations too quickly.
- Using a large batch size of 1024 (similar to the pretraining phase) and a cosine learning rate schedule with warmup, we found that a learning rate of 5e-5, which is more than 10 times lower than the one used during training, led to improved performance.
- While augmentations were helpful during fine-tuning, they did not provide significant improvements for linear probing.
- Contrary to many papers that suggest using no weight decay and the SGD optimizer during the fine-tuning phase, we achieved our best results with the AdamW optimizer and a low weight decay of 1e-4. Removing weight decay led to rapid overfitting.
- We also verified that the same set of hyperparameters was effective across different tasks and models. However, it remains uncertain if further tuning of individual models would result in additional improvements.

C Full results

Model Name	**n_records = 10**		**n_records = 100**		**n_records = 1,000**	
	Accuracy	**Kappa**	**Accuracy**	**Kappa**	**Accuracy**	**Kappa**
contrastive-transfo	0.6292	0.4718	0.7396	0.6243	0.7582	0.6518
contrastive-transfo-30k	0.5989	0.4468	<u>0.7438</u>	0.6353	0.7677	0.6697
contrastive-transfo-30k-25M	0.5724	0.4151	**0.7542**	**0.6568**	**0.7793**	**0.6926**
contrastive-unet	0.5449	0.3682	0.6676	0.5179	0.6888	0.5490
dino-transfo	0.6342	0.4727	0.7418	0.6275	0.7590	0.6538
dino-transfo-30k	0.5614	0.4003	0.7298	0.6122	0.7620	0.6607
dino-transfo-30k-25M	0.5675	0.3938	0.7430	<u>0.6419</u>	<u>0.7714</u>	<u>0.6801</u>
dino-transfo-untied-wd	<u>0.6389</u>	<u>0.4834</u>	0.7369	0.6194	0.7555	0.6480
dino-transfo-wd	**0.6414**	**0.4838**	0.7342	0.6148	0.7547	0.6471
dino-unet	0.6102	0.4383	0.7391	0.6229	0.7549	0.6475

Table 12: PSG staging Knn results.

Model Name	n_records = 10		n_records = 100		n_records = 1,000	
	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
contrastive-transfo	**0.6875**	**0.5676**	0.7706	0.6880	0.7256	0.6216
contrastive-transfo-30k	0.6772	0.5650	0.7404	0.6435	**0.7518**	**0.6673**
contrastive-transfo-30k-25M	0.6010	0.4554	**0.7788**	**0.7022**		
contrastive-unet	0.4864	0.3050	0.6524	0.5142	0.6409	0.4913
dino-transfo	0.5080	0.1750	0.7422	0.6513	0.7229	0.6175
dino-transfo-30k	0.5353	0.3617	0.6704	0.5351	0.7002	0.5836
dino-transfo-30k-25M	0.5590	0.3667	0.6695	0.5843		
dino-transfo-untied-wd	0.5777	0.4042	0.7102	0.6017	0.7443	0.6458
dino-transfo-wd	0.4941	0.3019	0.7132	0.5910	0.7048	0.5898
dino-unet	0.3059	-0.1045	0.6755	0.5338	0.7204	0.5955

Table 13: PSG staging linear probing results.

Model Name	n_records = 10		n_records = 100		n_records = 1,000	
	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
contrastive-transfo	**0.7197**	**0.6169**	0.7498	0.6607	0.7907	0.7123
contrastive-transfo-30k	0.6699	0.5463	0.7517	0.6595	0.7703	0.6901
contrastive-transfo-30k-25M	0.6172	0.4645	0.7528	0.6660	**0.8002**	**0.7249**
contrastive-unet	0.5849	0.4140	0.7453	0.6537	0.7530	0.6663
dino-transfo	0.6106	0.4858	**0.7618**	**0.6750**	0.7519	0.6597
dino-transfo-30k	0.6917	0.5463	0.7354	0.6427	0.7750	0.6987
dino-transfo-30k-25M	0.5590	0.3715	0.6685	0.5572	0.7040	0.5981
dino-transfo-untied-wd	0.5240	0.3561	0.7371	0.6412	0.7410	0.6474
dino-transfo-wd	0.6486	0.5107	0.7412	0.6522	0.7857	0.7039
dino-unet	0.5873	0.4483	0.6647	0.5379	0.6468	0.5139
transformer-sleep	0.4689	0.2609	0.6131	0.4044	0.7654	0.6808

Table 14: PSG staging fine-tuning results.

Model Name	Knn		Linear Probing		Fine-tuning	
	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
contrastive-transfo	**0.7120**	**0.5717**	0.7293	0.6092	0.7700	0.6677
contrastive-transfo-30k	0.7075	0.5685	**0.7513**	**0.6457**	0.7589	0.6581
contrastive-transfo-30k-25M	0.6763	0.5237			**0.8123**	**0.7456**
contrastive-unet	0.5665	0.3255	0.5525	0.3138	0.7348	0.6116
dino-transfo	0.6991	0.5570	0.7021	0.5665	0.7764	0.6781
dino-transfo-30k	0.6683	0.4961	0.6147	0.3875	0.7450	0.6378
dino-transfo-30k-25	0.6491	0.4791			0.7174	0.6144
dino-transfo-untied-wd	0.7009	0.5593	0.6583	0.5123	0.7522	0.6459
dino-transfo-wd	0.6952	0.5419	0.7218	0.5620	0.7506	0.6503
dino-unet	0.6992	0.5554	0.6294	0.4401	0.7068	0.5796

Table 15: Dreem staging results.

Model Name	n_records = 23				n_records = 235			
	F1	PPV	Sensitivity	Specificity	F1	PPV	Sensitivity	Specificity
contrastive-transfo	0.9116	0.8480	0.9855	0.5335	0.9542	0.9304	0.9793	0.8425
contrastive-transfo-30k	0.9073	0.8390	0.9878	0.5368	**0.9565**	0.9361	0.9779	0.7908
contrastive-unet	0.8523	0.7677	0.9579	0.2398	0.8417	0.8108	0.8750	0.5728
dino-transfo	0.9079	0.8391	0.9889	0.4854	0.9455	0.9243	0.9677	0.7830
dino-transfo-30k	**0.9137**	0.8521	0.9848	0.5514	0.9556	0.9463	0.9651	0.8313
dino-transfo-untied-wd	0.9125	0.8478	0.9879	0.4850	0.9482	0.9355	0.9613	0.8225
dino-transfo-wd	0.8887	0.8115	0.9822	0.5123	0.9433	0.9280	0.9592	0.8232
dino-unet	0.9081	0.8688	0.9512	0.6355	0.9433	0.9208	0.9669	0.7671

Table 16: Onhead detection Knn results.

Model Name	n_records = 2				n_records = 23				n_records = 235			
	F1	PPV	Sensitivity	Specificity	F1	PPV	Sensitivity	Specificity	F1	PPV	Sensitivity	Specificity
contrastive-transfo	0.5355	0.6591	0.4509	0.0000	0.8133	0.6855	0.8679	0.0181	0.9331	0.9314	0.9347	0.8556
contrastive-transfo-30k	0.6929	0.8639	0.5785	0.7895	0.9291	0.9248	0.9334	0.8439	**0.9644**	0.9736	0.9555	0.9347
contrastive-unet	0.6379	0.6912	0.5923	0.0000	0.6993	0.7117	0.6874	0.0025	0.7464	0.7101	0.7866	0.2758
dino-transfo	0.3574	0.6594	0.2305	0.1971	0.9147	0.8651	0.9704	0.6307	0.9440	0.9566	0.9318	0.9061
dino-transfo-30k	0.6339	0.5818	0.5778	0.0000	0.9223	0.9345	0.9104	0.8150	0.9332	0.9272	0.9394	0.8415
dino-transfo-untied-wd	0.7103	0.7874	0.6470	0.6010	0.9187	0.8959	0.9427	0.7216	0.9489	0.9520	0.9460	0.8822
dino-transfo-wd	**0.7195**	0.5619	0.7156	0.0000	**0.9376**	0.9461	0.9292	0.8823	0.9340	0.9279	0.9401	0.8058
dino-unet	0.0552	0.6260	0.0287	0.0000	0.6215	0.7111	0.5260	0.0000	0.7674	0.7044	0.8427	0.3035

Table 17: Onhead detection linear probing results.

Model Name	n_records = 2				n_records = 23				n_records = 235			
	F1	PPV	Sensitivity	Specificity	F1	PPV	Sensitivity	Specificity	F1	PPV	Sensitivity	Specificity
contrastive-transfo	0.6220	0.6983	0.5976	0.2918	0.9396	0.9792	0.9031	0.9382	0.9690	0.9658	0.9723	0.9017
contrastive-transfo-30k	0.4157	0.7596	0.2861	0.8157	**0.9556**	0.9928	0.9211	0.9807	**0.9758**	0.9904	0.9617	0.9765
dino-transfo	0.6707	0.6488	0.7025	0.2101	0.9362	0.9690	0.9057	0.9332	0.9484	0.9722	0.9257	0.9343
dino-transfo-30k	0.6520	0.7960	0.5522	0.6526	0.9307	0.9766	0.8888	0.9431	0.9527	0.9715	0.9347	0.9328
dino-transfo-untied-wd	**0.8502**	0.7413	0.9965	0.0029	0.9108	0.9833	0.8482	0.9584	0.9632	0.9652	0.9612	0.9119
dino-transfo-wd	0.6891	0.6604	0.7205	0.0669	0.9349	0.9834	0.8910	0.9568	0.9594	0.9718	0.9473	0.9191
dino-unet	0.0862	0.7490	0.0473	0.9644	0.8505	0.8353	0.8663	0.4687	0.8175	0.7832	0.8552	0.4290

Table 18: Onhead detection fine-tuning results.